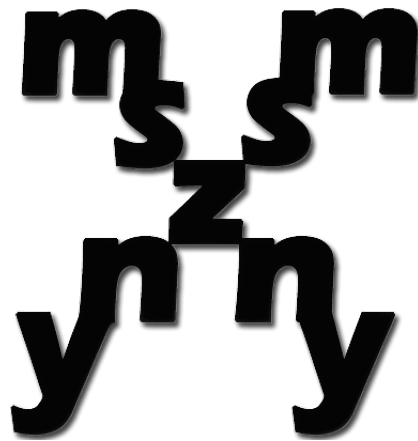


XXI. Magyar Számítógépes  
Nyelvészeti Konferencia



Szerkesztette:  
Berend Gábor  
Gosztolya Gábor  
Vincze Veronika

Szeged, 2025. február 6-7.

**Szerkesztette:**

Berend Gábor, Gosztolya Gábor, Vincze Veronika  
{berendg,ggabor,vinczev}@inf.u-szeged.hu

**Felelős kiadó:**

Szegedi Tudományegyetem  
TTIK, Informatikai Intézet  
6720 Szeged, Árpád tér 2.

**ISBN:** 978-963-688-034-7

Szeged, 2025. február

**Az MSZNY 2025 konferencia szervezője:**

HUN-REN-SZTE Mesterséges Intelligencia Kutatócsoport

**Támogatók:**

K&H Bank  
Neumann János Számítógéptudományi Társaság  
SZTE Informatikai Intézet

## Előszó

2025. február 6–7-én immáron huszonegyedik alkalommal kerül sor a Magyar Számítógépes Nyelvészeti Konferencia megrendezésére. A konferencia fő célkitűzése a kezdetek óta állandó: lehetőséget biztosítani a nyelv- és beszédtechnológia területén végzett kutatások eredményeinek ismertetésére és megvitatására, ezen felül pedig a különféle hallgatói projektek, illetve ipari alkalmazások bemutatására.

Az idei évben a 23 beküldött cikkből gondos mérlegelést követően 20 cikk került elfogadásra, melyek témája a nyelv- és beszédtechnológia számos szakterületét lefedi a legújabb nyelvi modellek bemutatásától kezdve a beszédtechnológia eredményein keresztül egészen a chatbotos alkalmazásokig.

Nagy örömet jelent számunkra, hogy Szondy Máté elfogadta meghívásunkat, aki plenáris előadását *“Munkatárs, partner vagy ellenség? A mesterséges intelligenciával való kapcsolat pszichológiai kérdései”* címmel fogja megtartani.

Az idei évben is díjazzuk a konferencia legjobb cikkét, mely a legjelentősebb eredményekkel járul hozzá a magyarországi nyelv- és beszédtechnológiai kutatásokhoz. Ezen felül immár hetedik alkalommal osztjuk ki a legjobb bíráló díját, amellyel a bírálók fáradtságos, ugyanakkor nélkülözhetetlen munkáját kívánjuk elismerni. A fenti díjak anyagi fedezetét a Neumann János Számítógéptudományi Társaság biztosítja. Az idei évben a K&H Bank szponzorációjának köszönhetően egy további különdíj kiosztására is sor fog kerülni, melynek odaítéléséről a K&H Bank által felállított bizottság fog dönteni.

A szervezőbizottság nevében,

Ács Judit,

Berend Gábor,

Gosztolya Gábor,

Ligeti-Nagy Noémi,

Nemeskey Dávid Márk,

Novák Attila,

Simon Eszter,

Sztahó Dávid,

Vincze Veronika



# Tartalomjegyzék

<b>Szemantika, pragmatika</b>	<b>1</b>
3 A kontextusablakon kihajolni nem veszélyes: jogi szövegek hatékony szemantikus keresése <i>Csányi Gergely Márk, Lakatos Dorina, Vadász János Pál, Nagy Dániel, Üveges István</i>	
17 A pragmatikai annotáció kontextusfüggősége nagy nyelvi modell esetében – felszólító alakok funkcióinak annotálása huBert modellel <i>Szécsényi Tibor, Virág Nándor</i>	
29 Internetes hírek automatikus osztályozása <i>Osváth Máttyás, Héja Enikő</i>	
41 Optimizing Abstractive Arabic Summarization via RLHF and DPO with Llama 2 <i>Mram Kahla, Zijian Győző Yang</i>	
<b>Beszédtechnológia, kommunikáció</b>	<b>57</b>
59 Különböző wav2vec 2.0 rétegekből nyert beágyazások használata sclerosis multiplex felismerésére <i>Gosztolya Gábor, Tóth László, Svindt Veronika, Bóna Judit, Hoffmann Ildikó</i>	
73 Imitált és valódi depresszió elkülönítése mély neurális háló alapú jellemzőkinyerési módszerekkel <i>Dalotti Ágoston, Kiss Gábor</i>	
87 Magyar nyelvű beszédleiratózó tanítása sok (tíz)ezer óra beszéddel <i>Dobsinszki Gergely, Kádár Máté Soma, Fegyő Tibor, Mády Katalin, Mihajlik Péter</i>	
97 RAG módszerek implementálásának kihívásai egy célorientált chatbot rendszer kontextusában <i>Csáki Csaba, Vándor Péter</i>	
<b>Nyelvmodellek</b>	<b>111</b>
113 A látens szemantikus tulajdonságok segítségével előtanított nyelvi modellek vizsgálata <i>Berend Gábor</i>	
123 Text cleaning with transformer language models for Hungarian <i>Gábor Madarász, András Holl, Noémi Ligeti-Nagy, Zijian Győző Yang, Tamás Váradi</i>	

- 137 Középkori nyelvek feldolgozása ingyenes és kereskedelmi generatív nyelvmodellekkel  
*Pethő Gergely, Swaroop Krishna*
- 153 PULI LlumiX modell: Egy folytatólagosan előtanított nagy nyelvi modell  
*Yang Zijian Győző, Dodé Réka, Ferenczi Gergő, Hatvani Péter, Héja Enikő, Lengyel Mariann, Ligeti-Nagy Noémi, Madarász Gábor, Sárossy Bence, Varga Kristóf, Varga Tamás, Váradi Tamás*

### **Korpusz, nyelvi elemzés**

**169**

- 171 A HuSpaCy és e-magyar elemzőláncok teljesítményének átfogó összehasonlítása országgyűlési szövegeken: a tokenizálástól a függőségi elemzésig  
*Skrabák Boglárka, Ligeti-Nagy Noémi*
- 185 HuAMR: A Hungarian AMR Parser and Dataset  
*Botond Barta, Endre Hamerlik, Milán Konor Nyist, Judit Ács*
- 197 HunSimpleNews: Az első autentikus magyar nyelvű szövegekből álló szövegegyyszerűsítési korpusz  
*Prótár Noémi, Nemeskey Dávid Márk*
- 219 A méret a lényeg? Morfológiailag annotált korpuszok összehasonlító kiértékelése  
*Dömötör Andrea, Indig Balázs, Nemeskey Dávid Márk*

### **Poszter, laptopos bemutató**

**231**

- 233 Zárt és nyílt súlyú LLM-ek teljesítményértékelése RAG-alapú felsőoktatási chatbot alkalmazás példáján  
*Csáki Csaba, Vándor Péter*
- 247 Magyar szerkezetár demó  
*Sass Bálint, Indig Balázs, Kalivoda Ágnes, Lagos Cortes Mátyás, Lipp Veronika, Makrai Márton, Pethő Gergely, Simon László, Vadász Noémi*
- 257 Evaluation Library for the Hungarian Language Understanding Benchmark (HuLU)  
*Péter Hatvani, Kristóf Varga, Zijian Győző Yang*
- 269 Egy magyar nyelvű táblázatos kérdésmegválaszolás kiértékelő adatbázis  
*Tóth Gábor, Farkas Richárd, Szántó Zsolt*







# SZEMANTIKA, PRAGMATIKA



# A kontextusablakon kihajolni nem veszélyes: jogi szövegek hatékony szemantikus keresése

Csányi Gergely Márk<sup>1</sup>, Lakatos Dorina<sup>1</sup>, Vadász János Pál<sup>2</sup>, Nagy Dániel<sup>1</sup>,  
Üveges István<sup>1</sup>

<sup>1</sup>MONTANA Knowledge Management Ltd.

<sup>2</sup>National University of Public Service

{csanyi.gergely,lakatos.dorina,uveges.istvan,nagy.daniel}@montana.hu  
vadasz.pal@uni-nke.hu

**Kivonat** Tanulmányunkban egy szemantikus keresést támogató rendszert mutatunk be magyar bírósági határozatokon, amely képes az azonos tényállású ügyek hatékony felismerésére és visszakeresésére. A kiértékelési rendszerünk alappilléret az úgynevezett „tényállásvázlatok” képezik, amelyek tömören foglalják össze egy-egy ügy kulcsfontosságú tényeit, ezáltal lehetőséget biztosítva a releváns jogi precedensek gyors és pontos azonosítására.

Vizsgálatunk során tizenkét korszerű szövegbeágyazási modellt vetettünk össze, amelyek közül a Cohere, a Beijing Academy of Artificial Intelligence és az OpenAI legújabb fejlesztései teljesítettek a legjobban, különösen a hosszú, zajos jogi szövegek feldolgozásában. A rövidebb kontextusablakkal bíró modellek esetében megvizsgáltunk több szegmentálási eljárást is, köztük az egyszerű feldarabolást (chunking), az átlapolt feldarabolást (striding) és a rövidebb utolsó chunk nagyobb hatását csökkentő (last chunk scaling) technikákat, amelyek javították a hosszú szövegek megbízható kezelhetőségét.

A metodika egyaránt alkalmazható más Q&A jellegű rendszereknél is, mivel pedig a legtöbb általunk kipróbált modell többnyelvű, így felhasználhatóság nem korlátozódik a magyarra.

**Kulcsszavak:** szemantikus keresés, chunking, striding, jogi Q&A rendszer, magyar bírósági határozatok

## 1. Bevezetés

A hasonló jogesetek megtalálása kiemelt feladat mind az ügyvédek, mind a bírák számára a napi munkavégzés során. Ez biztosítja, hogy a hasonló tények hasonló döntésekhez vezessenek, amely nagymértékben elősegíti a konzisztens jogi döntések meghozatalát. Hagyományosan a jogesetkeresés főként a pertárgy szerinti kategóriák alkalmazására koncentrál, amely ugyan hasznos, de egyben korlátozott megközelítés is (Csányi és mtsai, 2022). A pertárgy szerinti szűrés ugyan megkönnyíti a keresési folyamatot, de nem képes hatékonyan megragadni az egyes jogesetek közötti szemantikai kapcsolatokat. Ennek eredményeként a

folyamat időigényes marad, a releváns precedensek megtalálása pedig továbbra is jelentős munkaerő-ráfordítást igényel.

A jogi területen a szemantikai hasonlóság alapú keresés általában azt jelenti, hogy egy adott határozathoz hasonló további határozatokat szeretnénk hatékonyan megtalálni. Ebben a cikkben szemantikai hasonlóság alapú kereséssel foglalkozunk, melynek célja, hogy hasonló tényeket tartalmazó bírósági határozatokat találjunk. A kiértékeléshez *tényállásvázlatokat* használunk, amelyek rövid, néhány mondatos leírások, amelyek röviden összefoglalják az adott jogi probléma tényeit. Egy valóperes esetben például a tényállásvázlat röviden leírhat minden olyan információt, amely egy gyermekelhelyezési ügyben releváns lehet, mint például a felek bemutatása, vagy a gyermekek kapcsolata az egyes szülőkkel, emellett esetünkben egy kérdéssel is zárulhat.

A bírósági határozatok általában rendkívül hosszúak (átlagosan 3-5000 szó), és speciális jogi nyelvezetet, terminus technicusokat használnak, ami megnehezíti a jó minőségű szemantikai alapú reprezentációk létrehozását. Ennek egyik fő oka, hogy a szövegek nem férnek bele a transzformer-alapú beágyazási modellek kontextusablakába. Ezért egy gyakorlatban alkalmazható szemantikai keresőrendszer két fontos pillérre támaszkodik: 1) megfelelő beágyazási forma kiválasztása, amely hosszabb szövegeket is képes kezelni, valamint 2) a különböző megoldások kiértékelésére és összehasonlítására való képesség. A kiértékelés nem egyértelmű egy nagyobb adathalmaz esetén, mivel még ha emberek értékelik is a keresési eredményeket egy adott lekérdezési szcenárióban, nem garantált, hogy ott éppen megtaláltuk az adott ügyhöz kapcsolódó összes releváns további ügyet. Ezért az emberi kiértékelés inkább csak validációs célra használható, de nem alkalmazható hatékonyan az egyes módszerek rangsorolásához.

Cikkünkben összesen tizenkét beágyazási módszert hasonlítottunk össze, melyek közül tizenegy transzformer architektúrára épül (Vaswani és mtsai, 2017), egy pedig a fasttext alapú szövegbeágyazásra (Bojanowski és mtsai, 2017). Ez utóbbira csak mint baseline megoldásra tekintettünk. Ezen túlmenően a transzformer-alapú modellek esetében, amelyek kontextusablaka legfeljebb 512 token, hét különböző módon próbáltuk kezelni az ennél hosszabb dokumentumok problémáját. Ennek eredményeképpen átfogó összehasonlítást kívánunk nyújtani, amely jó alapot biztosít hasonló rendszerek fejlesztéséhez, továbbá megmutatja, hogy a jogi tények szemantikájának megragadása elegendő-e hasonló dokumentumok kereséséhez. Bár a kutatásunkban használt adathalmaz magyar nyelvű, a legjobban teljesítő modellek mind többnyelvűek, ami arra utal, hogy eredményeink valószínűleg más nyelveken is alkalmazhatók és hasznosak, túlmutatva a magyar nyelvi kontextuson. Emellett az ismertett módszertan hatékonyan követhető egy gyakorlatban alkalmazandó szemantikai keresőrendszer létrehozásakor is.

A cikk felépítése a következő: a 2. fejezet összegzi a legrelevánsabb korábbi kutatásokat. A 3. fejezet bemutatja a tanulmányban használt adathalmazt. A 4. fejezet ismerteti a vektorizálási folyamat során felmerülő nehézségeket és a kiértékelési metrikákat. Az eredményeket a 5. fejezetben mutatjuk be és tárgyaljuk. Végül a következtetéseket az 6. fejezetben vonjuk le.

## 2. Kapcsolódó irodalom

A hosszú szövegek közötti szemantikai hasonlóság alapú keresés régóta fennálló probléma a természetesnyelv-feldolgozásban, különösen az információ-visszakereső rendszerek megjelenése óta (Salton és McGill, 1986). A szövegek vektorizálása alapvető szerepet játszik a hasonlóságok azonosításában, például koszinusz távolság segítségével (Qian és mtsai, 2004).

A transzformer-alapú megoldások a kontextusablak rövidegéből adódó problémára többféle megoldási kísérletet is tettek. A feladatot Wang és mtsai (2023) például egy hosszú szövegek feldolgozására optimalizált modellel (LTR-BERT) oldották meg, amely a szövegeket paragrafusokra bontja. A szerzők egy lineáris réteggel kompresszálták a BERT rejtett rétegét és ebből történt egy Q&A-s finomhangolás.

Limsopatham (2021) összehasonlította egy osztályozási feladaton a hosszú jogi dokumentumok elejéről vagy végéről 512 tokent megtartó truncationt, a 200 token hosszú részekre darabolást, illetve ezek átlagos és maximális poolingját BERT modellekkel, jogi előtanítással és anélkül is. Összevetette továbbá a nagyobb kontextusablakkal bírót, de jogi adaton nem előtanított BigBird (Zaheer és mtsai, 2020) és Longformer (Beltagy és mtsai, 2020) alapú modelleket is. Az eredmények azt mutatták, hogy a truncation alkalmazása a leggyengébb teljesítményt eredményezte, míg mindkét pooling alapú darabolás jelentősen jobb eredményeket hozott, továbbá a jogi előtanítás is javított a teljesítményen. Mindezek ellenére a Longformer és a BigBird modellek felülmúlták az összes egyéb megközelítést, még akkor is, ha nem voltak a jogi doménre előtanítva.

Vatsal és mtsai (2023) szintén hosszú jogi szövegek osztályozásakor azt tapasztalták, hogy a feldarabolt szövegrészek közti átlapolással (striding) lehetett az adott feladatot a legpontosabban megoldani. A legjobb eredményeket 64 token hosszúságú stride használatával érték el.

A jelen tanulmány az új beágyazási modellek összehasonlítására és a hosszú szövegek feldolgozási kihívásainak kezelésére fókuszál.

## 3. Adathalmaz

A vizsgálatunk során használt adathalmaz 1172 magyar bírósági határozatból áll, amelyek minden bírósági szintet lefednek. Ezeket korábbi kísérletekben már alkalmaztuk teszhalmazként (Csányi és mtsai, 2024). A dokumentumokat egy LSTM alapú retorikai szerepeket figyelembe vevő neurális modell (Rhetorical Role Labeler - RLL) címkézte, amelyet a szerzők tanítottak be. Ez az RLL modell mondatokra bontja a dokumentumokat, és minden egyes mondatot az alábbi címkék egyikével lát el:

- **Tényállás:** minden mondat, amely leírja, hogy miről szól a jogvita.
- **Pertörténet:** mivel a bírósági határozatok minden szintjét vizsgáltuk (Kúria, Törvényszékek, Ítéletábrák, Közigazgatási Bíróságok stb.), a dokumentumok tartalmazhatnak információkat korábbi döntésekről is.
- **Felek érvei:** a felek érveiről és kérdéseiről szóló mondatok.

- **Döntés:** az ítéletet leíró mondatok.
- **Bíróság érvelése:** a bírói érvelés a döntés jogi alapját illetően.
- **Indokolás:** az ítélet indoklását, a bíróság érvelését tartalmazó mondatok.
- **Perköltség:** mondatok arról, hogy az ítélet alapján melyik félnek mennyit kell fizetnie perköltségként.
- **Rendelkező rész:** az ítélet gyakorlati következményeit leíró mondatok, például hogy az alperesnek X összeget kell fizetnie kártérítésként a felperes részére.
- **Egyéb:** egyik fenti kategóriába sem tartozó részek, például aláírások, a dokumentum fejléce, keltezés, fejezetcímek stb.

Mivel egy olyan rendszert kívántunk létrehozni, amely képes hasonló bírósági határozatokat megtalálni egy rövid esetleírás alapján, így ezekből most csak a *Tényállás* mondatokat használtuk fel.

### 3.1. Tényállásvázlatok

Azért, hogy a kiértékelés költséghatékony és reprodukálható legyen, a kereséshez kivonatokat készítettünk, amelyek egy-egy ügy tényállását írják le. Ezeket az adott dokumentumokból hoztuk létre, eltávolítva minden specifikus azonosítót, például helyek, szervezetek nevét, dátumokat stb. Erre az előfeldolgozásra azért volt szükség, hogy a keresés során a hasonlóság alapját ne konkrét részletek (pl. az eljáró bíró neve) adják, hanem az ügyek valódi, absztrakt jogi tartalma. Ez jó eséllyel pozitívan hat a gyakorlatban felmerülő keresések hatékonyságára is, hiszen a jogi munka során a cél a lényegileg hasonló ügyek megtalálása. Az így keletkezett kivonatokra a tanulmányban *tényállásvázlatokként* hivatkozunk. Ezek keletkezési módja miatt elvárható, hogy ha egy adott dokumentumra keresünk a belőle készített tényállásvázlat segítségével, akkor az a legelső találatok között szerepeljen. Ez lehetővé teszi a különböző szövegbeágyazási modellek összehasonlítását megfelelő metrikák, például a Mean Reciprocal Rank alapján.

Kétféle megközelítést próbáltunk ki a tényállásvázlatok elkészítése során. Először az OpenAI GPT 3.5 turbo modelljét használtuk az API `completion` végponton keresztül, hogy az adott dokumentum *Tényállás* részéhez tartozó mondatokból összefoglalót készítsünk. Az alkalmazott modell a `gpt-3.5-turbo-1106` volt. Ebben a fázisban az 1. példához hasonló promptokat használtunk.

**1. Példa.** *Foglald össze a "" jelek közötti szöveget maximum 3 mondatban úgy, ahogyan azt egy ügyvéd tenné, aki hasonló dokumentumokat keres egy jogi adatbázisban. Az összefoglaló ne tartalmazzon semmilyen konkrét információt, mint például földrajzi nevek, pontos dátumok, szervezetek nevei vagy egyszavas keresési kifejezések. Csak az átfogalmazott szöveget add vissza, amely legyen koherens, 2-3 mondatos, rövidebb az eredetinél, és ne másolja az eredeti szöveget. A szöveg: ""... ""*

Az instrukciók egyes változataiban közös elem volt például az, hogy a modell ne említsen konkrét névelemeket, valamint hogy inkább parafrázzálja a szöveget, mintsem hogy annak konkrét részeit másolja egybe.

Mivel azonban a modell sok esetben nem tartotta az utasításokat, így végül jogterületenként (közigazgatási, polgári, büntető, gazdasági és munkaügyi) 20-20 dokumentumhoz annotátorok kézzel készítették tényállásvázlatokat, összesen 100-at. Ezt három annotátor végezte úgy, hogy átnézték, és szükség esetén teljesen átfogalmazták a GPT által generált válaszokat. Azokban az esetekben, amikor a modell által készített eredeti összefoglaló is helyes volt, azt változatlanul hagyták.

### 3.2. Az adathalmaz jellemzői

Az adathalmaz token-szintű statisztikáit az 1. táblázat mutatja be. A *T* (vázlat) oszlop a kiértékeléshez kiválasztott 100 dokumentum tényállásainak eredményeit tartalmazza, míg a *T-vázlat* oszlop az annotátorok által készített vázlatok statisztikáit mutatja be. Minden egyéb eredményt a teljes korpuszon számoltunk ki. A *T* oszlopban a tényállásra vonatkozó adatok láthatók.

A pontos modellnevek, azok kontextusablak-méretei és a beágyazási dimenziók az 3. táblázatban találhatóak.

1. táblázat. Tokenek száma (átlagosan) és az egy tokenre jutó átlagos karakterek száma különböző tokenizálókkal mérve. Az egy tokenre jutó karakterek arányát az 1 172 dokumentumot tartalmazó teljes adathalmazon számoltuk ki.

	<b>T</b>	<b>T (vázlat)</b>	<b>T-vázlat</b>	<b>kar./tok.</b>
<b>hubert</b> <b>sbert_hubert</b> <b>danieleff</b>	969,17	697,36	82,02	4,694
<b>e5_large</b> <b>e5_base</b> <b>cohere</b> <b>bge_m3</b> <b>jina_v3</b>	1140,09	819,21	108,41	3,862
<b>mcontriever</b>	1492,35	1091,38	139,82	2,908
<b>openai_ada</b> <b>openai_3_large</b>	1902,46	1411,87	181,77	2,259

A különböző tokenizálók máshogyan tokenizálják a szöveget, azaz például átlagosan eltérő számú karaktert fednek le tokenenként. Ez egyrészt számít az API-n keresztül hívható modellek esetében, hiszen a hívás költségét tokenekben kell mérni, másrészt befolyásolja, hogy az egyes modellek kontextusablakai hány karaktert tudnak átlagosan lefedni.

A táblázatból látszik, hogy több modell is azonos tokenizálót használ. A huBERT tokenizálója teljesített a legjobban, míg az openai tokenizálója átlagosan kétszer annyi tokent használ fel ugyanarra a szövegre nézve. Az XLM-RoBERTa tokenizálója pedig közel 4 karaktert fed le 1 tokenrel.

A 8192 token szélességű kontextusablakot használó modellek (`openai`, `jina_v3`, `bge_m3`) esetében kiszámoltuk, hogy ez az ablak milyen arányban fedi le az egyes dokumentumokat. A részleteket a 2. táblázat mutatja be.

2. táblázat. A 8192 token szélességű kontextusablakot használó modellek által teljesen lefedett dokumentumok aránya és a kontextusablakot meghaladó dokumentumok esetén az egy kontextusablakkal lefedett tokenek aránya.

Modell	Lefedett dokumentumok [%]	Hosszú szövegek lefedett tokenjei [%]
<code>openai</code>	97,01	60,59
<code>bge_m3</code> , <code>jina_v3</code>	98,55	63,02

A magas lefedettség arány miatt valószínűtlen volt, hogy a részekre bontás jelentős hatással lenne a 8192 token szélességű kontextusablakot használó modellek eredményeire. Ezért a részekre bontást csak a `bge_m3` modell esetében számoltuk ki, míg a többi modellnél kizárólag truncated vektorformákat alkalmaztunk. Fontos megjegyezni, hogy a kisebb kontextusablakot használó modellek még a dokumentumok 50%-át sem fedték le teljesen. Ezért szükséges volt a problémát kezelni annak érdekében, hogy elkerüljük az alacsony hatékonyságú szövegrepresentációk használatát. A probléma megoldásának részleteit lásd a 4.1. fejezetben.

## 4. Alkalmazott módszerek

A fejezet röviden bemutatja a tanulmányban használt vektorizálási formákat, valamint az eredmények értékelésére alkalmazott módszereket.

### 4.1. Vektorizálás

A vektorizálás minősége kulcsfontosságú a vektoros keresés során. A tanulmányban kipróbált vektorizálási modelleket a 3. táblázat tartalmazza. A `cohere`, `openai_3_large`, `openai_ada` és `jina_v3` modelleket API-n keresztül értük el. Ez fontos információ lehet olyan alkalmazások fejlesztése során, ahol a feldolgozott adatokat harmadik fél nem ismerheti meg, hiszen az API hívások során az adatok mindenképpen eljutnak a feldolgozást végző szerverre, amely rejthet magában adatbiztonsági kockázatot. A `fasttext` modellt (Bojanowski és mtsai, 2017) 157 000 magyar bírósági határozat felhasználásával tanítottuk be a hivatalos `fasttext` Python csomag segítségével. A modell egy skip-gram típusú modell (Mikolov és mtsai, 2013), ahol a negatív mintavételezést 10-re, a tanulási rátát pedig 0,05-re állítottuk, illetve 100 dimenziós vektorokat generáltunk. Minden egyéb paramétert az alapértelmezett beállításon hagytunk és az így készült modell jelentette a baseline megoldásunkat.

<sup>1</sup> <https://txt.cohere.com/introducing-embed-v3/> (2024.09.25)

<sup>2</sup> <https://platform.openai.com/docs/guides/embeddings/embedding-models> (2024.09.25)



## 3. táblázat. A tanulmányban összehasonlított vektorizáló modellek

Model neve	Rövidítés	Kontextusablak	Dimenzió
fasttext modell 157 000 magyar jogeseten tanítva (Bojanowski és mtsai, 2017)	fasttext	-	100
NYTK/sentence-transformers-experimental-hubert-hungarian (Osváth és mtsai, 2023)	sbert_hubert	128	768
danieleff/hubert-base-cc-sentence-transformer	danieleff	512	768
SZTAKI-HLT/hubert-base-cc (Nemeskey, 2021)	hubert	512	768
intfloat/multilingual-e5-base (Wang és mtsai, 2024)	e5_base	512	768
intfloat/multilingual-e5-large (Wang és mtsai, 2024)	e5_large	512	1024
Cohere/Cohere-embed-multilingual-v3.0 <sup>1</sup>	cohere	512	1024
facebook/mcontriever-msmarco (Izacard és mtsai, 2021)	mcontriever	512	768
OpenAI text-embedding-ada-002 <sup>2</sup>	openai_ada	8191	1536
OpenAI text-embedding-3-large <sup>2</sup>	openai_3_large	8191	3072
BAAI/bge-m3 (Chen és mtsai, 2024)	bge_m3	8192	1024
jinaai/jina-embeddings-v3 (Sturua és mtsai, 2024)	jina_v3	8192	1024

A következő csoport magyar nyelvre előtanított modelleké volt. Itt megvizsgáltuk a finomhangolás nélküli huBERT (Nemeskey, 2021) modellt illetve, két másikat is: a mondat szintű szemantikai hasonlóságra S-BERT (Reimers és Gurevych, 2019) segítségével finomhangolt sbert\_hubert modellt (Osváth és mtsai, 2023), valamint a Q&A-re finomhangolt danieleff modellt. Az sbert\_hubert modellt a Hunglish 2.0 párhuzamos korpuszon (Varga és mtsai, 2007) került finomhangolásra, hogy hasonló eredményeket produkáljon, mint a bert-base-nli-stsb-mean-tokens (Reimers és Gurevych, 2019) modellt. A korpusz jelentős része jogi szövegekből állt. A danieleff modellt 170 kérdés-válasz párral finomhangolták, amelyeket egyetemi tanulmányi szabályzatok 1000-5000 karakter hosszú szakaszaiból vettek.

A következő csoport többnyelvű modelleket foglal magában, amelyek kérdés-válasz, szemantikai hasonlóság és információvisszakeresési feladatokra lettek finomhangolva: többnyelvű e5 modellek (Wang és mtsai, 2024), Cohere többnyelvű beágyazási modell<sup>1</sup>, BGE-M3 többnyelvű beágyazási modell (Chen és mtsai, 2024), a Jina AI többnyelvű beágyazási modellje (Sturua és mtsai, 2024), valamint a Facebook mContriever modellje (Izacard és mtsai, 2021). Ezeket többek között az MSMARCO Q&A adathalmazon (Bajaj és mtsai, 2016) finomhangolták.

Az mContriever modell egy többnyelvű BERT alapú megoldás (Devlin és mtsai, 2018), amelyet a CCNet adathalmazon Wenzek és mtsai (2019) tanítottak kontrasztív tanulás (Izacard és mtsai, 2021) segítségével. A kontrasztív tanulás célja, hogy a lekérdezési reprezentáció alapján megtalálja a pozitív dokumentumnak megfelelő reprezentációt a negatívak között. A modellt már sikerrel alkalmazták a ShunQA magyar nyelvű nyíltkérdés-megválaszoló rendszerben (Berkecz és mtsai, 2024).

Az e5 modellek XLM-RoBERTa alapú (Conneau és mtsai, 2019) többnyelvű modellek, amelyek angol E5 alapú szövegbeágyazások többnyelvű kiterjesztései. Ezeket gyengén felügyelt kontrasztív módon tanították elő szövegpárokon, majd kis mennyiségű magas minőségű címkézett adaton felügyelten finomhangolták őket Wang és mtsai (2024).

A cohere modellek beágyazási technikájáról nem érhetők el részletes publikációk kereskedelmi okok miatt, de egy blogbejegyzés<sup>1</sup> szerint ezek a beágyazások

a témák egyezését és a tartalom minőségét is figyelembe veszik, ami magasabb minőségű kereséseket eredményezhet.

A `bge_m3` modell szintén az XLM-RoBERTa alapú modellek közé tartozik, és több mint 100 nyelvre biztosít reprezentációkat különböző szinteken (szó, bekezdés, teljes szöveg akár 8192 tokenig). Képes kezelni sparse, dense és többvektoros visszakeresési feladatokat is.

A Jina AI v3 többnyelvű modellje (Sturua és mtsai, 2024) a legjobb nem angol nyelvű eredményeket éri el a szemantikai szöveg hasonlósági feladatokban az MTEB ranglistán<sup>3</sup> (Muennighoff és mtsai, 2022). Ez a modell szintén 8192 token hosszú kontextusablakkal rendelkezik, és számos feladatra alkalmazható, például dokumentum visszakeresésre, klaszterezésre, osztályozásra és szövegillesztésre. A Matryoshka Representation Learning (MRL) technológiának köszönhetően a beágyazási dimenzió akár 32-re is csökkenthető jelentős teljesítményvesztés nélkül.

Az OpenAI v3 beágyazási modelljeit szintén Matryoshka Representation Learning technikával tanították, amely lehetővé teszi a beágyazások lerövidítését anélkül, hogy azok fogalomreprezentáló tulajdonságai sérüljenek (Kusupati és mtsai, 2022).

Fontos még megemlíteni, hogy az egyes modellek esetében mindig azzal a módszerrel nyertük ki a beágyazásokat, amire a modell eredetileg tanítva lett. Így például a hubert esetében CLS vektorokat, a többi BERT modell esetében pedig token átlagokat alkalmaztunk.

**Hosszú szövegek kezelése** A transzformer-alapú modellek kiválóan alkalmasak szemantikai kapcsolatok reprezentálására. Azonban egyik jelentős hátrányuk, hogy csak rögzített tokenablak méretig képesek szövegeket feldolgozni. Az nem jelent problémát, ha a szöveg rövidebb, mint a modell kontextusablaka. Ezzel szemben kérdések merülnek fel, ha a szövegek nem férnek bele ebbe az ablakba. Ennek megoldására hét különböző megközelítést hasonlítottunk össze, köztük a *Last Chunk Scaling* (LCS) módszert, amely különösen hasznos, ha a szöveg nagyjából a kontextusablak méretének 2-5-szöröse. Az alkalmazott megközelítések a következők:

- **truncated:** ez a legegyszerűbb megközelítés, amely csak a dokumentum első kontextusablak-méretű részét tartja meg. Fontos megjegyezni, hogy különböző modellek eltérő számú karaktert képesek lefedni. Ez az alapértelmezett (baseline) módszer.
- **chunked:** a dokumentumot kontextusablak-méretű darabokra bontja úgy, hogy a szavak ne legyenek kettévágva, csak a szavak végén történjen a felosztás. Ezekre a darabokra kiszámítjuk a vektorokat, majd ezek átlagolásával egy dokumentumvektor jön létre.
- **stride:** a dokumentum darabolása hasonló a chunked megközelítéshez, de biztosítja, hogy az egyes darabok között legyenek átfedő tokenek, ahogy azt

<sup>3</sup> <https://huggingface.co/spaces/mteb/leaderboard>

Vatsal és mtsai (2023) is leírja. Ezek a darabok szintén átlagolásra kerülnek egy dokumentumvektor létrehozásához. Két különböző stride méretet teszteltünk: a kontextusablak 25%-át és a fix 16 tokent.

- **last chunk scaling (LCS)**: ez a technika alkalmazható a chunked vagy stride módszerek mellett. A dokumentum darabolása után az utolsó darab általában kevesebb tokent tartalmaz, mint a kontextusablak. Megoldásunk az utolsó darab vektorát az alábbi faktoriall skálázza:  $\frac{\text{last chunk tokenek száma}}{\text{kontextusablak mérete}}$ . Ezt a módszert a chunked és mindkét stride megközelítéssel (25% és fix 16 token) teszteltük.

A stride és az LCS módszerek célja a kontextusablaknyi részekre bontás hátrányainak enyhítése. A stride a tokenek megosztásával kezeli a részek elkülönülését, míg az LCS megakadályozza a kisebb darabok túlsúlyozását, amely torzíthatja az átlagvektort. Ez különösen fontos volt a mi esetünkben, ahol a Tényállások átlagosan 2-8 részből álltak, a vektorizálótól függően.

Összességében hét megközelítést teszteltünk a legfeljebb 512 tokent kezelő modelleken: *truncated*, *chunked*, *chunked+LCS*, *stride (25%)*, *stride (25%)+LCS*, *stride (fix 16)* és *stride (fix 16)+LCS*.

## 4.2. Kiértékelés: Mean Reciprocal Rank

A kiértékelés során a széles körben használt Mean Reciprocal Rank (MRR) mérőszámot alkalmaztunk.

Az MRR a következőképpen számítható:

$$MRR = \frac{1}{n} \sum_{i=1}^n \frac{1}{r_i} \quad (1)$$

ahol  $n$  a kiértékelés során használt dokumentumok száma (jelen esetben 100 dokumentum),  $r_i$  pedig a releváns dokumentum visszakeresett pozíciója. Az MRR olyan metrika, amely akkor alkalmazható, ha egy adott lekérdezéshez ismert az elvárt dokumentum, ami a 3.1. fejezetben ismertetett adathalmazra teljesül. Fontos kiemelni, hogy a keresés során mind az 1172 dokumentumban kerestünk, de csak 100 tényállásvázlattal.

## 5. Eredmények

Az MRR eredményeket a 4. táblázat ismerteti. Az MRR pontszámokat 100-zal megszoroztuk, így az elméleti maximális érték 100. A továbbiakban erre a megszorozott értékre hivatkozunk MRR-ként. Minden sor legjobb eredményét **félkövér** kiemeléssel jelöltük. A *chunking* sor az alkalmazott részekre bontási módszereket mutatja:

- **trunc**: a truncated módszert jelenti,
- **0**: a 0 token átfedéssel stride-ot jelenti, amely egyenértékű a chunked opcióval,
- **16**: a 16 tokenes átfedéssel stride beállítást jelenti,

- **25%**: a kontextusablak 25%-ának megfelelő átfedéses stride-ot jelenti,
- **LCS**: jelzi, alkalmaztuk-e a Last Chunk Scaling technikát.

Az OpenAI és Jina beágyazási modellek esetében azok nagy kontextusablaka miatt csak a truncated vektorokat számoltuk ki.

4. táblázat. MRR eredmények (100-zal szorozva)

chunking/stride	trunc	0	0	25%	25%	16	16
LCS	False	False	True	False	True	False	True
bge_m3	<b>93,67</b>	<b>93,67</b>	<b>93,67</b>	<b>93,67</b>	<b>93,67</b>	<b>93,67</b>	<b>93,67</b>
cohere	92,99	93,23	94,87	90,69	92,21	94,27	<b>95,03</b>
danieleff	77,09	77,09	<b>79,88</b>	76,46	78,35	78,23	<b>79,88</b>
fasttext	<b>53,72</b>	<b>53,72</b>	<b>53,72</b>	<b>53,72</b>	<b>53,72</b>	<b>53,72</b>	<b>53,72</b>
hubert	7,69	<b>12,12</b>	8,79	8,01	7,69	10,65	8,79
e5_base	85,13	86,81	88,01	86,67	87,45	87,42	<b>88,77</b>
e5_large	87,82	89,00	<b>92,45</b>	87,83	89,37	88,51	91,81
jina_v3	<b>93,08</b>	-	-	-	-	-	-
mcontriever	81,46	89,53	90,62	90,34	91,14	89,23	<b>91,42</b>
openai_3_large	<b>93,05</b>	-	-	-	-	-	-
openai_ada	<b>83,65</b>	-	-	-	-	-	-
sbert_hubert	34,81	<b>60,43</b>	57,31	57,17	57,91	58,56	58,11

A legrosszabb teljesítményt a **hubert** nyújtotta, amely rámutat a feladat-specifikus finomhangolás fontosságára. A következő csoport a kis kontextusablakkal rendelkező **sbert\_hubert** modellt, valamint a doménspecifikusan tanított **fasttext** modellt tartalmazta. Az **sbert\_hubert** modellt mondatok szemantikai információjának reprezentálására finomhangolták, így nem meglepő, hogy jobban teljesített, mint a nem finomhangolt **hubert**. A **fasttext** elkülönül az összehasonlított vektorizálási módszerektől, mivel ez az egyetlen nem transzformer-alapú modell. Ennek ellenére sikerült felülmúlnia a 128-as tokenablakú **sbert\_hubert** modell különböző beállításainak többségét, bár elmaradt a nagyobb kontextusablakkal rendelkező modellek teljesítményétől.

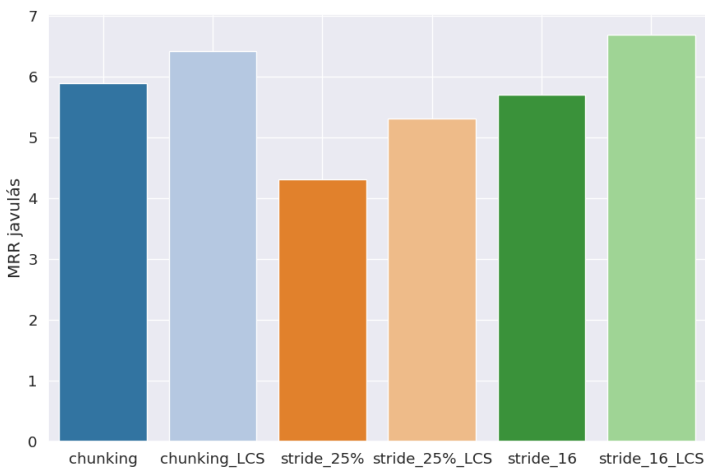
A következő szintet a **danieleff** modell képviselte, amelyet kérdés-válasz (Q&A) feladatra finomhangoltak, kevesebb, mint 200 példán. A modell kontextusablaka négyszer nagyobb (512), mint az **sbert\_hubert** modellé (128). Az eredmények arra utalnak, hogy a kontextusablak méretkülönbsége és a feladat-specifikus finomhangolás sokkal fontosabb tényező, mint a részekre bontási, stride és LCS módszerek összessége, mivel a **danieleff** modell 20 ponttal előzte meg az **sbert\_hubert** modellt.

Ugyancsak jelentős, kb. 8-12 pontos növekedés volt megfigyelhető az MRR-ben a többnyelvű **mcontriever** és E5 alapú modelleknél a **danieleff**-hez képest, elérve a 91,42-es és a 92,45-ös legjobb MRR értékeket. Ez azt igazolja, hogy pusztán a szemantikai tulajdonságokra orientált finomhangolás nem elegendő a hasonló tényállásokat felvonultató jogesetek keresése esetén. A passage-retrieval stílusú finomhangolás új szintre emeli a teljesítményt.

A legjobban teljesítő OpenAI modell a nemrégiben megjelent `openai_3_large` volt, amely 93,05-ös MRR-t ért el, 10 ponttal megelőzve a korábbi `openai_ada` modellt. A `jina_v3` modell 93,08-as MRR értéket ért el, szorosan megelőzve a `openai_3_large` modellt. A `bge_m3` modell a második legjobb MRR eredményt érte el 93,67-es eredménnyel. A legjobbnak a `cohere` beágyazás bizonyult, 95,03-as MRR értékkel, annak ellenére, hogy csak 512 token szélességű kontextusablakkal rendelkezik.

### 5.1. Hosszú szövegek kezelésére alkalmazott megközelítések összehasonlítása

Az 1. ábrán az átlagos MRR javulások láthatóak az egyes módszerek esetében a truncated megoldáshoz képest.



1. ábra: Átlagos MRR javulás a különböző hosszú szövegek kezelésére alkalmazott megközelítések esetében a truncated megközelítéshez képest

Az 1. ábrán bemutatott eredmények szerint bármely részekre bontási stratégia jelentősen javítja a modellek teljesítményét a truncated módszerhez képest. Ez az eredmény összhangban áll a korábbi kutatásokkal is (Vatsal és mtsai, 2023). Általánosságban elmondható, hogy amikor LCS-t alkalmaztunk, az eredmények jobbak lettek, mint anélkül. Az LCS alkalmazása 0,84-es növekedést eredményezett az átlagos MRR-ben (72,01→72,85). Azonban, ha csak a három legjobban teljesítő modellre (`cohere`, `e5_large` és `mcontriever`) fókuszálunk, azt láthatjuk, hogy az LCS átlagosan 1,81-es növekedést eredményezett (90,29→92,10). Átlagosan és a legjobb 3 modell esetében is a legjobb részekre bontási módszer a stride 16+LCS megközelítés volt, amelyet szorosan követett a chunked+LCS.

Meglepő, hogy a 25%-os stride nem javította az eredményeket. Ezzel szemben a stride 16 a legtöbb esetben javította a teljesítményt ez utóbbihoz képest.

## 6. Következtetések

Cikkünkben jogi szövegeken alapuló szemantikai hasonlóságkeresési feladatot vizsgáltunk meg egy 1172 magyar bírósági határozatot tartalmazó korpuszon. Létrehoztuk egy hasonló tényállásokat tartalmazó ügyek keresésére alkalmas szemantikai hasonlóságkereső-rendszer alapjait, amelyhez bemenetként elegendő egy rövid tényállásvázlat megadása.

A keresőrendszerek kiértékelése összetett feladat, hiszen a keresési teljesítmény megbízható értékelése megköveteli az összes visszakeresett dokumentum alapos vizsgálatát. Ez az emberi részvételt igénylő feladat azonban költséges, és az értékelési eredmények újrafelhasználtsága korlátozott lehet. Ezt a kihívást az OpenAI `gpt-3-turbo` modelljének segítségével kíséreltük meg megoldani, amellyel vázlatokat generáltunk a tényállásokhoz, ezeket pedig egyenként kézzel ellenőriztünk, módosítottunk. Ez lehetővé tette számunkra, hogy megmérjük az eredeti dokumentum helyezését a visszakeresett dokumentumok között, így a különböző vektorizációs módszerek mennyiségi szempontból összehasonlíthatóvá váltak.

Összehasonlítottunk egy jogi területre betanított `fasttext` modellt és tizenegy transzformer-alapú vektorizációs megoldást a szemantikus keresés vizsgálatához. Az eredményeink más visszakeresési feladatok, például a Retrieval Augmented Generation (RAG) alkalmazások szempontjából is hasznosak lehetnek.

A transzformer-alapú modellek kontextusablaka általában nem képes lefedni a teljes dokumentumot. Megvizsgáltuk és összehasonlítottuk, hogy a dokumentumok kontextusablak méretű részekre bontása, a stride technika (átfedő kontextusablak méretű részek létrehozása) és az utolsó vektor súlyozása, a last chunk scaling (LCS) hogyan befolyásolja a vektorok minőségét az első kontextusablak használatához képest. Eredményeink azt mutatták, hogy az LCS hatékonyan javította a vektorok minőségét, és minden részekre bontási módszer felülmúlta a truncated beállítást. A stride technika eredményei a stride hosszától függően változtak, ezért ezt a módszert az adott feladathoz igazodó különböző stride értékekkel kell tesztelni.

A legjobb eredményeket 16 token hosszú stride alkalmazásával értük el, amely a legjobban teljesítő Cohere és E5 modellek kontextusablak-méretének 3,125%-át tette ki. A legjobban teljesítő modellek sorrendben a következők voltak: Cohere `embed-multilingual-v3.0` modellje, a BAAI `bge-m3` modellje, Jina AI `jina-embeddings-v3` modellje, OpenAI `text-embedding-3-large` modellje és a Microsoft `multilingual-e5-large` modellje. Tehát a megközelítésünkkel sikerült egy 512 hosszú tokenablakkal bíró modellel jobb eredményt elérni, mint több 8192 hosszúságú modellel.

## Hivatkozások

- Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., és mtsai: Ms marco: A human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268 (2016)
- Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. CoRR abs/2004.05150 (2020), <https://arxiv.org/abs/2004.05150>
- Berkecz, P., Zombori, T., Banga, G., Szabó, G., Szántó, Z., Novák, A., Richárd, F.: Shunqa: egy nyíltkérdés-megválaszoló rendszer (2024)
- Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the association for computational linguistics 5, 135–146 (2017)
- Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., Liu, Z.: Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation (2024)
- Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (2019)
- Csányi, G.M., Vági, R., Gadó, K., Üveges, I., Nagy, D., Bajári, L., Megyeri, A., Fülöp, Anna, Vadász, J.P.: Building a production-ready, hierarchical subject matter classifier for legal decisions (2024), unpublished
- Csányi, G.M., Vági, R., Nagy, D., Üveges, I., Vadász, J.P., Megyeri, A., Orosz, T.: Building a production-ready multi-label classifier for legal documents with digital-twin-distiller. Applied Sciences 12(3), 1470 (2022)
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
- Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., Grave, E.: Unsupervised dense information retrieval with contrastive learning. arXiv preprint arXiv:2112.09118 (2021)
- Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., és mtsai: Matryoshka representation learning. Advances in Neural Information Processing Systems 35, 30233–30249 (2022)
- Limsopatham, N.: Effectively leveraging bert for legal document classification. In: Proceedings of the Natural Legal Language Processing Workshop 2021. pp. 210–216 (2021)
- Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
- Muennighoff, N., Tazi, N., Magne, L., Reimers, N.: Mteb: Massive text embedding benchmark. arXiv preprint arXiv:2210.07316 (2022)
- Nemeskey, D.M.: Introducing huBERT. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2021). p. TBA. Szeged (2021)

- Osváth, M., Yang, Z.G., Kósa, K.: Analyzing narratives of patient experiences: A bert topic modeling approach. *Acta Polytechnica Hungarica* 20(7), 153–171 (2023)
- Qian, G., Sural, S., Gu, Y., Pramanik, S.: Similarity between euclidean and cosine angle distance for nearest neighbor queries. In: *Proceedings of the 2004 ACM symposium on Applied computing. SAC04*, ACM (Mar 2004), <http://dx.doi.org/10.1145/967900.968151>
- Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019)
- Salton, G., McGill, M.J.: *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., USA (1986)
- Sturua, S., Mohr, I., Akram, M.K., Günther, M., Wang, B., Krimmel, M., Wang, F., Mastrapas, G., Koukounas, A., Wang, N., és mtsai: jina-embeddings-v3: Multilingual embeddings with task lora. *arXiv preprint arXiv:2409.10173* (2024)
- Varga, D., Halácsy, P., Kornai, A., Nagy, V., Németh, L., Trón, V.: Parallel corpora for medium density languages. *Amsterdam Studies In The Theory And History Of Linguistic Science Series 4* 292, 247 (2007)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* 30 (2017)
- Vatsal, S., Meyers, A., Ortega, J.: Classification of us supreme court cases using bert-based techniques. *arXiv preprint arXiv:2304.08649* (2023)
- Wang, J., Huang, J.X., Sheng, J.: An efficient long-text semantic retrieval approach via utilizing presentation learning on short-text. *Complex and Intelligent Systems* 10(1), 963–979 (Aug 2023), <http://dx.doi.org/10.1007/s40747-023-01192-3>
- Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., Wei, F.: Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672* (2024)
- Wenzek, G., Lachaux, M.A., Conneau, A., Chaudhary, V., Guzmán, F., Joulin, A., Grave, E.: Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359* (2019)
- Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., és mtsai: Big bird: Transformers for longer sequences. *Advances in neural information processing systems* 33, 17283–17297 (2020)



# A pragmatikai annotáció kontextusfüggősége nagy nyelvi modell esetében – felszólító alakok funkcióinak annotálása huBert modellel

Szécsényi Tibor<sup>1</sup>, Virág Nándor<sup>2</sup>

<sup>1</sup> SZTE BTK Általános Nyelvészeti Tanszék,  
MTA–SZTE–DE Elméleti Nyelvészeti és Informatikai Kutatócsoport  
szecsényi.tibor@szte.hu

<sup>2</sup> SZTE Nyelvtudományi Doktori Iskola,  
MTA–SZTE–DE Elméleti Nyelvészeti és Informatikai Kutatócsoport  
virag.nandor9910@gmail.com

**Kivonat:** Természetes nyelvi szövegekben a pragmatikai jellemzők annotálása során nagy kontextust kell figyelembe venni. Tanulmányunkban azt vizsgáljuk, hogy egy pragmatikai jellemző automatikus annotálása során milyen hatással van a rendelkezésre álló kontextus nagyságának a változása. A vizsgálat egy olyan korpuszon történt, melyen rendelkezésre állt egy manuálisan rögzített pragmatikai annotáció a felszólító alakok funkcióinak kategorizálására. A korpusz segítségével a huBert base nagy nyelvi modellt finomhangoltuk az annotációs feladat elvégzésére, és megvizsgáltuk, hogy a modell az annotálás során milyen mértékben támaszkodik a rendelkezésre álló kontextusra. A kontextuális hatást kétféleképpen vizsgáltuk: a tanítószekvenciák hosszának változtatásával hogyan befolyásolja a modell összesített pontosságát; illetve az annotálandó elemek szekvencián belüli elhelyezkedése milyen hatással van az annotáció pontosságára.

## 1 Bevezetés

Természetes nyelvi szövegekben az elemek nyelvi jellemzőinek meghatározása során a jellemzőtől függő mértékben kell különböző nagyságú kontextust figyelembe venni. Morfológiai jegyek vagy a POS-tagek meghatározásánál elegendő néhány szavas kontextus ismerete, szintaktikai jellemzőknél a mondatnyi kontextus ismeret lehet szükséges, szemantikai jegyeknél viszont már egy-két mondatnyi kontextusra és szükség lehet. Pragmatikai jellemzők esetében viszont sokszor a teljes szöveget figyelembe kell venni a nyelvi elemek helyes annotálásához. Tanulmányunkban azt vizsgáljuk, hogy egy pragmatikai jellemző automatikus annotálása során milyen hatással van a rendelkezésre álló kontextus nagyságának a változása.

Vizsgálatunkat a MedCollect Egészégügyi álhírkorpuszon (Németh T., 2023; Szécsényi és mtsai, 2024) végeztük, melyen rendelkezésre állt egy korábban kézi annotációval rögzített pragmatikai annotáció, a felszólító alakok funkcióinak azonosítása (Szécsényi és mtsai, 2024). A korpusz segítségével a huBert base nagy nyelvi modellt (Nemeskey, 2020; 2021) finomhangoltuk az annotációs feladat elvégzésére. A tanítás elsődleges célja az volt, hogy a manuális annotálást megközelítő pontosságú automa-

tikus annotáló eszközt hozunk létre, közvetett, de jelen tanulmány szempontjából fontosabb célja pedig az, hogy megvizsgáljuk, hogy a modell az annotálás során milyen mértékben támaszkodik a rendelkezésre álló kontextusra. Az a kiinduló hipotézisünk, hogy a megfelelő megbízhatósággal működő gépi annotálás során is jobb eredmény érhető el, ha az elemző eszköz számára nagyobb kontextus hozzáférhető.

A kontextuális hatást kétféleképpen vizsgáltuk. Egyrészt azt néztük meg, hogy a tanítás és a teszt során használt szekvenciák hossza befolyásolja-e az annotálás pontosságát: elvárásunk szerint a hosszabb tanító és teszt szekvenciák esetében nagyobb az annotálás pontossága. Másrészt azt vizsgáltuk, hogy az annotálandó kifejezéseknek a szekvencián belüli elhelyezkedése miként befolyásolja a kifejezés annotációjának a pontosságát. Elvárásunk szerint a szekvenciák elején és végén kisebb a pontosság a szekvencia közepéhez viszonyítva, továbbá a szekvencia belsejében a szekvencia elejétől mért távolsággal növekszik a pontosság, vagyis a baloldali kontextus nagyobb hatással van a felszólító alakok funkcióinak a meghatározásánál.

A következő szakaszban bemutatjuk a korpusz annotációs sémáját, a manuális annotálás jellemzőit, valamint a manuális annotálás pontosságát befolyásoló tényezőket. A 3. szakaszban a huBert base finomhangolásának részleteit tárgyaljuk. A 4. szakasz különböző szekvenciahosszúságú tanítóadatokkal finomhangolt eszköz annotációjának a pontosságát, fedését és F1 értékeit hasonlítjuk össze, amit az 5. szakaszban a felszólító alakok teszt szekvencián belül elfoglalt pozíciójának a feltáró elemzés követ.

## 2 A felszólító alakok funkcióinak kézi annotálása a MedCollect korpuszban

A felszólítás (direktíva) a nyelvi manipuláció eszköze, amellyel a beszédpartnert vagy a szöveg olvasóját próbáljuk meg rávenni egy olyan tevékenység jövőbeni elvégzésére, amelyet egyébként nem feltétlenül tenne meg. A felszólítás legközvetlenebbül felszólító alakú igét tartalmazó megnyilatkozásokkal történhet, de megvalósulhat indirekt módon is, például kérdő mondatlall: *Ide tudnád adni a sőt?*

A felszólító alakoknak azonban nem csak a felszólítás kifejezése lehet a funkciójuk. A MedCollect korpusz annotációja során a felszólító alakok különböző funkcióban való használatának azonosítása volt a cél, az annotálás során a felszólító alakokhoz egy-egy funkció lett rendelve értéként. Vegyük sorra ezeket az értékeket a korpuszban való előfordulásuk gyakorisága szerint!

**Nodirectiva** – olyan felszólító alakok, amelyek nem hajtanak végre felszólítást, és a többi nem felszólító funkciót sem kapják meg. Ezek leggyakrabban kötőmódban álló igék, pl. *Belefeáradtam, hogy állandóan maszkot viseljek.*

**Saját hangú** – olyan felszólítás, amelyben a szöveg alkotója szólítja fel a szöveg olvasóját valamire, pl. *Mindenki viseljen maszkot!*

**Közvetített** – olyan felszólítás, amely a szöveg olvasójára irányul, de a felszólítás forrása nem közvetlenül a szövegalkotó, hanem valaki más, a szövegalkotó csak közvetíti azt, de egyet is ért vele. Pl. *A szakértők szerint is viseljünk maszkot.* A saját hangú és a közvetített felszólítások valódi felszólítások.

**Meta** – olyan felszólítás, amit valaki intézett valaki felé, de az nem a szöveg olvasója (*Az USA elnöke felszólította a New York-iakat, hogy viseljének maszkot.*), vagy ha igen, akkor a szövegalkotó nem ért vele egyet (*Az oltásellenesek azt akarják, hogy ne viseljünk maszkot.*). Ezekben az esetekben a szöveg szerzője csak beszámol egy felszólításról.

**Szövegszervező** – lehet ugyan valamilyen felszólító erejük, elsődleges funkciójuk azonban a figyelemirányítás, a szöveg koherenciájának megteremtése, pl. *Lásd a következő példát!*

**Interakciós** – olyan felszólító alak, amelyek a beszélőnek vagy a hallgatónak a szöveg megértését segítő tevékenységgel, vagy a beszélő és a hallgató közötti viszonytal kapcsolatos, pl. *Gondoljunk csak bele! Hogy őszinte legyek...*

**Ambiguous** – olyan felszólító alak, amely a fentebb megadottak közül több funkcióval is rendelkezhet.

A korpusz maga 707 darab online elérhető nyilvános weboldalról származó hírszerű egészségügyi témájú szöveget tartalmazott, a korpusz összterjedelme kb. 370 000 token. A kézi annotálás során 2664 felszólító alak került címkézésre. Az annotálásban 22 annotátor és 2 kurátor vett részt, az annotátorok összmunkaideje kb. 400 óra volt. A kézi annotálás során a felszólító alakok funkcióinak megállapításán kívül még két másik jegy is jelölve lett (*source*: a felszólítás forrása; *target*: a felszólítás célzottja), de ezek a gépi tanulási kísérletek során nem lettek figyelembe véve.

A felszólító alakok funkcióinak meghatározása során az annotátoroknak olyan döntést kellett végrehajtaniuk, ami nem kizárólag az annotálandó elemek és közvetlen szöveggörnyezetük formai jellemzőire támaszkodik, hanem sokszor a tágabb kontextust és az olvasói háttértudást (világtudás) is felhasználó következtetéseket is figyelembe kellett venni (l. pl. Archer és mtsai, 2009; Milà-Garcia, 2018). A tágabb kontextus hatása például akkor jelentkezik, amikor egy hosszabb szövegben a felszólító alak megjelenési helyénél akár bekezdésekkel korábban jelzi a szöveg, hogy valakinek a véleményét idézi (például egy interjúban), és ezért a felszólító alakot meta vagy közvetített címkével kell ellátni. Annak eldöntéséhez pedig, hogy a két címke közül melyik a helyes, az olvasó (annotátor) háttértudására alapuló következtetés szükséges, ti. hogy a szöveg alkotója vajon egyetért-e a felszólítással (közvetített), vagy sem (meta). A felszólító alakok funkcióinak meghatározása során további tényezőként jelenik meg meg az annotátorok egyéni különbségei is: a felszólítás erősségének, vagy akár a felszólítás meglétének a megítélése is eltérő lehet annotátoronként.

A kontextus hatása, a háttértudás felhasználásának szükségessége és az annotátorok egyéni különbségei miatt a pragmatikai jegyek annotálása során sokkal kisebb az annotátorok közötti egyetértés mértéke, mint például a morfológiai vagy szintaktikai jellemzők megállapítása során. Azonban az annotátoroknak a kurátor által elfogadottól eltérő annotációja nem tekinthető feltétlenül annotálási hibának: az eltérések egy része tényleges hiba (pl. figyelmetlenség), más része viszont a lehetséges eltérő megítélésből fakad.

A felszólító alakok funkcióinak annotálása során az annotátorok átlagosan 0,824 pontossággal és 0,846 fedéssel dolgoztak, az F1 értékek átlaga pedig 0,830 volt. A megbízhatósági értékek számításánál a kurátor által elfogadott annotációkat tekintet-

tük helyesnek. Az 1. Táblázat azt mutatja, hogy az annotátorok átlagosan milyen pontossággal, fedéssel és F1 értékkel tudták az egyes jegyeket azonosítani.

1. Táblázat: A manuális felszólításannotálás megbízhatósági értékei jegyenként és összesítve

	<b>Pontosság</b>	<b>Fedés</b>	<b>F1</b>
<b>nodirectiva</b>	0,83	0,88	0,85
<b>sajat_hangu</b>	0,91	0,90	0,90
<b>kozvetített</b>	0,63	0,60	0,60
<b>meta</b>	0,61	0,58	0,58
<b>szovegszervezo</b>	0,75	0,82	0,76
<b>interakcios</b>	0,53	0,58	0,55
<b>ambiguous</b>	0,32	0,24	0,27
<b>összesítve</b>	0,82	0,85	0,83

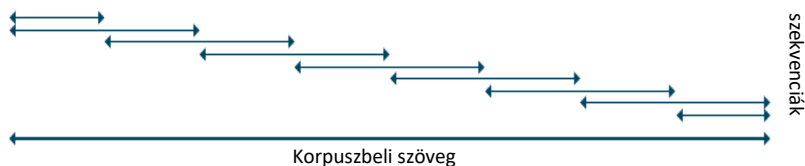
A rendelkezésre álló kézzel annotált korpusz segítségével megpróbáltunk egy nagy nyelvi modellt finomhangolni az annotációs feladat elvégzésére. Célkitűzésünk az volt, hogy egy olyan eszközt hozzunk létre, amely a manuális annotátorokhoz közelítő megbízhatósággal működik. Mivel a manuális annotáció megbízhatóságát befolyásoló egyik tényező a kontextus figyelembevétele volt, ezért az eszközön megvizsgáltuk a kontextus hatását is, ezt mutatjuk be a 4. és az 5. részben.

### 3 A huBert base finomhangolása

Az automatikus annotáció megoldásának fontossága kettős: egyrészt a tanítókorpuszként használt annotált korpusz bővítése során alkalmazható előannotálásra, vagy az annotátorok kiváltására (a kurátori döntést meghagyva), másrészt pedig közvetlenül használhatjuk a felszólítás funkcióit azonosítani képes modellt olyan alkalmazásokban, amelyekben szükség van ilyen információra. Ha az erre a feladatra finomhangolt nagy nyelvi modellt be akarjuk vetni éles terepen, számot kell adni azokról a potenciális problémákról, amelyekkel a humán annotátorok is szembesültek, konkrétan a kontextuális hatásról és a háttértudásról az automatikus annotáció során.

A kettő közül a háttértudás szerepének értelmezése egyértelműbb. Egy nagy nyelvi modell nem rendelkezik azzal a világról alkotott mögöttes tudással, amivel a humán elemző. Ennek a hiánya azonban nem feltétlenül jelentős, hiszen az emberi befogadó számára se feltétlenül szükséges a világról alkotott kép bevonása a felszólítások értelmezésében, mindössze támpont lehet a helyes kategorizáláshoz. Sokkal fontosabb a kontextuális tényezők felmérésére való képesség, ahogy azt korábban is említettük. A felszólítások funkcióinak azonosítása során a humán befogadók támaszkodnak a szöveggörnyezetre is, ezért felmerül a kérdés a nagy nyelvi modellek alkalmazása esetében, hogy „figyelembe veszik”-e megfelelő módon a rendelkezésre álló kontextust, ahhoz, hogy ez segítsen az annotálási feladatban. Ha a válasz igen, akkor pedig mennyire komoly szerepe van a kontextus nagyságának a modell megbízhatóságának mértékében?

A feladat elvégzéséhez a huBert base modellt (Nemeskey, 2020; 2021) használtuk, amely egy, a Common Crawl adatbázis magyar részkorpuszán és a Wikipédia magyar nyelvű szövegein tanított Bert modell. Ennek a finomhangolásához használtuk fel a MedCollect korpusz szövegeit. Ahhoz, hogy a kontextus méretéből adódó különbségeket tesztelni lehessen, eltérő tokenhosszúságú (64, 128, 256 és 512) szekvenciákra szeletelve kapta meg a modell az egyes verziók tanítása során. Mivel a pragmatikai annotálásnál releváns kontextuális információkat találhatunk a teljes szövegben is, ezért a modellt nem mondatokra szegmentált tanító adatokkal láttuk el, hanem a korpuszban található teljes szövegek adott tokenhosszúságú részeivel. Annak érdekében, hogy tesztelhesük a szekvencián belüli pozíció jelentőségét is, meg kellett oldanunk, hogy minden annotálandó szónak legyen kellő nagyságú bal- és jobboldali kontextusa, ezért a szegmentumokat átfedéssel, eltolással állítottuk elő. Ennek szematikus ábrázolása látható az 1. ábrán.



1. ábra. A korpusz szövegeinek szekvenciákra osztása 50%-os átfedéssel.

Az egymást követő szekvenciák közötti átfedés 50 százalékos volt, tehát például, ha 128 tokenes szegmentumokkal dolgoztunk, akkor az első szekvencia második 64 eleme megegyezett a következő szekvencia első 64 elemével. Ezáltal, ha egy annotálandó elem a szekvenciájának a legvégén helyezkedett el, és nem volt mellette kellő nagyságú jobboldali kontextus, a következő szekvenciában középtájon jelent meg, így megfelelő méretű kontextus esetén is látta a tanulás során a modell. Annak érdekében hogy minden szó ugyanannyiszor (kétszer) szerepeljen a tanító szekvenciákban, az első szekvencia első felét, és az utolsó második felét külön szekvenciaként is szerepeltettük.

A tokenhosszúságot leszámítva az egyes verziók minden másban megegyeztek egymással, hogy a kontextusméretet azonos körülmények között tudjuk vizsgálni. A modellt 32-es batch-mérettel 4 epochban tanítottuk 5e-5 tanulási rátával. Emellett tízszeres keresztvalidálást alkalmaztunk.

A modellt megbízhatóságát a humán annotátorokéhoz hasonlóan értékeltük: a kurátori verziót tekintjük a szövegek helyes annotálásának, és ahhoz viszonyítottuk, hogy mennyire tudja megbízhatóan megjósolni az annotálandó szavak címkéjét a seqeval (Nakayama, 2018) Python modul használatával. A kiértékelésnél az egyes jegyekre vonatkozó pontosság, fedés és F1 érték megállapításán felül a modell teljes megbízhatóságának jellemzésére az összes jegyre vonatkozó súlyozott átlagát használtuk. A következő szakaszban az egyes verziókra vonatkozó adatokat ismertetjük, illetve az ezekből levonható következtetéseket részletezzük.

## 4 A különböző szekvenciahosszú adatokkal tanított huBert pontossági adatainak összevetése

A szekvenciahosszúság, azaz az egyes annotációs döntések meghozásához rendelkezésre álló kontextus nagyságának hatásának vizsgálatára négy verziót készítettünk a finomhangolt huBert base modellből. A verziókra vonatkozó megbízhatósági értékek közül az egyes kategóriákra vonatkozó F1 értékeket, valamint ezek súlyozott átlagát az 2. Táblázatban láthatjuk.

2. Táblázat: A különböző tokenhosszú szekvenciákkal tanított modellek F1 értékei kategóriánként és összesítve

<b>szekvenciahossz</b>	<b>64</b>	<b>128</b>	<b>256</b>	<b>512</b>	<b>support</b>
<b>nodirectiva</b>	0,87	0,88	0,91	0,90	2828
<b>sajat_hangu</b>	0,86	0,86	0,85	0,89	2036
<b>kozvetített</b>	0,46	0,45	0,44	0,47	402
<b>meta</b>	0,37	0,42	0,45	0,38	334
<b>szovegszervezo</b>	0,74	0,78	0,76	0,78	189
<b>interakcios</b>	0,47	0,55	0,56	0,56	114
<b>ambiguous</b>	0,00	0,00	0,00	0,00	24
<b>súlyozott átlag</b>	0,80	0,81	0,82	0,83	5927

A verziók súlyozott átlagát tekintve láthatjuk, hogy az elérhető kontextus növelésével javult az automatikus annotáció megbízhatósága. A javulás azonban kis mértékű, 0,80-ról 0,83-ra emelkedett csak az F1 értékek súlyozott átlaga a 64-es maximális tokenhosszúságú verziótól az 512-es verzióig. A modell megbízhatóságáról azonban elmondható minden verzió esetében, hogy összehasonlítható a humán annotátorok által elért megbízhatóságával ( $F1 = 0,830$ ), tehát a munka elsődleges célját elértnek tekinthetjük.

A két leggyakoribb címke a tanítókorpuszban a *nodirectiva* és a *sajat\_hangu* voltak. Ennek köszönhető lehet, hogy a modell minden tokenhosszúságnál ezeknek az annotálását tudta a leginkább elsajátítani, és átlagon felüli, 0,9 körüli F-értéket elérni. A többi funkcióból egy nagyságrenddel kevesebb tanító adat állt rendelkezésre, így nem meglepő, hogy ezeket nem volt képes olyan jól elsajátítani. Inkább meglepő ilyen szempontból, hogy a *szovegszervezo* funkciót milyen magas megbízhatósággal sajátította el, tekintve, hogy ez volt az egyik legkisebb előfordulással álló kategória. Ennek az lehet az oka, hogy ez egy könnyen sémába rendezhető, véges szókészletet használó funkció, amelynek a mintázatát néhány példa alapján is be lehet azonosítani. Az egyes funkciók nem mutatják a súlyozott átlagon megfigyelhető monoton növekedést, de alapjaiban véve elmondható, hogy a nagyobb szekvenciahosszúság pontosabb annotációhoz vezetett összesítésben és jegyenként is. A modell egyszer sem jelölt az *ambiguous* címkével, minden alkalommal, amikor a tanító adatok között azt látta, választott neki egy olyan funkciót, amit felismert.

Annak ismeretében, hogy a rendelkezésre álló kontextusméret azonos körülmények között ugyan javítja a modell megbízhatóságát, azonban ez a javulás kis mértékű, felmerülhet a kérdés, hogy szükséges-e a nagyobb tokenhosszúságú verziók használata, vagy megelégedhetünk például a 128-as szekvenciahosszúsággal finomhangolt

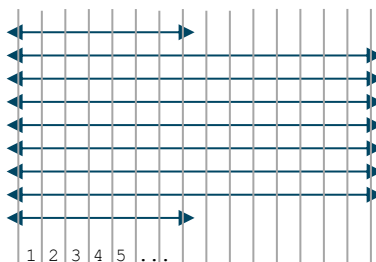
modellel is. A kísérlet további paramétereinek egyenlőségénél azt mondhatjuk, hogy igen, azonban a tanítás során megfigyeltük, hogy a kisebb maximális tokenhosszúságú verziók a tanulás végére az egyes lépésekben kevéssel javultak már (64-es tokenhosszúságnál a cumulative loss a harmadik epoch végén 0,0103, a negyedik epoch végén 0,0082), míg a nagyobb tokenhosszúságúak még lépésenként nagyobb javulást produkáltak (512-es tokenhosszúságnál a cumulative loss a harmadik epoch végén 0,0235, a negyedik epoch végén 0,0174). Emiatt felmerülhet a kérdés, hogy más tanítási feltételek mellett egy nagyobb modell megbízhatóbb eredményeket mutatott volna-e. Gondolhatunk itt az epochok számának növelésére, amellyel egy sokkal időigényesebb tanulási folyamat állna elő, de több lépés állna a modell rendelkezésére a nagyobb megbízhatóság elérésére, vagy eltérő átfedés alkalmazására, hogy több tanulható adatpont szerepeljen a tanítókorpuszban. A legkézenfekvőbb megoldás ezzel kapcsolatban pedig természetesen a tanítóadatok számának növelése lenne, ez azonban a jelenlegi helyzetben nem lett volna megoldható.

Összességében elmondható, hogy az első számú célunkat, az annotátorok eredményeihez hasonló megbízhatósági értékeket produkálni képes automatikus annotáló eszközt el tudtuk érni. A kontextus méretének vizsgálatában pedig arra jutottunk, hogy mutat ugyan javulást a modell az elérhető kontextus növelésével, ez a javulás nem nagy mértékű, így azonos feltételek mellett azt mondhatjuk, hogy a kontextus méretének hatása az automatikus annotálás megbízhatóságára nem nagy.

## 5 A szekvencián belüli pozíció hatása

A pragmatikai annotáció során a kontextusnak az annotáció megbízhatóságára gyakorolt hatását más módon is vizsgálhatjuk. Megnézhetjük azt is, hogy az annotáció során az egyes nyelvi elemeknek az annotációjának a pontossága/fedése hogyan változik a szekvencián belül elfoglalt helyétől. A vizsgálat során megállapíthatjuk, hogy a szekvenciák elején, közepén vagy a végén található elemek címkézése történt-e megbízhatóbb módon, illetve hogy ez a megbízhatóságváltozás szimmetrikus-e, azaz ugyanolyan hatása van-e a bal és a jobb oldali kontextusnak. Várakozásunk szerint a szekvenciák közepén nagyobb megbízhatósággal osztályoz az eszköz, mint a szélein, illetve a maximális megbízhatóság a szekvencia közepétől jobbra lesz megfigyelhető, vagyis nagyobb bal oldali kontextus szükséges a helyes osztályozáshoz, mint jobb oldali. A várható aszimmetria oka az, hogy a természetes nyelv használata során lineárisan, balról jobbra produkáljuk és dolgozzuk fel a szövegeket, és a feldolgozhatóság, azaz a megértés szempontjából előnyösebb, ha a bal oldali kontextus hordozza azokat az információkat, amelyek a megértéshez szükségesek. Például a felszólító alakok megértése szempontjából azt, hogy egy felszólító alak *nodirectiva*, azaz kötőmódú-e, legtöbbször a felszólító alakot tartalmazó tagmondatot megelőző tagmondat igéje határozza meg, pl. *Belefáradtam, hogy állandóan maszkot viseljek*. Ugyanakkor a jobb oldali kontextus is fontos lehet bizonyos esetekben, például idézések esetében gyakran az idézett szövegrész után jelenik meg az idézést kifejező szövegrészlet: *Mindenki viseljen maszkot – mondta az államtitkár*. A kérdés az, hogy a nyelvben megfigyelhető aszimmetria jelentkezik-e az automatikus annotáló eszköz működése során is.

Az 1. ábrán látható volt, hogy a modell tanítása során hogyan lettek kialakítva egy szöveg szavaiból 50 százalékos átfedéssel az egyes, tanításra és tesztelésre használt szekvenciák. A szekvencián belüli pozíció hatásának vizsgálatához pozícióként összehasonlítottuk az egyes szekvenciáknak a tanításhoz használt változatán megfigyelhető címkéit ugyanezen szekvenciának a tesztelés során megjósolt címkékkel, azaz gyakorlatilag a szekvenciákat a 2. ábrán látható módon egymás alá toltuk, és az első pozícióban megfigyelt és megjósolt címkék alapján számoltuk az első pozíció pontosságát, fedését és F-értékét, és így tovább: minden pozícióhoz meghatároztuk ezeket a megbízhatósági értékeket.



2. ábra. A nyelvi elemek címkézésének megbízhatóságmérése pozícióként.

A felszólító alakok funkcióinak meghatározása során szavakhoz lett címke rendelve, a szekvenciák viszont tokenekből állnak, ezért a vizsgált címkék a szó első tokenjének a pozíciójában lettek figyelembe véve.

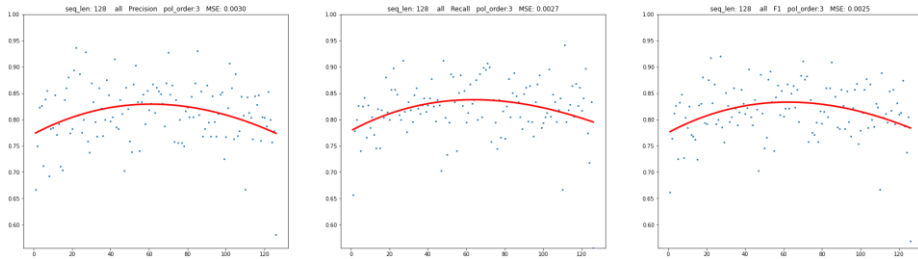
A megbízhatósági értékek funkcióként külön-külön és összesítve is meg lettek határozva. A funkciókénti számolásnál true pozitív lett, ha a megfigyelt és a megjósolt címke is az adott funkciócímke volt; false pozitív esetén a megjósolt címke az adott címke volt, de a tényleges címke ettől eltérő (vagy 0 címkéjű); false negatív esetében pedig a megfigyelt címke egyezett meg a vizsgált címkével, a jósolt pedig nem. Az összesített megbízhatóság esetében TP-nek az azonos (de nem nulla) címkézés, FP-nek a különböző (de nem 0 jósolt) címkézés, FN-nek pedig a különböző (de nem 0 tanított) címkézés számított, azaz az osztályozás mikroátlagot számítottuk. A megbízhatósági értékeket 64, 128, 256 és 512 szekvenciahosszra is meghatároztuk, ezek grafikonos ábrázolása a <https://github.com/szecsenyi/MSZNY2025> githubon található.

A pozíciókénti megbízhatósági értékeknek igen nagy a szórása. Ennek leginkább az az oka, hogy az egyes pozícióknál aránylag kevés felszólító alak található. Az 1. táblázatban is látható, hogy összesen mintegy 6000 felszólító alakot kell felcímkézni, ez 64 tokenhosszúságú szekvenciák esetében átlagosan 100 címkét jelent, 512 szekvenciahossznál azonban csak 10-et. Még kevesebb átlagos címkézés jut egy-egy pozícióra, ha csak az egyes értékek megbízhatóságát vizsgáljuk: *nodirectiva* és *saját hangú* címkékből 2000-3000 található, de a többiből már csak 500 alatti mennyiségű. Ez utóbbi esetekben 64, illetve 512 szekvenciahossznál átlagosan már csak 10, ill. 1 alatti mennyiség jut az egyes pozíciókra, ezek az eredmények így már valójában értékelhetetlenek.

A megbízhatósági értékek nagy szórása miatt a pozíció és a megbízhatóság közötti összefüggést a pontthalmazra illesztett görbékkel lehet jobban szemléltetni. Másodfo-

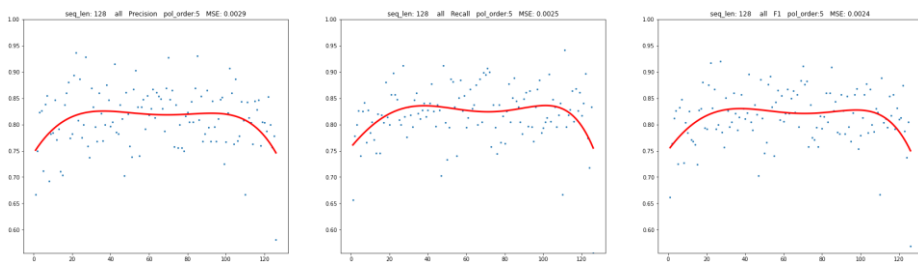


kú polinom illesztése esetén a kapott görbe a maximális értékhez viszonyítva szimmetrikus, ezért a bal és jobboldali kontextus hatásának a különbségét nem mutatja. Harmadfokú polinomnál már látható ez a különbség is, ezért a grafikonoknál ezt alkalmaztuk. Ötödfokú polinom esetében elég nagy pozíciókénti címkeszámnál még jobban megfigyelhető az összefüggés, ezért ilyen közelítéseket is ábrázoltunk. A polinomok illesztését a numpy python library `polyfit` függvényével végeztük. A polinomillesztés metrikájaként a polinom átlagos négyzetes eltérését (MSE) számítottuk, ami az ábrák feliratában is megjelenik.



**3. ábra.** 128 tokenhosszúságú szekvenciáknál az egyes pozíciókban mért P, R és F1 értékek az összes címkét figyelembe véve, illetve a pontokhoz illesztett harmadfokú polinom.

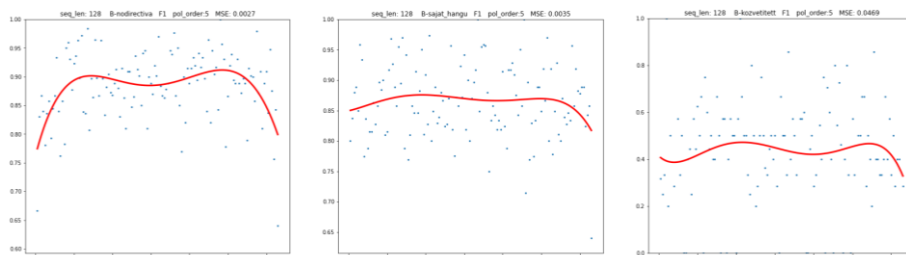
Az 3. ábrán látható, hogy mindhárom megbízhatósági érték esetében a széleken alacsonyabb értékek vannak, mint a szekvencia közepén. A regressziós görbék továbbá aszimmetriát is mutatnak, mivel a maximális érték nem az szekvencia közepére esik, és két oldalán nem ugyanolyan mértékű az értékek csökkenése. Az aszimmetria még jobban megfigyelhető ötödfokú polinom illesztése esetén (4. ábra): a szekvencia közepén található aránylag egyenes szakasz bal oldalán hosszabb, de laposabb emelkedés van, a jobb oldalán rövidebb, de meredekebb csökkenés.



**4. ábra.** 128 tokenhosszúságú szekvenciáknál az egyes pozíciókban mért P, R és F1 értékek az összes címkét figyelembe véve, illetve a pontokhoz illesztett ötödfokú polinom.

A grafikonok alapján a felszólító alakok funkcióinak megállapításakor a huBERT az annotált elemek bal oldalán nagyobb kontextust vesz figyelembe, mint a jobb oldalán, de a két (különböző méretű) kontextus hatása nagyjából megegyezik. Az 4. ábra grafikonjai alapján a releváns baloldali kontextus mérete kb. 25–30 token hosszúságú, a jobb oldal pedig 15–20.

A kontextusfüggőség az egyes funkcióknál is megfigyelhető, bár különböző mértékben (5. ábra). A *nodirectiva* értéknél nagyobb a kontextus hatása, mint a *saját hangú* értéknél, de a figyelembe vett kontextusok nagysága nem különböző. A közvetített funkció grafikonján észrevehető, hogy az F1 értékek sokkal jobban szóródnak, mint a másik funkciónál, és a regressziós görbe is jóval alacsonyabban van. Ennek oka az, hogy a korpuszban sokkal kevesebb ilyen címkéjű elem szerepel; ennek tudhatjuk be azt is, hogy ebben a grafikonban a pontok vízszintes csíkokba rendeződnek.



**5. ábra.** 128 tokenhosszúságú szekvenciáknál az egyes pozíciókban mért F1 értékek az a *nodirectiva*, *saját hangú* és a *közvetített* funkciónál, illetve a pontokhoz illesztett ötödfokú polinom.

## 6 Összegzés

A tanulmány azt vizsgálta, hogy a pragmatikai jegyek nagy nyelvi modellel történő automatikus annotációja során az eszköz mennyire veszi figyelembe a rendelkezésre álló kontextust. A vizsgálat a MedCollect egészségügyi álhírkorpuszban található felszólító alakok különböző pragmatikai funkcióinak kézzel annotált változatának segítségével történt.

A kontextuális hatás kétfajta módszerrel lett elemezve. Az egyiknél azt vizsgáltuk, hogy a tanításnál és tesztelésnél használt 50%-os átfedéssel kialakított szekvenciák hossza (64, 128, 256, 512 token) befolyásolja-e az eszköz megbízhatósági értékeit (P, R, F1). Azt tapasztaltuk, hogy a szekvenciahossz növelésével együtt nőtt az F1 érték is (0,80, 0,81, 0,82 és 0,83). A másik módszernél azt vizsgáltuk, hogy a felszólító alakok a tesztsekvencián belüli pozíciója hogyan befolyásolja a funkció meghatározásának megbízhatóságát. Azt tapasztaltuk, hogy a szekvenciák széléhez közeli pozíciókban alacsonyabbak voltak a megbízhatósági értékek, mint a szekvenciák közepén, vagyis a rendelkezésre álló kontextusnak volt hatása, továbbá hogy a kontextusfüggőség aszimmetrikus, vagyis nagyobb bal oldali kontextust (25–30 token) vesz figyelembe az eszköz, mint jobb oldalit (15–20 token). A figyelembe vett kontextus nagysága megmagyarázza azt is, hogy a szekvenciák hosszának növelése miért csak ilyen kis mértékben növeli a megbízhatóságot: ha a funkciók megállapításához csak  $\pm 25$  token kontextust vesz figyelembe az eszköz, akkor a szekvenciák közepén, ahol ez a kontextus rendelkezésre áll, állandó megbízhatóságú a kategorizá-

lás, amit csak a szekvenciák végeinek megbízhatósága ront le. Minél hosszabb a szekvencia, annál nagyobb a középső rész hossza a végekhez viszonyítva.

A korpuszban levő felszólító alakok kis száma miatt a kis előfordulású funkciók pozícionkénti megbízhatóságának mérése nem értelmezhető. Ez javítható lenne azzal, ha a szekvenciák átfedését megnövelnénk 75 vagy 90 százalékra.

A nagy nyelvi modell figyelembe veszik az automatikus pragmatikai annotáció során a rendelkezésre álló kontextust. A figyelembe vett kontextus azonban sokkal kisebb, mint a humán annotátorok esetében feltételezett.

## Bibliográfia

- Archer, D., Culpeper, J., Davies, M.: Pragmatic annotation. In: Lüdeling, A., Kytö, M. (szerk.) *Corpus Linguistics: An International Handbook*. pp. 613–542. Walter de Gruyter, Berlin; New York (2009)
- Milà-Garcia, A.: Pragmatic annotation for a multi-layered analysis of speech acts: A methodological proposal. *Corpus Pragmatics* 2, 265–287 (2018) <https://doi.org/10.1007/s41701-018-0037-z>
- Nakayama, H.: seqeval: A Python framework for sequence labeling evaluation. (2018)
- Nemeskey, D. M.: Introducing huBERT. In: Berend, G., Gosztolya, G., Vincze, V. (szerk.) XVII. Magyar Számítógépes Nyelvészeti Konferencia. pp. 3–14. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2021)
- Nemeskey, D. M.: *Natural Language Processing Methods for Language Modeling* (PhD Thesis). Eötvös Loránd University (2020)
- Németh T. E.: Álhírek, áltudományos nézetek nyelvészeti azonosítása. *Magyar Nyelv* 119, 490–496 (2023) <https://doi.org/10.18349/MagyarNyelv.2023.4.490>
- Szécsényi T., Nagy C. K., Németh T. E.: Felszólításannotálás a MedCollect egészségügyi állírkorpuszban. In: Berend G., Gosztolya G., Vincze V. (szerk.) XX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 159–170. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2024)



# Internetes hírek automatikus osztályozása

Osváth Máttyás<sup>1</sup>, Héja Enikő<sup>1</sup>

<sup>1</sup> HUN-REN Nyelvtudományi Kutatóközpont, 1068 Budapest, Benczúr u. 33.

{osvath.matyas, heja.eniko}@hun-ren.nytud.hu

**Kivonat:** Az elmúlt évtizedekben az online sajtó vált a hírek egyeduralkodó forrásává, amely egyre növekvő igényt teremt az automatizált tartalomkategorizálási módszerek iránt. Tudomásunk szerint ez az első olyan kutatás, amelynek célja magyar hírek téma szerinti osztályozása nagy nyelvi modellekkel. Jelen vizsgálat célja az online elérhető magyar hírekben megjelenő témák azonosítása és a hírek szövegeinek osztályozása. Az elemzés anyagát 77 magyar hírportál 5,225,195 szövege adta 2013-tól 2023-ig. A kategória-készlet elkészítésére két módszert vizsgáltunk meg: egy statikus szóbeágyazásokon alapuló módszert, valamint egy BERT alapú témamodellezést. A két módszer eredményeit manuálisan validáltuk, így jött létre a 19 elemből álló kategóriakészlet. A hírek osztályozásához három módszert alkalmaztunk: egy már létező címkeajánló rendszer mellett a nagy méretű XLM-RoBERTa és a PULI Llumix 32K Instruct modell performanciáját vizsgáltuk, ez utóbbit zero-shot tanítással. Bár az eredmények nagy szórást mutattak, azt gondoljuk, hogy a PULI Llumix 32K Instruct modell kis munkával továbbfejleszhető egy nagy pontosságú osztályozóvá.

## 1 Bevezetés

### 1.1 Célkitűzés

Az elmúlt évtizedekben a nyomtatott sajtó visszaszorulásával az online média vált a hírek egyeduralkodó forrásává. A jelenleg magyarul online elérhető hírek szövegeinek mérete már több milliárd tokenre rúg. Ez a tény illetve, hogy napi szinten keletkeznek újabb és újabb hírek, szükségessé teszi, hogy a híreket tartalmuk szerint kategorizálni tudjunk.

Amennyire tudjuk, ez az első olyan kutatás, amelynek célja magyar hírek téma szerinti osztályozása. Kutatásunkhoz legközelebb talán a sajtószövegek automatikus tematikus címkézését célzó címkéző algoritmus áll (Yang és mtsai, 2020), amely a hírszövegek nagy mennyiségű, manuálisan létrehozott címkéiből indult ki.

Így a jelen kutatás célja kettős: egyfelől egy nagy fedésű, ám viszonylag átlátható, jól definiált kategória rendszert kívánunk kialakítani, amely az osztályozás alapjául szolgálhat, másfelől a hagyományos gépi tanulásos osztályozó algoritmusokkal szemben (tartóvektor-gép, naiv-Bayes, véletlen erdő) azt kívánjuk vizsgálni, hogy az

újabb, nagy nyelvi modelleken alapuló eljárások milyen pontossággal képesek az osztályozás elvégzésére. Az elemzés anyagát kb. 80 magyar nyelvű hírportálon 2013 és 2023 között megjelent kb. 5 millió hír adta. A kategória-készlet elkészítése során két automatikus módszert használtunk. Az első módszer statikus szóbeágyazásokon alapult: az URL-ek releváns szegmenseinek word2vec reprezentációjából gráfokat építettünk, majd ezen gráfok részgráfjait vizsgálva automatikusan nyertük ki a kategória jelölteket. A második módszer a BERT alapú témamodellőzés volt, ahol a szövegek vektortérbeli beágyazása után dimenziócsökkentést és klaszterelemzést alkalmaztunk a témák kinyerésére. A következő lépésben az automatikusan kinyert témákat manuális validálás során kategóriákba szerveztük. A végső kategóriakészlet a két módszer eredményeinek manuális egyesítésével jött létre. A kutatás második szakaszában 500 db véletlenszerűen kiválasztott hírt soroltunk be a végső 19 kategória alá. A hírek osztályozása 3 féle módon történt. Először a már említett automatikus neurális címkeajánló modell használatára került sor. A második és harmadik kísérlet során két különböző modellt használtunk: egyfelől a Meta által kiadott nagy méretű XLM-RoBERTa-t, másfelől a PULI LlumiX 32K Instruct modellt zero-shot tanítással. Eredményeink azt mutatják, hogy a PULI LlumiX 32K Instruct megközelíti a címkeajánló pontosságát, míg az XLM-RoBERTa mindössze 18% pontosságot produkál.

## 1.2 Szakirodalmi háttér

Tudomásunk szerint a hazai kutatások elsősorban a témamodellőzésre koncentrálnak. Ezen kutatások főként Látens Dirichlet Allokációt (LDA) alkalmaztak (pl. Barna és mtsai, 2023 és Vági és mtsai 2023). De olyan kutatásokról is tudunk, amelyek BERT alapú témamodellóezést használtak (pl. Nagy, 2024 és Gelei és mtsai 2023). Általánosságban elmondható, hogy a szerzők témamodellőzésre használt módszereket alkalmaznak egy adott domain-specifikus korpuszra (pl. politika, vállalkozás, értékelések), melyek az így keletkezett témákkal leírhatóvá válnak. Azonban eddig nem történt olyan vizsgálat, amely a témamodellőzésre szolgáló technikákat összehasonlította volna, sőt olyan magyar vizsgálatról sem tudunk, amely magyar szövegek téma szerinti osztályozásával foglalkozott volna. Kutatásunkhoz legközelebb talán a sajtószövegek automatikus tematikus címkézését célzó algoritmus áll (Yang és mtsai, 2020), amely a HVG nyomtatott, illetve online verzióinak manuálisan létrehozott címkéiből indult ki. Mivel ezek a címkék általában nem követnek központi szerkesztési elveket, hanem újságírók hozzák létre őket eseti alapon, a címkékészlet nagyon nagy, és sokszor átgondolatlan. A HVG esetében Yang és mtsai több mint 75.000 címkét használtak.

Az angol nyelvű hírek esetében már számos olyan adatbázis áll rendelkezésünkre, ahol a hírek kategóriákba vannak szervezve. Ezek közül talán a legrégebbi és legszélesebb körben használt a 20 Newsgroups Dataset, amely kb. 20,000 hírt tartalmaz, amelyek megközelítőleg egyenletesen oszlanak meg 20 különböző hírkategória között. A híradatbázis érdekessége, hogy bár egészen a legutóbbi időkig intenzíven használták

osztályozási feladatok kiértékelésére<sup>1</sup>, amennyire tudjuk nem áll rendelkezésre részletes dokumentáció sem a híryanag gyűjtéséről, sem a kategóriák kialakításáról, sőt még a szerző sem biztos<sup>2</sup>. Egy további probléma a kategóriák kapcsán jelentkezik: bizonyos hírosztályok nagyon szorosan összetartoznak, míg mások sokkal távolabb vannak egymástól, és így jobban elkülöníthetőek. Egy másik ismert híradatbázistól, az AG's corpus of news articles<sup>3</sup>-től eltekintve, a kategorizált híradatbázisok jellemzően egy hírportál anyagait tartalmazzák, és kategóriarendszerük vagy kis számú, előre definiált elemből áll, például a BBC News (Bose, 2019) 5 előre definiált kategóriát használ, vagy az adott hírportál kategóriát használják fel, például a Guardian News (Kharosekar, 2023), de olyan is van, ahol nem derül ki egyértelműen, hogy a kategóriák hogyan kerültek kialakításra, például a multimodális N24News (Wang és mtsi, 2022), amely a New York Times híreit sorolja 24 kategóriába. A fenti megközelítések nemcsak az alkalmazott kategóriák számában térnek el, hanem abban is, hogy az osztályokba sorolt cikkek száma kategóriánként egyenletes eloszlást mutat-e vagy sem. A fentiek alapján azt mondhatjuk, hogy munkánk újszerűnek tekinthető, hiszen célja egy olyan általános kategóriarendszer részben adatvezérelt kidolgozása, amely portáltól függetlenül lehetővé teszi a hírek besorolását.

## 2 Munkafolyamat

### 2.1 Adatgyűjtés

A kutatás anyagát 5.225.195 online elérhető magyar cikk anyagából állt össze, amely a nyilvánosan elérhető Common Crawl<sup>4</sup> 2013 és 2023 közötti magyar nyelvű alkorpuszának a részét képezte. Az anyagot úgy állítottuk össze, hogy az URL-ekben leggyakrabban szereplő forrás domainekből kézzel kiválogattuk a híroldalakat – összesen 77-et. Az 1. Táblázat mutatja a 10 leggyakoribb forrásdomainhez tartozó URL-ek számát.

1. Táblázat: a 10 leggyakoribb forrás domain és az azokhoz tartozó URL-ek száma

#	Forrás domain	URL-ek száma
1	index.hu	654.583
2	24.hu	547.370
3	hvg.hu	301.694
4	prohardver.hu	191.579
5	velvet.hu	182.266
6	444.hu	175.886
7	femina.hu	167.298

<sup>1</sup> Pl. <https://paperswithcode.com/sota/text-classification-on-20news>

<sup>2</sup> Általában Lang (1995)-nek tulajdonítják, de a cikk nem említi expliciten ezt az adatbázist (vö. <http://qwone.com/~jason/20Newsgroups/>)

<sup>3</sup> [http://groups.di.unipi.it/~gulli/AG\\_corpus\\_of\\_news\\_articles.html](http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html)

<sup>4</sup> <https://commoncrawl.org/>

8	rangado.24.hu	153.913
9	starity.hu	131.728
10	infostart.hu	122.638

## 2.2 A kategóriarendszer kialakítása

A kutatás első részében statikus, illetve kontextuális szóbeágyazásokat is használtunk a kategóriakészlet kialakítására. A végső kategóriakészlet ezek összevonásából született meg a manuális validálás során.

### A kategória-készlet kialakítása statikus szóbeágyazásokkal

A statikus szóbeágyazásokon végzett kísérleteink azon az előfeltevésen alapultak, hogy a hírportálok esetében az URL-ek struktúrája tükrözi, hogy az adott sajtóorgánium milyen kategória alá sorolta be az adott hírt. Így az URL-ek koherensebb képet nyújtanak a kategóriákról, mint a címkék, hiszen ez utóbbiak sok esetben egyedi döntés nyomán születnek meg.

A munkaszakasz az alábbi lépéseket követte:

(1) Az összes URL-ből a megfelelő URL szakasz kivágásával létrehoztuk a kategória-jelölteket. A kategória-jelölteket a forrástörzset után közvetlenül követő legközelebbi két ‘/’ közötti karaktersorozatként azonosítottuk.

(2) A következő lépésben a kategória-jelölteket automatikusan ékezetesítettük (Laki és Yang 2020), melynek eredményeképpen 1234 kategóriajelölteket kaptunk (pl: *önidő, önleány, öregség, örökhagyás, örülünkvincent*).

(3) Ebben a lépésben, word2vec reprezentációt rendeltünk a kategória-jelöltekhez. Ehhez a NYTK-word2vec<sup>5</sup> modellt használtuk. Sajnos 263 esetben nem állt rendelkezésünkre word2vec reprezentáció (pl. *whistlermamája, dokuexport, tvtorrent, asparhelttükai*), így ezeket az URL töredékeket nem vettük figyelembe. Ezek jelentős része kitalált szóösszetétel volt.

(4) Héja és Ligeti-Nagy (2023) cikkben leírt módszertan alapján készítettük el a maradék 969 kategória-jelölt szomszédsági mátrixát, amelyet egy alkalmas küszöbérték (0.5) mentén binarizálva egy hurokél-mentes, irányítatlan G gráffá konvertáltunk. A G gráf 386 izolált csúcsot tartalmazott, tehát összesen 583 URL-fragmentum szolgált a kategóriák alapjául.

(5) Majd megvizsgáltuk, hogy a G gráfban vannak-e olyan részgráfok, amelyek kategória-jelölteket hírkategóriáknak megfelelő szemantikai osztályokba szervezik.

A kézi kiértékelés során összefüggő részgráfokat, teljes részgráfokat (klikkek) és k-klikk közösségeket vizsgáltunk ( $k=3$ ). Az összefüggő komponensek vizsgálata során azzal a problémával szembesültünk, hogy mindig van egy óriás komponens, amely több

<sup>5</sup> [https://nlp.nytud.hu/word2vec/cbow\\_3.tar.gz](https://nlp.nytud.hu/word2vec/cbow_3.tar.gz)



kategóriát is összevon, másfelől pedig a kisebb komponensek összetartozó kategóriákat szabdalnak fel (pl.  $\{unicredit, erste, mkb\}$  és  $\{raiffeisen, cib\}$ ).

A klikkek kiértékelése során azt találtuk, hogy a teljes részgráfok “túl érzékeny” struktúrák, vagyis ha egy teljes részgráfból valami miatt egy él hiányzik, a teljes gráf több klikkre esik szét a kategóriák indokolatlan szétDarabolását eredményezve (pl.  $\{2020, 2021, 2019, 2023\}$  és  $\{2021, 2022, 2023\}$ ).

A klikkhez közel álló, ám stabilabb reprezentációt nyújtó struktúra a k-klikk közösség, amely kevésbé érzékeny az élek elhagyására. Hiszen egy k-klikk közösség azon k méretű klikkek uniója, amelyek elérhető szomszédos (k-1 csúcsot megosztó) k-klikkeken keresztül.

Azt találtuk, hogy a k-klikk közösségek (k=3) eredményezik a legjobb szemantikai osztályokat. A 386 releváns URL szegmensből 352 került 3-klikk közösségekbe 37 csoportosítást eredményezve. Sajnos a 37 csoportból 2 szemantikailag inkoherens, feltehetőleg kisebb koherens csoportok összevonásából származik, így végül 35 szemantikailag koherens 3-klikk közösséget kaptunk, amelyek összesen 196 URL-szegmenst tartalmaztak. A 2. Táblázatban mutatunk be néhányat az automatikusan létrejött kategóriákból:

2. Táblázat: Néhány k-klikk közösségekkel létrehozott szemantikai osztály

#	Szemantikai osztályok
1	{'akciók', 'nyereményjátékok', 'promóció'}
2	{'akciók', 'programok', 'rendezvények'}
3	{'frizura', 'smink', 'szexi'}
4	{'allergia', 'cukorbetegség', 'gyermekbetegségek', 'kórkép', 'laktózérzékenység', 'savtúltengés'}
5	{'felhasználó', 'fogyasztó', 'szolgáltatás', 'termék', 'vásárló'}
6	{'baba', 'gyerek', 'kölyök', 'poronty'}
7	{'bank', 'deviza', 'devizaárfolyamok', 'hitelek', 'kötvény', 'pénzpiacok', 'részvény', 'részvényárfolyamok', 'állampapír', 'árfolyamok'}
8	{'menedzserek', 'szakértők', 'tervezők'}
9	{'blogger', 'celeb', 'klasszis', 'riporter', 'stílusikon', 'sztár', 'sztárok', 'sztárvilág', 'világsztár'}
10	{'nb1', 'nb2', 'nb3', 'rájátszás', 'tabella', 'teremfoci'}

Végül a létrejött szemantikai osztályokat kézzel validáltuk, melynek során bizonyos szemantikai osztályokat összevontunk, másokat pedig irrelevánsnak ítéltünk a feladat szempontjából, így ezeket elhagytuk.

### A kategória-készlet kialakítása BERT témamodellezéssel

Másodszor, a hírekben rejlő témák és kategóriák kinyerésére a BERT alapú témamodellezést is alkalmaztunk. A témamodellezés egy nem felügyelt gépi tanulási módszer, amelyet dokumentumokban látens témák azonosítására használnak. Jelen kutatásban Grootendorst (2022) által javasolt módszert követtük az alapbeállításokkal együtt, amely az alábbi lépésekből áll. A szövegek vektortérbeli beágyazására a BERT

modell (Devlin és mtsai, 2019) variánsait használtunk. A Nemeskey (2020) által létrehozott első magyar nyelvű BERT modell, a huBERT, és a Yang és munkatársai (2023) által fejlesztett 345 millió paraméteres PULI BERT modellt.

Következő lépésben dimenziócsökkentésre került sor az UMAP algoritmust felhasználva. Ezután egy sűrűség alapú hierarchikus klaszterezési algoritmust, a HDBSCAN-t használtuk a klasztereket létrehozására, azzal a megkötéssel, hogy egy klaszter minimum 15 szót tartalmazhat, ami az a minimális elemszám, amely egy csoport klaszterként való azonosításához szükséges; az ennél kisebb elemszámú csoportokat az algoritmus zajként azonosítja. A módszer egyik fontos előnye, hogy nem szükséges előre meghatározni a klaszterek számát. A témák azonosításához c-TF-IDF (class-based TF-IDF) módszerrel végeztünk témakiválasztást. Az utolsó lépésben kézzel validáltuk az eredményeket.

### 2.3 Osztályozás

A kutatás második részében háromféle osztályozót alkalmaztunk a hírek kategóriákhoz való hozzárendelésére. A hírekből véletlenszerű mintavétellel választottunk ki 500 darabot, és ezeket a híreket manuálisan a kategóriákhoz rendeltük.

Először a Bevezetésben már említett automatikus neurális címkeajánló modell használatára került sor (Yang és mtsai, 2020). A modell címkeket rendel a bemeneti hírekhez a hozzájuk tartozó valószínűségi értékekkel. Mivel a modell címkekészlete és a jelenlegi kategóriarendszer között csak nagyon kicsi az átfedés (több, mint 75.000 címke vs. 19 kategória), ezért minden olyan prediktált kategóriát helyesnek tekintettünk, ahol a címke az általunk létrehozott kategóriába tartozott (pl. a pénz címke a gazdaság kategóriába tartozik).

Másodszor, egy előtanított többnyelvű RoBERTa modell továbbfejlesztett változatát használtunk osztályozásra az általunk kialakított kategóriák valószínűségeinek a meghatározásához zero-shot tanítási környezetben (Liu és mtsai, 2019). A Meta által kiadott, nagyméretű XLM-RoBERTa modellt alkalmaztuk, amely 2,5 TB méretű adaton lett előtanítva. A tanító adat 100 nyelvet tartalmazott, köztük magyart. A modell BERT architektúrára alapul és 560 millió paraméterrel rendelkezik (Conneau és mtsai, 2020). Az osztályozás finomhangolás nélkül történt, mivel még nincs megfelelő finomhangolásra alkalmas adathalmaz a magyarra.

Harmadszor egy generatív nyelvmodellt használtunk, a Nyelvtudományi Kutatóközpont által fejlesztett PULI LlamaX 32K Instruct modellt (Yang és mtsai, 2024) zero-shot tanítási környezetben. A modell azt a feladatot kapta, hogy az előre definiált kategóriák listájából a legvalószínűbb témát jósolja meg. Az alábbi prompt használtuk:

```
Milyen kategóriába tartozik az alábbi szöveg: {szöveg}.  
Egy szóval válaszolj! A lehetséges kategóriák:  
{kategóriák}
```

### 3 Eredmények

#### 3.1 Kategóriarendszer elkészítése

A statikus szóbeágyazások esetében a kézi validálás eredménye a következő 22 téma lett: *autó, motor, sport, életmód, történet, bulvár, szórakozás, média, társadalom, oktatás, gazdaság, kultúra, politika, pszichológia, tudomány, világ, egészség, betegség, ünnepek, család, természet és táplálkozás.*

A huBERT és PULI BERT által feldolgozott szövegek nagyságrendileg azonos, 4.732 és 4.741 témát adtak eredményként. Ezután, a kapott témák jól, közepesen és rosszul definiált kategóriákra való szétbontására került sor manuálisan, mivel az algoritmus hajlamos nehezen értelmezhető vagy inkoherens témákat eredményezni (Grootendorst, 2022). Egy téma jól-definiált, ha kizárólag jól formált szavakat tartalmazott és szemantikailag is koherens volt. Például az egyik hazai ellátást leíró téma az alábbi kulcsszavakat adta: *ellátás, ellátó, ellátások, ellátáshoz, egészségügy, sürgősségi, családtámogatás, gyermekegészségügy és üzemorvos.* A közepesen vagy rosszul definiált témák szavai hasonló részsztringeket tartalmaznak, de szemantikailag nem koherensek. Például az egyik téma az alábbi kulcsszavakat foglalta magában: *hálózat, halott, hálósoba, hálóraj és halottasház.* A modellek által adott kimenetek kiértékelése során érdekes megfigyelés volt, hogy a két eltérő BERT modell által detektált témák nagyfokú átfedést mutattak: számos témát tehát mindegyik BERT modell felismert. Összesen 440 jól-definiált témát sikerült azonosítani (vagyis a témák kb. 10%-a volt jól-definiált). A 440 téma közül a 3. táblázatban 10 téma kerül bemutatásra. A hírek közt számos téma megjelent, köztük időjárás, közszereplők, cégek, szociális média, turizmus és asztrológia. A témák további differenciálása kérdéses.

3. Táblázat: Azonosított témák és kulcsszavak

#	Kategória	Kulcsszavak
1	Közszereplő	Orbán Viktor, Trump, Merkel, Hugh Jackman, John Kennedy
2	Szociális média	Instagram, Facebook, Netflix, Youtube
3	Cég	Apple, Coca cola, Disney, Ikea, Lego, Unicef, IBM, NVIDIA, Spotify
4	Időjárás	meteorológia, meteor, őszi, havazás, eső, riasztás, előrejelzés
5	Turizmus	turisztika, szezonális, húzóágazat, kormánybiztos, belföld, külföld, árbevétel
6	Foci	magyar, foci, futball, futballista, liga, fc, ferencváros, nemzetközi foci
7	Asztrológia	csillagjegy, horoszkóp, szex horoszkóp, pénz horoszkóp, kínai horoszkóp
8	Ezotéria	vénusz, rejtély, férfi, erogén zóna

9	Szex	szextrükk, szexfüggőség, szexvarázsló, szexpozíció, szexpóz, szerelem és szex
10	Kritika	filmkritika, cinematrix, pókverzum, romantikus, vadidegen, avengers, színdarab, paródia

A statikus szóbeágyazásokon alapuló és a BERT témamodellőzés során is azt találtuk, hogy számos téma csoportosítható egy nagyobb, felettes kategóriába, így mindkét módszer esetében a szemantikailag összetartozó témákat kategóriákba vontuk össze. A megbízhatóság növelése érdekében a kategóriák iteratív megbeszélések révén kerültek kiválasztásra. Összességében, a szóvektorok által adott témák definiáltsága megelőzte a témamodellőzését: az értelmezhető kategóriák aránya meghaladta a BERT alapú témamodellőzését, de a témamodellőzés eredményeként új kategóriák is létrejöttek. A végleges kategóriarendszer a két módszerrel előállított témák metszetéből került előállításra. A 4. táblázatban található a 19 kategória és a hozzájuk tartozó témák.

A hírekben azonosított témák és kategóriák a magyar média által hasznosnak vélt, rendszeresen előforduló témáit mutatja be. A témamodellőzés egyik hátránya, hogy feltételezi, hogy egy hír egy témáról szól, ami nem feltétlenül teljesül, és emiatt a hírekben rejlő egyedi témák felismerése problémákba ütközhet.

4. Táblázat: Témák összevonása kategóriákba

#	Kategória	Témák
1	Média	Belföld, külföld, klímaváltozás, időjárás, természeti katasztrófa, büntetés, baleset, építkezés, magazin
2	Bulvár	Sztárok, celebek, hírességek, életrajz, közszereplő, horoszkóp, ruházat
3	Ünnepek	Húsvét, karácsony, születésnap
4	Politika	Elnökválasztás, parlament, jobboldal, baloldal, demokrácia, Fidesz, választás, tiltakozás, sztrájk, miniszterelnök
5	Technológia	bio-tech, cpu, hardver, notebook, internet, robot, technológiai áttörés, telefon
6	Tudomány	Biológia, nobel-díj, űrkutatás, agrártudomány, univerzum
7	Kultúra	Zene, mozi, film, fesztivál, múzeum, fotózás, képregény, könyv, sorozat, művész
8	Gazdaság	Bank, pénz, tőzsde, segély, vállalat, részvény, kötvény, befektetés, biztosítás, hitel, kata, ingatlan, olaj, adomány, babaváró
9	Család	Szülők, anya, gyerek, baba, fiatalok, örökbefogadás, házasság, szülés
10	Egészség és életmód	Vitamin, vegan, vega, gluténmentes, táplálkozás, szépségápolás, fogyókúra, bodypozitivitás, dohányzás, testmozgás, immunrendszer, fogyókúra

11	Betegség	Covid, koronavírus, tünetek, vírus, melanoma, leukémia, cukorbetegség, allergia, influenza
12	Természet	Vulkán, óceán, természeti katasztrófák
13	Oktatás	Felsőoktatás, közoktatás, egyetem, iskola, érettségi, tudományegyetem
14	Világ	Európa, Amerika, Afrika, Törökország, Britannia, Hollywood, Izrael, Ukrajna, Oroszország
15	Sport	Foci, kézilabda, kosárlabda, jégkorong, válogatott, tenisz, világbajnokság, olimpia, stadion
16	Járművek, közlekedés	Autó, motor, bmw, mercedes, légi közlekedés
17	Társadalom	Lakás, élelmiszer, rasszizmus, transzneműség, homoszexualitás, migráció, munka, munkanélküliség, áldozat
18	Pszichológia	Párkapcsolat, depresszió, terápia, molesztálás, stressz, szakítás, álomfejtés
19	Szórakozás	Póker, szerencsejáték, gaming

### 3.2 Osztályozás

A hírek osztályozásához három különböző módszert alkalmaztunk egy véletlenszerűen összeállított 500 hírt tartalmazó mintán. Az automatikus neurális címkeajánló modell érte el a legnagyobb pontosságot: a hírek 79,7%-át osztályozta helyesen.

A PULI-Llumix 32k Instruct modell szintén figyelemre méltó eredményeket produkált zero-shot tanítással, hiszen a hírek 70,1%-át osztályozta helyesen, míg az XLM-RoBERTa mindössze 18,4%-os pontosságot ért el. Az alacsony pontosság háttérben áll, hogy nem találtunk magyarul elérhető adathalmazt a modell finomhangolásához.

Ugyanakkor az automatikus címkeajánló rendszer hátránya, hogy a mindössze 500 hírhez 215 különböző címkét használt fel, ami önmagában nem teszi lehetővé a hírek tartalmára vonatkozó széleskörű általánosítást. A címkék vizsgálata során azt találtuk, hogy számos címke redundáns vagy túlságosan specifikus volt ahhoz, hogy a helyes kategóriába sorolható legyen. A rendszer nagy mennyiségű, nem egységesített címkehalmazon került tanításra, ami magyarázhatja felhasznált címkék számosságát.

## 4 Következtetés

A magyar hírek automatikus kategorizációja a tartalmak szervezését és könnyebb hozzáférését segíti, hozzájárulva az információs ökoszisztéma jól-szervezettségéhez. Jelen tanulmány során 77 magyar nyelvű hírportál 2013 és 2023 között keletkezett híreit vizsgáltuk. A későbbi osztályozás alapjául szolgáló kategóriarendszer kialakításához egyfelől statikus szóbeágyazásokból épített gráfstruktúra részgráfjait elemeztük,

másfelől témamodellezést alkalmaztunk két különböző BERT alapú modellel. Ez a munkaszakasz a keletkezett csoportok manuális validálásával zárult, melynek során 19 kategória került kialakításra.

A második munkaszakaszban három neurális módszert vizsgáltunk abból a szempontból, hogy milyen pontossággal képesek a hírek osztályozására: egy már létező neurális címkeajánló modellt, egy BERT alapú többnyelvű modellt, valamint egy generatív utasításkövető modellt. A modellek által kimenetként adott témák kiértékeléséhez emberi munka volt szükséges. Az osztályozásra használt modellek közül a feladat-specifikus címkeajánló rendszer adta a legjobb eredményt.

Kutatásunk jelentősége, hogy egy nagy fedésű, átlátható és jól definiált kategória rendszer kialakításával hatékony módszert kínál a hírek automatikus osztályozására.

Ugyanakkor láttuk, hogy egy jelenleg is létező, elsősorban más célokat szolgáló címkéző egyelőre pontosabb eredményeket ér el a hírek osztályozásában, mint az általunk megvizsgált módszerek. Így a jelen vizsgálat egy kezdeti lépésként értékelendő, amelyet a jövőben tovább kívánunk fejleszteni. A kódbázis és osztályozásra szolgáló annotált korpuszt ennek részeként tervezzük nyilvánosságra hozni. Elsődleges célunk a jövőben annak a vizsgálata, hogy a state-of-the-art nyelvmodellek milyen eredményeket érnek el az osztályozási feladaton valamint, hogy a különböző tanítási scenáriók és a modellek különböző beállításai milyen hatással vannak az eredményekre. További irány a klaszterelemzés által adott eredmények vizuális értelmezése, címkék eloszlásának értelmezése és a különböző modelteljesítményt mérő metrikák felhasználása. Összességében kiderült, hogy a kialakított címkékészlet és osztályozás alkalmasnak tűnik arra, hogy egy nagyobb hírkorpusz híreit automatikusan címkézzük. Remélhetőleg a kutatásunk a közeljövőben hozzájárul majd az automatikus szövegelemzési technológiák fejlődéséhez, különös tekintettel a nagy szöveges adathalmazok kezeléséhez és értelmezéséhez.

## Köszönetnyilvánítás

A kutatást az MTA "Tudomány a Magyar Nyelvért Nemzeti Program" támogatta.

## Bibliográfia

- Barna I, Knap Á. Analysis of the Thematic Structure and Discursive Framing in Articles about Trianon and the Holocaust in the Online Hungarian Press Using LDA Topic Modelling. *Nationalities Papers*. 2023;51(3):603-621. doi:10.1017/nps.2021.67
- Bose, Bijoy. BBC News Classification. <https://kaggle.com/competitions/learn-ai-bbc>, 2019. Kaggle.
- Common Crawl: <https://commoncrawl.org/>
- Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. "Unsupervised Cross-Lingual Representation Learning at Scale." arXiv, April 8, 2020. <https://doi.org/10.48550/arXiv.1911.02116>.

- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." arXiv, May 24, 2019. <https://doi.org/10.48550/arXiv.1810.04805>.
- Gelei, Andrea & Fodor, Szabina & Ternai, Katalin, 2023. "Az ipar 4.0 felkészültség értékelési rendszere a témamodellzés segítségével - középpontban a kis- és középvállalatok Közgazdasági Szemle, Közgazdasági Szemle Alapítvány vol. 0(11), pages 1230-1260.
- Grootendorst, Maarten. "BERTopic: Neural Topic Modeling with a Class-Based TF-IDF Procedure." arXiv, March 11, 2022. <https://doi.org/10.48550/arXiv.2203.05794>.
- Héja, Enikő; Ligeti-Nagy, Noémi. Magyar melléknevek polisém jelentéseinek automatikus kinyerése gráfokkal. In: Berend, Gábor; Gosztolya, Gábor; Vincze, Veronika (szerk.) XIX. Magyar Számítógépes Nyelvészeti Konferencia, MSZNY-2023. Szeged, Magyarország : Szegedi Tudományegyetem TTIK, Informatikai Intézet (2023) 469 p. pp. 63-76. , 14 p.
- Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." arXiv, July 26, 2019. <https://doi.org/10.48550/arXiv.1907.11692>.
- Kharosekar Aditya <https://www.kaggle.com/datasets/adityakharosekar2/guardian-news-articles-2023>, Kaggle
- Nagy, Mihály. Testing an Inductive Mixed-method Computational Approach to News Frame Analysis: An analysis of Hungarian online reporting of the 2014 Russia-Ukraine conflict, IFLA International News Media Conference 2024
- Nemeskey, Dávid Márk. "Natural Language Processing Methods for Language Modeling." ELTE, 2020. [https://hlt.bme.hu/en/publ/nemeskey\\_2020](https://hlt.bme.hu/en/publ/nemeskey_2020).
- Vági, Renátó and Úveges, István and Megyeri, Andrea and Fülöp, Anna and Vadász, János Pál and Nagy, Dániel and Csányi, Gergely Márk (2023) Increasing access to legal information with unsupervised solutions. HUNGARIAN JOURNAL OF LEGAL STUDIES, 64 (3). pp. 456-471. ISSN 2498-5473
- Yang, Zijian Győző, Attila Novák, and László János Laki. "Automatic Tag Recommendation for News Articles." Eger, Hungary, 2020.
- Yang Zijian Győző, Dodé Réka, Ferenczi Gergő, and Váradi Tamás. "Jönnek a nagyok! BERTLarge, GPT-2 és GPT-3 nyelvmodellek magyar nyelvre." Szeged, 2023.
- Yang, Zijian Győző, Réka Dodé, Gergő Ferenczi, Péter Hatvani, Enikő Héja, Gábor Madarász, Noémi Ligeti-Nagy, et al. "The First Instruct-Following Large Language Models for Hungarian," 247–52. Debrecen, Hungary: University of Debrecen, 2024.
- Zhen Wang, Xu Shan, Xiangxi Zhang, and Jie Yang. 2022. N24News: A New Dataset for Multimodal News Classification. In Proceedings of the Thirteenth Language Resources and Evaluation Conference, pages 6768–6775, Marseille, France. European Language Resources Association.





# Optimizing Abstractive Arabic Summarization via RLHF and DPO with Llama 2

Mram Kahla<sup>1</sup>, Zijian Győző Yang<sup>2</sup>

<sup>1</sup>Pázmány Péter Catholic University, Faculty of Information Technology and Bionics  
kahla.mram@itk.ppke.hu

<sup>3</sup>HUN-REN Hungarian Research Centre for Linguistics  
yang.zijian.gyozo@nytud.hun-ren.hu

**Abstract.** Given the advantages observed with Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO) in English, it is promising to explore their effectiveness for abstractive summarization in languages with complex morphological and syntactic features, such as Arabic. In this study, we fine-tune the Llama 2 model, which demonstrates a significant capability to enhance summarization results. We highlight how Llama 2, combined with advanced techniques like RLHF and DPO, markedly improves the quality of Abstractive Arabic summarization, showcasing the model’s superior performance in this challenging task. Furthermore, the AraSum corpus plays a critical role in achieving outstanding results, highlighting its effectiveness in improving the performance of summarization models. While this work focuses on Arabic, the techniques and insights presented are language-agnostic, offering broader applications for abstractive summarization in other languages. Additionally, we introduce the AraRLHF and AraDPO datasets, which will be made publicly available to support reproducibility and advance research in Arabic NLP.

**Keywords:** abstractive summarization, Reinforcement Learning, Arabic, RLHF, DPO, Direct Preference Optimization, Llama 2

## 1 Introduction

In Natural Language Processing (NLP), automatic text summarization stands as a pivotal task, catering to the ever-increasing volume of information available in today’s digital age.

Unlike extractive summarization (Zhang et al., 2018) which selects and rephrases existing segments from the original text, abstractive summarization (See et al., 2017) involves generating novel sentences that capture the essence of the source material. This process demands a deep understanding of semantics, contextual nuances, and linguistic structures to produce coherent and concise summaries.

Specifically in the context of the Arabic language, abstractive summarization holds significant promise and challenges due to the language’s intricate syntax, rich morphology, and diverse linguistic features.

In recent years, reinforcement learning (RL) has emerged as a promising paradigm for enhancing sequence generation tasks in NLP, such as abstractive summarization and question-answering. RL enables models to align outputs with human preferences (Ziegler et al., 2019) and leverage human feedback to improve factual accuracy and user alignment (Nakano et al., 2021). With its ability to learn optimal decision-making policies through interaction with an environment, RL offers an effective approach to refining abstractive summarization models, particularly when the goal is to align generated summaries with human preferences.

The outcomes obtained in the English language summarization through Reinforcement Learning from Human Feedback (RLHF) demonstrate significant improvements in the quality of the generated text (Stiennon et al., 2020) offering a clear advantage over larger supervised models that rely solely on traditional training methods.

While RLHF has proven effective in adjusting model outputs to better reflect human preferences, it is not without its limitations. A major limitation of RLHF is that its process is considerably more complex than traditional supervised learning. To address this complexity, methods like Direct Preference Optimization (DPO) (Rafailov et al., 2023) have been introduced as simpler training paradigms. DPO enables language models to be trained from human preferences without the added complexity of reinforcement learning while performing as well as or even better than existing RLHF algorithms.

The objective of this research is to explore the application of Reinforcement Learning from Human Feedback and Direct Preference Optimization to the task of abstractive Arabic text summarization. Our main contribution lies in applying Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO) to the task of abstractive text summarization for the Arabic language. We demonstrate how LLaMA 2, when combined with these advanced techniques and the AraSum corpus, significantly enhances the quality of Arabic text summarization. To foster reproducibility and encourage further research in Arabic NLP, we will release the AraRLHF and AraDPO datasets, which consist of human preference data specifically tailored for RLHF and DPO models. The datasets will be available on our GitHub<sup>1</sup>.

The rest of the paper is structured as follows: Section 2 reviews related work. Section 3 outlines the methodology. Section 4 discusses the corpora used, and Section 5 details the models used. Section 6 presents our experiments and results, and finally, Section 7 concludes the paper.

## 2 Related work

Reinforcement learning from human feedback (RLHF), originally developed for training simple robots in simulated environments and Atari games (Christiano et al., 2017; Ibarz et al., 2018).

---

<sup>1</sup> <https://github.com/ppke-nlpg/AraSum>

In terms of reinforcement learning with human feedback to train text summarization models, Böhm et al. (2019) learn a reward function from 2,500 human judgments of CNN/DM (Nallapati et al., 2016) summaries that are used in a reinforcement learning setting.

A similar method of recursive task decomposition was used for summarizing books in (Wu et al., 2021). They combine learning from human feedback with recursive task decomposition by using models trained on smaller parts of the task to assist humans in giving feedback on the broader task.

Ziegler et al. (2019) fine-tune pre-trained language models with reinforcement learning by exploiting a reward model trained from human preferences. Then the model is used to generate summaries over Reddit TL;DR, and CNN/DM datasets. The limitation of their framework is that their labelers prefer extractive summaries and there are low agreement rates between labelers and researchers.

Stiennon et al. (2020) followed their previous work on learning from human feedback and proposed to gather a dataset composed of human preferences between pairs of summaries as the first step. Then the prediction of the human-preferred summary is generated by a reward model (RM) trained via supervised learning. Lastly, the score produced by the RM is maximized as much as possible by a policy trained via reinforcement learning. This approach significantly outperforms both human reference summaries and much larger models fine-tuned with supervised learning alone.

Although RLHF has proven effective in aligning model outputs with human preferences, it has certain limitations, such as the high cost and complexity of training reward models, and the potential for misalignment between the reward model and human preferences (Casper et al., 2023). To address the complexity of RLHF optimization, Rafailov et al. (2023) introduced Direct Preference Optimization (DPO) as an alternative approach. Unlike RLHF, DPO eliminates the need for training a reward model and instead directly trains the language model based on human preferences using a simple binary cross-entropy objective.

Human feedback has been utilized to improve various AI systems across different tasks. For instance, in dialogue systems, Jaques et al. (2019) employed crowd-sourced human labeling to judge whether dialogue generated by an offline RL agent was fluent and amicable. Similarly, in the translation task, Kreutzer et al. (2018) collected both explicit and implicit human feedback to improve a machine translation model by integrating the feedback into a reinforcement learning framework. In review generation, Cho et al. (2018) developed models of coherence from existing texts and used these models as RL rewards to enhance long-form generation. For question-answering, Nakano et al. (2021) fine-tuned GPT-3 to answer long-form questions within a web-browsing environment. This setup enabled the model to navigate the web and incorporate human feedback to optimize answer quality through imitation learning. Additionally, human feedback has been applied to other tasks, such as evidence extraction (Perez et al., 2019), story generation (Zhou and Xu, 2020), and semantic parsing (Lawrence and Riezler, 2018).

The successful integration of RLHF into language technology was notably advanced by the development of ChatGPT (Ouyang et al., 2024). This research achieved significant improvements in the model’s ability to generate responses that align more closely with human-like communication. The approach began with a supervised fine-tuning phase, where the large language model was trained on prompts containing specific instructions. This was followed by an additional fine-tuning phase using reinforcement learning, further enhancing the model’s response quality and alignment with human preferences.

Regarding abstractive summarization in the Arabic language, one study by Azmi and Altmami (2018) introduced a four-stage abstractive summarization framework where the core of the system is an extractive summarizer. Training a model specifically for headline generation was presented in (Al-Maleh and Desouki, 2020). Another research by Elmadani et al. (2020) utilized the PreSumm approach along with a multilingual BERT model for fine-tuning both extractive and abstractive models. AraBART Introduced by Kamal Eddine et al. (2022), a pre-trained encoder-decoder model designed for abstractive summarization tasks tailored to the Arabic language. Furthermore, an analysis by Chouikhi and Al-suhaibani (2022) conducted a comparison analysis of various Arabic language models’ performance in the task of text summarization.

There are two additional experiments conducted as part of the abstractive Arabic summarization task. In the first experiment, Kahla et al. (2021) created the first monolingual, human-written corpus for abstractive Arabic text summarization and used it to fine-tune several language models: m-BERT, AraBERT, and m-BART-50. To enhance the performance of the baseline systems, a cross-lingual knowledge transfer method was applied. In the second experiment, Kahla et al. (2022) extended the Arabic summarization corpus, AraSum<sup>2</sup>, and made it publicly available. This expanded corpus contains approximately 50,000 Arabic articles with their corresponding leads. The experiment involved pre-training monolingual and trilingual BART models for Arabic, as well as fine-tuning these models and the mT5 model for abstractive summarization using the AraSum corpus. Results showed that the models trained on AraSum performed well, surpassing the state-of-the-art XL-Sum (Hasan et al., 2021) model at the time of publication.

In terms of Reinforcement Learning from Human Feedback and Direct Preference Optimization for the Arabic language, there is a noticeable scarcity of existing research. Leveraging RLHF and DPO presents a powerful technique that deserves application within such complex linguistic contexts.

### 3 Methodology

This research explores the application of RLHF and DPO to the task of abstractive Arabic text summarization.

---

<sup>2</sup> <https://github.com/ppke-nlpg/AraSum>

### 3.1 Reinforcement Learning experiments

The RLHF approach we adopt is based on OpenAI’s methodology (Stiennon et al., 2020), consisting of three main steps:

- **Step 1: Collect demonstration data, train a supervised policy, and send comparisons to humans.**

Humans are provided with reference texts and summaries generated by fine-tuning a language model. They are then asked to choose the best summary from the given samples.

- **Step 2: Collect comparison data, and train a reward model (RM).** A reward model is trained using the human feedback collected in the first step. Based on the annotations provided by the human evaluators, this model predicts the likelihood (log odds) that a given summary is preferred.
- **Step 3: Optimize a policy against the reward model using Proximal Policy Optimization (PPO).**

The output of the reward model serves as a reward measure. The supervised policy will be fine-tuned to maximize this reward using reinforcement learning, with the Proximal Policy Optimization (PPO) algorithm guiding the optimization process.

### 3.2 Direct Preference Optimization experiments

For the DPO approach, we adopt the method proposed by Rafailov et al. (2023), which simplifies the RLHF process by eliminating the need to fit a reward model. Instead, DPO directly trains language models based on human preferences. The DPO approach consists of the following steps:

- **Step 1: Collect preference data from human evaluators.**

Human evaluators are provided with multiple summaries for a given input and asked to select the one they prefer.

- **Step 2: Apply Direct Preference Optimization.**

DPO bypasses the need for a reward model and directly utilizes the human preference data to train the language model. The model is optimized by applying a binary cross-entropy objective, where it learns to assign higher probabilities to the summaries preferred by the human evaluators.

- **Step 3: Fine-tune the language model based on preferences.**

The language model is fine-tuned to generate summaries that better align with human preferences, achieving this without the need for reinforcement learning algorithms.

## 4 Corpora used

For our experiments, two datasets are required: The first dataset is used in RLHF to train the reward model to assess summary quality, while in DPO, it directly guides the optimization of the language model based on human preferences. The second dataset is used in the final step of both methodologies, which involves fine-tuning the models based on the collected preferences.

#### 4.1 Human Preference Dataset

The first dataset, named AraRLHF and AraDPO, is utilized in the initial step of both RLHF and DPO, focusing on collecting preference data from human evaluators. This dataset is then employed in Step 2 of each methodology. In RLHF, the AraRLHF dataset is used to train the reward model (RM), which predicts the quality of generated summaries based on the collected human preferences. In DPO, the AraDPO dataset is used directly to train the language model based on these preferences, without the need for a reward model.

To create this dataset, we utilized manual evaluation results from our previous research (Kahla et al., 2021), where we fine-tuned transformer models for abstractive Arabic text summarization using the first version of AraSum. This corpus includes 21,508 articles and their corresponding leads. The transformer models evaluated were as follows:

- m-BERT model (Devlin et al., 2019): fine-tuned for Arabic.
- AraBERT model (Antoun et al., 2020): fine-tuned for Arabic.
- m-BART-50 model (Tang et al., 2020): fine-tuned for Arabic.
- m-BERT+HUN model (Yang et al., 2021): originally fine-tuned for Hungarian and then fine-tuned for Arabic.
- m-BERT+ENG model: first fine-tuned for English and then fine-tuned for Arabic.
- m-BART-50+RUS model: first fine-tuned for Russian then fine-tuned for Arabic.

The evaluation involved three human evaluators who evaluated the outputs of these six models, indicating their preferred summaries for a given input by assigning scores to each summary from 100 random samples, see figure 1.

The human evaluation data underwent preprocessing and was restructured to be suitable for training the reward model in RLHF and for direct use in DPO.

The AraRLHF dataset consists of 1,746 samples, randomly shuffled and divided into 80% for training and 20% for testing. Similarly, the AraDPO dataset contains 29,682 samples, also shuffled and split into 80% for training and 20% for testing.

The AraRLHF and AraDPO datasets will be made publicly available upon publication of this paper to support reproducibility and encourage further research in Arabic NLP. The datasets will be accessible at the following link: <https://github.com/ppke-nlpg/AraSum>

#### 4.2 Dataset for Fine-tuning Llama 2

For fine-tuning the Llama 2, we used the extended version of the AraSum corpus (Kahla et al., 2022), which contains 49,604 articles along with their corresponding leads. In addition, we used the Arabic portion of the multilingual XL-Sum corpus (Hasan et al., 2021), which consists of 46,897 articles and their corresponding leads. Both datasets are designed for abstractive text summarization.

For instruction fine-tuning, we used the prompt template recommended by the Stanford Alpaca research (Taori et al., 2023):

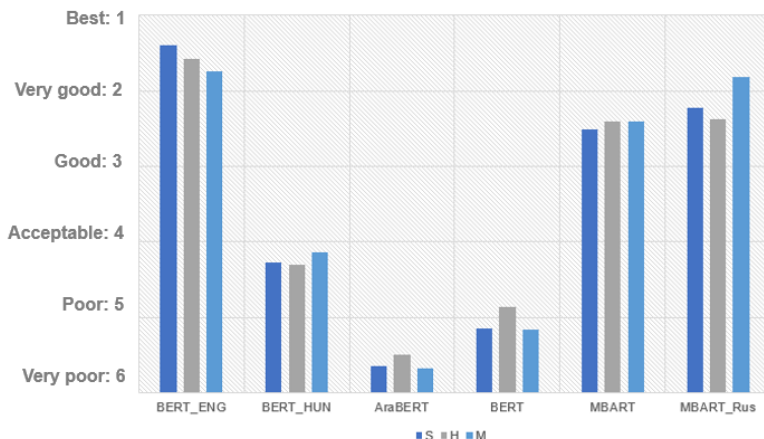


Fig. 1: Human evaluation results from our previous study (Kahla et al., 2021), where H, S, and M represent the human evaluators.

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:

Summarize the article written in Arabic below.

### Input:

[article text]

### Response:

[article summary]

Because the Llama 2 model is English-centric, we used an English template.

## 5 Experiments and Results

In our first experiment, we fine-tuned state-of-the-art Arabic summarization models with RLHF, specifically the mT5++ models from our previous research (Kahla et al., 2022).

In the next experiment, we performed supervised fine-tuning (SFT) on the Llama 2 model for Arabic summarization. Following that, we applied RLHF and DPO fine-tuning to the SFT model.

Llama 2 (Touvron et al., 2023b) is an advanced large language model developed by Meta, marking the second iteration of the LLaMA series (Touvron et al., 2023a). It represents a significant advancement in natural language processing. Llama 2 is available in various sizes: a 7-billion-parameter model, a 13-billion-parameter model, and a 70-billion-parameter model. For our experiments, we

fine-tuned the smallest model with 7 billion parameters. For supervised fine-tuning, we used the Stanford Alpaca implementation (Taori et al., 2023). The training hyperparameters are as follows: learning rate =  $2e-5$ ; global batch size = 256; epoch = 3; warmup ratio = 0.03; sequence length = 1800; bf16; deep-speed. For this task, we utilized eight NVIDIA A100 GPUs, each with 80GB of memory.

For RLHF experiments, we applied the Transformer Reinforcement Learning X implementation from CarperAI (Havrilla et al., 2023). The training hyperparameters are as follows: learning rate =  $1e-5$ ; global batch size = 4; epoch = 3; sequence length = 1800; number layers unfrozen = 2.

For the reward model, we fine-tuned an XLM-RoBERTa-large (Conneau et al., 2020) model for the Arabic summarization quality prediction model. For this task, we use the Hugging Face implementation<sup>3</sup>. The training hyperparameters are as follows: learning rate =  $2e-5$ ; global batch size = 32; epoch = 10; sequence length = 1024. We also conducted experiments with the mT5 base and large models (Xue et al., 2021), but they only achieved a Pearson correlation of 10–20.

In Table 1, we can see the results of the reward model experiments. The evaluation metrics are the same as those used in the research by Yang and Laki (2023): Pearson Correlation, Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). We achieved the highest Pearson correlation of **88** with 5 epochs. We used this checkpoint in subsequent experiments.

	Pearson correlation $\uparrow$	MAE $\downarrow$	RMSE $\downarrow$
XLM-RoBERTa-base	81.25	0.83	1.04
<b>XLM-RoBERTa-large</b>	<b>88.00</b>	<b>0.69</b>	<b>0.86</b>
mT5-base	10.73	1.53	1.75
mT5-large	21.49	1.77	1.97

Table 1: Reward model experiments

For the DPO experiments, we utilized the Hugging Face implementation<sup>4</sup>, which is based on the original DPO research (Rafailov et al., 2023). The training hyperparameters are as follows: learning rate =  $5e-4$ ; global batch size = 16; epoch = 3; sequence length = 1800.

In both the RLHF and DPO experiments, we tested different hyperparameters, with the best ones described above. For these tasks, we utilized a single NVIDIA A100 GPU with 80GB of memory.

The models that were experimented with and evaluated are as follows:

<sup>3</sup> <https://github.com/huggingface/transformers/tree/main/examples/pytorch>

<sup>4</sup> [https://huggingface.co/docs/trl/dpo\\_trainer](https://huggingface.co/docs/trl/dpo_trainer)



- **mT5++**: The state-of-the-art mT5-small model from the study of Kahla et al. (2022), fine-tuned using the AraSum corpus, and using the XL-Sum Arabic corpus.
- **'mT5++' + RLHF**: The fine-tuned mT5++ model is further fine-tuned with the RLHF approach, where a reward model is trained from human feedback, followed by Proximal Policy Optimization (PPO) for policy refinement on the AraSum corpus, and the XL-Sum corpus.
- **'mT5++' + DPO**: The fine-tuned mT5++ model is further fine-tuned with the Direct Preference Optimization (DPO) approach using the human-evaluated dataset.
- **Llama 2**: The Llama 2 model with 7 billion parameters, supervised fine-tuned (SFT) using the AraSum corpus, and the XL-Sum Arabic corpus.
- **Llama 2 + RLHF**: The SFT Llama 2 model fine-tuned with the RLHF approach using the development set of AraSum and XL-Sum Arabic corpus, and the fine-tuned XLM-RoBERTa-large reward model.
- **Llama 2 + DPO**: The SFT Llama 2 model is fine-tuned with the Direct Preference Optimization approach using the human-evaluated dataset.

We evaluated the system output using the ROUGE-N and ROUGE-L metrics. ROUGE-1 and ROUGE-2 assess the overlap of word unigrams and bigrams, respectively, while ROUGE-L measures the overlap of the longest common subsequence between two texts. ROUGE-L sum extends this by applying the ROUGE-L calculation at the sentence level and then aggregating the results for the final score.

It should be noted that the specific ROUGE scores presented here were calculated using the latest version of the ROUGE (Lin, 2004) library that was implemented by Hugging Face<sup>5</sup>, with the following setting: `use_stemmer=True`. Using the latest version, we were unable to reproduce the original values published in Kahla et al. (2022) and Hasan et al. (2021). We also tried using the implementation of XL-Sum<sup>6</sup> and the original implementation by Google<sup>7</sup>, but neither worked. The main objective is to demonstrate the enhanced performance resulting from our experiments; therefore, we used the values from the latest version of the Hugging Face Evaluate library. For better readability, we used the ROUGE \* 100 values. For better transparency, the old and new ROUGE values for the mT5++ models are presented as follows (in the order: ROUGE-1/ROUGE-2/ROUGE-L):

- old values of mT5++ Arasum Test Set: 33.172/13.914/24.782
- new values of mt5++ Arasum Test Set: 4.560/0.344/4.509
- old values of mT5++ XL-Sum Test Set: 29.128/11.049/24.070
- new values of mt5++ XL-Sum Test Set: 1.489/0.043/1.483

As with other fine-tuning experiments, we needed to determine the optimal number of epochs. Figure 2 shows experiments conducted with different epoch

<sup>5</sup> <https://huggingface.co/docs/evaluate/index>

<sup>6</sup> [https://github.com/csebuatnlp/xl-sum/tree/master/multilingual\\_rouge\\_scoring](https://github.com/csebuatnlp/xl-sum/tree/master/multilingual_rouge_scoring)

<sup>7</sup> <https://github.com/google-research/google-research/tree/master/rouge>

counts. In both the RLHF and DPO experiments, optimal performance was observed across three epochs: 0.3, 0.6, and 1, with epoch 1 yielding the best results.

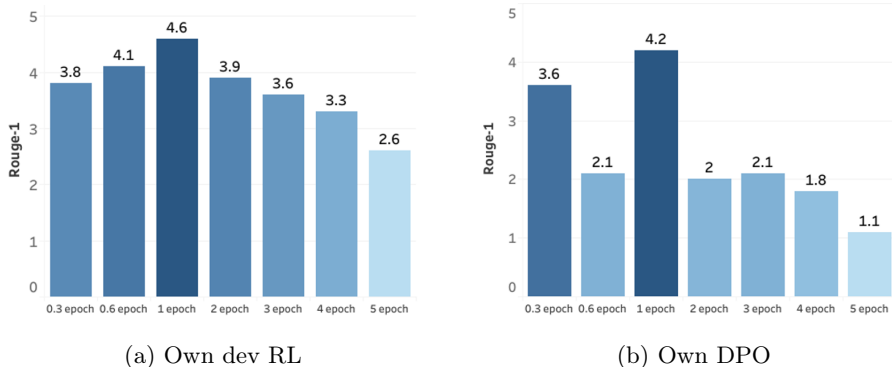


Fig. 2: Performance of ROUGE-1 across Epochs variations. In both datasets, we can see a performance improvement when the epoch is 1.

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-L sum
<b>AraSum Test Set</b>				
mT5++	4.560	0.344	4.509	4.537
'mT5++' + RLHF	3.464	0.245	3.435	3.444
'mT5++' + DPO	2.813	0.248	2.819	0.543
<b>Llama 2</b>	<b>4.636</b>	<b>0.414</b>	<b>4.618</b>	<b>4.616</b>
<b>Llama 2 + RLHF</b>	<b>4.947</b>	<b>0.486</b>	<b>4.957</b>	<b>4.949</b>
<b>Llama 2 + DPO</b>	<b>4.719</b>	<b>0.470</b>	<b>4.659</b>	<b>4.664</b>
<b>XL-Sum Arabic Test Set</b>				
mT5++	1.489	0.043	1.483	1.481
'mT5++' + RLHF	0.633	0.014	0.626	0.635
'mT5++' + DPO	0.534	0.029	0.540	0.543
<b>Llama 2</b>	<b>2.241</b>	<b>0.102</b>	<b>2.225</b>	<b>2.223</b>
<b>Llama 2 + RLHF</b>	<b>2.344</b>	<b>0.104</b>	<b>2.339</b>	<b>2.325</b>
<b>Llama 2 + DPO</b>	<b>2.447</b>	<b>0.112</b>	<b>2.440</b>	<b>2.431</b>

Table 2: ROUGE scores on the AraSum, and the XL-Sum Arabic test sets.

Table 2 presents the experimental results. The ROUGE scores reveal several significant insights across the models and fine-tuning approaches. Llama 2, with

its 7 billion parameters, significantly outperforms the mT5++ model across all metrics, demonstrating Llama 2's superior capabilities. Both RLHF and DPO contribute to improved performance, with Llama 2 + RLHF achieving the highest scores on the AraSum dataset, indicating a substantial boost in performance. In contrast, the 'mT5++' + RLHF model performs the worst across all metrics, suggesting that mT5 struggles to benefit from the RLHF approach. Additionally, the ROUGE scores for XL-Sum are significantly lower compared to AraSum across all models, highlighting the strength and quality of the dataset AraSum in achieving better summarization performance.

## 6 Conclusion

In this paper, we applied Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO) to the task of abstractive text summarization for the Arabic language. By fine-tuning the state-of-the-art LLaMA 2 model, we observed a remarkable enhancement in summarization quality, particularly when RLHF was used with the AraSum dataset. The performance improvements highlight the strength of LLaMA 2, especially when combined with RLHF on our dataset. Moreover, the AraSum corpus played a crucial role in achieving superior results, consistently surpassing models fine-tuned on the XL-Sum dataset. This study demonstrates that advanced techniques like RLHF and DPO, in combination with a robust dataset such as AraSum and a highly capable large language model such as LLaMA 2, can significantly elevate the quality of abstractive Arabic text summarization. While our focus was on Arabic, the techniques and insights presented in this study are inherently language-agnostic. They have the potential to be applied to other languages, particularly those with complex morphological and syntactic features, making this work relevant for broader multilingual NLP tasks. In addition, we are committed to publicly releasing the AraRLHF and AraDPO datasets to promote reproducibility and further advancements in Arabic NLP.

In the meantime, the Llama 3 models have been released. We aim to continue our experiments with these new models and anticipate achieving further advancements in performance through their utilization.

## References

- Al-Maleh, M., Desouki, S.: Arabic text summarization using deep learning approach. *Journal of Big Data* 7, 1–17 (2020)
- Antoun, W., Baly, F., Hajj, H.: AraBERT: Transformer-based model for Arabic language understanding. In: Al-Khalifa, H., Magdy, W., Darwish, K., Elsayed, T., Mubarak, H. (eds.) *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. pp. 9–15. European Language Resource Association, Marseille, France (May 2020), <https://aclanthology.org/2020.osact-1.2>

- Azmi, A.M., Altmami, N.I.: An abstractive Arabic text summarizer with user controlled granularity. *Information Processing and Management* 54(6), 903–921 (2018), <https://www.sciencedirect.com/science/article/pii/S030645731730417X>
- Böhm, F., Gao, Y., Meyer, C.M., Shapira, O., Dagan, I., Gurevych, I.: Better rewards yield better summaries: Learning to summarise without references. arXiv preprint arXiv:1909.01214 (2019)
- Casper, S., Davies, X., Shi, C., Gilbert, T.K., Scheurer, J., Rando, J., Freedman, R., Korbak, T., Lindner, D., Freire, P., et al.: Open problems and fundamental limitations of reinforcement learning from human feedback. arXiv preprint arXiv:2307.15217 (2023)
- Cho, W.S., Zhang, P., Zhang, Y., Li, X., Galley, M., Brockett, C., Wang, M., Gao, J.: Towards coherent and cohesive long-form text generation. arXiv preprint arXiv:1811.00511 (2018)
- Chouikhi, H., Alsuhaibani, M.: Deep transformer language models for Arabic text summarization: A comparison study. *Applied Sciences* 12(23) (2022), <https://www.mdpi.com/2076-3417/12/23/11944>
- Christiano, P.F., Leike, J., Brown, T., Martic, M., Legg, S., Amodei, D.: Deep reinforcement learning from human preferences. *Advances in neural information processing systems* 30 (2017)
- Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V.: Unsupervised cross-lingual representation learning at scale. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. pp. 8440–8451. Association for Computational Linguistics, Online (Jul 2020), <https://aclanthology.org/2020.acl-main.747>
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), <https://aclanthology.org/N19-1423>
- Elmadani, K.N., Elgezouli, M., Showk, A.: Bert fine-tuning for Arabic text summarization. ArXiv abs/2004.14135 (2020)
- Hasan, T., Bhattacharjee, A., Islam, M.S., Mubasshir, K., Li, Y.F., Kang, Y.B., Rahman, M.S., Shahriyar, R.: XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. pp. 4693–4703. Association for Computational Linguistics, Online (Aug 2021), <https://aclanthology.org/2021.findings-acl.413>
- Havrilla, A., Zhuravinskyi, M., Phung, D., Tiwari, A., Tow, J., Biderman, S., Anthony, Q., Castriato, L.: trlX: A framework for large scale reinforcement learning from human feedback. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. pp. 8578–8595.

- Association for Computational Linguistics, Singapore (Dec 2023), <https://aclanthology.org/2023.emnlp-main.530>
- Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., Amodei, D.: Reward learning from human preferences and demonstrations in Atari. *Advances in neural information processing systems* 31 (2018)
- Jaques, N., Ghandeharioun, A., Shen, J.H., Ferguson, C., Lapedriza, À., Jones, N., Gu, S., Picard, R.W.: Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *CoRR abs/1907.00456* (2019), <http://arxiv.org/abs/1907.00456>
- Kahla, M., Novák, A., Yang, Z.G.: Fine-tuning and multilingual pre-training for abstractive summarization task for the Arabic language. *Annales Mathematicae et Informaticae* (2022), <https://ami.uni-eszterhazy.hu>
- Kahla, M., Yang, Z.G., Novák, A.: Cross-lingual fine-tuning for abstractive Arabic text summarization. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. pp. 655–663. INCOMA Ltd., Held Online (Sep 2021), <https://aclanthology.org/2021.ranlp-main.74>
- Kamal Eddine, M., Tomeh, N., Habash, N., Le Roux, J., Vazirgiannis, M.: AraBART: a pretrained Arabic sequence-to-sequence model for abstractive summarization. In: Bouamor, H., Al-Khalifa, H., Darwish, K., Rambow, O., Bougares, F., Abdelali, A., Tomeh, N., Khalifa, S., Zaghouani, W. (eds.) *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*. pp. 31–42. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Hybrid) (Dec 2022), <https://aclanthology.org/2022.wanlp-1.4>
- Kreutzer, J., Khadivi, S., Matusov, E., Riezler, S.: Can neural machine translation be improved with user feedback? *arXiv preprint arXiv:1804.05958* (2018)
- Lawrence, C., Riezler, S.: Improving a neural semantic parser by counterfactual learning from human bandit feedback. *arXiv preprint arXiv:1805.01252* (2018)
- Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: *Text Summarization Branches Out*. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (Jul 2004), <https://www.aclweb.org/anthology/W04-1013>
- Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al.: Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021)
- Nallapati, R., Zhou, B., Ma, M.: Classify or select: Neural architectures for extractive document summarization. *arXiv preprint arXiv:1611.04244* (2016)
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.: Training language models to follow instructions with human feedback. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. pp. 27730–27744. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2024)

- Perez, E., Karamcheti, S., Fergus, R., Weston, J., Kiela, D., Cho, K.: Finding generalizable evidence by learning to convince q&a models. arXiv preprint arXiv:1909.05863 (2019)
- Rafailov, R., Sharma, A., Mitchell, E., Manning, C.D., Ermon, S., Finn, C.: Direct preference optimization: Your language model is secretly a reward model. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), <https://arxiv.org/abs/2305.18290>
- See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1073–1083. Association for Computational Linguistics, Vancouver, Canada (Jul 2017), <https://www.aclweb.org/anthology/P17-1099>
- Stiennon, N., Ouyang, L., Wu, J., Ziegler, D.M., Lowe, R., Voss, C., Radford, A., Amodei, D., Christiano, P.F.: Learning to summarize from human feedback. CoRR abs/2009.01325 (2020), <https://arxiv.org/abs/2009.01325>
- Tang, Y., Tran, C., Li, X., Chen, P.J., Goyal, N., Chaudhary, V., Gu, J., Fan, A.: Multilingual translation with extensible multilingual pretraining and fine-tuning (2020), <https://arxiv.org/abs/2008.00401>
- Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., Hashimoto, T.B.: Stanford Alpaca: An Instruction-following LLaMA model (2023), [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca)
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023a)
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T.: Llama 2: Open foundation and fine-tuned chat models (2023b), <https://arxiv.org/abs/2307.09288>
- Wu, J., Ouyang, L., Ziegler, D.M., Stiennon, N., Lowe, R., Leike, J., Christiano, P.F.: Recursively summarizing books with human feedback. CoRR abs/2109.10862 (2021), <https://arxiv.org/abs/2109.10862>
- Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., Raffel, C.: mT5: A massively multilingual pre-trained text-to-text transformer. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 483–498. Association for Computational Linguistics, Online (Jun 2021), <https://aclanthology.org/2021.naacl-main.41>

- Yang, Z.G., Agócs, Á., Kusper, G., Váradi, T.: Abstractive text summarization for Hungarian. *Annales Mathematicae et Informaticae* 53, 299–316 (2021)
- Yang, Z.G., Laki, L.J.: Enhancing machine translation with quality estimation and reinforcement learning. *Annales Mathematicae et Informaticae* 58, 180–190 (2023)
- Zhang, X., Lapata, M., Wei, F., Zhou, M.: Neural latent extractive document summarization. arXiv preprint arXiv:1808.07187 (2018)
- Zhou, W., Xu, K.: Learning to compare for better training and evaluation of open domain natural language generation models. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 34, pp. 9717–9724 (2020)
- Ziegler, D.M., Stiennon, N., Wu, J., Brown, T.B., Radford, A., Amodei, D., Christiano, P., Irving, G.: Fine-tuning language models from human preferences. arXiv preprint arXiv:1909.08593 (2019)





# BESZÉDTECHNOLÓGIA, KOMMUNIKÁ- CIÓ



# Különböző wav2vec 2.0 rétegekből nyert beágyazások használata sclerosis multiplex felismerésére

Gosztolya Gábor<sup>1,2</sup>, Tóth László<sup>2</sup>, Svindt Veronika<sup>3</sup>,  
Bóna Judit<sup>4</sup>, Hoffmann Ildikó<sup>3,5</sup>

<sup>1</sup> HUN-REN–SZTE Mesterséges Intelligencia Kutatócsoport, Szeged

<sup>2</sup> Szegedi Tudományegyetem, Informatikai Intézet

<sup>3</sup> HUN-REN Nyelvtudományi Kutatóközpont, Budapest

<sup>4</sup> ELTE Eötvös Loránd Tudományegyetem,

Alkalmazott Nyelvészeti és Fonetikai Tanszék, Budapest

<sup>5</sup> Szegedi Tudományegyetem, Pszichiátriai Klinika

ggabor @ inf.u-szeged.hu

**Kivonat** A sclerosis multiplex (SM) a központi idegrendszer krónikus gyulladással megbetegedése, mely többek között az alanyok beszédére is hatással van. Emiatt az automatikus beszédfeldolgozás relatíve egyszerű, olcsó és találokzásmentes (távoli) módot nyújthat arra, hogy követhessük a betegek beszédproduktójának változását. Egy ilyen rendszer tanítása során problémát jelent a rendelkezésre álló beszédanyag mennyiségének és (főleg) változatosságának korlátozottsága (pl. kevés beszélő), ami miatt általában nem praktikus egyetlen (end-to-end) mély hálót alkalmazni az SM detektálására. Ugyanakkor az járható megközelítés lehet, hogy osztályozásra valamilyen hagyományosabb módszert (pl. SVM-et vagy véletlen erdőt) alkalmazunk, mély hálók beágyazásait használva jellemzőként. Egy korszerű mély háló (pl. wav2vec 2.0) esetén kézenfekvő a konvolúciós és a finomhangolt blokkok utolsó rétegét választani, azonban a közbülső rétegek is hasznosnak bizonyulhatnak. Jelen tanulmányban azt vizsgáljuk meg, hogy a közbülső (belső) rétegekből vett beágyazások használatával javíthatunk-e az SM automatikus felismerésének pontosságán. Eredményeink alapján a 24 finomhangolt rejtett réteg mélyebben fekvő kb. egyharmada bizonyult a leghasznosabbnak, statisztikailag szignifikáns javulást is eredményezve a blokkok utolsó rejtett rétegéhez képest.

## 1. Bevezetés

A sclerosis multiplex (SM) a központi idegrendszer krónikus gyulladással megbetegedése, mely számos különféle kognitív és nyelvi tünettől jár együtt (Szirmai, 2006). A betegség előrehaladása jelentősen eltérhet az egyes alanyoknál, valamint időben is változhat. A betegség előrehaladtával romlás következhet be a beteg járásában, mozgáskoordinációjában, az alany fáradékonyabbá válhat, valamint

romolhatnak a kognitív és nyelvi funkciói is (akár dizartria is kialakulhat (Orsolya és mtsai, 2020)). Mindezek alapján az automatikus beszédelemzés potenciálisan alkalmazható a betegség automatikus, kontaktmentes és (viszonylag) olcsó előszűrésére, vagy a beteg állapotának romlásának követésére.

Az elmúlt bő egy évtizedben az automatikus beszédelemzés a beszédtechnológia egy önálló területévé fejlődött. Ide tartozik a *paralingvisztika* (*computational paralinguistics*), amelynek célja a beszélő érzelmi vagy fizikai állapotának megállapítása a beszédjelből (például érzelemfelismerés (Grósz és mtsai, 2023; Kondratenko és mtsai, 2023), a beszélő korának és nemének azonosítása (Kumar és mtsai, 2016), az alany pillanatnyi álmoságának mértékének megbecslése (Huckvale és mtsai, 2020), vagy dadogás detektálása (Grósz és mtsai, 2022)).

Az automatikus beszédelemzés másik nagy területe az *orvosi célú beszédfeldolgozás* (*pathological speech processing*). Itt a feladat annak megállapítása, hogy a beszélő szenved-e valamely konkrét betegségben (pl. Parkinson-kór (Jenei és mtsai, 2022; Klumpp és mtsai, 2022), Alzheimer-kór (Ivanova és mtsai, 2023; Pérez-Toro és mtsai, 2022), depresszió (Fara és mtsai, 2023; Kiss és mtsai, 2016)). Az utóbbi években a mély neurális hálók alkalmazása is rutinszerűvé vált a területen (Fara és mtsai, 2023; Jenei és mtsai, 2022; Pérez-Toro és mtsai, 2022).

Az önfelügyelt tanulás (*self-supervised learning*) megjelenésével napjaink talán legnépszerűbb beszédfeldolgozó architektúrájává a wav2vec 2.0 vált (Baeviski és mtsai, 2020). Természetesen bizonyos feladatokon közvetlenül is alkalmazhatóak ilyen mély hálók (pl. beszédfelismerésre (Mihajlik és mtsai, 2022) vagy érzelemfelismerésre (Chen és Rudnicky, 2023)), ugyanakkor korlátozottabb mennyiségű tanítóadattal sajnos nem triviális a betanításuk. Ilyen esetekben is lehetőségünk van azonban wav2vec 2.0 hálók alkalmazására, például valamely más feladaton (pl. beszédfelismerés vagy beszélőazonosítás) tanított hálók kiértékelésével és a kapott aktivációk (*beágyazások* (*embeddings*)) jellemzőkként használatával; ekkor a szigorúan vett osztályozási feladatra pl. SVM-et vagy véletlen erdőt alkalmazhatunk (Pepino és mtsai, 2021). Az orvosi beszédfeldolgozási területen extrém módon kevés a tanítóadat mennyisége, így a mély hálók az ilyen feladatokon általában ez utóbbi módon, jellemzőkinyerésre vannak felhasználva (Egas-López és mtsai, 2023; Pérez-Toro és mtsai, 2022; Thienpondt és mtsai, 2023; Cai és mtsai, 2025).

Egy neurális háló jellemzőkinyerésre használatához ki kell választanunk egy (vagy több) rejtett réteget, amelyből a beágyazásokat nyerjük. Egy wav2vec 2.0 architektúrájú háló két fő blokkból áll: egy alsó *konvolúciós* és egy felső *finomhangolt* (vagy *kontextualizált*) blokkból, és kézenfekvőnek tűnik a blokkok utolsó rejtett rétegének használata (Gosztolya és mtsai, 2023; Kodali és mtsai, 2023). Jelen dolgozatunkban ennél alaposabb vizsgálatot végzünk: azt vizsgáljuk, hogy kaphatunk-e hasznosabb jellemzőket valamely közbülső rejtett réteg használatával. Az osztályozási feladat magyar nyelvű sclerosis multiplex alanyok megkülönböztetése lesz (szintén magyar nyelvű) egészséges kontroll alanyoktól, a beágyazásokat pedig két, magyar nyelvre finomhangolt wav2vec 2.0 hálóból nyerjük: ebből az egyik beszédfelismerésre, a másik pedig beszélőfelismerésre lett finomhangolva.

## 2. Hangfelvételek

A vizsgálatokra a budapesti Uzsoki Utcai Kórház Neurológiai Osztályán és az (akkor még) Eötvös Loránd Kutatóhálózat Nyelvtudományi Kutatóközpontjában került sor. A vizsgálatot az Uzsoki Utcai Kórház etikai bizottsága hagyta jóvá, és a Helsinki Nyilatkozatnak megfelelően végeztük el. Kísérleteinket 23 SM alany (18 nő és 5 férfi) és 22 kontroll beszélő (16 nő és 6 férfi) felvételein végeztük. Az SM alanyok mindegyike a relapszáló-remittáló (*relapsing-remitting*, RRMS) altípusba tartozott. A két csoport statisztikailag illesztett életkor, iskolázottság és nem szerint (azaz  $p > 0,05$ ).

A felvételi protokoll során összesen 17 különféle feladatot rögzítettünk az alanyokkal. Jelen tanulmányunkban kísérleteinket a *szövegösszefoglalás* feladat hangfelvételein végeztük; ennek során az alanyoknak egy kétperces, számukra korábban ismeretlen tudományos ismeretterjesztő szöveg meghallgatása után minél pontosabban el kellett azt mesélniük. A feladat végrehajtása számos kognitív funkció (figyelemösszpontosítás, munkamemória, időbeli orientáció, rendszerezés) összehangolt működését igényli (Mar, 2004).

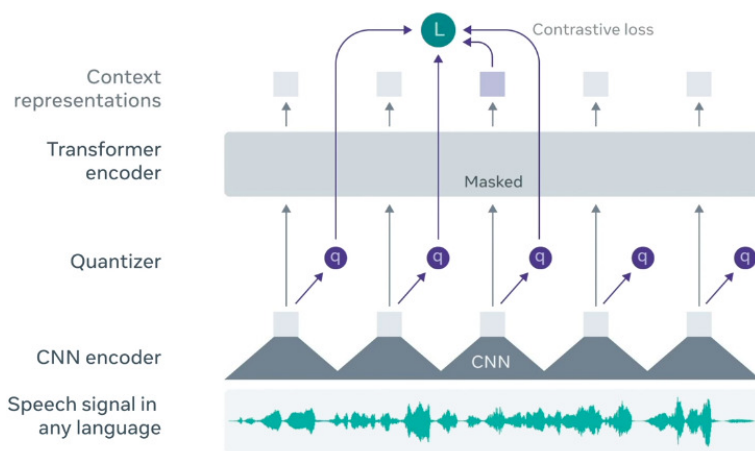
Az alanyok válaszait egy Sony PCM-A10 digitális diktafonnal, valamint csíptetős mikrofonnal rögzítettük. Az eredetileg sztereoó, 48 kHz mintavételű felvételeket az automatikus feldolgozás előtt 16 kHz mintavételezésű, monó formátumra konvertáltuk.

## 3. wav2vec 2.0

A **wav2vec** architektúra egy konvolúciós neurális háló, melynek bemenete a nyers beszédhang-jel, kimenete pedig egy olyan reprezentáció, mely alkalmas egy beszédfelismerő rendszer bemenetének. Tanítása önfelügyelt módon történik, az adott felvétel következő szegmensének predikciójára (Schneider és mtsai, 2019). A **wav2vec 2.0** architektúra ezen a megközelítésen úgy javított, hogy a tanítás során – a modell zajtűrésének javítása érdekében – maszkolást (a bemenet egyes részeinek nulla értékkel helyettesítését) is alkalmazott. További különbség a kontrasztív tanítás (contrastive learning) alkalmazása, melynek során a modell célja annak felismerése is, hogy két különbözőféleképpen transzformált reprezentáció (esetünkben a finomhangolt réteg kimenete, valamint a konvolúciós réteg kimenetének diszkretizált (kvantált) formája) azonos bemenethez tartozik-e (Baevski és mtsai, 2020). A wav2vec 2.0 architektúra az 1. ábrán látható.

### 3.1. Jellemzőkinyerés wav2vec 2.0 segítségével

Amennyiben egy wav2vec 2.0 hálót paralingvisztikai vagy orvosi célú beszédelemzési feladatban jellemzőkinyerésre szeretnénk használni, a legkézenfekvőbb választás vagy a konvolúciós, vagy a finomhangolt blokk utolsó rétegéből venni az aktivációkat (Fan és mtsai, 2021; Kodali és mtsai, 2023). Mivel ezek a beágyazások keretszintűek, a kinyert vektorok száma egyenesen arányos lesz a felvétel hosszával, így ebben a formában nem használhatóak jellemzőkként (legalábbis a



1. ábra: Egy finomhangolt wav2vec 2.0 háló struktúrája. Forrás: <https://ai.meta.com/blog/wav2vec-20-learning-the-structure-of-speech-from-raw-audio> .

számunkra most releváns feladatban). Hogy felvételszintű jellemzőket kapjunk belőlük, valamilyen módon aggregálni kell őket a teljes hangfelvételen, például venni az átlagukat és a szórásukat (Gosztolya és mtsai, 2022; Pérez-Toro és mtsai, 2022; Vaessen és Van Leeuwen, 2021).

Jelen tanulmányunkban a finomhangolt blokk *belső* rétegeire koncentrálnak. Egy standard XLSR-53 hálóban (lásd (Babu és mtsai, 2022)) 24 ilyen transzformer réteg van, mely kiegészülve a konvolúciós blokk utolsó rejtett rétegével, 25 lehetőséget jelent. Közismert, hogy egy mély háló mélyebben fekvő rétegei jellemzően alacsonyabb szintű információkat tárolnak (beszédfeldolgozás esetén pl. csend, különböző zajok vagy további akusztikai paraméterek (pl. visszhang mértéke)), míg a magasabban található rétegekben elsősorban magasabb szintű (esetünkben például, a jellemzőkinyerésre használt háló függvényében, fonetikai vagy az aktuális beszélőre jellemző) információk találhatóak. Sajnos ezen kívül további kapaszkodó nem áll rendelkezésünkre a megfelelő rejtett réteg kiválasztására, így mind a 25 variációt végig fogjuk próbálni.

## 4. Kísérletek

### 4.1. Jellemzőkinyerés

Két wav2vec 2.0 modellt használtunk a jellemzőkinyerési lépés során, melyek a Facebook által előtanított XLSR-53 modell finomhangolásával keletkeztek (Babu és mtsai, 2022). Ebben az architektúrában a konvolúciós réteg utolsó rejtett rétege 512 neuronból áll, míg az ezt követő kontextualizált blokk mint a 24 rejtett rétege 1024 neuront tartalmaz. A finomhangolt modellek közül az első a nyilvánosan elérhető wav2vec2-large-xlsr53-hungarian, melyet jonatasgrosman

finomhangolt a Mozilla Common Voice 6.1 adatbázis magyar részhalmazán (8 órányi adaton) (Grosman, 2021). A továbbiakban erre a modellre mint *leírató* modellre fogunk hivatkozni.

A másik főhasznált modellünk saját tanítású volt, ugyanúgy a XLSR-53 modellből kiindulva. Ezt a (magyar nyelvű) BEA Spontánbeszéd-adatbázis egy részhalmazán finomhangoltuk (Neuberger és mtsai, 2014). 165 beszélőt választottunk ki; a felvételekből automatikusan kivágtuk azokat a részeket, melyekben a felvételvezető hangja is hallható, így összesen körülbelül 60 órányi beszédanyagot használtunk. Az eredeti sztereó, 44,1 kHz-en mintavételezett bemondásokat monó, 16 kHz-es formátumra konvertáltuk. Az utolsó rejtett réteg aktivációit kiátlagoltuk (*average pooling*), az ezt követő kimeneti rétegben pedig szoftmax aktivációt használtunk 165 neuronnal. A betanult modell beszélőosztályozási hibája a teszhalmazon (de értelemszerűen ugyanezen beszélőkön) 1,92% volt. A továbbiakban erre a modellre mint *beszélő-azonosító* modellre fogunk hivatkozni.

A kinyert beágyazások mindkét háló és minden réteg esetén keretszintűek voltak, melyeket az időtengely mentén vett átlaggal és szórással konvertálunk felvételszintűvé. Mivel jelen munkánkban elsősorban a finomhangolt blokkból nyert beágyazásokra fókuszáltunk, a legtöbb teszt során 2048 felvételszintű jellemzőt használtunk; ez alól a konvolúciós blokk utolsó rétegét vizsgáló tesztejünk voltak kivételek, ahol 1024 jellemzőnk volt.

Természetesen a két wav2vec 2.0 hálóval kapott eredményeket nem lehet közvetlenül összehasonlítani, hiszen azok tanítása más-más adaton (és vélhetőleg más hiperparaméterekkel) történt. Véleményünk szerint azonban alapvetően így is alkalmasak arra, hogy demonstrálják a különböző célokra tanított hálók eredményességét, amikor jellemzőkinyerésre használjuk azokat.

## 4.2. Osztályozás

Osztályozási kísérleteink egy hagyományos sémát követtek: mivel az orvosi beszédfeldolgozási területen egy beszélő felel meg egy gépi tanulási példának, adathalmazunk gépi tanulási szempontból nagyon kisméretűnek számít. Ezért nem definiáltunk külön tanító-, fejlesztési- és teszhalmazt, hanem keresztvalidációt alkalmaztunk. Minden csoportba (*fold*ba) egy SM és egy kontroll alany felvételei kerültek, így összesen 23 csoportot kaptunk. Az osztályozás minőségét a ROC görbe alatti terület (AUC) metrikával mértük, melynek használata szintén igen elterjedt a területen (Carvajal-Castaño és mtsai, 2022; Gosztolya és mtsai, 2022).

Osztályozó eljárásnak SVM-et használtunk (LibSVM implementáció (Chang és Lin, 2011)). A nu-SVM változatot használtuk lineáris kernellel; a  $C$  hiperparaméter értékét  $10^{\{-5, \dots, 1\}}$  között változtattuk. Beágyazott keresztvalidációt alkalmaztunk (Cawley és Talbot, 2010): a  $C$  hiperparamétert minden tanítás esetén egy további (belső, 22-szeres) keresztvalidációs lépés segítségével választottuk ki, a legjobb ROC görbe alatti terület (AUC) érték alapján.

Az AUC értékek robusztusságának ellenőrzése érdekében minden osztályozási kísérletet ötször végeztünk el, melyek a 23 beszélőcsoport (véletlenszerű) kialakításában tértek el egymástól. Az ábrákon és a táblázatokban az öt így kapott AUC érték átlagát adjuk meg, míg a modellek robusztusságának vizsgálatához

wav2vec 2.0 modell	Beágyazás	AUC		
		Átlag	Szórás	Terjedelem
Leiratózó	Utolsó konvolúciós	0,724	0,008	[ 0,712; 0,733 ]
	Utolsó finomhangolt	0,806	0,014	[ 0,787; 0,824 ]
Beszélő-azonosító	Utolsó konvolúciós	0,716	0,020	[ 0,690; 0,741 ]
	Utolsó finomhangolt	0,402	0,059	[ 0,324; 0,461 ]

1. táblázat. A mért átlagos AUC értékek, azok szórása és terjedelme ([min; max]) a konvolúciós és a finomhangolt blokkok utolsó rétegeiből nyert beágyazások használatával.

az AUC értékek szórását, valamint az értékek terjedelmét ([min; max]) tüntetjük föl.

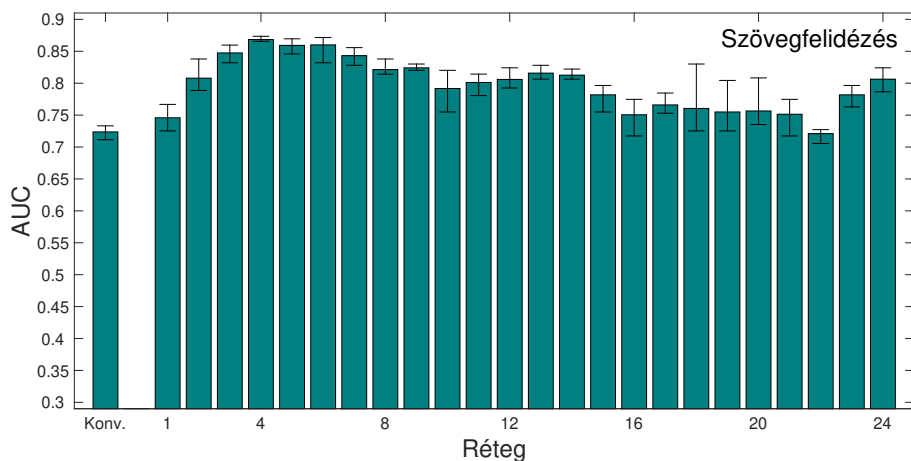
## 5. Az utolsó rétegekkel kapott eredmények

Az 1. táblázat mutatja a konvolúciós és finomhangolt blokkok utolsó rejtett rétegeinek használatával kapott eredményeket. A **leiratózó** modellel kapott eredmények elfogadhatónak tekinthetők, és igen hasonlóak a korábban a szakirodalomban megjelent AUC értékekhez (Gonzalez-Machorro és mtsai, 2023; Gosztolya és mtsai, 2023). A szórás is elég alacsony mindkét típusú beágyazás esetén, amely elég robosztus felismerést jelez. A **beszélő-azonosító** háló esetén a konvolúciós beágyazásokkal nagyon hasonló értékeket kaptunk, mint a leiratózó modellel: a 0,716-os átlagos AUC érték vállalhatónak tekinthető, bár a modellek AUC értékeinek szórása és terjedelme valamivel nagyobb. A finomhangolt blokk utolsó rejtett rétege esetén azonban az osztályozási teljesítmény igen rossznak bizonyult: a 0,402-es átlagos AUC érték még a véletlen találgatást is alulmúlja. Ez valószínűleg betudható annak, hogy a két háló eltérő feladatra lett tanítva, bár, mivel a tanítás számos paramétere (adatbázis, keretrendszer, hiperparaméterek) biztosan vagy vélhetően eltér, ezt érdemes fenntartással kezelni.

## 6. A közbülső rétegekkel kapott eredmények

A 2. ábra mutatja a **leiratózó** wav2vec 2.0 hálóból nyert beágyazások használatával kapott átlagos AUC értékeket. (*Konv.* jelöli a konvolúciós blokk utolsó rétegéből nyert beágyazásokhoz tartozó eredményeket, míg 1..24 a finomhangolt blokk megfelelő rejtett rétegét. A 24. réteg a blokk utolsó rétege, amelyhez tartozó eredmények az 1. táblázatban is szerepelnek.) A hibásávok a legkisebb és legnagyobb kapott értékeket mutatják. Meglepő módon az *összes* belső rejtett réteggel jobb eredményeket kaptunk, mint az utolsó konvolúciós rétegre támaszkodva. A különbségeket a Mann-Whitney U teszttel (lásd (Mann és Whitney, 1947)) elemezve azt látjuk, hogy a javulás a 23 esetből 20-ban bizonyult statisztikailag is szignifikánsnak (csak a 16., 18. és 22. rétegek képeznek kivételt).





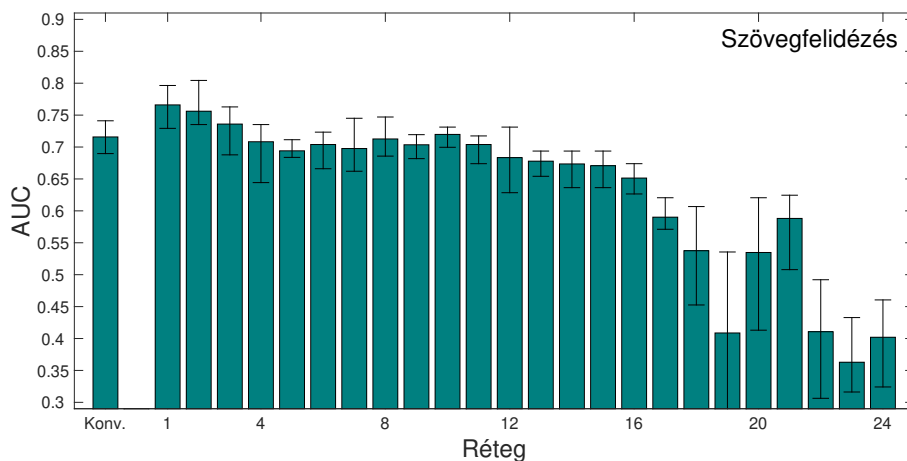
2. ábra: Az átlagos AUC értékek és a [min; max] terjedeleme (hibasávokkal jelölve) a konvolúciós blokk utolsó rejtett rétege (*Konv.*), valamint a finomhangolt blokk összes rejtett rétege esetén, a *leiratozó wav2vec 2.0* hálót használva.

A finomhangolt blokk utolsó rejtett rétegéhez hasonlítva viszont már csak hat esetben kaptunk statisztikailag szignifikáns (vagy időnként bármilyen) javulást; ezek mindegyike a finomhangolt blokk mélyebben fekvő részén (1...9. rétegek) található. Ez alapján érdemes lehet egy *wav2vec 2.0* háló belső rejtett rétegeiből nyerni az osztályozáshoz használt beágyazásokat, de tanácsos lehet a mélyebben fekvő rétegeket választani, legalábbis *sclerosis multiplex* felismerése esetén.

A **beszélő-azonosító** háló esetén (ld. 3. ábra) az átlagos AUC értékek teljesen másként alakultak. A konvolúciós blokk utolsó rétegét ugyan néhány belső rejtett réteg túlszárnyalja (ebből az első három réteggel kapott javulás bizonyult szignifikánsnak), a későbbi rétegek használatával azonban csak azonos SM-azonosítási teljesítményt tudunk elérni. A finomhangolt blokk magasabban található rejtett rétegei pedig egyenesen a pontosság csökkenéséhez vezettek. (Ebben az esetben nem láttuk sok értelmét, hogy a finomhangolt blokk utolsó rétegéhez is hozzámérjük a kapott eredményeket.)

A 2. táblázatban tüntettük föl a néhány kiválasztott rejtett réteg esetén kapott AUC értékeket; a szignifikáns javításokat „\*” ( $p < 0,05$ ) és „\*\*” ( $p < 0,01$ ) jelzi, míg „—” esetén nincs ilyen különbség. A per szimbólum („/”) előtti és utáni szimbólumok a konvolúciós, illetve a finomhangolt blokk utolsó rejtett rétegéhez viszonyított javulásra vonatkoznak. A **leiratozó** hálóból származó beágyazások esetén mind a négy kiválasztott réteggel szignifikáns javulást tudunk elérni a konvolúciós blokk utolsó rejtett rétegéhez képest<sup>1</sup>, míg az utolsó finomhangolt réteget a 4. és 6. rejtett réteggel tudtuk jelentősen túlszárnyalni. Ezen két réteg között nincs nagy különbség az átlagos AUC értékben, bár a 4. réteg esetén ez

<sup>1</sup> Valamint még további 16 réteggel



3. ábra: Az átlagos AUC értékek és a [min; max] terjedelem (hibasávokkal jelölve) a konvolúciós blokk utolsó rejtett rétege (*Konv.*), valamint a finomhangolt blokk összes rejtett rétege esetén, a *beszélő-azonosító* wav2vec 2.0 hálót használva.

wav2vec 2.0 modell	Beágyazás	AUC		
		Átlag	Szórás	Terjedelem
Leiratózó	2. finomhangolt <sup>**/-</sup>	0,808	0,022	[ 0,789; 0,838 ]
	4. finomhangolt <sup>**/**</sup>	0,868	0,004	[ 0,866; 0,874 ]
	6. finomhangolt <sup>**/**</sup>	0,860	0,016	[ 0,832; 0,872 ]
	8. finomhangolt <sup>**/-</sup>	0,821	0,010	[ 0,814; 0,838 ]
	Utolsó konvolúciós	0,724	0,008	[ 0,712; 0,733 ]
	Utolsó finomhangolt	0,806	0,014	[ 0,787; 0,824 ]
Beszélő-azonosító	2. finomhangolt <sup>**/**</sup>	0,756	0,028	[ 0,735; 0,804 ]
	4. finomhangolt <sup>-/**</sup>	0,708	0,038	[ 0,644; 0,735 ]
	6. finomhangolt <sup>-/**</sup>	0,704	0,024	[ 0,666; 0,723 ]
	8. finomhangolt <sup>-/**</sup>	0,713	0,026	[ 0,686; 0,747 ]
	Utolsó konvolúciós	0,716	0,020	[ 0,690; 0,741 ]
	Utolsó finomhangolt	0,402	0,059	[ 0,324; 0,461 ]

2. táblázat. Az átlagos AUC értékek, szórásuk és terjedelmük ([min;max]) néhány kiválasztott belső rejtett réteg esetén. A statisztikailag szignifikáns javulást „<sup>\*\*</sup>” ( $p < 0,05$ ) és „<sup>\*\*\*</sup>” ( $p < 0,01$ ) jelzi, míg „-” esetén nincs ilyen különbség.

valamivel magasabb (0,868 vs. 0,860), valamint az alacsonyabb szórás és szűkebb terjedelem robusztusabb működésre is utal.

A *beszélő-azonosító* wav2vec 2.0 háló esetén nem ennyire jók az eredmények: a konvolúciós blokk utolsó rejtett rétegével kapott osztályozási teljesít-

wav2vec 2.0 modell	Beágyazás	AUC		
		Átlag	Szórás	Terjedelem
Leiratozó	1...8. finomhangolt <sup>**/*</sup>	0,832	0,040	[ 0,740; 0,872 ]
	9...16. finomhangolt <sup>**/-</sup>	0,798	0,026	[ 0,741; 0,828 ]
	17...24. finomhangolt <sup>**/-</sup>	0,762	0,033	[ 0,719; 0,817 ]
Beszélő- azonosító	1...8. finomhangolt <sup>-/**</sup>	0,722	0,037	[ 0,664; 0,793 ]
	9...16. finomhangolt <sup>-/**</sup>	0,686	0,029	[ 0,632; 0,730 ]
	17...24. finomhangolt <sup>-/-</sup>	0,479	0,107	[ 0,311; 0,621 ]

3. táblázat. Az átlagos AUC értékek, szórásuk és terjedelmük (5. és 95. percentilisek) a finomhangolt blokk mélyebben, középen és magasabban fekvő régiói esetén. A statisztikailag szignifikáns javulást „\*” ( $p < 0.05$ ) és „\*\*” ( $p < 0.01$ ) jelzi, míg „—” esetén nincs ilyen különbség.

ményt csak a finomhangolt blokk 2. rejtett rétege tudta túlteljesíteni ( $p = 0,003$ ), a többi kiemelt réteg már csak ekvivalens eredményekhez vezetett, ráadásul valamivel nagyobb szórás-értékek mellett. (A 2. finomhangolt rétegre épülő osztályozó modellek robusztussága megegyezik a konvolúciós rétegre építettekével.) A finomhangolt blokk utolsó rejtett rétegét sikerült mind a négy esetben szignifikánsan túlteljesíteni, de annak pontossága eleve nagyon alacsony volt. Összességében a beszélő-azonosító modellel kapott eredmények elmaradtak a leiratozó modellel kapottaktól, még a mélyebben fekvő rejtett rétegek esetén is.

Végül a 3. táblázatba szedtük össze finomhangolt blokk alsó, középső és legfőbb egyharmadának összesített teljesítményét. (Ebben az esetben a terjedelmet a 40 ( $8 \times 5$ ) AUC érték 5. és 95. percentilisével számítottuk.) A **leiratozó** háló esetén mindegyik régió szignifikánsan jobb eredményeket adott, mint a konvolúciós blokk utolsó rejtett rétege ( $p < 0,01$ ), a finomhangolt blokk utolsó rejtett rétegéből kapott beágyazásoknál azonban csak az alsó régió bizonyult jobbnak ( $p = 0,038$ ), míg a legfőbb régió szignifikánsan rosszabb eredményeket adott. A **beszélő-azonosító** háló esetén, meglepő módon, még a mélyebb régió is csupán azonos teljesítményt tudott nyújtani, mint a konvolúciós réteg (0,722 és 0,716 átlagos AUC értékek,  $p = 0,691$ ), a középső és legfőbb régiók pedig szignifikánsan rontottak ( $p = 0,038$  és  $p < 0,001$ ). A legtöbb esetben sikerült túlszárnyalni a finomhangolt blokk utolsó rejtett rétegét, azonban ez a referencia-érték eleve igen alacsony volt.

Összességében azt kaptuk, hogy mindkét háló esetén általában a finomhangolt blokk mélyebben fekvő rétegeiből érdemes jellemzőket kinyerni, mert azok alkalmasabbak a sclerosis multiplex felismerésére. A kétfajta hálótípus közül a beszédfelismerésre tanított (*leiratozó*) sokkal hatékonyabbnak bizonyult, mint a beszélők felismerésére tanított (*beszélő-azonosító*), azonban ezt fenntartással kell kezelnünk, hiszen a két háló eltérő (bár egyaránt magyar nyelvű) adaton és eltérő körülmények között lett betanítva.

## 7. Konklúzió

Jelen tanulmányunkban azt vizsgáltuk, hogy a sclerosis multiplex milyen mértékben detektálható automatikusan az alanyok beszédéből. Ehhez egy viszonylag hagyományos rendszert építettünk, ahol a jellemzőket egy wav2vec 2.0 hálóból nyertük, osztályozásra pedig SVM-et használtunk. 45 anyanyelvi magyar beszélő felvételeit használtuk (23 relapszáló-remittáló altípusú SM alanyttó és 22 demográfiailag illesztett egészséges kontrolltól). A wav2vec 2.0 háló konvolúciós és finomhangolt (kontextualizált) blokkjainak utolsó rejtett rétegeiből vett jellemzők mellett a finomhangolt blokk további 23 rétegét is megvizsgáltuk. Két különböző módon (beszédfelismerésre, valamint beszélőfelismerésre) tanított wav2vec 2.0 hálót is teszteltünk.

A beszédfelismerő (vagy leiratózó) háló esetén a konvolúciós blokk utolsó rejtett rétegéhez képest majdnem minden esetben statisztikailag szignifikáns javulást sikerült elérni, és a finomhangolt blokk utolsó rejtett rétegét is túlszárnyaltuk a mélyebben fekvő rejtett rétegekkel. A beszélő-azonosító háló esetén azonban, bár néhány mélyebben fekvő rejtett réteg javulást eredményezett, a legtöbb esetben azonos vagy egyenesen rosszabb eredményeket kaptunk. Ezekből a megfigyelésekből arra következtethetünk, hogy egy jellemzőkinyerésre használt wav2vec 2.0 hálót érdemesebb lehet a hagyományos beszédfelismerési feladatra tanítani, mint beszélőazonosításra. Ugyanakkor mindkét tanítási kritérium esetén a finomhangolt blokk mélyebben fekvő rejtett rétegekre érdemes támaszkodni.

Az elvégzett kísérletek hiányossága, hogy a két wav2vec 2.0 háló, bár architektúrájukban és a finomhangolás előtti súlyaikban megegyeztek, más adaton, eltérő keretrendszert használva és eltérő hiperparaméterekkel lettek tanítva. Bár a saját csapatunk által, beszélőazonosításra tanított háló (beszélő)osztályozási pontossága kellően alacsonynak tűnik (2% alatti), nem zárhatjuk ki, hogy valamilyen technikai probléma vezetett a látványosan alacsonyabb teljesítményhez, mikor a rejtett rétegek aktivációit használtuk jellemzőkként.

Megjegyeznénk még, hogy az egyes rejtett rétegek más jellegű információkat tárolnak, így a belőlük kinyert beágyazások kombinálása tovább javíthatja az osztályozási pontosságot. Sajnos az ilyen kombinációs kísérletek korrekt kiértékeléséhez valószínűleg több alanyra van szükség, mint a jelenleg rendelkezésünkre álló 45 fő (amely egyébként az orvosi célú beszédfeldolgozás területén elegendőnek számít). Ennek ellenére a közeljövőben tervezzük ilyen kombinációs kísérletek elvégzését.

## 8. Köszönetnyilvánítás

A kutatást részben a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal támogatta a NKFIH K-132460 pályázat keretében. A kutatást (amelyet a Szegedi Tudományegyetem valósított meg) az Innovációs és Technológiai Minisztérium és a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal is támogatta a TKP2021-NVA-09 pályázat és a Mesterséges Intelligencia Nemzeti Laboratórium (RRF-2.3.1-21-2022-00004) keretében.

## Hivatkozások

- Babu, A., Wang, C., Tjandra, A., Lakhota, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., Auli, M.: XLSR: Self-supervised cross-lingual speech representation learning at scale. In: *Proceedings of Interspeech*. pp. 2278–2282 (2022)
- Baevski, A., Zhou, Y., Mohamed, A., Auli, M.: wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems* 33, 12449–12460 (2020)
- Cai, J., Song, Y., Wu, J., Chen, X.: Voice disorder classification using wav2vec 2.0 feature extraction. *Journal of Voice* p. to appear (2025)
- Carvajal-Castaño, H.A., Pérez-Toro, P.A., Orozco-Arroyave, J.R.: Classification of Parkinson’s Disease patients – a deep learning strategy. *Electronics* 11(17), 2684 (2022)
- Cawley, G.C., Talbot, N.L.C.: On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research* 11(Jul), 2079–2107 (2010)
- Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2(3), 1–27 (2011)
- Chen, L.W., Rudnicky, A.: Exploring wav2vec 2.0 fine tuning for improved speech emotion recognition. In: *Proceedings of ICASSP*. Rhodes Island, Greece (2023)
- Egas-López, J.V., Svindt, V., Bóna, J., Hoffmann, I., Gosztolya, G.: Automated multiple sclerosis screening based on encoded speech representations. In: *Proceedings of Interspeech*. pp. 3003–3007. Dublin, Ireland (Aug 2023)
- Fan, Z., Li, M., Zhou, S., Xu, B.: Exploring wav2vec 2.0 on speaker verification and language identification. In: *Proceedings of Interspeech*. pp. 1509–1513 (2021)
- Fara, S., Hickey, O., Georgescu, A., Gorla, S., Molimpakis, E., Cummins, N.: Bayesian Networks for the robust and unbiased prediction of depression and its symptoms utilizing speech and multimodal data. In: *Proceedings of Interspeech*. pp. 1728–1732. Dublin, Ireland (2023)
- Gonzalez-Machorro, M., Hecker, P., Reichel, U.D., Hammer, H.N., Hoepner, R., Pedrotti, L., Zmutt, A., Sagha, H., van Beek, J., Eyben, F., Schuller, D.M., Schuller, B.W., Arnrich, B.: Towards supporting an early diagnosis of multiple sclerosis using vocal features. In: *Proceedings of Interspeech*. pp. 1518–1522 (2023)
- Gosztolya, G., Egas-López, J.V., Svindt, V., Bóna, J., Hoffmann, I.: Sclerosis multiplex felismerése spontán beszédből wav2vec 2.0 modellekből kinyert jellemzőkkel. In: *Proceedings of MSZNY*. pp. 33–43. Szeged, Hungary (2023)
- Gosztolya, G., Tóth, L., Svindt, V., Bóna, J., Hoffmann, I.: Using acoustic Deep Neural Network embeddings to detect Multiple Sclerosis from speech. In: *Proceedings of ICASSP*. pp. 6927–6931. Singapore (May 2022)
- Grosman, J.: Fine-tuned XLSR-53 large model for speech recognition in Hungarian. <https://huggingface.co/jonatasgrosman/wav2vec2-large-xlsr-53-hungarian> (2021)

- Grósz, T., Porjazovski, D., Getman, Y., Kadiri, S., , Kurimo, M.: Wav2vec2-based paralinguistic systems to recognise vocalised emotions and stuttering. In: Proceedings of ACM Multimedia. pp. 7026–7029. Lisboa, Portugal (Oct 2022)
- Grósz, T., Virkkunen, A., Porjazovski, D., Kurimo, M.: Discovering relevant subspaces of BERT, Wav2Vec 2.0, ELECTRA and ViT embeddings for humor and mimicked emotion recognition with integrated gradients. In: Proceedings of MuSe. pp. 27–34. Ottawa, Canada (2023)
- Huckvale, M., Beke, A., Ikushima, M.: Prediction of sleepiness ratings from voice by man and machine. In: Proceedings of Interspeech. pp. 4571–4575. Shanghai, China (Oct 2020)
- Ivanova, O., Martínez-Nicolás, I., Meilán, J.J.G.: Speech changes in old age: Methodological considerations for speech-based discrimination of healthy ageing and Alzheimer’s disease. *International Journal of Language & Communication Disorders* 59(1), 13–37 (2023)
- Jenei, A.Z., Kiss, G., Sztahó, D.: Detection of speech related disorders by pre-trained embedding models extracted biomarkers. In: Proceedings of SPECOM. pp. 279–289. Gurugram, India (2022)
- Kiss, G., Tulics, M.G., Sztahó, D., Vicsi, K.: Language independent detection possibilities of depression by speech. In: Proceedings of NoLISP. pp. 103–114 (2016)
- Klumpp, P., Arias-Vergara, T., Vásquez-Correa, J.C., Pérez-Toro, P.A., Orozco-Arroyave, J.R., Batliner, A., Nöth, E.: The phonetic footprint of Parkinson’s disease. *Computer Speech & Language* 72(Mar), 101321 (2022)
- Kodali, M., Kadiri, S.R., Alku, P.: Classification of vocal intensity category from speech using the wav2vec2 and whisper embeddings. In: Proceedings of Interspeech. pp. 4134–4138 (2023)
- Kondratenko, V., Karpov, N., Sokolov, A., Savushkin, N., Kutuzov, O., Minkin, F.: Hybrid dataset for speech emotion recognition in Russian language. In: Proceedings of Interspeech. pp. 4548–4552 (2023)
- Kumar, N., Nasir, M., Georgiou, P., Narayanan, S.S.: Robust multichannel gender classification from speech in movie audio. In: Proceedings of Interspeech. pp. 2233–2237. San Francisco, CA, USA (Sep 2016)
- Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics* 18(1), 50–60 (1947)
- Mar, R.A.: The neuropsychology of narrative: Story comprehension, story production and their interrelation. *Neuropsychologia* 42(10), 1414–1434 (2004)
- Mihajlik, P., Balog, A., Grácsi, T.E., Kohári, A., Tarján, B., Mády, K.: BEA-Base: A benchmark for ASR of spontaneous Hungarian. In: Proceedings of LREC. pp. 1970–1977 (2022)
- Neuberger, T., Gyarmathy, D., Grácsi, T., Horváth, V., Gósy, M., Beke, A.: Development of a large spontaneous speech database of agglutinative Hungarian language. In: TSD. pp. 424–431 (2014)
- Orsolya, K., Alinka, T., Katalin, J., Péter, K.: A beszédsebesség vizsgálata parkinson-kór-, sclerosis multiplex, valamint stroke-eredetű dysarthriák ese-

- tében. *Rehabilitáció: A magyar rehabilitációs társaság folyóirata* 30(1), 3–10 (2020)
- Pepino, L., Riera, P., Ferrer, L.: Emotion recognition from speech using wav2vec 2.0 embeddings. In: *Proceedings of Interspeech*. pp. 3400–3404. Brno, Czechia (2021)
- Pérez-Toro, P., Klumpp, P., Hernandez, A., Arias, T., Lillo, P., Slachevsky, A., García, A., Schuster, M., Maier, A., E.Nöth, Orozco-Arroyave, J.: Alzheimer’s detection from English to Spanish using acoustic and linguistic embeddings. In: *Proceedings of Interspeech*. pp. 2483–2487 (2022)
- Schneider, S., Baeviski, A., Collobert, R., Auli, M.: wav2vec: Unsupervised pre-training for speech recognition. In: *Proceedings of Interspeech*. pp. 3465–3469 (2019)
- Szirmai, I.: *Neurológia. Medicina*, Budapest (2006)
- Thienpondt, J., Speksnijder, C.M., Demuyneck, K.: Behavioral analysis of pathological speaker embeddings of patients during oncological treatment of oral cancer. In: *Proceedings of Interspeech*. pp. 3018–3022 (2023)
- Vaessen, N., Van Leeuwen, D.A.: Fine-tuning wav2vec2 for speaker recognition. In: *Proceedings of ICASSP*. pp. 7967–7971 (2021)





# Imitált és valódi depresszió elkülönítése mély neurális háló alapú jellemzőkinyerési módszerekkel

Dalotti Ágoston<sup>1</sup>, Kiss Gábor<sup>1</sup>

<sup>1</sup>Budapesti Műszaki és Gazdaságtudományi Egyetem,  
Távközlési és Mesterséges Intelligencia Tanszék  
dalottiagoston@edu.bme.hu, kiss.gabor@vik.bme.hu

**Kivonat** Korunk egyik leggyakoribb mentális betegsége a depresszió. Az utóbbi években számos kutatás foglalkozott a depresszió automatikus felismerésével. A depresszió hatással van a betegek beszédképzésére, és így a megváltozott beszéd elemzése lehetőséget biztosít a diagnózis automatikus támogatására, vagy betegség automatikus szűrésére. Jelen kutatásban azt vizsgáljuk, hogy a depressziót felismerő modellek, hogyan teljesítenek különböző érzelmi töltetű illetve depressziós beszédet imitáló minták esetében. Ennek a vizsgálata azért fontos, mert eddig főleg úgy vizsgálták az ilyen modellek teljesítményét, hogy egészséges - semleges érzelmi töltetű, illetve depressziós személyek beszédén tesztelték azokat. Ezek tükrében, jelen cikkben a legújabb mély neurális háló alapú jellemzőkinyerő eljárások segítségével bemutatunk egy nem-együtműködő páciens feltételező - depresszió súlyosságát becslő - depressziót felismerő modellt. Az eredmények alapján kijelenthető, hogy a különböző érzelmi töltetek illetve a depresszió imitálása megnehezíti a depresszió felismerését, ugyanakkor bemutatjuk, hogy megfelelő modell kialakításával javítható a felismerés pontossága.

**Kulcsszavak:** Depresszió, X-Vektor, ECAPA, ResNet, Speechbrain, SVR, Random Forest, MLPRegression, Imitáció, Diagnózis

## 1. Bevezetés

Az emberi beszéd egy komplex kifejező eszköz, mely a nyelvi tartalom túlmenően nonverbális információkat is magában hordoz. Kutatások sora foglalkozott a beszédben rejlő nonverbális lehetőségek kiaknázásával, úgy, mint beszélőazonosítással (Gráczai és mtsai, 2022), diarizációval (Dawalatabad és mtsai, 2021) vagy akár orvosi diagnózis támogatásával (Sztahó és mtsai, 2022)(Salih és Sztahó, 2023). Ez utóbbihoz kapcsolódva számos kutatás foglalkozott a depresszió - beszédelemzés alapján történő - diagnózisának támogatásával (Low és mtsai, 2020). Az eddigi kutatások elsősorban semleges érzelmi töltetű, együtműködő egészséges személyek és depressziós személyek beszédmintáit elemezték. Jelen kutatásban azt vizsgáljuk a korábbiaktól eltérő új beszédminták felhasználásával, hogy a depresszió illetve különböző érzelmi állapotok imitálása mennyiben

nehezíti meg a depressziós állapot automatikus felismerését. A kutatás során először elkészítettünk egy magyar nyelven működő modellt, amely közelíti a legkorszerűbb együttműködő pácienszt feltételező modellek teljesítményét. Ennek megvizsgáltuk a működését az imitált mintákon. Majd modern, mély neurális hálókön alapuló jellemzőkinyerő eljárások segítségével javaslatot teszünk arra, hogyan lehet javítani a gépi modelleken, hogy ilyen jellegű mintákon is jobban teljesítsenek, úgy, hogy közben a klasszikus együttműködő mintákon se romoljon a teljesítményük.

## 1.1. Depresszió

A depresszió<sup>1</sup> korunk egyik legmeghatározóbb gyógyítható betegsége. A WHO (World Health Organization) statisztikái szerint a teljes populáció 3,8% -át, míg a felnőttek 5% -át érinti (WHO, 2023). Továbbá predikciójuk szerint az unipoláris depresszió 2030-ra benne lesz a 3, világszinten legtöbb embert érintő betegségben (a HIV/AIDS és a szívproblémák mellett) (Marcus és mtsai, 2012). A depresszió jelentősen befolyásolja a betegek életminőségét, szomatikus panaszok mellett komoly kognitív, viselkedésszerű és motivációs tünetek jelentkeznek. A korai tünetek viszont annyira különfélék lehetnek, hogy a háziorvoshoz forduló betegek ritkán kapnak megfelelő diagnózist (Torzsa és mtsai, 2009). A depresszió diagnosztizálására csak egy szűk, magasan képzett orvosi-pszichológusi réteg képes.

**BDI-II - Beck Depression Inventory-II** (továbbiakban BDI) olyan tünetek klinikai megfigyelésén alapul, amelyek depressziós személyeknél gyakran, míg egészséges személyeknél ritkábban jelentkeznek (Beck és mtsai, 1988).

Az egyes tünetekre adott értékek összegzéséből alakul ki a BDI pontszám.

A határokat irodalomtól függően egy kicsit máshol találjuk, mi az alábbiak szerint dolgoztunk (Egas-López és mtsai, 2022):

- 0-13: minimális depresszió (egészséges)
- 14-19: enyhe depresszió
- 20-28: mérsékelt depresszió
- 29-63: súlyos depresszió

A BDI egy elfogadható, és nemzetközileg használt skála a depresszió súlyosságának becslésére. Kimutatható, hogy felhasználásával a gépi tanuló modellek pontosabb becslést képesek adni a vizsgált alany betegségének súlyosságát illetően. Bár a BDI pontszám manuális kitöltéssel elérhető, az automatikus beszédelemzés olyan esetekben is támogatja a diagnózist, ahol a pontszám nem áll rendelkezésre. A teszt egyszerű elvégzése lehetővé teszi a depressziós beszédatadabázisok gyors növelését, amely elengedhetetlen a jelenlegi modellek pontosságának fejlesztéséhez (Hajduska-Dér és mtsai, 2022).

---

<sup>1</sup> Major Depressive Disorder - továbbiakban depresszió

## 2. Adatbázisok

### 2.1. Magyar Depressziós Beszédadatbázis (MDB)

A kutatásunk kezdetekor elérhető legnagyobb, magyar nyelvű felvételeket tartalmazó, depressziós beszédadatbázist használtuk, amelyben a vizsgált személyek egy körülbelül 1 perc hosszú rögzített szöveget olvastak fel ("Az Északi Szél és a Nap").

A felvételek csendes helyiségben kerültek rögzítésre Lexicon Alpha típusú külső hangkártya segítségével, 44,1 kHz mintavételi frekvenciával, 16 bites kvantálással, tömörítetlen PCM kódolással, ATR3350 típusú csiptetős mikrofonnal. A mikrofon a vizsgált személy mellkasához volt rögzítve (Kiss, 2019).

Erre az adatbázisra a későbbiekben "Magyar Depressziós Beszédadatbázis (MDB)" néven hivatkozunk. Az adatbázis összesen 414 személytől tartalmaz pontosan egy olvasott felvételt, 214 egészséges és 200 depressziós embertől. A 1. táblázatban foglaltuk össze a vizsgált alanyokkal kapcsolatos legfontosabb információkat.

1. táblázat. Az MDB vizsgált alanyainak adatai.

	Alanyok száma (nők/férfiak)	Alanyok BDI szerinti átlaga és szórása	Alanyok életkorának átlaga és szórása
Depressziós alanyok:	200 (134/66)	27,1 (±9,1)	39,25 (±14,25)
Egészséges alanyok:	214 (133/81)	5,48 (±5,1)	52,47 (±20,13)
Összesített:	414 (267/147)	15,75 (±13,15)	46 (±18,8)

### 2.2. Magyar Depressziós Imitációs Beszédadatbázis

A depresszió és különböző érzelmi állapotok imitálásához új beszédadatbázist rögzítettünk, az MDB rögzítési körülményeivel megegyező módon.

A felvételek elkészítése előtt minden vizsgált alanyt tájékoztattunk az adatkezelési kötelességeinkről, és kitöltöttünk egy nyilatkozatot, miszerint tárolhatjuk és vizsgálhatjuk az általuk származó hangfelvételeket (Nemzeti Adatvédelmi és Információszabadság Hatóság, 2016).

Az adatbázis elkészítéséhez hivatásos és amatőr színészeket kértünk fel. Minden vizsgált alannal 5 felvételt készítettünk, az alábbiak szerint:

- Kontroll felolvasás: Természetes módon (semleges érzelmi állapotban) történő felolvasás. Erre a továbbiakban "Kontroll minta"-ként hivatkozunk.
- Depressziót imitáló felolvasás. Fontos megjegyezni, hogy ezen kívül semmilyen instrukciót nem adtunk, a vizsgált alany döntése volt, hogy ez mit jelent. Erre a továbbiakban "Depressziót imitáló minta"-ként hivatkozunk.

- Dühöt imitáló felolvasás. Erre a továbbiakban "Dühöt imitáló minta"-ként hivatkozunk.
- Szomorúságot imitáló felolvasás. Erre a továbbiakban "Szomorúságot imitáló minta"-ként hivatkozunk.
- Vidám hangulatot imitáló felolvasás. Erre a továbbiakban "Vidámságot imitáló minta"-ként hivatkozunk.

Továbbá minden vizsgált alannyal kitöltöttünk egy BDI tesztet, melynek eredményére a továbbiakban "Saját BDI pontszám"-ként hivatkozunk.

Erre az adatbázisra a későbbiekben "Magyar Depressziós Imitációs Beszédadatbázis (MDIB)" néven hivatkozunk. Az adatbázisban összesen 135 felvétel van, 27 különböző színésztől.

A vizsgált alanyok releváns adatait a 2. táblázatban foglaltuk össze.

2. táblázat. Az MDIB vizsgált alanyainak adatai.

Alanyok száma (nők/férfiak)	Saját BDI pontszámok átlaga és szórása	Alanyok életkorának átlaga és szórása
27	8,63	31,3
(13/14)	(±5,35)	(±16,57)

Az adatbázis létrehozása során igyekeztünk minél több hivatásos színészt megkeresni. Végül a minták 48%-a származik hivatásos színházi színészekről (Katonai József Színház, Örkény István Színház, Vígszínház színészei, és a Színház- és Filmművészeti Egyetem hallgatói).

**Az érzelmek imitálásáról:** Fontos megjegyezni hogy az MDIB-ben rögzített felvételek imitált érzelmekkel töltött bemondásokat tartalmaznak. Saját tapasztalatunk, és korábbi kutatások is azt igazolják, hogy a színészek által előadott érzelmek gyakran hevesebbek, mint ahogy azt a valóságban tapasztaljuk (Fegyő, 2019). Ennek ellenére az érzelmek beszéd alapú detektálásával foglalkozó kutatásokban bevett szokás, hogy színészek bemondásaival dolgoznak.

### 3. Módszerek

#### 3.1. Mély neurális háló alapú jellemzőkinyerés

A kutatás során három különböző mély neurális háló alapú beszédjellemezőkinyerő modellt használtunk; X-Vektor implementációt<sup>2</sup>, az ennek továbbfejlesztéseként értelmezhető ECAPA-TDNN implementációt<sup>3</sup>, és a ResNet architektúrának az implementációját<sup>4</sup>. Mindhárom esetben a Speechbrain által előtanított megvalósítást használtuk (Ravanelli és mtsai, 2021). A modellek tanítása a VoxCeleb1-2 adatbázisokkal történt (Chung és mtsai, 2018).

<sup>2</sup> <https://huggingface.co/speechbrain/spkrec-xvect-voxceleb>

<sup>3</sup> <https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb>

<sup>4</sup> <https://huggingface.co/speechbrain/spkrec-resnet-voxceleb>

**X-Vektor technológia** David Snyder nevéhez fűződik (Snyder, 2020). Az eredeti architektúra beszélőazonosításhoz készült, de korábbi kutatások bemutatták, hogy a 7. (szegmens) réteg beágyazásait lényegesen szélesebb célra fel lehet használni (Egas-López és mtsai, 2021) (Egas-López és mtsai, 2022).

3. táblázat. X-Vektor TDNN architektúra (Snyder és mtsai, 2018).

Réteg	Réteg kontextus	Teljes kontextus	Bement x Kimenet
1. (keret)	$[t-2, t+2]$	5	120x512
2. (keret)	$\{t-2, t, t+2\}$	9	1536x512
3. (keret)	$\{t-3, t, t+3\}$	15	1536x512
4. (keret)	$\{t\}$	15	512x512
5. (keret)	$\{t\}$	15	512x1500
Stat. összegzés	$[0, T)$	T	1500Tx3000
6. (szegmens)	$\{0\}$	T	3000x512
7. (szegmens)	$\{0\}$	T	512x512
Softmax	$\{0\}$	T	512xN

A mély neurális háló szerkezetét a 3. táblázatban mutatjuk be. Megfigyelhető, hogy a keret szintű rétegek az adott keret szűk időbeli kontextusát is felhasználják a kimenet megalkotásához, így Time-Delay Neural Network (TDNN)-ről beszélünk.

A statisztikai összegzés után már tud szegmens szinten dolgozni a neurális háló és megfigyelhető, hogy itt a kimenetein már fix méretű (512) jellemzővektorok jelennek meg. A 7. (szegmens szintű) réteg kinyeréséből áll elő a későbbiekben felhasznált jellemzővektor.

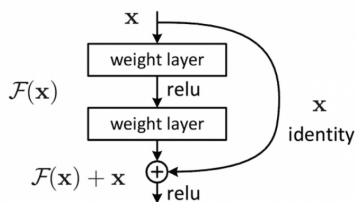
**Emphasized Channel Attention, Propagation and Aggregation**, a továbbiakban ECAPA, az x-vektor modell alapjaira építkezik, és három területen vezet be fejlesztéseket.

Először, bevezeti a csatorna- és kontextusfüggő statisztikai összegzést, amely kiterjeszti a rövid időbeli figyelmet a csatornafigyelemre. A módosítás azt teszi lehetővé a hálózat számára, hogy a beszélőspecifikus jellemzőkre fókuszáljon, elkerülve a hasonló vagy egyidejű időpontokban történő aktivációt.

Másodszor, a modell bevezeti az 1-dimenziós Squeeze and Excitation (1D SE) használatát, ezzel kiterjesztve a keretszintű jellemzőket. Az eredeti x-vektor modellben ezek a jellemzők egy 15 keretnyi időablakra korlátozódtak. Azonban az ECAPA modellben, az 1D SE bevezetésével a modell képessé válik az egész hangfelvétel globális tulajdonságainak megragadására. Az architektúrát a 1. ábra mutatja be.

Továbbá egy reziduális blokkot is bevezettek, amely az 1D SE komponenst összefűzi az x-vektor modellel. Az ECAPA különböző rétegekből származó jellemzőtérképeket integrál, felismerve, hogy még a magasabb rétegek is gazdagíthatják a későbbi beágyazásokat. Ezt a Multilayer Feature Aggregation (MFA)





2. ábra: Alap építőelem a ResNet architektúrában (He és mtsai, 2016).

hez, majd egy újabb ReLU aktiválás következik. Kutatásunk során 34 reziduális blokkot tartalmazó modellel dolgoztunk. A ResNet34 topológiáját a 4. táblázat foglalja össze.

4. táblázat. ResNet34 architektúra 16kHz-es mintavételezés esetén. T a szekvencia hosszúságát jelöli (Villalba és mtsai, 2020).

Réteg	Kimenet mérete	Kimenet csökken	Csatornák	Blokkok	Kernel
konv1	40 x T	Nem	16	-	7x7
resblock-1x	40 x T	Nem	16	3	3x3
resblock-2x	20 x T/2	Igen	32	4	3x3
resblock-3x	10 x T/4	Igen	64	6	3x3
resblock-4x	5 x T/8	Igen	128	3	3x3
átlag	1 x T/8	-	128	-	

### 3.2. Jellemzők előfeldolgozása

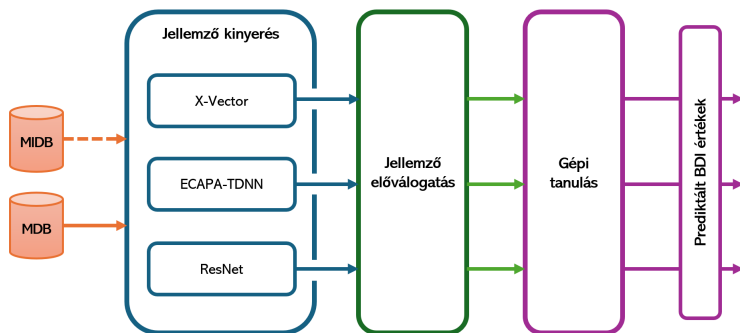
A tútanulás elkerülése végett pearson korreláció alapú jellemző kiválasztást hajtottunk végre, így elhagyva azokat a jellemzőket, amelyek zajt vagy random fluktuációt reprezentálnak (Pudjihartono és mtsai, 2022), illetve, hogy a tanítás időigényét csökkentsük (Yu és Liu, 2004).

Mindhárom jellemzőkinyerő eljárás után kapott jellemző halmazból azt az 50 jellemzőt választottuk ki, amelyek a legnagyobb korrelációt mutatták a BDI pontszámokkal. A kiválasztott jellemzőket z-normalizáltuk.

### 3.3. Modell tanítása és tesztelése

Kutatásunk során a 3. ábrán látható lépések mentén haladva készítettük el a modelleket Python környezetben.

**Referencia Modell:** Megvalósítottunk egy korszerű együttműködő páciens feltételező depresszió súlyosságát becslő modellt. Az MDB mintáin elvégeztük mindhárom neurális hálóval a beszédjellemzők kinyerését, majd végrehajtottuk



3. ábra: A folyamat általános blokkábrája.

a 3.2. fejezetben részletezett lépéseket. Az így kapott jellemzőkkel tanítottunk egy Szupport Vektor Regressziós modellt a BDI pontszámok prediktálására. A tanításhoz RMSE hibafüggvényt használtunk. A tanítás során rbf kernelt használtunk, amely paramétereinek értékeit keresztvalidáció segítségével optimalizáltuk ( $2^{-10} - 2^{10}$  között).

Ezzel párhuzamosan az MDIB mintáin is elvégeztük a jellemzőkinyerést, és az előfeldolgozást, de az így kapott jellemzőket csak a teszteléshez használtuk fel.

**Imitálásra robosztus modell:** A referencia modellhez képest két jelentős változtatást tettünk meg. Egyrészt az előfeldolgozás lépéseit az MDB és MDIB közös adatbázisán végeztük el, ezáltal olyan beszédjellemzők is bekerültek a kiválogatott halmazba, amelyek a referencia modell esetében nem. Továbbá a gépi modell tanítását is az MDB és MDIB közös adatbázisával végeztük el.

Három különböző gépi modellt próbáltunk ki, a Random Forest Regressziót, a Multi-Layer Perceptron neurális háló alapú regressziót, és a Szupport Vektor Regressziót. Mindhárom esetben 20Fold x 20Fold beágyazott keresztvalidációval végeztük el egyrészt a modell optimalizálását, másrészt a tesztelését.

Fontos megjegyezni, hogy az MDIB mintái között személyenként 5 felvétel szerepel. A tanulás során ezeket a mintákat csoportként kezeltük, hogy mind az 5 minta egyszerre kerüljön a tanító vagy teszt halmazba.

Az elvégzett kísérletekből egyértelműen kiderült, hogy a feladatra a három gépi tanuló modell közül a Szupport Vektor Regressziós gép a legalkalmasabb. A modellben rbf kernelt alkalmaztunk, a cost és gamma paraméterek  $2^{-10} - 2^{10}$  hatványok közötti optimalizálásával.

## 4. Eredmények

Az eredmények kiértékelése során a prediktált BDI pontszámok alapján a depresszió osztályozását is elvégeztük. A 13,5 BDI pontszám feletti értékeket depressziósnak, az alattiakat pedig egészségesnek vettük.



#### 4.1. Referencia modell

A 3.3. fejezetben bemutatott (referencia modell) architektúra kiértékelése során kapott eredményeket a 5. táblázatban foglaltuk össze. Az MDB alanyaira a tanítás során<sup>5</sup>, míg az MDIB alanyaira a tesztelés során születtek a BDI predikciók, amelyek alapján a metrikákat kiszámoltuk.

5. táblázat. Baseline eredmények az MDB alanyokra és MDIB alanyokra prediktált BDI pontszámokból.

	MAE	RMSE	Pearson korrelációs együttható	Szenzitivitás	Specificitás	Pontosság	AUC
X-Vektor	6,91	9,23	0,66	79%	88%	84%	0,89
ECAPA	8,82	11,01	0,48	74%	69%	71%	0,79
ResNet	8,09	10,4	0,53	81%	68%	74%	0,81

Az eredmények alapján kijelenthető, hogy az x-vektor alapú modell működött a legjobban. Jelentősen jobb eredményt ért el a másik két mély neurális alapú jellemzőkinyerő eljárással szemben.

**Statisztikai próba:** Az MDIB alanyaira prediktált BDI pontszámok egy szemléletes prezentációja az azonos személy kontroll mintája és az adott érzelmet, vagy depressziót imitáló mintája között számolt párosított t-próba eredménye. A statisztikai próba során a nullhipotézisünk az alábbi volt:

*Az imitáció nincs hatással a depresszió súlyosságát prediktáló modellek működésére.*

A nullhipotézist az alábbi szignifikancia szintek mellett vizsgáltuk:

- $p > 0,05$ : nem szignifikáns a két pontthalmaz eloszlásának eltérése
- $0,05 > p > 0,01$ : \*
- $0,01 > p > 0,001$ : \*\*
- $0,001 > p$ : \*\*\*

Az elvégzett párosított t-próba eredményeit a 6. táblázatban foglaltuk össze. Általánosan elmondható, hogy a modell kevésbé robusztus az imitálásra.

A 4. ábrán megfigyelhető az x-vektor jellemzőkinyerő módszer segítségével elkészített regressziós modell eredményei. Megfigyelhető, hogy nem-együtműködő páciensre (depressziót imitáló minták) jellemzően magasabb BDI pontszámokat prediktál, mint együtműködő páciens (kontroll minta) esetében.

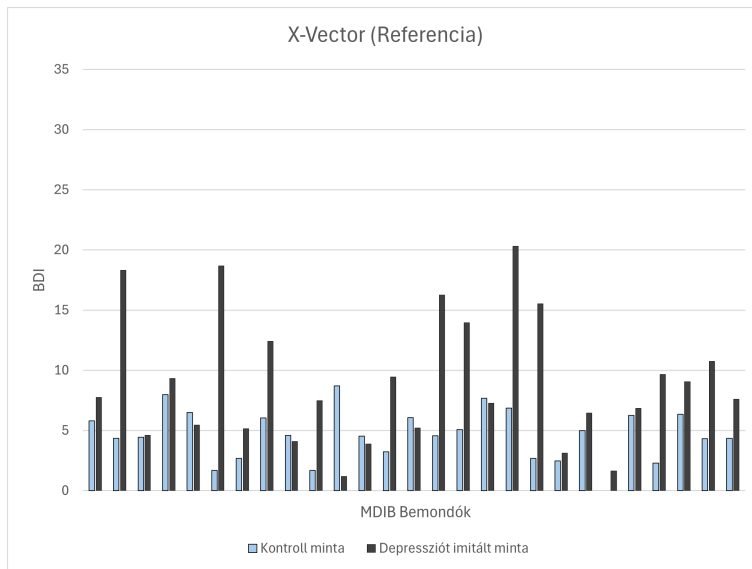
#### 4.2. Imitálásra robusztus modell

A predikciókból számított metrikák eredményeit a 7. táblázat foglalja össze. Ebben az esetben is az x-vektor jellemzőit felhasználva sikerült a legalacsonyabb

<sup>5</sup> A tanítás során elvégzett beagyazott keresztvalidáció külső validációs halmazba esett alanyokra prediktált BDI pontszámokat rögzítettük.

6. táblázat. Baseline párosított t-próba eredmények az MDIB alanyokra prediktált BDI pontszámokból.

	kontroll minta - depressziót imitáló minta	kontroll minta - dühöt imitáló minta	kontroll minta - szomorúságot imitáló minta	kontroll minta - vidámságot imitáló minta
X-Vektor	(***)	(***)	nem szignifikáns	(**)
ECAPA	(***)	nem szignifikáns	(***)	nem szignifikáns
ResNet	(***)	(*)	(**)	(*)



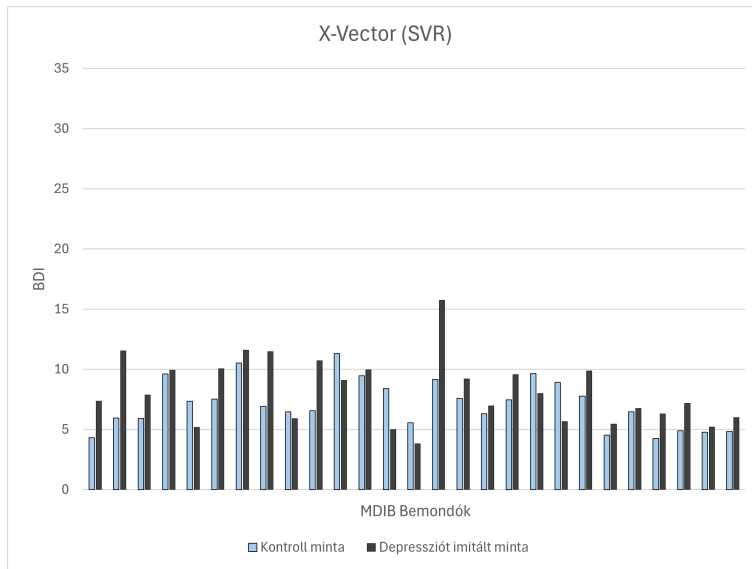
4. ábra: Az MDIB alanyainak kontroll és depressziót imitáló felvételeire (x-vektor jellemzőkkel tanított) referencia modell által prediktált BDI pontszámok.

RMSE-t elérni. Általánosan elmondható, hogy ezzel a modellel jobb eredmények születtek, mint a viszonyítási verzióként bemutatott modellel. (Ld. 5. táblázat.) A 5. ábrán az MDIB alanyainak kontroll és depressziót imitáló mintáira prediktált BDI pontszámokat mutatjuk be. Megfigyelhető, hogy az x-vektor jellemzők felhasználásával született predikciók közel esnek egymáshoz.

**Statisztikai próba:** Az MDIB mintáira prediktált BDI pontszámokra elvégzett párosított t-próbák eredményeit a 8. táblázatban foglaltuk össze. Összevetve a referencia modell eredményeivel, belátható, hogy a modell lényegesen pontosabb predikciókat végez az imitált mintákra, tehát a célunk, miszerint létre szeretnénk hozni egy érzelmi imitációra robosztus modellt, sikeresnek tekinthető. A depressziót imitált minták esetében is javultak az eredmények, de további erőfeszítésre van szükség, ha azt is elfogadható mértékre szeretnénk emelni.

7. táblázat. SVR gépi tanulás eredményei az MDB alanyokra és MDIB alanyokra prediktált BDI pontszámokból.

	MAE	RMSE	Pearson korrelációs együttható	Szenzitivitás	Specifititás	Pontosság	AUC
X-Vektor	6,78	9,11	0,66	78%	92%	86%	0,89
ECAPA	8,43	10,58	0,49	84%	58%	69%	0,80
ResNet	8,35	10,69	0,49	81%	68%	74%	0,81



5. ábra: Az MDIB alanyainak kontroll és depressziót imitáló felvételeire az imitációra optimalizált (és x-vektor jellemzőkkel tanított) modell által prediktált BDI pontszámok.

## 5. Konklúzió

Jelen kutatásban megvizsgáltuk, hogy a beszéd alapján depresszió súlyosságát becselő modellek működésére milyen hatással van, ha a páciens szándékosan olyan beszédmintát hoz létre, amelyben depressziót vagy különböző érzelmeket imitál. A vizsgálathoz két beszédatadabázist használtunk fel. Egyrészt a Magyar Depressziós Beszédatadabázist, amelyben 414 különböző - egészséges illetve depressziós személytől származó felvételek kerültek rögzítésre. Másrészt az újonnan létrehozott Magyar Depressziós Imitációs Beszédatadabázist, amelyben 27 különböző egészséges színésztől 135 felvétel került rögzítésre.

A gépi tanuláshoz a beszédjellezőket három korszerű neurális háló alapú jellemzőkinyerő eljárással (x-vektor, ECAPA-TDNN, ResNet) és további előfeldolgozási lépésekkel készítettük elő. Létrehoztunk egy referencia modellt, amellyel a legkorszerűbb együttműködő páciens feltételező modellek performanciáját kö-

8. táblázat. SVR párosított t-próba eredmények az MDIB alanyokra prediktált BDI pontszámokból.

	kontroll minta - depressziót imitáló minta	kontroll minta - dühöt imitáló minta	kontroll minta - szomorúságot imitáló minta	kontroll minta - vidámságot imitáló minta
X-Vektor	(*)	n.sz.	n.sz.	n.sz.
ECAPA	(***)	n.sz.	(***)	n.sz.
ResNet	(**)	n.sz.	(**)	n.sz.

zelítettük. Továbbá három különböző imitálásra robotsztus gépi tanuló modellt készítettünk el (Szupport Vektor Regresszió, Random Forest Regresszió, Multi-Layer Perceptron Regresszió), amelyek közül az SVR által elért, legkiemelkedőbb eredményeket mutattuk be.

A kiértékelés során bemutattuk, hogy a referencia modell hogyan teljesít az imitált minták kiértékelése során. Az x-vektor jellemzőkkel értük el a legjobb eredményeket (RMSE: 9,23), viszont az elvégzett statisztikai próbából kiderült, hogy a modell az imitálásra kevésbé robotsztus. Az optimalizált modell kiértékelését is elvégeztük. Ebben az esetben is az x-vektor jellemzőkkel értük el a legjobb eredményeket (RMSE: 9,11). Az elvégzett statisztikai próbából kiderült, hogy a modell a különböző érzelmeket imitált mintákra robotsztus lett, illetve a depressziót imitáló mintákra is jobban teljesített.

Ez alapján kijelenthető, hogy a beszéd alapján depresszió súlyosságát becsülő modelleket fel lehet készíteni arra, hogy a páciensek különböző érzelmi töltettel mondják be a szöveget. Ezáltal növelni lehet ezen modellek objektivitását, amely egy rendkívül fontos szempont a diagnosztikát támogató rendszerek esetében. A modellek depresszió imitálására történő felkészítéséhez további erőfeszítésekre van szükség.

## 6. Köszönetnyilvánítás

A K143075 számú projekt a Nemzeti Kutatási Fejlesztési és Innovációs Alapból biztosított támogatással, az K pályázati program finanszírozásában valósult meg.

## Hivatkozások

- Aziz, D., Sztahó, D.: Automatic cross-and multi-lingual recognition of dysphonia by ensemble classification using deep speaker embedding models. Expert Systems p. e13660 (2024)
- Beck, A.T., Steer, R.A., Carbin, M.G.: Psychometric properties of the beck depression inventory: Twenty-five years of evaluation. Clinical psychology review 8(1), 77–100 (1988)
- Chung, J.S., Nagrani, A., Zisserman, A.: Voxceleb2: Deep speaker recognition. Interspeech, arXiv preprint arXiv:1806.05622 (2018)

- Dawalatabad, N., Ravanelli, M., Grondin, F., Thienpondt, J., Desplanques, B., Na, H.: Ecapa-tdnn embeddings for speaker diarization. Proc. Interspeech Conf. pp. 3560–3564 (2021)
- Desplanques, B., Thienpondt, J., Demuynck, K.: ECAPA-TDNN: emphasized channel attention, propagation and aggregation in TDNN based speaker verification. In: Meng, H., Xu, B., Zheng, T.F. (szerk.) Interspeech 2020. pp. 3830–3834. ISCA (2020)
- Egas-López, J.V., Balogh, R., Imre, N., Tóth, L., Vincze, V., Pákáski, M., Kálmán, J., Hoffmann, I., Gosztolya, G.: Enyhe kognitív zavar detektálása beszédhangból x-vektor reprezentáció használatával. Magyar Számítógépes Nyelvészeti Konferencia, 17 pp. 147–156 (2021)
- Egas-López, J.V., Kiss, G., Sztahó, D., Gosztolya, G.: Automatic assessment of the degree of clinical depression from speech using x-vectors. In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8502–8506. IEEE (2022)
- Fegyő, T.: Érzelmek automatikus felismerése a beszéd akusztikus jellemzői alapján. Beszédkutatás p. 151 (2019)
- Grácsi, T.E., Fejes, A., Krepsz, V., Huszár, A.: Speaker recognition over the course of 10 years and across speech styles. Alkalmazott Nyelvtudomány (SI), 94–109 (2022)
- Hajduska-Dér, B., Kiss, G., Sztahó, D., Vicsi, K., Simon, L.: The applicability of the beck depression inventory and hamilton depression scale in the automatic recognition of depression based on speech signal processing. Frontiers in psychiatry 13, 879896 (2022)
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
- Kiss, G.: A depressziós beszéd akusztikai-fonetikai jellemzőinek vizsgálata. Ph.D.-értekezés, Budapest University of Technology and Economics (Hungary) (2019)
- Low, D.M., Bentley, K.H., Ghosh, S.S.: Automated assessment of psychiatric disorders using speech: A systematic review. Laryngoscope investigative otolaryngology 5(1), 96–116 (2020)
- Marcus, M., Yasamy, M.T., van Ommeren, M.v., Chisholm, D., Saxena, S.: Depression: A global public health concern. WHO paper written for the World Federation of Mental Health, World Mental Health Day. World Health Organization (2012)
- Nemzeti Adatvédelmi és Információszabadság Hatóság: Tájékoztató a munkahelyi adatkezelések alapvető követelményeiről (2016)
- Pudjihartono, N., Fadason, T., Kempa-Liehr, A.W., O’Sullivan, J.M.: A review of feature selection methods for machine learning-based disease risk prediction. Frontiers in Bioinformatics 2, 927312 (2022)
- Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.C., Yeh, S.L., Fu, S.W., Liao, C.F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y.,

- Mori, R.D., Bengio, Y.: SpeechBrain: A general-purpose speech toolkit (2021), arXiv:2106.04624
- Salih, A.D.A.H., Sztahó, D.: Cross-lingual dysphonic speech detection using pre-trained speaker embeddings (2023), ISBN: 978-963-306-912-7
- Snyder, D.: X-Vectors: Robust neural embeddings for speaker recognition. Ph.D.-értékezés, Johns Hopkins University (2020)
- Snyder, D., Garcia-Romero, D., McCree, A., Sell, G., Povey, D., Khudanpur, S.: Spoken language recognition using x-vectors. In: *Odyssey 2018*. pp. 105–111 (2018)
- Sztahó, D., Jenei, A.Z., Valálik, I., Vicsi, K.: The effect of speech fragmentation and audio encodings on automatic parkinson’s disease recognition. *Journal of Biomedical Science and Engineering* 15(1), 6–25 (2022)
- Torzsa, P., Szeifert, L., Dunai, K., Kalabay, L., Novák, M.: A depresszió diagnosztikája és kezelése a családorvosi gyakorlatban. *Orvosi Hetilap*, 150 (36). pp. 1684–1693 (2009), ISSN 0030-6002
- Villalba, J., Chen, N., Snyder, D., Garcia-Romero, D., McCree, A., Sell, G., Borgstrom, J., García-Perera, L.P., Richardson, F., Dehak, R., Torres-Carrasquillo, P.A., Dehak, N.: State-of-the-art speaker recognition with neural network embeddings in NIST SRE18 and speakers in the wild evaluations. *Computer Speech Language* 60, 101026 (2020)
- WHO: Depressive disorder. <https://www.who.int/news-room/fact-sheets/detail/depression> (2023), [Utolsó megtekintés 2025.01.10]
- Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research* 5, 1205–1224 (2004)

# Magyar nyelvű beszédleiratózó tanítása sok (tíz)ezer óra beszéddel

Dobsinszki Gergely<sup>1,2</sup>, Kádár Máté Soma<sup>1,2</sup>, Fegyó Tibor<sup>1,2</sup>, Mády Katalin<sup>3</sup>,  
Mihajlik Péter<sup>2,3</sup>

<sup>1</sup>SpeechTex Kft.

<sup>2</sup>Budapesti Műszaki és Gazdaságtudományi Egyetem,  
Távközlési és Mesterséges Intelligencia Tanszék

<sup>3</sup>HUN-REN Nyelvtudományi Kutatóközpont  
{dobsinszki, kadar, fegyo, mihajlik}@tmit.bme.hu, mady@nytud.hu

**Kivonat** Jelen tanulmány célja a magyar nyelvű beszédfelismerés pontosságának növelése nagy mennyiségű tanítóanyag és önfelügyelt tanulás (Self-Supervised Learning, SSL) módszerének alkalmazásával. Kísérleteink során kisebb, nyilvános és nagyobb, nem publikus adathalmazokon is teszteltük a önfelügyelt előtanítás hatékonyságát. Az SSL technika alkalmazása kis méretű, magyar nyelvű tanító adathalmazok esetén látványos teljesítményjavulást eredményezett a beszédfelismerő modellek pontosságában. A nagy, több ezer órás adathalmazon történő finomhangolás esetén is felgyorsította a tanulási konvergenciát, azonban a jelenlegi eredmények alapján nem múlja felül az angol nyelvű felügyelt előtanítással elért pontosságot. Mindemellett a minden korábbinál nagyobb, tisztán magyar nyelvű halmazzal felügyelten finomhangolt modellek több független kiértékelő halmazon is minden korábbit megelőző pontosságot értek el.

**Kulcsszavak:** mélytanulás, akusztikus modellezés, ASR, SSL

## 1. Bevezetés

Korábbi kutatások bizonyítják, hogy a neurális hálók tanításánál a konvergálási sebességet javíthatja, ha nem véletlen súlyokkal kezdünk egy tanítást, hanem előtanított (pretrained) súlyokból indulunk ki [Cho és mtsai \(2020\)](#). Nincs ez másképp az automatikus beszédfelismerés (ASR, Automatic Speech Recognition) területén sem, ahol egy a célnyelvtől eltérő nyelvre tanított háló súlyai is jó kiindulási állapotot jelenthetnek [Huang és mtsai \(2020\)](#).

Az egyre nagyobb méretű neurális modellek egyre nagyobb adatigényének kiszolgálása nem lehetséges drága és lassan előállítható manuális(an ellenőrzött) címkéssel. Erre a problémára jelenthet megoldást az önfelügyelt tanítás (SSL, Self-Supervised Learning), aminek célja kimeneti címkék használata nélkül olyan kiindulási modell/súlyok előállítása, melyek képesek magas fokú tudásreprezentációra és így egy relatíve sekély ráépülő réteggel együtt történő finomhangolás révén lerövidíti a tanulást és javítja a végeredményeket az adott konkrét (pl. beszéd-szöveg átalakítási) feladatra [Yang és mtsai \(2022\)](#), [Lee és mtsai \(2024\)](#).

Munkánk során az volt a célunk, hogy megvizsgáljuk, magyar nyelvű beszédfelismerésnél milyen mértékben segíthet a célzottan magyar nyelvű hanganyagokon végzett önfelügyelt előtanítás. Nagyméretű, közel húszezer órányi magyar nyelvű hanganyagot tanítottunk SSL akusztikus modelleket, majd különféle jellegű és méretű, leiratozott hanganyagokon finomhangoltuk és nyilvános adatokon értékeltük ki őket. Eredményeinket összehasonlítva más, magyar nyelvre elérhető nyílt modellekkel elértéssel, láthatóan általánosabban használható, esetenként lényegesen magasabb beszédfelismerési pontosságot sikerült elérni.

## 2. Adatok

Első lépésként audio adatok gyűjtésére volt szükség az SSL tanításhoz. Ennél a módszernél a felügyelt tanításhoz képest nagyságrendekkel több adatra van szükség. Az adatok volumenéből adódóan fontos szempont lett a hangfájlok tömörítésének kiválasztása. Az adatok könnyebb kezelése végett tarred<sup>1</sup> gyűjteményeket hoztunk létre az egyes fájlokból. Mivel ezek a gyűjtemények több fájlt foglalnak egyszerre magukba, így kevesebb fájlolvasásra van szükség, optimálisabb tanítási folyamatot biztosítva.

### 2.1. SSL

Az önfelügyelt tanításhoz tehát nagy mennyiségű magyar nyelvű beszédatra volt szükség, aminek beszerzése önmagában nem jelentett kihívást, de a kísérletek könnyebb megismételhetőségének és az eredmények jobb hasonlíthatósága érdekében előnyben részesítettük a nyilvánosan elérhető adatbázisokat.

A tanítóadatok nagyját a Voxpopuli V2 Wang és mtsai (2021) halmaz tette ki. Ez a Meta/ Facebook Research által karbantartott halmaz több nemzet különböző nyelvű európai parlamenti felszólalását tartalmazza. Hosszabb (több perces) felvételek is akadnak, de mivel az ASR rendszerek tanításához rövidebb hangszegletekre van szükség, feldaraboltuk a hanganyagokat 20 másodpercnél nem hosszabb egységekre. Többnyire automatikusan detektált, csendes, beszédet nem tartalmazó részeknél igyekeztünk a vágásokat megtenni. Továbbá, a hosszabb szüneteket kivágva csökkentettük a teljes halmaz hosszát. A végső halmaz 17.470 (tizenhétezer-négyszázhetven) órányi magyar nyelvű felvételt tesz ki.

Egy relatíve kisebb, "házon belüli" (In House, IH) halmazt is használtunk az önfelügyelt tanításainkhoz. Az IH halmaz főképpen rádiós beszélgetőműsorokat tartalmaz. Előkészítésénél a Voxpopuli halmazhoz hasonlóan jártunk el, csend részeknél darabolva, vágva, így összesen 3.36 ezer órányi anyagot kaptunk. Bár terjedelemben lényegesen rövidebb a Facebook Research által rendszerezett halmaznál, a beszéd stílusát és jellegét tekintve más területhez tartozik, így jelentős hozzáadott értéket képviselt az SSL tanítási folyamat során, ahol a Voxpopuli + IH = SSL halmazon történt minden tanítás.

<sup>1</sup> Tarred adathalmazok - <https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/datasets.html#tarred-datasets>



## 2.2. Mozilla Common Voice (CV-16)

Finomhangolás és kiértékelési kísérleteinkhez a szabadon hozzáférhető adatok közül a Mozilla Common Voice magyar nyelvű, 16.1 verziójú részhalmazát használtuk (CV-16) [Ardila és mtsai \(2019\)](#).

Az adott verzió összesen 92 órányi rögzített, ellenőrzött adatmennyiséget tartalmaz, melyből a tanító, validációs és teszt adathalmaz mérete órában kifejezve rendre 52.5, 16.8, 17.7.

## 2.3. BEA

A további finomhangolási kísérletekhez olyan adatbázis kerestünk, amiben társalgási beszéd található. A BEA (BEszélt nyelvi Adatbázis) [Gósy \(2013\)](#); [Neuberger és mtsai \(2014\)](#) a Nyelvtudományi Kutatóközpont által gyűjtött és karbantartott adathalmaz, mely spontán (monologikus és dialogikus), valamint olvasott beszédet is tartalmaz több egyedi beszélőtől. A BEÁ-n belül a BEA-Base részhalmazt [Mihajlik és mtsai \(2023\)](#) használtuk, ami kifejezetten beszédfelismerési modellek kiértékelése céljából készült. Az adathalmaz kutatási célokra ingyenesen hozzáférhető.

Az általunk tanításra használt (train-114) halmazának hossza 68 óra, validációs halmaza (dev-spont) 3.8, teszt halmaza (eval-spont) pedig 4.75 óra hosszú.

## 2.4. BNC

Az általunk BNC-nek (Broadcast News and Conversations) nevezett adatgyűjtemény egy privát leiratozott, több ezer órányi, magyar nyelvű televíziós műsorok hangsávját tartalmazó halmaz. Túlnyomó többségben többrésztvevős beszélgetéseket tartalmaznak a felvételek, kisebb részben pedig olvasott híreket vagy beszédeket. Felhasználása előtt a hanganyagok darabolására és normalizálására volt szükség, így biztosítva a konzisztenciát és összehasonlíthatóságot a többi tanító és kiértékelő halmazokkal.

A BNC esetén az általunk létrehozott tanítóadat hossza 2703 óra, validációs halmaza 113 óra és teszt halmaza 41 órányi hanganyagot tartalmaz.

## 3. Kísérletek

Kísérleteink fókuszában az állt, hogy hogyan tudjuk a nagyméretű beszédatokat a legjobban felhasználni beszédfelismerő tanításához. A nagy adatbázisokkal való manipuláció természetesen nehézkes, ezért a hiperparamétereket a kis-méretű CV-16 és BEA adatokon kíséreltük meg beállítani/optimalizálni. Ez esetekben tudatosan kerültük a kiértékelő halmazok használatát, azokat későbbre, a nagy (BNC) halmazon történő tanítás kiértékelésére tartogattuk.

### 3.1. Kísérleti környezet

Minden kísérletet az Nvidia NeMo Toolkit 1.23-as verziójával végeztünk [Harper és mtsai \(2023\)](#). A környezet több modellarchitektúrát is támogat, multi-gpu és multi-node tanítást tesz lehetővé. SSL tanítás is konfigurálható a segítségével.

A NeMo Toolkit támogatja a SLURM ütemező rendszert, ami segítségével az SSL tanítások történtek a Komondor szuperszámítógépen.

### 3.2. Modellek

A neurális architektúra kiválasztásánál arra ügyeltünk, hogy legalább egy angol nyelvű előtanított súlymátrix elérhető legyen. Ezen súlyokat felhasználva tudtunk olyan tanításokat végezni melyekhez az SSL alapú kísérleteinket hasonlítottuk. Önfelügyelt tanításnál először egy általános modellt készítettünk címkézetlen adatok segítségével. Ezen modelltől kiindulva pedig beszédfelismerési feladatra finomhangoltuk a modellt különböző annotált adatokat felhasználva.

A kutatásainkhoz egy konkrét akusztikus modell architektúrát kerestünk. Mivel az SSL tanítás egy rendkívül számításigényes folyamat, igyekeztünk relatíve kis paraméterszámú, gyorsan tanítható modellt kiválasztani, szem előtt tartva a megfelelően magas pontosság mellett a lehetséges ipari felhasználási szempontokat is (véges GPU memória, számítási kapacitások, stb.). Választásunk így a NeMo toolkit-ben [Harper és mtsai \(2023\)](#) kényelmesen használható *FastConformer Large (CTC)*<sup>2</sup>-re esett, melynek paraméterszáma kb. 121 millió [Rekesh és mtsai \(2023\)](#). A Conformer architektúra [Gulati és mtsai \(2020\)](#) lényegében egy transformer encoder [Vaswani és mtsai \(2017\)](#), amiben a szokásos attention és előrecsatolt (fully connected) réteg mellett egy 1D konvolúciós réteg is megtalálható blokkonként a lokális kontextus hangsúlyosabb figyelembe vétele érdekében. Ez az architektúra talán a legnépszerűbb az end-to-end mélyneuronháló beszédfelismerési alkalmazásában. A FastConformer csak annyiban tér el a klasszikus Conformer hálózattól, hogy hagyományos 1D konvolúció helyett ún. "time-channel separable" konvolúciót [Kriman és mtsai \(2020\)](#) alkalmaz, ami voltaképpen egy paraméter- és számíthatékony közelítő változata az utóbbinak.

A FastConformer hálózat kimenete a skálázható szótárméretű szótöredék (subwords unit) valószínűségeloszlása 80ms-onként, erre került a végső CTC [Graves és mtsai \(2006\)](#) réteg/költségfüggvény. A finomhangoláshoz használt hanganyagok szöveges leírataák tokenizálására a SentencePiece [Kudo és Richardson \(2018\)](#) eljárást alkalmaztuk alapértelmezett 1024-es szótármérettel.

### 3.3. SSL tanítás

AZ SSL tanításoknál Contrastive [Baevski és mtsai \(2020\)](#) és Masked Language Modeling (MLM) [Hsu és mtsai \(2021\)](#) hibafüggvényekkel kísérleteztünk. A kontrasztív megközelítések esetén a modellek úgy tanulnak, hogy különbséget tesznek

<sup>2</sup> [https://github.com/NVIDIA/NeMo/blob/main/examples/asr/conf/fastconformer/fastconformer\\_ctc\\_bpe.yaml](https://github.com/NVIDIA/NeMo/blob/main/examples/asr/conf/fastconformer/fastconformer_ctc_bpe.yaml)

a valódi és a zavaró látens reprezentációk között, míg az MLM esetén a szöveges BERT modellek [Devlin és mtsai \(2019\)](#) tanításához hasonlóan a kimaszkolt audio részletet az ismert (nem maszkolt) kontextusból kell megbecsülnie a modellnek.

Az SSL tanítás során a NeMo adott verziójának alapértelmezett beállításait használtuk, csak azon paramétereket tüntetjük fel a továbbiakban, melyek ettől eltértek.

Az SSL tanításhoz 2 darab node-ot használtunk, összesen 16 darab A100 SXM4 GPU-val, GPU-nként 40 GB VRAM-al. Node-onként 64 CPU core és 256 GB RAM állt rendelkezésre. Egy epoch ideje nagyjából 1 óra volt. Tanításainknál mindenhol az AdamW optimizert [Kingma és Ba \(2014\)](#), [Loshchilov és Hutter \(2019\)](#) alkalmaztuk.

## 4. Eredmények

A következőkben a 2. fejezetben felsorolt halmazokon végzett SSL módon előtanított súlyokból kiindult tanítások eredményeit közöljük, melyek mellé az NVIDIA által előtanított angol nyelvű akusztikus modellel inicializált "cross-language transfer learning" eredményeket is megadjuk referenciának. A magyar nyelvű SSL modelleket és az angol nyelvű (felügyelten előtanított) modelleket minden esetben valamely magyar nyelvű adatbázison tanítottuk tovább (finomhangoltuk), mely során a validációs halmazt használtuk a tanítás monitorozására. A végső kiértékelést az előzőektől független tesztadatokon végeztük.

### 4.1. SSL + CV-16

Különböző SSL tanított modellek kipróbálása történt ebben a környezetben. A kezdeti súlyokat leszámítva, minden tanítási paraméter fix, megegyezik a NeMo FastConformer-Large tanítási receptjével. A modellek a CV-16 tanító halmazán tanultak (tovább), 100 epoch-on keresztül,  $2 * 10^{-4}$ -es LR, 256-os tokenizer mérettel.

Experiment Name	SSL Loss Type	SSL Epoch	Num Negatives	Val WER
en_weights	NA	NA	NA	16.49 %
ssl_weights_1	Contrastive	70	40	15.56 %
ssl_weights_2	Contrastive & MLM	100	40	13.20 %
ssl_weights_3	Contrastive & MLM	100	100	13.43 %

1. táblázat. Angol nyelvű és különböző SSL modellek finomhangolása CV-16-on.

### 4.2. SSL + BEA

Ebben a kísérletsorozatban már csak egyetlen SSL modellel dolgoztunk. Elsősorban learning rate (LR) hangolására fókuszáltunk. CV-16-os kísérletekkel ellentétben itt nem volt elég a kiindulási SSL modell-re lecserélni a kezdeti angol nyelvű

előtanított súlyokat jobb eredmények eléréséhez: ahogy a 2. táblázat mutatja, a tanulási ráta görbéjének alakítása jelentős hatással volt a validációs halmazon mért hibára.

Initial weights	Initial LR	Minimum LR	Warmup Steps	Val WER
en_weights	5e-4	5e-8	12500	17.11 %
ssl_weights_3	5e-4	5e-6	1000	17.47 %
ssl_weights_3	7e-4	5e-8	1000	18.56 %
ssl_weights_3	4e-4	5e-8	2000	16.46 %
ssl_weights_3	3e-4	5e-8	2000	15.99 %
ssl_weights_3	2e-4	5e-8	2000	15.32 %
ssl_weights_3	1e-4	5e-8	2000	14.81 %
ssl_weights_3	9e-5	5e-8	2000	14.53 %
ssl_weights_3	8e-5	5e-8	2000	14.76 %
en_weights	9e-5	5e-8	2000	26.73 %

2. táblázat. Különböző SSL modellek finomhangolása BEA-n és kiértékelésük BEA dev\_spont-on.

Az 2. táblázatban látható `ssl_weights_3` az 1. táblázat azonos nevű modell súlyait jelöli. A választás azért erre az előtanított modellre esett, mert stabilabban voltak alacsonyok a validációs WER értékek a CV-16-on végzett finomhangolás során (a táblázatban csak a végső értékek vannak feltüntetve) mint a többi próbálkozásnál.

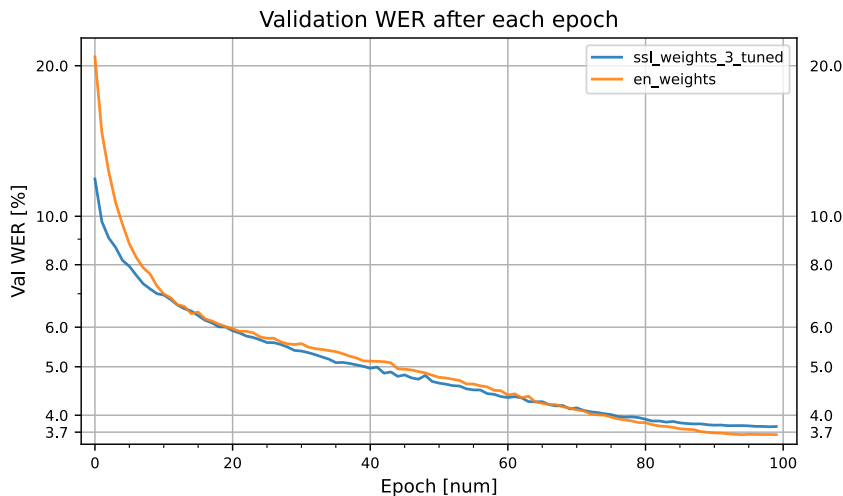
### 4.3. SSL + BNC

A harmadik, egyben legfontosabb kísérletként nagy mennyiségű leiratozott adattal, a BNC-el tanítottuk tovább a kiválasztott előtanított modellt.

A BNC-n történő finomhangolást a 20.000 órányi magyar nyelvű hanganyagon önfelügyelten előtanult, valamint az angol nyelven az NVIDIA által felügyelten előtanított modelleken is elvégeztük. Noha az előzetes tesztek során – lásd az 1. és 2. táblázatot – az angol súlyokkal történő inicializálást mindig felülmúlta a magyar SSL inicializálás, ezúttal más eredményt kaptunk.

Bár az SSL előtanítás a konvergenciát gyorsította, de végeredményben nem hozott javulást az angol nyelvű előtanuláshoz képest. Az 1. ábrán láthatóak a hiperparaméter-hangolt SSL alapú és angol nyelvű hálóból kiindult tanítások eredményei. Látható, hogy az SSL alapú gyorsabban konvergált, "jobban illeszkedett" a tanítási adathalmazra de végeredményben marginálisan gyengébb eredményt ért el a kiértékelő halmazon is (3. táblázat). Ennek oka a BNC adatokra szabott hiperparaméter-optimalizáció hiánya is lehet.

A BNC adatokkal finomhangolt modelleket összehasonlítottuk más, magyar nyelvű neurális háló alapú leiratozó rendszerekkel. Az egyik ilyen a BEAST2, amely limitált mennyiségű adaton, a BEA-n tanult, a BEA teszt halmazán legjobb eredményt elérve. A másik rendszer a szabadon hozzáférhető OpenAI által



1. ábra. BNC adatokon történő tanítás, angol (en\_weights) és SSL módon tanított súlyokból (ssl\_wights\_3\_tuned) kiindulva. Látható, hogy az SSL tanított gyorsabban konvergál a tanítás elején, végül az angol, felügyelten tanított súlyokkal éri el a pontosabb eredményt.

Initial Weights	Val WER (%)	Test WER (%)
en_weights	3.66	3.83
ssl_weights_3	3.75	3.96

3. táblázat. Az magyar nyelvű adatokkal tanított SSL és az angol nyelven tanított súlyokból indított finomhangolás validációs és teszt eredményei az BNC halmazon.

Model	LM	CV-16 WER (%)	FLEURS WER (%)	BEA WER (%)
BEAST2	Van	19.53	25.63	<b>10.98</b>
Whisper large-v3	Implicit	13.4	12.9	21.70
FastConformer HU-SSL + BNC (saját)	Nincs	7.43	9.98	13.06
FastConformer EN + BNC (saját)	Nincs	<b>7.32</b>	<b>9.74</b>	12.70

4. táblázat. A BEAST2 [Kádár és mtsai \(2023\)](#) rendszer, a Whisper large-v3 [Radford és mtsai \(2022\)](#) és a saját megoldásaink eredményei (magyar SSL és angol előtanított hálókból kiindulva), három, magyar nyelvet is tartalmazó, szabadon vagy kutatási célra hozzáférhető kiértékelő halmazon.

fejlesztett Whisper architektúrának 3. iterációja. A Whisper több nyelven képes leiratozni, igény szerint az elhangzott hanganyag alapján nyelvdetektálást is végez, és fordítani is képes. A különböző megoldások pontosságát a CV-16, FLEURS [Conneau és mtsai \(2023\)](#) és BEA halmazokon vizsgáltuk, ezek a 4. táblázatban láthatóak.

Mindkét megoldásunk messze felülmúlja a másik kettő megvalósítást, teszi ezt nyelvmódel (LM: Language Model) használata, azaz előzetes írott nyelvi ismeretek nélkül. Mindemellett a legkisebb paraméterszámú architektúra a három megoldás közül. Egyedül a BEA halmazon teljesít rosszabbul a BEAST2-es modellnél, ez magyarázható többek közt azzal, hogy a tanításhoz nem lett felhasználva a BEA tanítóhalmaz, így "out of domain" ez a teszhalmaz számára. Ennek ellenére messze felülmúlja a Whisper large-v3 által elért eredményeket ezen a halmazon, mely számára szintén ismeretlen a BEA halmaz.

## 5. Összefoglalás

Az eredményeink nyomán megállapítható, hogy kis méretű adathalmazokkal való továbbtanításkor az SSL előtanítás jelentős javulást tud elérni a az idegen nyelven előtanított súlyokhoz képest. Ugyanakkor az általunk vizsgált modellnél ez az előny elveszik, amikor nagy mennyiségű leiratozott anyaggal történik a továbbtanítás.

Az önfelügyelt tanításhoz 20 ezer óránál is kevesebb anyagot használtunk, ami relatíve kevésnek tekinthető az ilyen jellegű megközelítéseknél. A jövőben szeretnénk ezt a halmazt bővíteni, és egy ilyen modellt finomhangolni felügyelten.

Szeretnénk továbbá nagyobb méretű modellekkel megvizsgálni az önfelügyelt tanítást. Ezek a modellek több VRAM-ot igényelnek, tovább tanulnak, így a jelenlegi munkánk kereteibe nem fértek bele. Viszont kutatások igazolják, hogy az SSL tanítás nagyobb méretű, nagy paraméterszámú modelleknél tudja igazán kifejteni hatását, így szeretnénk a FastConformer család egyik nagyobb modelljét megvizsgálni, a FastConformer XL-t, ami nagyjából 5-ször annyi paramétert tartalmaz, mint az általunk vizsgált architektúra.

## Köszönetnyilvánítás

A munka részben az NKFI Alap által támogatott NKFIH K143075 és K135038, valamint az NKFIH-828-2/2021 (MILAB) projektek támogatásával valósult meg. Köszönettel tartozunk továbbá a KIFÜ-nek (Kormányzati Informatikai Fejlesztési Ügynökség <sup>3</sup>), amiért hozzáférést biztosítottak a Magyarországon található Komondor szuperszámítógéphez, illetve az NVIDIA Akadémiai Hardver Támogatásnak (NVIDIA Academic Hardware Grant).

<sup>3</sup> <https://ror.org/01s0v4q65>

## Irodalomjegyzék

- Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F.M., Weber, G.: Common voice: A massively-multilingual speech corpus. arXiv preprint arXiv:1912.06670 (2019)
- Baevski, A., Zhou, H., Mohamed, A., Auli, M.: Wav2vec 2.0: A framework for self-supervised learning of speech representations (Oct 2020), <https://arxiv.org/abs/2006.11477>
- Cho, E., Li, J., Kim, S., Jinyu, L.: Cross-language transfer learning and domain adaptation for end-to-end automatic speech recognition. In: Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 7494–7498. IEEE (2020)
- Conneau, A., Ma, M., Khanuja, S., Zhang, Y., Axelrod, V., Dalmia, S., Riesa, J., Rivera, C., Bapna, A.: Fleurs: Few-shot learning evaluation of universal representations of speech. In: 2022 IEEE Spoken Language Technology Workshop (SLT). pp. 798–805. IEEE (2023)
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019), <https://arxiv.org/abs/1810.04805>
- Gósy, M.: Bea—a multifunctional Hungarian spoken language database. *Phonetica* 105, 50–61 (2013)
- Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning. pp. 369–376 (2006)
- Gulati, A., Qin, J., Chiu, C.C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., és mtsai: Conformer: Convolution-augmented transformer for speech recognition. arXiv preprint arXiv:2005.08100 (2020)
- Harper, E., Majumdar, S., Kuchaiev, O., Li, J., Zhang, Y., Bakhturina, E., Noroozi, V., Subramanian, S., Koluguri, N., Huang, J., Jia, F., Balam, J., Yang, X., Livne, M., Dong, Y., Naren, S., Ginsburg, B.: Nemo: a toolkit for conversational ai and large language models (2023), <https://nvidia.github.io/NeMo/>
- Hsu, W.N., Bolte, B., Tsai, Y.H.H., Lakhotia, K., Salakhutdinov, R., Mohamed, A.: Hubert: Self-supervised speech representation learning by masked prediction of hidden units (2021), <https://arxiv.org/abs/2106.07447>
- Huang, J., Kuchaiev, O., O’Neill, P., Lavrukhin, V., Li, J., Flores, A., Kucsko, G., Ginsburg, B.: Cross-language transfer learning, continuous learning, and domain adaptation for end-to-end automatic speech recognition (2020), <https://arxiv.org/abs/2005.04290>
- Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
- Kriman, S., Beliaev, S., Ginsburg, B., Huang, J., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Zhang, Y.: Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6124–6128. IEEE (2020)

- Kudo, T., Richardson, J.: Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint arXiv:1808.06226 (2018)
- Kádár, M.S., Dobsinszki, G., Mády, K., Mihajlik, P.: „feeding the beast” – a bea speech transcriber továbbfejlesztése és integrálása neurális nyelvmodellel. In: XIX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 135–143 (2023)
- Lee, Y., Willette, J.R., Kim, J., Hwang, S.J.: Visualizing the loss landscape of self-supervised vision transformer (May 2024), <https://arxiv.org/abs/2405.18042>
- Loshchilov, I., Hutter, F.: Decoupled weight decay regularization (2019), <https://arxiv.org/abs/1711.05101>
- Mihajlik, P., Gráczai, T.E., Kohári, A., Tarján, B., Balog, A., Mády, K.: A bea továbbfejlesztése és alkalmazása kontrasztív gépi beszédfelismerési kísérletekre. In: Általános Nyelvészeti Tanulmányok XXXIV. Akadémiai Kiadó (2023)
- Neuberger, T., Gyarmathy, D., Gráczai, T.E., Horváth, V., Gósy, M., Beke, A.: Development of a large spontaneous speech database of agglutinative Hungarian language. In: International Conference on Text, Speech, and Dialogue. pp. 424–431. Springer, Cham (2014)
- Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust speech recognition via large-scale weak supervision (Dec 2022), <https://arxiv.org/abs/2212.04356>
- Rekesh, D., Koluguri, N.R., Krizan, S., Majumdar, S., Noroozi, V., Huang, H., Hrinchuk, O., Puvvada, K., Kumar, A., Balam, J., Ginsburg, B.: Fast conformer with linearly scalable attention for efficient speech recognition (2023), <https://arxiv.org/abs/2305.05084>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* 30 (2017)
- Wang, C., Riviere, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., Dupoux, E.: VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. In: Zong, C., Xia, F., Li, W., Navigli, R. (szerk.) *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pp. 993–1003. Association for Computational Linguistics, Online (Aug 2021), <https://aclanthology.org/2021.acl-long.80>
- Yang, H., Zhao, J., Haffari, G., Shareghi, E.: Self-supervised rewiring of pre-trained speech encoders: towards faster fine-tuning with less labels in speech processing. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (szerk.) *Findings of the Association for Computational Linguistics: EMNLP 2022*. pp. 1952–1959. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022), <https://aclanthology.org/2022.findings-emnlp.141>



# RAG módszerek implementálásának kihívásai egy célorientált chatbot rendszer kontextusában

Csáki Csaba<sup>1</sup> és Vándor Péter<sup>1</sup>

<sup>1</sup> Budapesti Corvinus Egyetem, Fővám tér 8,  
H-1093 Budapest, Magyarország  
peter.vandor@stud.uni-corvinus.hu  
csaki.csaba@uni-corvinus.hu

**Kivonat:** A Retrieval Augmented Generation (RAG) megoldás fókuszált tartalmú adatok és szövegek bevonásával javítja a nagy nyelvi modellek (LLM) által generált válaszok minőségét, és kiküszöbölheti hibáikat. A RAG egyre népszerűbb alkalmazása azonban felszínre hozott számos technikai kihívást, melyek megoldása nem mindig egyértelmű: a bemeneti adatforrások összetettek lehetnek, a válaszok pontossága felé magasak az elvárások, illetve meg kell tudni mondani hol található a forrásban a legjobb válasz egy kérdésre. Ezekre a feladatokra számos lehetséges alternatív technika közül lehet választani. A kutatás során egy célorientált chatbot fejlesztése keretében vizsgáltuk a RAG folyamat egyes lépéseit, és kipróbáltunk többféle szöveg beviteli (OCR, LLM vision), szöveg feldarabolási (rekurzív, ágensalapú) és válasz keresési módszert (hasonlósági vektor, BM25). Új feldolgozási módszereket vezetünk be a fentiek kombinálásával, melyek javítják az eredmény minőségét. A cikkben rámutatunk, hogy akár nyílt modellekre is támaszkodhatunk, mivel egyes esetekben a szabályalapú, standard feldolgozóknál pontosabb, strukturáltabb eredményt adnak, és zárt társaik a tartalmi szűrések miatt néhány esetben nem végezték el a feladatot.

**Kulcsszavak:** nyíltkérdés-megválaszolás, RAG, LLM, generatív válaszolás

## 1 Bevezetés

A nagy nyelvi modellek (LLM) rohamos terjedése mellett megjelentek olyan megoldások is, melyek e modellek gyengeségeinek kezelését tűzték ki célul. Az egyik legnépszerűbb és gyorsan felkapottá vált technológia a Retrieval Augmented Generation (RAG), mely lehetővé teszi többek között saját, lokális vagy célorientált (angolul 'custom') chatbotok létrehozását (Lewis és mtsai, 2020). A megközelítéssel csökkenthető a költségek, saját tartalmakat lehet kereshetővé tenni, és úgy csökkenthető a hallucináció jelensége, hogy a saját tartalmakat nem kell feltölteni a nyelvi modell szolgáltatójának felhőjébe (Shuster és mtsai, 2021).

Ilyen RAG megoldással kereshetővé tehető saját anyagok, elsősorban például PDF-ben vagy szövegesen tárolt dokumentumok. Mivel a megoldás lokálisan üzemeltethető, így a funkciók is kontrollálhatók, és a keresési adatok (felhasználói promptok és megtalált válaszok leírása és statisztikái) is elérhetőek. Ezek a jellemzők később

felhasználhatók mind a RAG továbbfejlesztésére, mind a tartalom javítására, mind a kérdezők promptolási technikáinak képzésére.

Egy RAG tudásbázisra épített saját chatbot kiépítése során azonban számos technikai kihívással kell szembenézni különösen szakkönyvek, tankönyvek, oktatási jegyzetek, vagy tudományos cikkek feldolgozása esetén, mivel ilyen jellegű dokumentumokra jellemző a nem szöveges elemek rendszeres jelenléte. Egyrészt képek, diagramok, folyamatábrák, szövegdobozok, illetve táblázatok szerepelnek benne, de problémát jelenthet a fejléc és láblécek jelenléte is. Másrészt ezek az elemek meg is szakítják a fő szöveg folytonosságát, így előfordulhat, hogy egy adott szövegrész egy vagy több oldallal később folytatódik. Képek, diagramok vagy akár egész oldalalak gyakran képként szerepelnek a PDF formátumú dokumentumokban, még akkor is, ha tartalmazznak szöveges információt. A RAG technika alkalmazáskor ezek feldolgozása és tudásbázisba történő betöltése komoly kihívást jelent, melyek kezeléséhez speciális technikák bevezetésére van szükség, pedig a képek és táblázatok tartalmának megfelelő integrálása, különösen a szövegrészek összekapcsolása kulcsfontosságú a RAG rendszer pontossága és hatékonysága szempontjából. Emellett egy ilyen megoldás igen költséges is lehet, hiszen a fejlesztési költségek (pl. programozói bérek) mellett mind a feldolgozás mind a lekérdezés során jelentős tétel az LLM API meghívása, ahol a fizetés alapvetően token-alapon történik. Lehetőség van azonban egyrészt különböző piaci modellek alkalmazására, másrészt szóba jöhetnek ingyenesen elérhető megoldások is.

Az itt bemutatott kutatás elsődleges célja az volt, hogy olyan új módszereket és optimalizációkat dolgozzunk ki, melyek úgy javítják a bot teljesítményét, hogy lehetőség szerint nem növelik sem a fejlesztés sem az üzemeltetés költségeit. A kutatásban figyelmet fordítottunk a feldolgozási és lekérdezési folyamatok teljesítményére, de költségeire is, hiszen ezek jelentős hatással vannak a chatbot fenntarthatóságára és alkalmazhatóságára. A forrás feldolgozás során a szöveg feldarabolásának pontosságára összpontosítottunk és az LLM API meghívásának költségeire, míg a lekérdezési folyamat során a válaszok keresésének gyorsaságát és pontosságát értékeltük.

A módszerek kipróbálását, továbbfejlesztését, tesztelését és eredményeik összehasonlítását egy olyan felsőoktatási chatbot kialakítás során végeztük el mely lehetővé teszi jegyzetek, tankönyvek, vagy nagy mennyiségű tanulmányok (elsősorban PDF és szöveges fájlok) feltöltését, és támogatja a hallgatókat az adott tananyagra vonatkozó kérdéseik megválaszolásában. További célkitűzés volt, hogy részletes statisztikákat biztosítson az oktatók számára, hogy a tananyagokat és a teszteredmények alapján továbbfejleszthessék, és támogathassák a személyre szabott tanulást.

A RAG folyamat lépéseinek optimalizálása során egy robosztus, általánosítható megoldás keresése hajtotta a kutatást. E tanulmány a PDF betöltés és feldolgozás (1), a szöveg feldarabolás (2), és a válasz keresés, sorbarendezés (3) során felmerülő kihívásokat és az azokra kidolgozott megoldásokat mutatja be.

## 2 Módszertan

Kutatásunk során PDF formátumú, sok szkennelt képet tartalmazó, nem strukturált dokumentumok feldolgozását vizsgáltuk zárt és nyílt LLM-ek segítségével a következő

feladatok mentén: szövegfoltonosság biztosítása, szemantikailag koherens feldarabolás, vektor adatbázisba töltés és a releváns szövegrészek megtalálása (Q&A).

A klasszikus, szabályalapú PDF olvasók (PyMuPDF) és OCR felismerők (Tesseract) mellett teszteltük különböző méretű és licenszű, vision képességekkel rendelkező nagy nyelvi modellek használhatóságát ezen a területen, valamint ezen eszközök kombinációját figyelembe véve az eredmény pontosságát, költségét és időigényét.

A feldarabolásnál a hagyományos rekurzív karakteralapú elválasztáson kívül kipróbáltuk a szemantikai vágást és a teljesen LLM-ekre bízott ágensalapú szövegfelbontást.

A releváns szövegrészek megtalálását elősegítő technikákat is alkalmaztunk a hallucinációk csökkentése és a minél pontosabb válaszok érdekében. Ilyen módszer a vektor adatbázisban történő szemantikai keresés, a BM25 alapú kulcsszó egyezés, a lekérdezés átfogalmazás (Ma és mtsai, 2020), kontextus beágyazás (Anthropic, 2024b) és a normalizált relevanciapontszám-alapú sorbarendezés (Cormack és mtsai, 2009). A komplex keresést BM25- és beágyazásalapú módszerek kombinálásával valósítottuk meg (Berkecz és mtsai, 2024). A PDF feldolgozáshoz a következő vision modelleket alkalmaztuk: a GPT-4o (OpenAI, 2024), Claude Sonnet 3.5 (Anthropic, 2024a), és Google Gemini Pro 1.5 (Gemini Team, 2024) zárt, illetve a Qwen2 72B (Yang és mtsai, 2024), Llama 3.2 11B, Llama 3.2 90B (Meta, 2024) és MiniCPM-Llama3-v2.5 8B (Yao és mtsai, 2024) nyílt modellek.

A kérdés-felelet módszerek tesztelésénél használtuk a GPT-4o, GPT-4-turbo és GPT-4o-mini (OpenAI, 2024) modelleket, majd a legjobb módszer részletes tesztjénél ezen túl a Claude Sonnet 3.5 (Anthropic, 2024a), Qwen2 72B (Yang és mtsai, 2024), valamint a Llama 3.1 405B (Llama Team, 2024), Mixtral-8x22B-Instruct (Jiang és mtsai, 2024; Mistral AI team, 2024) és Gemma 2-27B (Gemma Team, 2024) csak szöveges bemenetet kezelő modelleket is.

A teljes RAG folyamat tesztelését egy oktatási chatbot fejlesztésének kontextusában végeztük: egy angol nyelvű, 649 oldalas tankönyvet (Laudon, 2021) használtunk PDF formátumban, amelynek 393 oldala szkennelt, teljes oldalt kitöltő képként szerepelt, beleértve az összes olyan oldalt, amelyen ábrák is voltak. A dokumentumban gyakoriak voltak a magyarázó ábrák, diagramok, valamint minden fejezet tartalmazott egy-két, szövegdobozban kiemelt esettanulmányt is. Ilyen dokumentumok esetében kritikus a szöveges tartalom helyes kinyerése és összekapcsolása, hogy biztosítsuk a folytonos és pontos információ visszakeresést a chatbot rendszerekben.

### 3 A RAG folyamat problémái és megoldási technikák

A RAG folyamat során a nem strukturált dokumentumok tudásbázisba történő betöltése számos kihívással jár, amelyek kezelésére speciális technikák bevezetésére van szükség. A fő problémák közé tartozik a nem szöveges elemek feldolgozása, mint például a képek, diagramok, folyamatábrák, illetve táblázatok kezelése. Ezek az elemek gyakran szkennelt formában, képként szerepelnek a PDF formátumú dokumentumokban, még akkor is, ha tartalmaznak szöveges információt. A dokumentumban szereplő diagramok és folyamatábrák esetében a szövegkinyerés, valamint a táblázatok adatainak feldolgozása jelentős kihívás.

További problémát jelent a fejléc- és láblécek kihagyása, valamint a korábban említett szövegfolytonosság biztosítása. Emellett szükséges a szövegbeosztás automatikus felismerése is, azaz a fejezetek és alfejezetek elkülönítése és szemantikai egységben tartása. A szöveg feldolgozása során törekedni kell az oldalszámok pontos tárolására is (esetünkben ez megkönnyíti a hallgatók számára a releváns információ megtalálását a tananyagban).

A fenti kihívások kezelésére négyféle technikai megközelítést próbáltunk ki, amelyek részben vagy egészben megoldást nyújthatnak ezekre a problémákra. Az 1. megközelítés a szabályalapú PDF olvasók használata, mint például a PyPDF2 (Fenniak és mtsai, 2014), Camelot (Camelot Developers, 2021), PyMuPDF és PyMuPDF4llm (McKie és Liu, 2016). A 2. technika az optikai karakterfelismerés (OCR) alkalmazása, amelyhez a Tesseract nevű széles körben alkalmazott szoftvert használtuk (Smith, 2007). A 3. megközelítés a nyelvi modellek (LLM) vision képességeinek kihasználása, amely során a korábban említett modelleket használtuk. Végül a 4. megoldás a különböző technikák kombinálásával keletkezett, amikor PyMuPDF vagy Tesseract által felismert szöveget az LLM-k promptjába illesztettük feltételezve, hogy növeli az eredmény pontosságát.

A fenti eszközök kimeneti formája is alapvetően eltérő. Míg az OCR és a PDF olvasók jellemzően Markdown kimenetet biztosítanak, addig az LLM-alapú megoldások JSON szerkezetbe is képesek betölteni a felismert szövegeket. Természetesen sima szöveget minden technikai megoldás tud, de ez a fajta kimenet mindkét fenti másiktól ismertén kevesebb információt tartalmaz, ezért nem is vettük számításba.

A PyMuPDF4llm kimenetével megegyezően először az LLM-eket is Markdown formátum visszaadására kértük. Ekkor azonban kiderültek a megoldás korlátai: a címek szintjének meghatározása nem mindig volt pontos, konzisztens és a félbeszakított szövegek megjelölése is erősen hiányos volt. A Markdown elsősorban egy megjelenítési formátum, amivel nehéz algoritmikusan tovább dolgozni. A különböző szekciók LLM általi megjelölése (tagging) túl sok hibát tartalmazott.

```
"page": {
  "header": "Part Three Key...",
  "footer": "",
  "page_number": "468",
  "text_block": [
    {
      "type": "text",
      "title": "Continue from previous page",
      "body_text": "that every photo of..."
    },
    {
      "type": "text",
      "title": "Intelligent Agents",
      "body_text": "Intelligent agents are ..."
    }
  ]
}
```

1. ábra: JSON struktúra (saját munka)

A JSON struktúrát (1. ábra) úgy alakítottuk ki, hogy minden tartalmilag elkülöníthető elemét leírhatóvá tegye az oldalnak, és a későbbi algoritmikus feldolgozás során

biztosítani tudja a több oldalra szétszórt, de egy fejezethez tartozó szövegrészek összekötését. Az oldal (page) elem pontosan 1 oldal tartalmát írja, amelyhez 1 fejléc, lábléc és oldalszám információ tartozhat. Az oldalon belül kép, táblázat és törzsszöveg típusú szövegblokkok követik egymást. Minden blokk rendelkezik címmel, amely a fejezet címe törzsszöveg, a kép vagy táblázat címe a megfelelő esetén. Ha a szövegblokk az előző oldalakon kezdődő fejezet folytatása, akkor kezdetben egy speciális címet kap, ami jelzi ezt a tulajdonságát, és később az összekötésnél kapja meg a végleges címét.

#### 4 PDF feldolgozó módszerek és modell-specifikus kihívások

Összehasonlítva a hagyományos szabályalapú és OCR-alapú módszereket a szkennelt oldalak feldolgozására a nagy nyelvi modellek vision képességeivel, azt tapasztaltuk, hogy az egyértelmű nyertes az LLM lett, mivel eredményei (1) pontosabbak, (2) jobban követik a szöveg szerkezetét, és (3) kizárólag az LLM képes részletes leírásokat nyújtani folyamatábrákról és diagramokról. Minden OCR program esetén fellépnek karakter tévesztések, még a fizetős rendszereknél is (Pethő és mtsai, 2024). A klasszikus OCR megoldás sok esetben az ábrákon explicit szereplő szöveget sem tökéletesen nyerte ki, továbbá az oldalon belüli sorrendet sem követte több esetben. Az implicit információk, mint például a folyamatábrákon szereplő nyilak iránya és hogy ezek mely felsorolt elemeket kötik össze, teljes egészében elvesztek a tisztán OCR megoldással.

1. táblázat: Az LLM-ek és a Tesseract szoftver OCR feldolgozási eredményei

OCR feldolgozási kimenetek összehasonlítása és teljesítménye	GPT-4o	Claude Sonnet 3.5	Google Gemini PRO 1.5	Tesseract OCR	Qwen2 72B	Llama 3.2 11B Vision	Llama 3.2 90B Vision	Mini CPM-Llama3
Hiányzó szavak	0	0	13	8	1	8	8	316
Rosszul felismert szavak	0	0	13	11	1	2	1	37
Egyező szavak	776	776	763	768	775	768	768	460
Összes változás	0	0	26	19	2	10	9	353
<b>Pontosság (%)</b>	<b>100</b>	<b>100</b>	<b>98,32</b>	<b>98,97</b>	<b>99,87</b>	<b>98,96</b>	<b>98,96</b>	<b>59,28</b>
<b>Koszinusz hasonlóság (%)</b>	<b>100</b>	<b>100</b>	<b>98,32</b>	<b>98,78</b>	<b>99,87</b>	<b>99,35</b>	<b>99,42</b>	<b>72,27</b>

Az 1. táblázat mutatja az OCR feldolgozási eredmények összehasonlítását a különböző nyílt és zárt LLM modellek esetében, valamint a Tesseract programmal összevetést, amely iparági szabványnak számít az OCR területén. A mérés a legtöbb feldolgozási hibát okozó 463. oldalon történt. Látható, hogy a GPT-4o és a Claude Sonnet 3.5 modellek teljesítettek 100%-os pontossággal. A többi esetben több szó hiányzott az eredményből, vagy az eredeti szövegben nem szereplő új, rosszul felismert szavakat találtunk elszórvva a bekezdésekben. A legrosszabbul a Mini CPM Llama3 modell teljesített (2. ábra), amely egy kis, 8 milliárd paraméterrel rendelkező és ezért lokálisan futtatott nyelvi modell volt (Pondhouse Data, 2024). Más kisméretű modelleket is

kipróbáltunk, de az eredmények azt mutatták, hogy a kevés paraméter jelentősen rontja a felismerés képességét. Ezen kívül a túl hosszú vagy komplex prompt megzavarja a kisebb modelleket (pl. Tesseract OCR szöveg hozzáadása a prompthoz), és önismétlésbe, végtelen ciklusba kezdenek a kimeneten. A zárt modellek következetesen figyelmen kívül hagyták a segítségképpen elküldött, már felismert szöveget, és így annak csak költség- és idő növelő hatását érzékeltük.

Az OCR tesztelésnél csak a vision képességekkel rendelkező LLM-k használhatók. A Tesseract eredményéhez képest fontos megjegyezni, hogy az általunk elvárt JSON sémának megfelelően felbontott eredményt kizárólag az LLM-ek tudták szolgáltatni, hasonlóan a kép-, folyamatábra- és diagramleírások elkészítéséhez. Ezért bár a táblázatban a Tesseract pontosnak tűnik, de nem tudja a későbbi feldolgozáshoz szükséges alapstruktúrát kitölteni. A Mini CPM számára is megoldhatatlan akadályt jelentett a JSON séma és a Llama 3.2 kis, 11 milliárdos modellje hiába volt pontos, mivel a JSON kimenetbe egy a sémában nem szereplő részt illesztett, és megismételte benne a szöveg egyik bekezdést. Ez utóbbi algoritmikusan a későbbi feldolgozás során nem kezelhető.

that every photo of a Facebook user is connected to that person's Facebook account. **Count**. Computer vision systems are also used in autonomous vehicles such as drones and self-driving cars (see the chapter-ending **case case study**), industrial machine vision systems (e.g., inspecting bottles), military applications, and robotic tools. In **health care healthcare**, computer vision technology is being used for tumor evaluations . ... using **computerized computed** tomography (CT) scans after treatment. How a tumor reacts to **treatment treatments**

**2. ábra:** OCR feldolgozó eredményének összehasonlítása a Mini CPM Llama3 8B modell hibái pirossal, a GPT-4o helyesen felismert szavai zölddel (saját munka)

Az LLM-ek hajlamosak a magyarázat adásra, azaz a RAG válaszokat rendszeresen további szövegrészekkel egészítik ki – ami nehézséget okoz, egyrészt a lehetséges hallucináció, másrészt a válasz pontosságának romlása, végül a potenciálisan felesleges szövegrészek miatt. Ezeket a hőmérséklet (temperature) paraméter 0-ra állításával és célzott, felszólító módú utasítások megfogalmazásával a promptban kiküszöböltük.

A Claude és Gemini modellek esetén teljesen más irányból merült fel új probléma, amely a nem generatív rendszerek esetén elképzelhetetlen. Mindkét modell még 1-1 oldal feldolgozáskor jól teljesített, de 4-5 oldal elküldése esetén leállította a feldolgozást arra hivatkozva, hogy a modelleket nem tartalmak szó szerinti visszaadására (Gemini – recitation, Claude – regurgitation) készítették. Vélhetően ez összeköthető azokkal a bírósági perekkel, amelyeket egyes sajtótermékek, tartalomgyártó vállalatok indítottak a nagy nyelvi modelleket készítő óriáscégek ellen. Az Anthropic hivatalos bejegyzése szerint az intézkedés célja, hogy megakadályozzák, hogy Claude már létező anyagokat másoljon vagy visszamondjon. Az Anthropic felhasználási feltételei és szabályzatai tiltják a szolgáltatás olyan módon történő használatát, amely sérti a szellemi tulajdont vagy egyéb jogokat (Anthropic Privacy Center, 2024).

A feldolgozás sebessége a Tesseract és a lokálisan (GeForce 4090 24GB GPU-n) futtatott Mini CPM esetében volt a legrosszabb. A többi LLM esetén is aszinkron hívásokat kellett alkalmazni a feldolgozás jelentős felgyorsításához. Ebben az esetben

tömegesen küldünk időben szinte egyszerre hívásokat az LLM kiszolgálórendszerre felé, amit a szolgáltatás elérhetőségének fenntartása érdekében minden szolgáltató lekérdézési korlátozásokkal (rate limit) véd. Ennek következtében megfelelő késleltetést kellett alkalmaznunk az OCR folyamat során.

A GPT-4o esetén az OpenAI biztonsági intézkedései a modell kimentének kontrollására azt jelentették, hogy az OCR felismerési feladat során egyes oldalak megakadtak a tartalomszűrő (content filter) funkció miatt. Ilyen volt például az arcfelismerő szoftverekről szóló esettanulmány, amit veszélyes tartalomnak ítélt meg az OpenAI modellje, és ezért nem adta vissza az oldal tartalmát. A prompt módosításával sem sikerült kikerülni a szűrőt, ahogyan az OCR szöveg hozzáadásával sem lehet megkerülni a biztonsági funkciót. Két megközelítés alkalmazható a gyakorlatban: a teljes áttérés a nyílt modellekre vagy a zárt modell használata egy nyílt modellel kombinálva, mint tartalék módszer. Ez utóbbit választottuk az oktatási Chatbot rendszer megvalósítása során. Amennyiben tartalmi szűrő hibával tér vissza (finish reason) az OpenAI nyelvi modellje, akkor a hibakezelő részben automatikusan a TogetherAI rendszerén keresztül a Llama 3.2 90B Vision modell segítségével próbáljuk kinyerni az oldal tartalmát.

Amíg a szabályalapú rendszerek ingyenesek vagy egyszeri licenstdíjat kell fizetni értük, addig a zárt és nyílt LLM-k költsége jelentős, és a tokenek számától függ elsősorban. Nyílt LLM-k esetén a kiszolgáló hardver erőforrás kerül pénzbe, és ennek költséghatékony megvalósítása egy központi szolgáltatón (pl. TogetherAI) keresztül igénybevétele tokenalapú árazással. A további lehetőségek: virtuális gép bérlése vagy hardver vásárlása nagyságrendileg magasabb befektetést igényelnek. Habár az árak folyamatosan csökkennek, az új, erősebb modellek megjelenése újra felfelé viszi a költségeket (lásd GPT-o1-preview). A nyílt LLM-k tokenalapú árazása tükrözi a képességeiket, de még a legerősebb modellek is alatta maradnak a zárt LLM-ek költségeinek. A Claude Sonnet 3.5 nyújtotta a legrosszabb ár-érték arányú szolgáltatást, mivel a GPT-4o ugyanolyan mértékű limitjeinek eléréséhez sokkal nagyobb kezdeti, előre kifizetett API összeg volt szükséges. Mindig a feladathoz illeszkedő szintű modellt jó választani, de az OCR feladat biztosan nem a legfejlettebb képességű modellt igényli.

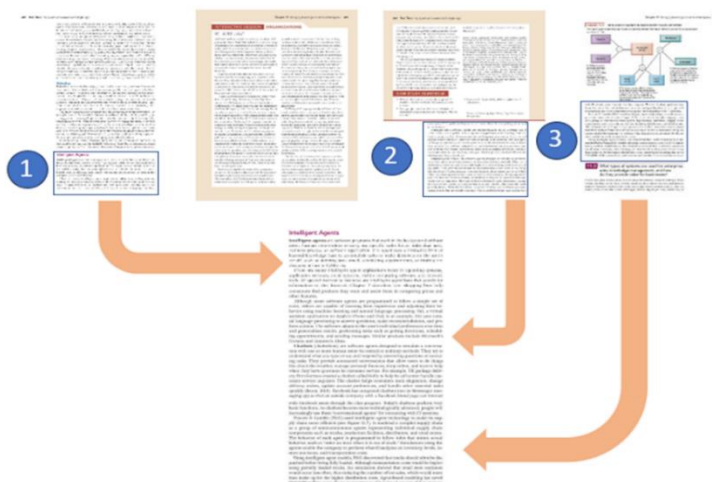
## 5 Összefűzés, a szöveg szemantikai koherenciájának biztosítása

A tankönyvek szövegét gyakran félbeszakítják folyamatábrák, diagramok, táblázatok vagy többoldalas esettanulmányok. A teszteléshez használt PDF könyvben számtalan példát találtunk arra, amikor a mondat közepén megszakad a szöveg folytonossága, és egy közbeékel, bekeretezett tanulmányt követően akár 2 oldallal később került lezárásra a félbehagyott mondat. A hagyományos szabályalapú vagy OCR-t használó programok elsősorban a pontos karakterfelismerése, a szövegblokkok oldalon belüli sorrendjének kitalálására koncentrálnak (Smith, 2007). Az előbb említett eset számukra kezelhetetlen, és érdekes módon nem találtunk olyan más módszert, amely a szöveg folytonosságnak ilyen mértékű megszakításával foglalkozott volna. Minden esetben a RAG módszert alkalmazó algoritmusok feltételezték, hogy az oldal jobb alsó sarkában lévő mondatot, szavakat a következő oldal bal felső sarkában kezdődő szöveghez kell illeszteni. Esetünkben ez inkább a kivétel volt, mint a szabály. A problémára

mindenképp találnunk kellett megoldást, mert könnyen belátható, hogy a rosszul összfűzött szövegrészek értelme teljesen más, és a további szemantikai kereséseknél kisebb eséllyel találjuk meg a legrelevánsabb szövegrészletet a kérdéssel kapcsolatban.

A klasszikus PDF beolvasó, feldolgozó programok OCR esetén tapasztalatunk szerint egyszerű szöveget adtak vissza vagy legfeljebb Markdown formátumban formázott címeket és táblázatokat. Ennek hatására először az LLM-ektől a címek Markdown formázását kértük és a félbemaradt mondatok, bekezdések megjelölését egy speciális karaktorsorozattal, hogy az összefűzés a címek, jelölések mentén megvalósítható legyen. Ez azonban teljesen véletlenszerűen működött, így nem adott megbízható eredményt. Ezért áttértünk a JSON formátumú szövegblokk alapú felismerésre, amikor minden oldalt címmel ellátott szövegblokkokra osztunk fel. A szövegrész típusa lehet kép, táblázat vagy szöveg. A tartalomszerkesztési módszerek alapelveiből következően a szövegeket címekkel tagoljuk, és a következő címig tartó rész egy szemantikai egységet alkot. Ezekbe az egységekbe ékelődhetnek be képek, táblázatok, példaként felhozott esettanulmányok, amelyek önálló címmel rendelkeznek. Ha olyan szövegblokkot találunk egy oldalon, amelynek nincs címe, akkor az valamelyik előző oldalon megnyitott fejezet folytatása, amellyel összekötjük az aktuális oldal cím nélküli részét.

Az összekötés algoritmusában figyelembe veszi, hogy több befejezetlen szövegblokk esetén mindig a később megkezdett tartalmi rész zárul le először. Ezért a párosítás során az előző oldal utolsó blokkjától visszafelé indulunk el és kötjük össze az új oldal első folytatólagos bekezdéseivel. Majd ezután tovább haladunk visszafelé és folytatjuk a párosítást. Figyelembe vesszük az előző oldalon már korábban párosított folytatólagos szövegrészeket, így ennek megfelelően az összekötés során hosszabb, láncolt listákra jellemző szerkezet is kialakulhat, ahogy azt a 3. ábrán is láthatjuk.



**3. ábra:** Az alfejezet szövege négy oldalon nyúlik át több megszakítással (saját munka).

A 3. ábrán a 468. oldalon kezdődik egy alfejezet. Ezt követi egy közbevetett tanulmány, amely a 469. oldalon kezdődik, és a 470. oldalon ér véget. A 468. oldal alján



elkezdett fejezet csak itt a 470. oldal alján folytatódik, majd a 471. oldal közepén ér véget egy beékelődő folyamatára után. A visszafelé kereső algoritmus megtalálja a fejezet 3 különböző oldalon szétszórt részét, és a megfelelő sorrendben konkatenálja mindhárom részt. A kép és táblázat típusú blokkokat önálló szemantikai egységként kezeljük.

További gondot jelentett, hogy az LLM-ek alapvető jellemzőikből következően kényszeresen befejezték a félbehagyott mondatokat, és több esetben a cím nélküli első bekezdéseket teljesen kihagyták a kimenetből. Ezeket a problémákat a promptban megfogalmazott, célzott mondatokkal sikerült megoldani. Az ilyen típusú szövegfolytonosság garantáltan megőrzi a szemantikai egységeket, és el is különíti őket egymástól. Valójában már belelóg a következő (szöveg feldarabolása) lépés elejébe, mivel ezeket a létrehozott egységeket további kisebb részekre feldaraboljuk, és nem a teljes összeragasztott szöveget alkalmazzuk a szövegdaraboló technikákat.

## 6 Szöveg feldarabolása

A fejezetek független szemantikai egységeket alkotnak, amelyek túl hosszúak, ezért további felosztásra van szükség kisebb, egyedileg értelmezhető darabokra a válaszgenerálás érdekében. Három módszert teszteltünk, mindegyikre 1000 karakteres határt megszabva.

A rekurzív karakteres felosztás a szöveget természetes határoknál (pl. új sorok, szóközök) osztja fel, amíg el nem éri a célméretet. Ez jellemzően új szövegdarabot jelent egy új bekezdés esetén, de egyes esetekben két-három bekezdés még egy szövegdarabot alkotott. A rekurzív felosztás LangChainben implementált, elterjedt szövegdarabolási megoldás.

A szemantikai feldarabolás arra az ötletre épül, hogy már a szövegdarabolás is a mondatokhoz tartozó beágyazások (embedding) szemantikai hasonlósága alapján dőljön el. A módszer egy 3 mondatos, 1 mondatonként csúszó ablakot használ a blokkok közötti szemantikai távolság összehasonlítására koszinusz hasonlóság alkalmazásával, és a kiválasztott percentilis (itt 95%) feletti távolságoknál vágja el a szöveget.

Az eljárás hátránya, hogy a határérték kiválasztása önkényes és kiszámíthatatlan, pontosan hol történik vágás. Ezáltal az eljárás nehezen általánosítható és automatizálható. Akár egy bekezdés végi mondatot is külön szövegdarabnak vesz, amennyiben az egy másik szövegrészre utal, és a bekezdések végét gyakran nem veszi figyelembe.

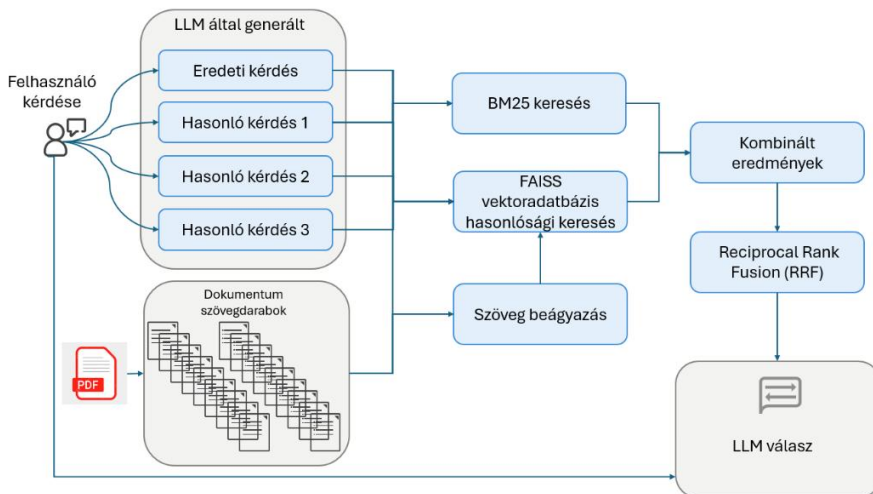
Az ágensalapú szövegdarabolás a kulcsfontosságú kritériumok (pl. szemantikai koherencia) körvonalazása után a promptban, teljes mértékben az LLM-re bízva a szöveg szegmentálását és a darabok létrehozását. A prompt a PDF feldolgozásnál használnál sokkal rövidebb, tartalmazza pl. az 1000 karakteres limitet, specifikálva azt is, hogy ez általában egy-két bekezdést jelent – ezzel további támpontot nyújtva az LLM-nek a feladat elvégzéséhez. 500 karakter alatti szövegdarabok elkerülése is szerepel a promptban, ami azt eredményezte, hogy az ágensalapú szövegdarabolás nyújtotta a legkonzisztensebb szövegdarab méreteket. A rekurzív karakter felosztáshoz hasonlóan követte a bekezdések végét a szövegdarabok létrehozása során.

Mindhárom módszer esetén a képeket és táblázatokat külön tartjuk, mert azokat nem érdemes további részekre bontani. Minden képleírás és kinyert táblázat alapvetően egy szemantikai egységet képez, amelyek hossza nem kiemelkedő a képzett 1000 karakteres szövegdarabokhoz képest.

A megtalált releváns szövegdarabok pontos oldalszámait és a hozzájuk tartozó oldalak képét megmutatjuk a felhasználó számára, hogy látható legyen az LLM milyen kontextus alapján határozta meg választát. A darabolás során a nagyobb szövegrészből kinyert kisebb szövegdarabok oldalszámai eltérhetnek a valós helyzethez képest, mivel egy-egy fejezet több oldalt is magába foglalhat. Ezért a kisebb szemantikai egységekre bontás legvégén minden szövegrészt végigkeresünk az eredeti, oldalanként egy sort tartalmazó adattáblában, ahol a megtalált előfordulás oldalszáma lesz a megfelelő.

## 7 Keresési módszerek és rangsorolás

A feldarabolt szövegrészek betöltésre kerülnek a FAISS vektoradatbázisba a kiválasztott beágyazásokkal, és a BM25 korpusz is létrehozásra kerül (Robertson és Zaragoza, 2009). Az ezekben történő kereséssel 3 különböző módszer tesztelése valósult meg. Az alapeset (baseline) módszer csak a vektoradatbázist használja fel, és a három leginkább hasonló szövegrésztletet adja vissza koszinusz hasonlóság alapján.



4. ábra: A felhasználói kérdések feldolgozásának lépései (saját munka)

Ezt egy jóval komplexebb kereséssel hasonlítjuk össze, amelynek teljes folyamata a 4. ábrán látható. A második módszer 3 szemantikailag hasonló lekérdezést generál az LLM segítségével (Ma és mtsai, 2023), és mindegyik lekérdezésre lefut a vektoradatbázis és BM25 korpusz keresés 3-3 eredménnyel visszatérve, a relevancia pontszámokkal együtt. A pontszámok normalizálásra kerülnek 0 és 1 közötti skálán, majd

csökkenő sorrend szerinti rendezést követően a két lista összekapcsolása történik meg az azonos indexű szövegdarabokat csak egyszer hozzáadva a közös listához. A szövegdarabok sorrendjének rangsorolása a Reciprocal Rank Fusion technikával valósul meg, amely a  $1 / (rank + k)$  képlet szerint rendezi a listát, ahol a  $k$  fix 60 (Cormack és mtsai, 2009). Ez a fajta rangsorlás biztosítja, hogy azon szövegdarabok, melyeket mindkét keresési módszer fontosnak ítélt a kérdés megválaszolásához, a lista elejére kerüljön. Az RRF rangsorolás felülmúlja a többi rangsorolási technikát (Cormack és mtsai 2009), valamint az Azure AI Search nagyvállalati megoldás is alkalmazza.

A harmadik módszer az Anthropic által ajánlott ún. kontextus beágyazás (Contextual Embedding) készítése (Anthropic, 2024b). Ezek a kontextust nyújtó szövegrészek minden szövegdarabhoz hozzácsatolásra kerülnek. Az LLM-ek feladata az volt, hogy készítsenek a teljes fejezet alapján egy tömör összefoglalót, amely a szövegdarabot a fejezeten belül elhelyezi, hogy a szöveg tágabb összefüggései is felismerhetőek legyenek a válaszadás során. Ez esetben csak a vektoradatbázis keresést alkalmazzuk, ami lehetővé teszi az alap módszerrel történő közvetlen összehasonlítást.

Mindegyik módszer esetén a szövegdarabokkal együtt tárolt metaadatok egy részét (például fejezet, kép és táblázat címek) is átadtuk a végső LLM lekérdezésnél, hogy még jobb eredményt érjünk el.

## 8 Kérdés-válasz teszt eredmények

A választott tankönyv 11. fejezetére vonatkozó 69 hivatalos tesztkérdést értékeltünk ki automatikusan, mivel ezen kérdésekhez rendelkezésre álltak a válaszok is. A tesztadatbázis 43 db feleletválasztós, 16 db igaz/hamis és 10 db esszé kérdést tartalmazott. Ennek megfelelően a kérdések nagy részét mintaalapú algoritmikus egyezéssel lehetett azonosítani (választott betűjel, igaz/hamis szavak előfordulása), míg az esszé kérdésekre a GPT-4o-t alkalmaztuk a bemeneten megadva az eredeti kérdést, az aktuálisan vizsgált LLM-től kapott választ és a hivatalos megoldást. Arra kértük, hogy 0-tól 1-ig tartó intervallumon állapítsa meg a válasz helyességét összehasonlítva a megoldással. Manuális ellenőrzés után a határt 0.8-nál húztuk meg, mert a tapasztalat azt mutatta, hogy ezek már biztosan jó megoldások. Tapasztalataink alapján az értékelést akkor csökkentette a kiértékelő GPT-4o, ha a válasz ugyan egyezett, de sokkal terjedelmesebb volt vagy további állításokat is tartalmazott.

Három módszert kiértékelését végeztük el (2. táblázat): (1) csak vektoradatbázis keresés, (2) kontextus beágyazás, és (3) egy kombinált vektoradatbázis, BM25 és RRF megközelítés. A kombinált RRF módszer következetesen felülmúlta a többi módszert, míg a kontextus beágyazás a 2. helyen végzett. Mindhárom RAG módszer esetében a legjobb eredmény 100%-os egyezés vizsgálata esetén 91% felett volt. A legjobbnak a GPT-4-turbo bizonyult 94,2%-os eredménnyel. A 100% azt jelenti, hogy az esszé típusú kérdések GPT-4o általi kiértékelésénél 1.0 értéknél húztuk meg a határt, ami a modell szerint a hivatalos megoldókulccsal teljesen egyező lényegi tartalmú választ jelentette. A 80%-os egyezést csak a kombinált RRF módszerre teszteltük, ahol a GPT-4-turbo 98,55%-os eredménnyel végzett az élen, ezzel a legjobb modellnek bizonyult.

A kombinált RRF módszert a korábban felsorolt 4 zárt és 4 nyílt LLM modellen teszteltük (GPT-4-turbo, GPT-4o, GPT-4o-mini, Claude Sonnet 3.5, Google Gemma 27B, Qwen2-72B, Llama-3.1-405B, Mixtral-8x22B), ahol a 80%-os egyezést elfogadva a legjobb eredményt a GPT-4-turbo érte el továbbra is 98,55%-kal, de a második a nyílt Qwen2 72B LLM lett 95,7%-kal.

2. táblázat: A 3 RAG módszer vizsgálata a 69 tesztkérdésen futtatva

	GPT-4o	GPT-4o-mini	GPT-4-turbo
<b>Eredmények: 100%-os egyezés</b>			
Csak vektor adatbázis keresés	<b>91,3%</b>	85,5%	87,0%
Kontextus beágyazás	85,5%	87,0%	<b>91,3%</b>
Vektor adatbázis, BM25 és RRF kombinált keresés	92,8%	91,3%	<b>94,2%</b>
<b>Eredmények: 80%-os egyezés</b>			
Vektor adatbázis, BM25 és RRF kombinált keresés	94,2%	94,2%	<b>98,55%</b>

## 9 Tanulságok összefoglalása

A nem strukturált, szöveget és képet is tartalmazó dokumentumok feldolgozására, tudásbázisba helyezésére és a tudásbázis lekérdezésére egy több lépéses RAG folyamatot valósítottunk meg, és teszteltünk egy felsőoktatási chatbot fejlesztésének kontextusában. A beolvasáshoz a vision képességgel rendelkező LLM-ek használata felülmúlja a szabályalapú és OCR megoldások teljesítményét, ahol a nyílt súlyú modellek a zárt modellekben megtalálható tartalmi szűrő nélkül biztosítják a szöveg teljességét több 100 oldalas dokumentumok esetén is. A nagy nyelvi modellek képesek pontosan JSON struktúrába illeszteni a szövegrészeket, lehetővé téve a félbeszakított szövegrészek algoritmikus összefűzését és a komplex oldalstruktúrák kezelését. A további feldarabolás céljára az ágensalapú módszer egy testre szabható, teljes dokumentumra konzisztens megoldás. A kérdés-válasz (Q&A) feladatra egy kombinált megoldás (a kérdések többszöri átfogalmaztatása, a BM25 és vektor adatbázis keresés kombinálása, majd RRF szerinti sorbarendezés) adta a legjobb eredményt minden tesztelt modell esetén. A kérdés-válasz teszten 1. helyen végzett GPT-4-turbo 7,2 százalékpontos javulást ért el a kombinált RRF módszerrel az egyszerű vektor hasonlósági kereséshez képest.

Az ingyenesen elérhető nyílt súlyú modellek és a zárt modellek összehasonlítása révén megállapítottuk, hogy a nyílt megoldások számos esetben megfelelő alternatívát nyújthatnak, különösen a költségek és a pontosság arányát tekintve. Nyílt megoldások alkalmazása jelentős költségmegtakarítást eredményezhet, miközben a chatbot teljesítménye versenyképes marad, így ez a megoldás egyértelműen vonzó lehet a felsőoktatási intézmények számára, amelyek hatékony eszközöket keresnek a tanulók támogatására és az oktatás továbbfejlesztésére. Kutatásaink alapján az optimális megoldás egy olyan megfelelő méretű nyelvi modell alkalmazása, melynek költségei minimálisak.

## Köszönetnyilvánítás

Az EKOP-24-2-003 azonosítószámú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási, Fejlesztési és Innovációs Alapból nyújtott támogatásával a 2024/2025 tanévre meghirdetett Egyetemi Kutatói Ösztöndíj Programjának a finanszírozásában valósult meg.

A chatbot fejlesztését és szerver szolgáltatását a Webra Kft. ([webra.hu](http://webra.hu)) támogatta.

## Bibliográfia

- Anthropic. (2024a). Claude 3.5 Sonnet Model Card Addendum. <https://paperswithcode.com/paper/claude-3-5-sonnet-model-card-addendum>
- Berkecz, P., Zombori, T., Banga, G., Szabó, G., Szántó, Zs., Novák, A. & Farkas, R. SHunQA: egy nyíltkérdés-megválaszoló rendszer. XX. Magyar Számítógépes Nyelvészeti Konferencia online kiadás, Magyarország, Szegedi Tudományegyetem (2024) pp. 73-84., 12 p.
- Camelot Developers. 2021. Camelot. <https://github.com/camelot-dev/camelot>. Letöltve: 2024.12.23.
- Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009*. <https://doi.org/10.1145/1571941.1572114>
- Fenniak, M., Kulkarni, A. & Witham, S. 2014. PyPDF. <https://github.com/py-pdf/pypdf>. Letöltve: 2024.12.23.
- Gemini Team Google. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv.org*. <https://doi.org/10.48550/arxiv.2403.05530>
- Gemma Team, Google DeepMind. (2024). Gemma 2: Improving Open Language Models at a Practical Size. *arXiv.org*. <https://arxiv.org/abs/2408.00118v3>
- Introducing Contextual Retrieval. (2024b). Anthropic.com. <https://www.anthropic.com/news/contextual-retrieval>
- Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., Casas, D.D., Hanna, E.B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L.R., Saulnier, L., Lachaux, M., Stock, P., Subramanian, S., Yang, S., ... Sayed, W.E. (2024). Mixtral of Experts. *arXiv*, abs/2401.04088. *arXiv.org*. <https://arxiv.org/abs/2401.04088>
- Laudon, K. és Laudon, J. (2021). *Management Information Systems: Managing the Digital Firm*. 17th (Global) edition. Pearson Education. Harlow UK.
- Lewis, P. S. H., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Neural Information Processing Systems*, 33, 9459–9474.
- Llama Team. (2024). The Llama 3 Herd of Models. *arXiv.org*. <https://arxiv.org/abs/2407.21783v2>
- LLM Document Extraction: How to use AI to get structured data from legacy documents. (2021). Pondhouse-Data.com. <https://www.pondhouse-data.com/blog/document-extraction-with-llms>
- Ma, X., Gong, Y., He, P., Zhao, H., & Duan, N. (2023). Query Rewriting in Retrieval-Augmented Large Language Models. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. <https://doi.org/10.18653/v1/2023.emnlp-main.322>

- McKie, J. X.; és Liu, R. 2016. PyMuPDF. <https://github.com/pymupdf/PyMuPDF>. Letöltve: 2024.12.23.
- Meta. (2024). Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta.com. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>
- Mistral AI team. (2024). Cheaper, Better, Faster, Stronger. Mistral.ai. <https://mistral.ai/news/mixtral-8x22b/>
- OpenAI. (2024). GPT-4o System Card. *arXiv.org*. <https://arxiv.org/abs/2410.21276>
- Pethő, G., Sass, B., Simon, L., Lipp, V. OCR-hibák kvantitatív elemzése több szövegválogatott összehasonlításával. XX. Magyar Számítógépes Nyelvészeti Konferencia online kiadás, Magyarország, Szegedi Tudományegyetem (2024) pp. 17-29., 13 p.
- Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389. <https://doi.org/10.1561/15000000019>
- Shuster, K., Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). Retrieval Augmentation Reduces Hallucination in Conversation. *Conference on Empirical Methods in Natural Language Processing*.
- Smith, R. (2007). An overview of the Tesseract OCR engine. *Proceedings of the International Conference on Document Analysis and Recognition*, 629–633. <https://doi.org/10.1109/icdar.2007.4376991>
- Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., ... Fan, Z. (2024). Qwen2 Technical Report. *arXiv.org*. <https://arxiv.org/abs/2407.10671v4>
- Yao, Y., Yu, T., Zhang, A., Wang, C., Cui, J., Zhu, H., Cai, T., Li, H., Zhao, W., He, Z., Chen, Q., Zhou, H., Zou, Z., Zhang, H., Hu, S., Zheng, Z., Zhou, J., Cai, J., Han, X., & Zeng, G. (2024). MiniCPM-V: A GPT-4V Level MLLM on Your Phone. *arXiv.org*. <https://arxiv.org/abs/2408.01800>
- Why am I receiving an “Output blocked by content filtering policy” error? Anthropic Privacy Center. (2024). Anthropic.com. <https://privacy.anthropic.com/en/articles/10023638-why-am-i-receiving-an-output-blocked-by-content-filtering-policy-error>

# NYELVMODELLEK





# A látens szemantikus tulajdonságok segítségével előtanított nyelvi modellek vizsgálata

Berend Gábor

Szegedi Tudományegyetem, Informatikai Intézet  
berendg@inf.u-szeged.hu

**Kivonat** A szóalakok helyett fogalmi egységek mentén előtanított nyelvi modellek mintahatékonyágát több korábbi munka is igazolta. Ezek a vizsgálatok azonban a létrehozott modelleket csupán finomhangolhatóságuk szempontjából vizsgálták. Jelen cikkben a látens szemantikus tulajdonságok jelentette kváziszimbolikus információ mentén történő előtanításnak a nyelvi modellezés elvégzésére gyakorolt (negatív) hatását mutatjuk be, illetve javaslunk két módszert is, amely használatával a modellek nyelvi modellezési képességei javíthatók a fogalmakra való érzékenységének megtartása mellett.

## 1. Bevezetés

A tradicionálisan előtanított maszkolt nyelvi modellek létrehozásuk során olyan kimeneti eloszlások meghatározásával tudják a veszteségfüggvényüket minimalizálni, amelyek a teljes valószínűségi tömeget pontosan egy megadott – az aktuális mondatban a kimaszkolást megelőzően az adott szó helyén eredetileg ott álló – szóra helyezik. Vagyis ha a maszkolás után előáll *Péternek [MASK] gyereke van.* mondatban az eredetileg a [MASK] token helyén álló szó a *három* számnév volt, úgy a nyelvi modell abban az esetben tudja minimalizálni az adott maszk tokenre vonatkozó veszteségét, ha annak behelyettesítésével kapcsolatosan kizárólag a *három* szó ottlétéhez rendel nullától különböző valószínűséget, holott a valóságban más szavak is alkalmasak lennének a mondatból kitakart szó helyettesítésére. A betanított maszkolt nyelvi modellek kimeneti eloszlásai természetesen már nem annyira szélsőségesek, mint amennyire szélsőséges kimeneti eloszlások minimalizálják a veszteségfüggvényüket, amit annak köszönhetnek, hogy az előtanításuk nagy mennyiségű, változatos szövegeken történik, és hosszú távon az egyes kimenetek mentén elvárt szélsőséges eloszlások „átlagát” adják vissza.

Shani és mtsai (2023) véleménycikkükben amellett érveltek, hogy a nyelvi modellek viselkedése szempontjából kívánatos lenne, ha azok működése a konkrét szóalakok helyett emberi fogalmak mentén történne. Egy ilyen megoldást adott (Berend, 2024), amely egy már tradicionálisan betanított nyelvi modell rejtett reprezentációi segítségével határozta meg azon látens szemantikai tulajdonságok körét, amely alapján aztán az enkóderalapú neurális nyelvi modellek tanításának elvégzését javasolta. A módosított előtanítással nyert neurális nyelvi modellek mintahatékonyak bizonyultak abban az értelemben, hogy megegyező mennyiségű szövegen történő előtanítást követően, változatos célfeladatok mentén való

finomhangolásuk során a hagyományos módon előtanított modelleknél átlagosan jóval kedvezőbb eredmények elérésére voltak képesek.

A Berend (2024) által létrehozott modellek mintahatékonysága azonban nem terjedt ki a létrejövő modellek maszkolt nyelvi modellezési képességeire, hiszen azáltal, hogy a javasolt előtanítás teljességgel függetlenítette magát a konkrét szóalakok előrejelzésétől, így ennek a feladatnak az elvégzésére teljesen alkalmatlanok voltak a modelljei.

A korábban javasolt látens szemantikus tulajdonságok mentén történő előtanítás egy további tulajdonsága, hogy egy már előtanításon átesett segéd nyelvi modellre támaszkodik. Azon túl, hogy egy már előtanított modell létezésének feltételezése egy új modell létrehozása során nem feltétlen elegáns, maga után vonja azt a negatív következményt is, hogy a látens szemantikus tulajdonságokkal operáló neurális nyelvi modell szótárának szükségképpen a segédmodell szótárával megegyezőnek kell legyen, ezzel egy potenciálisan komoly korlátot állítva a módszer gyakorlati alkalmazhatósága elé.

Jelen cikkben olyan kiterjesztését, illetve módosítását adjuk a maszkolt látens szemantikus modellezési (MLSM) előtanítási eljárásnak, amely segítségével létrehozott neurális nyelvi modellek a tokenek fogalmi csoportosításán túlmenően a konkrét maszkolt nyelvi modellezési feladat ellátására is képesek. A javasolt új módszerek közül az egyik még azzal a további kedvező tulajdonsággal is rendelkezik, hogy nem igényli egy már korábban betanított nyelvi modell meglétét, ezáltal lehetőséget teremt arra is, hogy akár teljesen új – a számunkra releváns új doménhez adaptált – tokenizálóval hozzunk létre neurális nyelvi modelleket mintahatékonny módon.

## 2. Kapcsolódó munkák

A maszkolt nyelvi modellezés során sokáig a BERT (Devlin és mtsai, 2019) modell létrehozása során lefektetett stratégiára támaszkodtak, ami az inputban található szótöredékek véletlenszerűen kiválasztott 15%-a alapján hajtja végre az előtanítást. A maszkolt nyelvi modellezéssel előtanított nyelvi modellek feladata az, hogy az eredeti szövegek 15%-ának lecserélését követően legyen képes helyreállítani a szövegből kikapart szóalakok pontos kilétét. A közelmúltban azonban számos alternatívát javasoltak a rekonstruálandó szótöredékek véletlenszerű kiválasztása helyett (Joshi és mtsai, 2020; Levine és mtsai, 2021; Yang és mtsai, 2023). Wettig és mtsai (2023) az inputszekvencia 15%-ára irányuló kiválasztási stratégiát vizsgálta, és azt találták, hogy érdemes az inputban található tokeneknek a megszokottnál nagyobb hányadát rekonstruálandó tokenként kezelni.

Berend (2023) egy olyan előtanítási feladatot vezetett be, amely során a tanulás tárgyát nem a kiválasztott inputtokenek rekonstruálása képezi, hanem az azok kontextusa mentén meghatározott látens szemantikus jellemzések meghatározása a célja. A szemantikus jegyek meghatározására egy segédmodell rejtett reprezentációi alapján került sor ritka kódolás használatával. A módszer sajátossága, hogy mivel az előtanítás során alkalmazott modellezés fókuszát áthelyezi a szóalakokról a kváziszimbolikus szemantikus jegyekre, így a klasszikus nyelv-

vi modellezés feladatát nem képesek ellátni az ezzel a módszerrel létrehozott modellek.

### 3. Vizsgált módszerek

Cikkünkben a klasszikus maszkolt nyelvi modellezéssel előtanított transzformeralapú enkóder modelleknél jobb mintahatékonyssággal rendelkező maszkolt látens szemantikus modellezési előtanítási eljárás továbbfejlesztési lehetőségeit vizsgáljuk. Célunk, hogy a látens szemantikus kategóriák kezelésére, valamint a szó(be)helyettesítési feladat elvégzésére *egyaránt* képes modelleket hozzunk létre. A következőkben áttekintjük a kísérleteinkben előforduló előtanítási eljárásokat, illetve azoknak az általunk javasolt továbbfejlesztéseit.

#### 3.1. Maszkolt nyelvi modellezés (MLM)

A klasszikus maszkolt nyelvi modellezési feladat során a kategorikus keresztent-rópia hibátagot használjuk annak számszerűsítésére, hogy az előtanított modell aktuálisan mekkora outputvalószínűséget rendel a megadott pozíción az eredeti inputban szereplő token visszahelyettesítésével kapcsolatban. Ez a klasszikus előtanítási paradigma húzódik meg számos modell, pl. a BERT (Devlin és mtsai, 2019), vagy a magyar huBERT (Nemeskey, 2021) létrehozása mögött.

#### 3.2. Maszkolt látens szemantikus modellezés (MLSM)

Az MLSM egy segédmodellre támaszkodik a maszkolásra véletlenszerűen kiválasztott tokenek kváziszimbolikus látens szemantikus profiljának meghatározása során.

Az eljárás első előkészítő lépése a segédmodell  $h$ -dimenziós rejtett állapotaiból mintavételezéssel létrehozott  $\mathbf{X} \in \mathbf{R}^{h \times n}$  mátrixra támaszkodik, amelyre megkeressük a

$$\min_{\mathbf{D}, \boldsymbol{\alpha} \in \mathbb{R}_{\geq 0}^{k \times n}} \frac{1}{2} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1, \quad (1)$$

feladat optimumát, és amely alapján egy korábbiakban nem látott  $\mathbf{x}_i \notin X$  rejtett reprezentációval rendelkező vektorra a  $\mathbf{D} \in \mathbf{R}^{h \times k}$  mátrixot már ismertnek tekintve az

$$\arg \min_{\boldsymbol{\alpha}_i \in \mathbb{R}_{\geq 0}^k} \frac{1}{2} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1 \quad (2)$$

módon határozható meg annak látens szemantikus jellemzését előállító  $\boldsymbol{\alpha}_i \in \mathbb{R}_{\geq 0}^k$  vektora.

A  $\lambda$  regularizációs tag azt eredményezi, hogy a  $\boldsymbol{\alpha}_i$ -beli együtthatók többsége pontosan 0 lesz, míg a nemnulla együtthatók pozíciói olyanok lesznek, hogy azok – a kísérleteink által is a későbbiekben alátámasztott módon – nagyfokú együttállást mutatnak a szemantikus tulajdonságaikkal.

**UT-MLSM: Utólagos tanításon átesett maszkolt látens szemantikus modellezés** A kváziszimbolikus látens szemantikus tulajdonságok segítségével előtanított modellek a modellezés hiperparamétereként megválasztott  $k$  darab látens szemantikus tulajdonságra vonatkozó eloszlásokat produkálnak. Amennyiben a látens tulajdonságok valóban a szavak fogalmi tulajdonságait ragadják meg, és az MLSM előtanítás megfelelően konvergált, úgy feltételezhető, hogy az MLSM célfüggvényvel előtanított modellek által szolgáltatott eloszlásoknak alkalmasak a konkrétan kimaszkolt szavak rekonstruálására is.

Ezen hipotézis ellenőrzésének érdekében egy olyan modellvariánst hoztunk létre, amely egy hagyományos MLSM előtanításon átesett modellt olyan módon egészít ki, hogy annak paramétereinek lefagyasztását követően egy további osztályozó fejjel egészíti ki a modellt. Ennek az osztályozó fejnek a feladata az, hogy a látens szemantikus tulajdonságok teréből a modell által használt szótárára vonatkozó logit értéket hozzon létre, vagyis lényegében egyetlen, a modellezés során használt látens tulajdonságok számaszor a modellben megkülönböztetett tokenek számaszoros lineáris leképezést tanulunk a modellhez úgy, hogy mindközben a korábbiakban az előtanításon átesett súlyait érintetlenül hagyjuk.

**SM-MLSM: Segédmodellmentes maszkolt látens szemantikus modellezés** A következőnek javasolt modellvariánk fontos tulajdonsága, hogy az nem igényli egy az előtanítás megkezdésének pillanatában már rendelkezésre álló betanított neurális nyelvi modell létezését. A segédmodellmentes maszkolt látens szemantikus modellezés (SM-MLSM) elvén működő módszerünk a maszkolt nyelvi modellezés és a látens szemantikus kategóriák kialakításának alternáló végrehajtásának az elvét követi.

Az SM-MLSM módszer egy teljesen véletlenszerű súlyokkal inicializált neurális nyelvi modelltől indul ki, majd az előtanítás során elvégzendő lépések számának az első meghatározott (esetünkben 12,5%-nak választott) lépésében kizárólagosan a klasszikus maszkolt nyelvi modellezés feladatát látja el. Az előtanítás a hiperparaméterben meghatározott darabszámú kisebb előtanítási szakaszból áll, és minden egyes szakasz végén, az adott szakaszban meghatározott utolsó 200 kötegben előfordult tokeneknek a transzformer modell utolsó rétegében előálló rejtett vektorai alapján egy (1)-ben található alakú optimalizálás problémát old meg annak érdekében, hogy a következő előtanítási szakaszban – a klasszikus maszkolt nyelvi modellezési feladat ellátása mellett párhuzamosan – a látens szemantikus tulajdonságok modellezésére vonatkozó előtanítási feladatot is el tudja látni a modell a frissített szótármátrixra támaszkodva.

Mivel a modell korai verziói még vélhetőleg kevésbé hasznos rejtett állapotokat produkálnak, ezért a maszkolt nyelvi modellezés és a látens szemantikus tulajdonságokra támaszkodó modellezésből jövő veszteség egymáshoz való viszonyát az előtanítás előre haladtával szakaszosan változtatjuk. A legelső szakaszban kizárólagosan csak a maszkolt nyelvi modellezésből jövő veszteségfüggvényt vesszük figyelembe a modell frissítése során, míg a következő szakaszokban ezen komponens azonos ütemben csökkenve egyre kisebb szerephez jut, az előtanítás végére elérve egy előzetesen meghatározott arányt (kísérleteink során 0,25-et).

## 4. Kísérletek

A következőkben bemutatjuk az előtanítással kapcsolatos kísérleti beállításainkat, majd a modellekkel végzett kísérleteinket közöljük.

### 4.1. Az előtanítás

Modelljeink előtanítása során az input szövegek maximális szekvenciahosszát 128 szubtokenben maximalizáltuk, 16-os gradiens akkumuláció mellett 64-es kötegméretet, valamint  $1e-4$  maximális tanulási rátát alkalmaztunk. A maszkolásra kiválasztott tokenek arányát megemeltük 0,15-ről 0,25-re, összhangban (Wetting és mtsai, 2023) ajánlásaival. Minden elkészült modellünknek elkészítettük az előtanítási lépéseinek 10%-a, 25%-a, 50%-a, valamint 100%-a után előálló változatát. Az előtanítást minden esetben a huBERT (Nemeskey, 2021) tokenizálójára és a Webcorpus2.0 (Nemeskey, 2020) szövegeire támaszkodva végeztük el. A korpusz hozzávetőlegesen tizenhatmilliárd tokenből áll.

**UT-MLSM előtanítások** Saját UT-MLSM modellünk<sup>1</sup> létrehozása során (Barend, 2024) publikusan elérhetővé tett, MLSM módszerrel előtanított DeBERTa architektúrájú (He és mtsai, 2021) modelljéből<sup>2</sup> indultunk ki. Mivel az UT-MLSM modell esetén a kiinduló modell súlyai lefagyasztásra kerültek, és éppen arra voltunk kíváncsiak, hogy az eredeti modell által visszaadott látens szemantikus tulajdonságokra vonatkozó tokenreprezentációk mennyire alkalmasak a maszkolós nyelvi modellezési feladat hatékony megtanulására, ebben az esetben mindössze 10000 előtanítási lépést hajtottunk végre, ami a Webcorpus2.0 körülbelül 7,5%-ának felel meg.

**SM-MLSM és tradicionális MLM előtanítások** Mivel ezek a modellek nem egy már korábbi előtanított modell továbbtanításából jöttek létre, itt az előtanításra fordított frissítési lépéseink számát százezerre állítottuk. Mindez hozzávetőlegesen tizenkétfélmilliárd token (a Webcorpus2.0 hozzávetőlegesen háromnegyedének) a fölhasználását jelentette.

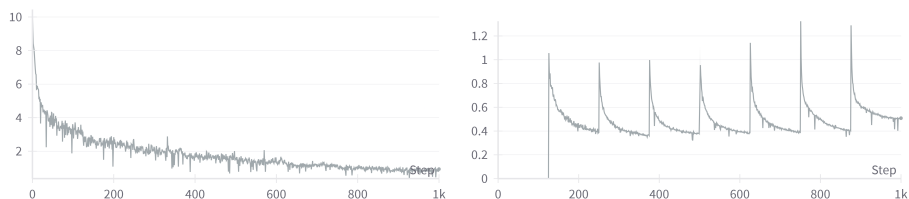
A tradicionális MLM modell<sup>3</sup> előtanításra vonatkozó hiperparamétereink nem tértek el a korábban megadottaktól, míg a segédmodellmentes MLSM modell<sup>4</sup> számára specifikus hiperparaméterek a következők szerint alakultak: a szótármátrix frissítését 12500 lépésként (a teljes előtanítás ideje alatt tehát összesen 8 alkalommal) végeztük el, az alkalmazott látens kategóriák számát a modell rejtett vektorainak dimenziószámának kétszeresére (1536-nak) választottuk, és a maszkolós nyelvi modellezés veszteségfüggvényének relatív súlya az előtanítás végére a 0,25-ös értéket vette fel. Az SM-MLSM előtanítás tulajdonképpen egy többfeladatos tanulási problémaként értelmezhető. Az aggregált veszteségfüggvényünk egyes komponenseinek alakulását az 1. ábrában mutatjuk be.

<sup>1</sup> <https://huggingface.co/SzegedAI/huDeBERTa-MLSM-UT>

<sup>2</sup> <https://huggingface.co/SzegedAI/huDeBERTa-MLSM>

<sup>3</sup> [https://huggingface.co/SzegedAI/huDeBERTa-MLM-v2\\_100k](https://huggingface.co/SzegedAI/huDeBERTa-MLM-v2_100k)

<sup>4</sup> <https://huggingface.co/SzegedAI/huDeBERTa-aux-free-MLSM>

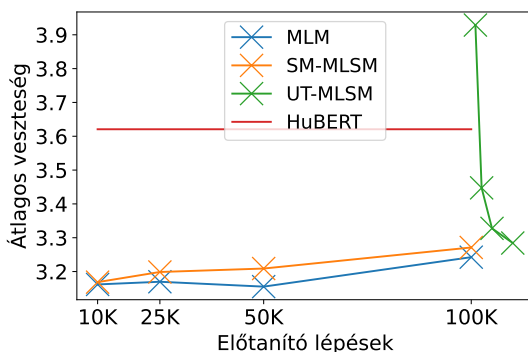


(a) Az MLM veszteségfüggvény alakulása (b) Az MLSM veszteségfüggvény alakulása

1. ábra: A segédmodell fölhasználása nélkül létrehozott SM-MLSM modell veszteségének komponenseinek alakulása a tanítás előrehaladtának függvényében: (a) a klasszikus MLM tanítás által (is) használt kategorikus keresztentrópia hibatarag, valamint a (b) MLSM által használt látens szemantikus tulajdonságok céleloszlásához viszonyított Kullback-Leibler divergencia célfüggvényértékének változása.

## 4.2. A modellek maszkolt nyelvi modellezési képességei

Mint azt korábban említettük, a kváziszimbolikus alapokon nyugvó eredeti MLSM modellezéssel előtanított modellek egyáltalán nem alkalmasak a maszkolt nyelvi modellezési feladat ellátására, vagyis a kimaszkolt tokenek helyére nem képesek a token helyén álló valódi tokenekre vonatkozó kimeneti eloszlásokat produkálni. Az UT-MLSM és SM-MLSM módszerekkel előtanított modelljeink különböző mechanizmusok útján ezek képességgel kapcsolatos hiányosságaira adnak egy megoldást a látens szemantikus tulajdonságok segítségével előtanított modellek viselkedésével kapcsolatban. A modellek tanítására nem használt fejlesztési korpusz mentén a különböző módon előtanított modellek átlagos veszteségfüggvényértékét közöljük a 2. ábrán.

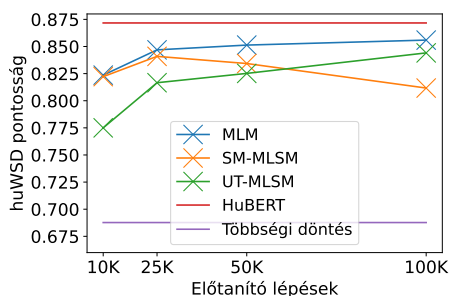


2. ábra: A különböző előtanított modelleknek a tanításukra föl nem használt szövegeken elért veszteségfüggvényének átlagos értéke a kumulált előtanító lépések számának függvényében. Az UT-MLSM modell esetén egy már előzetesen 100 ezer előtanítási lépésen átesett modelltől indulunk ki.

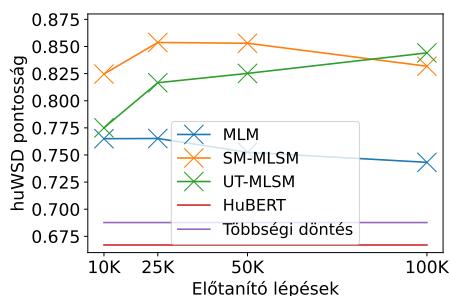
Az 1. ábra alapján megállapíthatjuk, hogy az előtanítás során a tokenek látens fogalmi tulajdonságait is figyelembe vevő modelljeink maszkolt nyelvi modellezési képességei összehasonlíthatók a kizárólag a maszkolós nyelvi modellezési feladat ellátását elvégezni képes tradicionális MLM módon előtanított modellével. Az UT-MLSM modellel kapcsolatban az a tény, hogy már a teljes Webcorpus2.0 kevesebb, mint 1%-ának a fölhasználásával nemtriviális nyelvi modellezési képességekre tett szert (4.0 alatti veszteségfüggvény-érték) azon hipotézisünket támasztja alá, hogy a huDeBERTa-MLSM modell által előzetesen megalkotott látens fogalmi hierarchiája hasznos kiindulási alapot tudott biztosítani a maszkolt nyelvi modellezési képességek elsajátítására nézve.

### 4.3. Jelentésegértelműsítési eredmények

Megvizsgáltuk a különböző modellek által a huWSD jelentésegértelműsítési feladaton elért eredményeket. A nyelvi modellek által létrehozott reprezentációk jelentésegértelműsítési feladatba történő felhasználásának egy egyszerű, de meglepően jó teljesítményt produkálni képes változata az, amikor az egyes ismert jelentéskategóriákhoz egy-egy centroidot rendelünk az adott kategóriába tartozó tanítóhalmazbeli szavakhoz a nyelvi modell által rendelt rejtett reprezentációkból, és tesztelés során egy egyszerű legközelebbi szomszédsági elven döntünk (Loureiro és Jorge, 2019). Az ezzel a módszerrel kapott eredményeinket a különböző módszerekkel előtanított nyelvi modelljeink eltérő elkészültségi fokai mellett a 3a. ábra tartalmazza. Látható, hogy mindegyik modell kifejezetten magas pontosság elérésére képes.



(a) Utolsó rejtett réteg vektora



(b) Modell kimeneti eloszlása

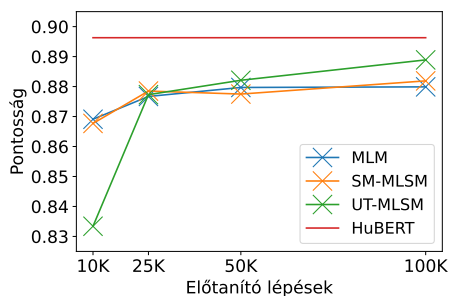
3. ábra: A különböző vizsgált modelleknek a huWSD adatbázisán elért jelentésegértelműsítési eredményei: (a) a modell utolsó rétegéből jövő rejtett állapotok, illetve (b) a modell kimeneti eloszlásának fölhasználásával elért eredmények.

Egy kevésbé gyakori megközelítés, ugyanakkor a modellezést a kváziszimbolikus látens szemantikus fogalmi tulajdonságok mentén (is) elvégző előtanítási

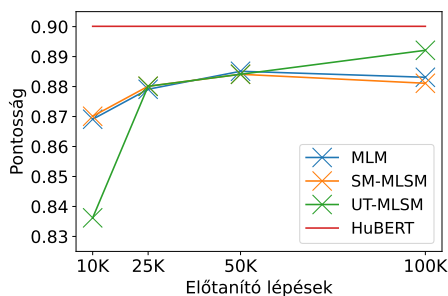
módszerek esetén azonban kínálja magát az a megoldás is, hogy a modell belső rejtett reprezentációi helyett konkrétan magára a modell kiementén produkált eloszlásvektorokra hagyatkozva hozzuk meg a döntéseinket. Ebben az esetben a legközelebbi szomszédság megítélését az Hellinger-távolság kiszámolásával végeztük. Az így kapott eredményeinket a 3b. ábra foglalja össze. Az látható, hogy a látens tulajdonságokat alkalmazó modellek ebben a fajta kiértékelésben sokkal jobb teljesítmény elérésére voltak képesek a látens szemantikus tulajdonságokat az előtanítás során figyelembe nem vevő társaikhoz képest.

#### 4.4. Finomhangolós eredmények

Noha kutatásunkban a modelleket elsődlegesen nem finomhangolhatóságuk szempontjából vizsgáltuk, a következőkben közöljük az opinhubank (Miháltz, 2013) szentimentosztályozási feladatán elért eredményüket. A modelleket 5 alkalommal finomhangoltuk, és közöljük az egyes modellek mentén kapott pontosság átlagát (4a), valamint a független finomhangolással létrehozott modellek szavaztatásával kapott eredményeinket (4b) a különböző módszerrel előtanított neurális nyelvi modelljeinkre. A kapott finomhangolási eredményeinkről elmondható, hogy azok mindegyike összehasonlítható, vagyis a javasolt módosításokkal is a finomhangolásra nézve jól teljesítő nyelvi modelleket kaphatunk.



(a) Egyedi modellek eredménye



(b) Modellek szavaztatásának eredménye

4. ábra: A különböző vizsgált modelleknek az opinhubank szentimentosztályozási adathalmazon elért eredményei.

## 5. Konklúzió

Cikkünkben a látens szemantikus tulajdonságokra támaszkodva megalkotott neurális nyelvi modellek azon hiányosságaira adtunk megoldásokat, miszerint azok nem alkalmasak a klasszikus maszkolós nyelvi modellezési feladat ellátására. A javasolt módszereink egyike (SM-MLSM) ráadásul nem csak ezen limitációján segít a látens kategóriák mentén történő nyelvi modellezésnek, hanem azt



is lehetővé teszi, hogy a nyelvi modell előtanításának megkezdésekor ne kelljen rendelkezésünkre állnia egy betanított már átesett segéd nyelvi modellnek.

## Köszönetnyilvánítás

A cikk a Bolyai János Kutatási Ösztöndíj támogatásával készült. A kutatás az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében. A MILAB SZTE nyelvtechnológia projekt nevében köszönetet mondunk az ELKH Cloud (lásd: Héder és mtsai (2022); <https://science-cloud.hu/>) használatáért, ami hozzájárult a publikált eredmények eléréséhez.

## Hivatkozások

- Berend, G.: Masked latent semantic modeling: an efficient pre-training alternative to masked language modeling. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (szerk.) Findings of the Association for Computational Linguistics: ACL 2023. pp. 13949–13962. Association for Computational Linguistics, Toronto, Canada (Jul 2023), <https://aclanthology.org/2023.findings-acl.876>
- Berend, G.: Neurális nyelvi modellek látens szemantikus információ alapján történő maszkolásmentes előtanítása. In: XX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 85–95. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2024), <https://acta.bibl.u-szeged.hu/78418/>
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), <https://www.aclweb.org/anthology/N19-1423>
- He, P., Liu, X., Gao, J., Chen, W.: DeBERTa: Decoding-enhanced BERT with disentangled attention. In: International Conference on Learning Representations (2021), <https://openreview.net/forum?id=XPZlaoTutsD>
- Héder, M., Rigó, E., Medgyesi, D., Lovas, R., Tenczer, S., Török, F., Farkas, A., Emódi, M., Kadlecik, J., Mező, Gy., Pintér, Á., Kacsuk, P.: The past, present and future of the ELKH cloud. Információs Társadalom 22(2), 128 (aug 2022), <https://doi.org/10.22503/inftars.xxii.2022.2.8>
- Joshi, M., Chen, D., Liu, Y., Weld, D.S., Zettlemoyer, L., Levy, O.: SpanBERT: Improving pre-training by representing and predicting spans. Transactions of the Association for Computational Linguistics 8, 64–77 (2020), <https://aclanthology.org/2020.tacl-1.5>
- Levine, Y., Lenz, B., Lieber, O., Abend, O., Leyton-Brown, K., Tennenholtz, M., Shoham, Y.: PMI-masking: Principled masking of correlated spans. In: International Conference on Learning Representations (2021), <https://openreview.net/forum?id=3Aoft6NWFej>

- Loureiro, D., Jorge, A.: Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation. In: Korhonen, A., Traum, D., Màrquez, L. (szerk.) Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 5682–5691. Association for Computational Linguistics, Florence, Italy (Jul 2019), <https://aclanthology.org/P19-1569>
- Miháltz, M.: OpinHuBank: szabadon hozzáférhető annotált korpusz magyar nyelvű véleményelemzéshez. In: IX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 343–345. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2013), [http://acta.bibl.u-szeged.hu/58859/1/msznykonf\\_009\\_343-345.pdf](http://acta.bibl.u-szeged.hu/58859/1/msznykonf_009_343-345.pdf)
- Nemeskey, D.M.: Natural Language Processing Methods for Language Modeling. Ph.D.-értékezés, Eötvös Loránd University (2020), [https://hlt.bme.hu/en/publ/nemeskey\\_2020](https://hlt.bme.hu/en/publ/nemeskey_2020)
- Nemeskey, D.M.: Introducing huBERT. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2021). pp. 3–14. Szeged (2021)
- Shani, C., Vreeken, J., Shahaf, D.: Towards concept-aware large language models. In: Bouamor, H., Pino, J., Bali, K. (szerk.) Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 13158–13170. Association for Computational Linguistics, Singapore (Dec 2023), <https://aclanthology.org/2023.findings-emnlp.877>
- Wettig, A., Gao, T., Zhong, Z., Chen, D.: Should you mask 15% in masked language modeling? In: Vlachos, A., Augenstein, I. (szerk.) Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics. pp. 2985–3000. Association for Computational Linguistics, Dubrovnik, Croatia (May 2023), <https://aclanthology.org/2023.eacl-main.217>
- Yang, D., Zhang, Z., Zhao, H.: Learning better masking for better language model pre-training. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (szerk.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 7255–7267. Association for Computational Linguistics, Toronto, Canada (Jul 2023), <https://aclanthology.org/2023.acl-long.400>

# Text cleaning with transformer language models for Hungarian

Gábor Madarász<sup>1</sup>, András Holl<sup>2</sup>, Noémi Ligeti-Nagy<sup>1</sup>, Zijian Győző Yang<sup>1</sup>,  
Tamás Váradi<sup>1</sup>

<sup>1</sup>HUN-REN Hungarian Research Centre for Linguistics  
surname.(middle name).forename@nytud.hun-ren.hu

<sup>2</sup>Library and Information Centre, Hungarian Academy of Sciences  
holl.andras@konyvtar.mta.hu

**Abstract.** In language technology, clean data is fundamental for training high-quality models, yet large corpora often contain substantial noise due to OCR errors, missing diacritics, and various user-generated inconsistencies. This paper presents a comprehensive text cleaning pipeline tailored for Hungarian, leveraging transformer-based language models optimized for three key tasks: OCR error correction, diacritic restoration, and filtering grammatically incorrect sentences. We introduce huT5, a Hungarian adaptation of the mT5 model, which significantly reduces model parameters and resource demands while maintaining strong performance on Hungarian-specific text cleaning tasks. The huT5 models were fine-tuned on carefully constructed Hungarian corpora for each task and benchmarked against state-of-the-art methods, demonstrating competitive results, particularly in OCR error correction and diacritic restoration. Our pipeline offers an efficient, freely accessible solution to enhance data quality for Hungarian NLP applications, setting a new standard in resource-efficient, language-specific text cleaning.

**Keywords:** Text cleaning, Transformer Language Models, Hungarian NLP, OCR correction, Diacritic restoration, huT5 model

## 1 Introduction

Corpus cleaning is an ongoing challenge in language technology. Clean data is essential for training language models, yet with the large datasets needed for deep neural networks, a significant portion often requires cleaning. Text data can be messy for various reasons. A prominent research area is the cleaning of OCR-processed texts, which are often prone to errors. Encoding issues, missing diacritics, and grammatical mistakes due to user habits or conventions are also common issues. Cleaning these issues is complex, and no single solution fits all cases. While large language models like ChatGPT hold promise for handling many tasks, they are not perfect and often demand substantial resources, which can be impractical in some scenarios.

To tackle these challenges, we are developing targeted models and applications that can efficiently handle various text cleaning tasks with minimal resources while achieving high accuracy.

Our research prioritizes three main tasks for Hungarian: OCR cleaning, diacritic restoration, and filtering grammatically incorrect texts.

This paper presents a collaborative project between the HUN-REN Hungarian Research Centre for Linguistics (HUN-REN NYTK) and the Library and Information Centre of the Hungarian Academy of Sciences (MTA KIK), aimed at making the contents of the REAL Repository more accessible to researchers and easier to curate and enhance for the MTA KIK.

## 2 Related Work

The importance of improving texts processed through Optical Character Recognition (OCR) is emphasized by the creation of specialized competitions, such as the post-OCR error correction challenge instituted at the International Conference on Document Analysis and Recognition (ICDAR) since 2017<sup>1</sup>. These competitions have catalyzed progress in OCR error rectification by utilizing cutting-edge methodologies. The method that secured victory in 2019, termed Context-based Character Correction (CCC, Rigaud et al., 2019), integrated a convolutional neural network with a BERT model for the purpose of error identification, coupled with a bidirectional Long Short-Term Memory (LSTM) encoder-decoder framework incorporating an attention mechanism for error rectification. Other scholars, including Nguyen et al. (2020), adapted CCC by incorporating Named Entity Recognition (NER) for the identification phase and substituting the LSTM-based correction model with a neural machine translation architecture known as OpenNMT. Schaefer and Neudecker (2020) also employed a dual-phase methodology, utilizing a bidirectional LSTM for error identification and a separate LSTM-based model for rectification.

Contemporary advancements have increasingly centered around transformer-based methodologies. Duong et al. (2021) addressed OCR correction without employing a specific detection module, instead opting to train a transformer model on a dataset encompassing both accurately transcribed text and OCR-induced erroneous text. They produced synthetic OCR errors through automated techniques, including random error insertion and parallel corpus training, which involved pairing pristine texts with their artificially flawed counterparts. In a similar vein, Mei et al. (2016) confronted scenarios involving multiple OCR correction models by devising a statistical approach to rank correction candidates predicated on attributes such as Levenshtein distance and lexicon validation.

Pethő et al. (2024) conducted an extensive study on types of OCR errors specific to Hungarian. Building on this, Laki et al. (2022) used neural language models to correct OCR errors, developing a two-part system with a Context-based Character Correction (Nguyen et al., 2020) detection module and an encoder-decoder model fine-tuned to perform the corrections. They also created a manually annotated Gold Standard training corpus for model training.

Numerous studies have explored the examination and cleaning of erroneous Hungarian texts using modern language technology tools. For instance, Dömötör

<sup>1</sup> <https://sites.google.com/view/icdar2017-postcorrectionocr>

and Yang (2018) conducted research on Hungarian corpora to identify the prevalence of non-standard errors, focusing primarily on spoken language and personal subcorpora. Their findings highlighted that the most frequent errors include the omission of punctuation, diacritics, and capitalization. This observation aligns with the fact that Hungarian researchers have undertaken separate studies to address each of these issues. Concerning language correctness, the HuLU benchmark kit includes the Hungarian Corpus of Linguistic Acceptability (HuCOLA) (Ligeti-Nagy et al., 2024), which specifically assesses the grammatical accuracy of Hungarian sentences.

In Hungarian punctuation research, Tündik et al. (2018) addressed punctuation correction using RNN networks, while Yang (2021) approached it with a transformer-based machine translation method.

Diacritic restoration in Hungarian has also been a longstanding area of research. Németh et al. (2000) presented a text-to-speech application that handled words without accents. Later, Ács and Halmai (2016) developed an n-gram-based statistical system that restores diacritics without relying on language-specific dictionaries. Several solutions use machine translation techniques to restore accents: Novák and Siklósi (2015) employed statistical machine translation, while Laki and Yang (2020a,b) trained neural machine translation models to solve this problem for Hungarian and other languages.

Despite the progress, most of these models are either not freely accessible or do not align well with our dataset, necessitating the development of custom models to meet our specific needs.

### 3 Corpora

In our task, we needed to clean a variety of different types of documents in a given library. The Library of the Hungarian Academy of Sciences was established in 1826 and has been serving the members of the Academy and the broader Hungarian research community ever since. The digital collections – in the form of an open access repository – were created in 2008. This repository – named REAL – has diverse holdings, mirroring the printed collection of the library. Its collection includes materials from multiple sources, such as manuscripts, books, and scientific papers, available in formats like printed copies and digital-born versions. This diversity of input channels has resulted in a rich, mixed collection – spanning scanned documents, born-digital files, publishers’ PDFs, accepted manuscripts, as well as various handwritten documents and images. For this project, we plan to use a modern text corpus comprising about 1 billion words. The REAL Repository contains over 250,000 documents, approximately half of which are suitable for our work.

#### 3.1 Training corpora for the cleaning models

For the OCR task, we constructed a custom corpus to train our language model. To ensure accurate error detection, the corpus includes a balanced mix of erroneous and error-free text segment pairs, with a distribution of 66.4% to 33.6%,

respectively. For this, we used the corresponding error-free electronic versions of the OCR-processed texts. Table 1 presents the main characteristics of the training and test corpora. The average Character Error Rate (CER) across the entire training dataset is 12.35%, and the Word Error Rate (WER) is 11.74%, measured against the reference data (error-free sentences).

Dataset		Segments	Tokens	Avg Length
Training Set	Source	1,374,665	56,551,620	43.05
	Target	-	47,029,140	35.80
Test Set	Source	6,780	124,401	18.35
	Target	-	115,217	16.99

**Table 1.** Training and test corpora characteristics for the OCR cleaning task

For diacritic restoration, we used the same corpora as Laki and Yang (2020a) in their research on Hungarian. They chose the online available parallel corpus, Open Subtitles<sup>2</sup>, which contains texts written in 62 languages, including Hungarian. It consists of movie subtitles with many shorter, informal sentences in them. Table 2 shows the main characteristics of the training and test corpora. Diacritic characters make up 6.59% of the total characters, and 35.75% of the words contain at least one diacritic.

Dataset	Segments	Tokens	Avg Length
Training Set	28,704,830	177,588,069	6.19
Test Set	3,000	18,635	6.21

**Table 2.** Training and test corpora characteristics for the diacritic restoration task

For filtering incorrect sentences, we used the HuCOLA corpus from the HuLU benchmark collection (Ligeti-Nagy et al., 2024). The corpus contains 9076 sentences labelled for their grammaticality. Here we used the training and the validation set, with 7276 and 900 sentences, respectively. Table 3 presents the main characteristics of the training and test corpora, where the distribution of incorrect and correct sentences is 21.6% and 78.4%, respectively.

Dataset	Segments	Tokens	Avg Length
Training Set	7,276	50,068	6.88
Test Set	900	6,145	6.75

**Table 3.** Characteristics of the HuCOLA training and test corpora

<sup>2</sup> <http://opus.nlpl.eu/OpenSubtitles-v2018.php>

## 4 Models and Experiments

### 4.1 huT5 models

Based on related work (Laki et al., 2022; Laki and Yang, 2020a), we identified the encoder-decoder architecture (Vaswani et al., 2017) as the most effective for our needs. However, no pretrained encoder-decoder language models specifically for Hungarian were available. Consequently, we developed a custom model by adapting the mT5 (Xue et al., 2021) for Hungarian. We followed guidance on single-language adaptation from a multilingual T5 model<sup>3</sup>, using methods to prune redundant embeddings and reduce parameter counts with minimal quality loss. Additionally, we optimized vocabulary size based on Abdaoui et al. (2020). This process resulted in two Hungarian-specific mT5 models (huT5) in both base and large versions, which will be made available on our Huggingface site<sup>4</sup>.

Once the huT5 models were created, we fine-tuned them for OCR cleaning, diacritic restoration, and incorrect sentence filtering.

### 4.2 Text cleaning pipeline

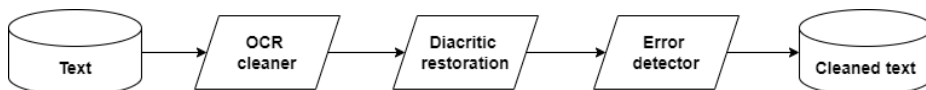


Fig. 1: Architecture of text cleaning pipeline

Using our fine-tuned, task-specific models, we can build a text cleaning pipeline to address our task. Figure 1 shows the architecture of our text cleaning pipeline. In our task, the primary errors originate from OCR; therefore, our first module is an OCR cleaner. The second major source of errors is missing or incorrect diacritics, so our second module is a diacritic restoration model. After these two modules complete the cleaning, if errors still remain, we use an erroneous sentence detector to mark or filter out the incorrect sentences.

In the next section, we will provide a detailed introduction to each of these modules.

### 4.3 Modules and models

For the OCR cleaning task, we initially tested the model from Laki et al. (2022), but it did not perform consistently across all text types in our dataset. Similarly,

<sup>3</sup> <https://towardsdatascience.com/how-to-adapt-a-multilingual-t5-model-for-a-single-language-b9f94f3d9c90>

<sup>4</sup> <https://huggingface.co/NYTK>

previous models for diacritic restoration (Laki and Yang, 2020a) were not freely available, prompting us to train custom models for these tasks.

Therefore, for these tasks, we fine-tuned custom, task-specific models based on our huT5 models. The large models were trained using two NVIDIA A100 GPUs (80GB each), while the base models were trained on four NVIDIA GeForce GTX 1080 GPUs (11GB each). We utilized the PyTorch Seq2SeqTrainer from the Transformers library<sup>5</sup> for fine-tuning. For comparison, we also trained the original mT5 models on these tasks.

The training hyperparameters for the OCR cleaning models are as follows: learning rate = 5e-5; global batch size = 256; epoch = 1; sequence length = 256.

The training hyperparameters for the diacritic restoration models are as follows: learning rate = 5e-5; global batch size = 512; epoch = 1; sequence length = 256.

For both OCR cleaning and diacritic restoration tasks, we tailored the batch sizes to the available GPU and used gradient accumulation to achieve the global batch size. Due to the larger corpus size in diacritic restoration, we selected a larger global batch size for this task.

The training hyperparameters for the HuCOLA models are as follows: learning rate = 5e-6; global batch size = 32; epoch = 10; sequence length = 128. In the HuCOLA task, we trained for 10 epochs to ensure reliable comparisons, selecting the best-performing checkpoint. In the case of the HuCOLA experiment, we did not use gradient accumulation. For better comparison, the global batch size was kept the same as in the experiment conducted by Yang et al. (2023).

In the final text cleaning pipeline, the best-performing models constitute the OCR cleaning, diacritic restoration, and erroneous filtering modules.

To evaluate our huT5 models comprehensively, we also fine-tuned both huT5 and mT5 models on the Hungarian Corpus of Linguistic Acceptability (HuRTE) and Hungarian version of the Stanford Sentiment Treebank (HuRTE) benchmarks (Ligeti-Nagy et al., 2024).

## 5 Results

Our first objective was to evaluate the huT5 models. Table 4 provides a comparison between our huT5 and mT5 models. A key result is the significant reduction in both the number of parameters (base: ~42%; large: ~68%) and model size (base: ~42%; large: ~67%) following the conversion. For quality evaluation, we used the HuCOLA, HuSST, and HuRTE benchmarks, along with accuracy as the evaluation metric. The primary focus here is on the performance difference between the two model types rather than on outperforming existing Hungarian models.

The results indicate that in all cases, the huT5 models perform similarly or better than the mT5 models, despite having fewer parameters.

Consistent with findings from Yang (2022); Yang and Váradi (2021), the encoder-decoder architecture tends to be more suitable for sequence-to-sequence

<sup>5</sup> <https://github.com/huggingface/transformers/tree/main/examples/pytorch>



tasks than for classification. This is reflected in our experiments with Hungarian benchmarks, where the models showed lower performance on classification tasks.

Model	Parameters	Size	HuCOLA	HuRTE	HuSST
huT5 base	244 million	977 MB	<b>80.98</b>	<b>55.00</b>	58.37
mT5 base	580 million	2.33 GB	<b>80.98</b>	52.66	57.94
huT5 large	820 million	3.28 GB	80.88	54.00	<b>58.79</b>
mT5 large	1.2 billion	4.92 GB	80.88	53.50	58.02

**Table 4.** Comparison of huT5 and mT5 on Hungarian benchmarks

Given that encoder-decoder models generally perform better on text generation tasks, we expect this model to be well-suited for our OCR cleaning and diacritic restoration tasks. In these experiments, we compared our models with Hungarian state-of-the-art (SOTA) models in this field, with benchmark results available in Laki et al. (2022) and Laki and Yang (2020a). The results can be seen in Table 5.

For evaluation, we used ROUGE-L See et al. (2017) (where higher scores indicate better quality  $\uparrow$ ), Word Error Rate (WER, with lower values indicating better performance  $\downarrow$ ), and Character Error Rate (CER, with lower values indicating better performance  $\downarrow$ ) Morris et al. (2004) for OCR cleaning. For diacritic restoration, we assessed model performance using precision and recall metrics.

As you can see, our huT5 large model achieved the best performance in OCR cleaning, outperforming the SOTA model. In the diacritic restoration task, none of our models surpassed the SOTA models, but we achieved competitive results and will publish our models. Notably, our huT5 models, with fewer parameters, outperformed or achieved competitive performance compared to the mT5 models.

Model	OCR Cleaning			Diacritic Restoration	
	ROUGE-L $\uparrow$	WER $\downarrow$	CER $\downarrow$	Precision $\uparrow$	Recall $\uparrow$
SOTA model	93.44	17.38	8.72	99.38	99.28
huT5 base	95.12	11.10	7.05	97.64	97.75
mT5 base	95.26	10.61	6.57	97.57	97.49
huT5 large	<b>95.66</b>	<b>10.01</b>	<b>6.46</b>	97.95	98.18
mT5 large	95.28	11.56	9.33	98.61	98.61

**Table 5.** Comparison of OCR cleaning and diacritic restoration results for huT5 and mT5 models

The third module is an erroneous sentence detector. After the previous two modules have corrected the text, any remaining errors can, depending on the task, be marked or filtered out by an error detector. For this task, the HuCOLA benchmark is the most suitable. As shown in Table 4, our model achieved only

about 80% accuracy. In comparison, the models trained by Yang et al. (2023) performed better, with the HuBERT and PULI models achieving approximately 90% and 91% accuracy, respectively. Consequently, in our final pipeline, we use the fine-tuned PULI Bert-Large model instead of our fine-tuned huT5 or mT5.

We also performed error analysis on the OCR and diacritic restoration models, which achieved the best performance.

In Table 6, the error analysis of the diacritic restoration model is presented. The typical errors are similar to those reported by Laki and Yang (2020a), with only the ratios differing. In our case, we observed a higher proportion of real errors.

There are two main categories included in the correct output: equivalent forms and correct replaceable outputs. The equivalent form (e.g., *hova-hová* (where), *tied-tiéd* (yours)) refers to cases where both forms of the given word are usable, with no difference in meaning. The replaceable outputs refer to cases where the given words have different meanings, but both are correct either in the given context or without additional context. In the case of real errors, approximately 30% of the errors stem from proper nouns, as the model is unable to correctly restore names.

Error type	Ratio	Examples (reference (ref) - prediction (pred))
<b>Correct output</b>	<b>37.5%</b>	
Equivalent form		<i>hova - hívá</i> ('where'), <i>tied - tiéd</i> ('yours')
Replaceable output		ref: <i>Fogjátok meg!</i> ('Catch him/her!')
		pred: <i>Fogjatok meg!</i> ('Catch me!')
		ref: <i>Az InStyle magazinnal.</i> ('With InStyle magazine.')
		pred: <i>Az InStyle magazinnál.</i> ('At InStyle magazine.')
Wrong reference		ref: <i>Két kijárat, egy elől</i> ('from'), <i>egy hátul.</i>
		pred: <i>Két kijárat, egy elől</i> ('in front'), <i>egy hátul.</i>
<b>Real errors</b>	<b>62.5%</b>	
Proper noun		<i>Livúró - Liuról, Ramával - Rámával</i>
Wrongly replaced		<i>még - meg, teli - téli</i>

**Table 6.** Error analysis of the diacritic restoration model

In Table 7, the error analysis of the OCR cleaning model is presented. Notably, 21% of the reported errors are actually false positives, where the reference text was incorrect, and the model provided the correct output. The remaining 79% of the errors represent real mistakes. These errors can be classified into three main categories: insertion, deletion, and replacement:

- Insertion: This occurs when the model adds an incorrect character to a word. Common cases involve the addition of extra punctuation marks or entirely incorrect letters. Additionally, the model may erroneously insert a space within a word, leading to segmentation errors. Finally, there are cases where the model inserts words that did not originally exist in the text.

- Deletion: In this category, the model removes a character that should have been retained. Similar to the insertion category, these deletions can involve punctuation marks, letters, spaces, or entire words. Deletion of spaces often results in the unintended merging of two words, while the deletion of whole words leads to a loss of information.
- Replacement: This category encompasses two primary types of errors: the substitution of an incorrect punctuation mark or letter. A notable subtype of replacement errors involves the mishandling of diacritic marks, where the model incorrectly replaces accented characters. Additionally, there are cases where the model replaces an entire word with a completely unrelated one, resulting in significant semantic deviations.

This analysis highlights specific patterns of error generation, providing insights for targeted model improvements.

Error type	Ratio	Examples (reference (ref) - prediction (pred))
<b>Correct output</b>	<b>21.38%</b>	<i>pcdig - pedig</i> <i>minder.kinek - mindenkinek</i> <i>színház- színház</i>
<b>Real errors</b>	<b>78,62%</b>	
Insertion	23,83%	<i>írva - írva:</i> (punctuation mark) <i>darab - dakrab</i> (letter) <i>mennybéli - menny béli</i> (space) <i>németalföldi - német 376 alföldi</i> (words)
Deletion	24,44%	<i>halastó, - halastó</i> (punctuation mark) <i>vesszővel - vesszvel</i> (letter) <i>már most - mármost</i> (space) <i>Hozott Isten! - [ ]</i> (words)
Replacement	30,35%	<i>bort, - bort;</i> (punctuation mark) <i>előbb - elébb</i> (letter) <i>színes - szines</i> (diacritic) <i>Mi - Ali</i> (word)

**Table 7.** Error analysis of the OCR cleaning

## 6 Challenges and limitations

Adapting mT5 to huT5 presented several challenges, particularly in addressing the unique linguistic features of Hungarian, such as complex morphology and extensive diacritic use. These characteristics required substantial pre-processing to enhance model robustness, especially for tasks like diacritic restoration and grammatical error detection. Additionally, as Hungarian is relatively low-resource, the limited availability of high-quality annotated datasets constrained training, which may affect model performance on domain-specific or informal text corpora.

Another challenge was the computational demand required for model adaptation and fine-tuning. Despite efforts to reduce parameter count, the large huT5 version remains resource-intensive, potentially limiting its accessibility to groups without advanced hardware. Additionally, adapting huT5 to specific tasks required careful tuning to avoid overfitting due to dataset limitations.

Lastly, current evaluation metrics like WER, CER, and ROUGE-L offer a partial view of improvements in text quality and readability. Developing refined evaluation metrics tailored to Hungarian text cleaning could help better capture the model’s impact.

## 7 Conclusion

In this study, we addressed the essential task of text cleaning for Hungarian using custom transformer-based models. The results of our OCR cleaning, diacritic restoration, and incorrect sentence filtering tasks highlight the effectiveness and adaptability of our huT5 models, showing both improved performance and resource efficiency over existing models. By adapting the mT5 model specifically for Hungarian, we achieved substantial reductions in parameter counts and model size, with a remarkable 42% reduction for the base model and 68% for the large model. This optimization not only maintained but also, in many cases, improved the model’s performance on Hungarian benchmarks, particularly in sequence-to-sequence tasks.

Our results confirm that the huT5 models are well-suited for a range of text cleaning tasks. Compared to the original mT5, the huT5 models consistently achieved better scores on Hungarian OCR cleaning and diacritic restoration, as shown by lower WER and CER values and higher ROUGE-L scores. Additionally, the high precision and recall in diacritic restoration and the solid performance across all metrics make the huT5 models a strong candidate for state-of-the-art solutions in Hungarian text cleaning.

This work also provides a valuable, freely accessible alternative for Hungarian-language tasks, meeting gaps left by existing models. The encoder-decoder architecture used in our approach effectively addresses both sequence-to-sequence and error detection needs, presenting a refined tool for improving data quality in Hungarian NLP applications.

Future work may extend these results by testing the huT5 models’ adaptability to additional languages or dialects. Nonetheless, this study establishes a new standard for Hungarian text cleaning, showing that transformer-based approaches, when tailored to the specific language requirements, can achieve both high accuracy and efficiency.

## Acknowledgment

The research was supported by the MTA “Tudomány a Magyar Nyelvért Nemzeti Program” (Science for the Hungarian Language National Programme).

## References

- Abdaoui, A., Pradel, C., Sigel, G.: Load what you need: Smaller versions of multilingual BERT. In: Moosavi, N.S., Fan, A., Shwartz, V., Glavaš, G., Joty, S., Wang, A., Wolf, T. (eds.) *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. pp. 119–123. Association for Computational Linguistics, Online (Nov 2020), <https://aclanthology.org/2020.sustainlp-1.16>
- Duong, Q., Hämäläinen, M., Hengchen, S.: An unsupervised method for OCR post-correction and spelling normalisation for Finnish. In: *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*. pp. 240–248. Linköping University Electronic Press, Sweden, Reykjavik, Iceland (Online) (May 31–2 Jun 2021)
- Dömötör, A., Yang, Z.Gy.: Így írtok ti: nem sztenderd szövegek hibátípusainak detektálása gépi tanulásos módszerrel [This is how you write: detecting error types in non-standard texts using machine learning]. XIV. Magyar Számítógépes Nyelvészeti Konferencia pp. 305–316 (2018)
- Laki, L.J., Kőrös, A., Ligeti-Nagy, N., Nyéki, B., Vadász, N., Yang, Z.Gy., Váradi, T.: OCR-hibák javítása neurális technológiák segítségével [Correcting OCR errors using neural technologies]. In: XVIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2022). pp. 417–430. Szegedi Tudományegyetem, Informatikai Intézet, Szeged, Hungary (2022)
- Laki, L.J., Yang, Z.Gy.: Automatic Diacritic Restoration With Transformer Model Based Neural Machine Translation for East-Central European Languages. In: *Proceedings of the 11th International Conference on Applied Informatics (ICAI 2020)*. pp. 190–202. Eger, Hungary (2020a), <http://ceur-ws.org/Vol-2650/>
- Laki, L.J., Yang, Z.Gy.: Automatikus ékezetvisszaállítás transzformer modellen alapuló neurális gépi fordítással [Automatic accent restoration using neural machine translation based on transformer model]. XVI. Magyar Számítógépes Nyelvészeti Konferencia pp. 181–190 (2020b)
- Ligeti-Nagy, N., Ferenczi, G., Héja, E., Laki, L.J., Vadász, N., Yang, Z.G., Váradi, T.: HuLU: Hungarian language understanding benchmark kit. In: Calzolari, N., Kan, M.Y., Hoste, V., Lenci, A., Sakti, S., Xue, N. (eds.) *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. pp. 8360–8371. ELRA and ICCL, Torino, Italia (May 2024), <https://aclanthology.org/2024.lrec-main.733>
- Mei, J., Islam, A., Wu, Y., Moh’d, A., Milios, E.E.: Statistical learning for OCR text correction. CoRR abs/1611.06950 (2016), <http://arxiv.org/abs/1611.06950>
- Morris, A., Maier, V., Green, P.: From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. pp. 2765–2768 (10 2004)
- Németh, G., Zaikó, Cs., Fekete, L., Olaszy, G., Endrédi, G., Olaszi, P., Kiss, G., Kis, P.: The design, implementation, and operation of a hungarian e-mail

- reader. *International Journal of Speech Technology* 3(3), 217–236 (Dec 2000), <https://doi.org/10.1023/A:1026567216832>
- Nguyen, T.T.H., Jatowt, A., Nguyen, N.V., Coustaty, M., Doucet, A.: Neural Machine Translation with BERT for Post-OCR Error Detection and Correction. In: *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*. p. 333–336. JCDL '20, Association for Computing Machinery, New York, NY, USA (2020)
- Novák, A., Siklósi, B.: Automatic diacritics restoration for Hungarian. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pp. 2286–2291. Association for Computational Linguistics, Lisbon, Portugal (Sep 2015), <https://www.aclweb.org/anthology/D15-1275>
- Pethő, G., Sass, B., Simon, L., Lipp, V.: OCR-hibák kvantitatív elemzése több szövegváltozat összehasonlításával [Quantitative analysis of OCR errors through the comparison of multiple text versions]. In: *XX. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2024)*. pp. 17–29. Szegedi Tudományegyetem, Informatikai Intézet, Szeged, Hungary (2024)
- Rigaud, C., Doucet, A., Coustaty, M., Moreux, J.P.: Icdar 2019 competition on post-ocr text correction. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. pp. 1588–1593 (2019)
- Schaefer, R., Neudecker, C.: A two-step approach for automatic OCR post-correction. In: *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*. pp. 52–57. International Committee on Computational Linguistics, Online (Dec 2020)
- See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: Barzilay, R., Kan, M.Y. (eds.) *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 1073–1083. Association for Computational Linguistics, Vancouver, Canada (Jul 2017), <https://aclanthology.org/P17-1099>
- Tündik, M.A., Tarján, B., Szaszák, G.: Televíziós feliratok írásjeleinek visszaállítása rekurrens neurális hálózatokkal [Restoration of punctuation marks in television subtitles using Recurrent Neural Networks]. *XIV. Magyar Számítógépes Nyelvészeti Konferencia* pp. 183–195 (2018)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is All you Need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 30, pp. 5998–6008. Curran Associates, Inc. (2017), <http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>
- Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., Raffel, C.: mT5: A massively multilingual pre-trained text-to-text transformer. In: Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y. (eds.) *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

- pp. 483–498. Association for Computational Linguistics, Online (Jun 2021), <https://aclanthology.org/2021.naacl-main.41>
- Yang, Z.Gy.: Automatikus írásjelek visszaállítása és Nagybetűsítés statikus korpuszon, transzformer modellen alapuló neurális gépi fordítással [Automatic punctuation restoration and capitalization on a static corpus using transformer-based neural machine translation]. pp. 225–232. Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szeged (2021)
- Yang, Z.Gy.: BARTerezzünk! - Messze, messze, messze a világtól, - BART kísérleti modellek magyar nyelvre [Let's BART! - Far, far, far away from the world, - BART experimental models for Hungarian]. In: XVIII. Magyar Számítógépes Nyelvészeti Konferencia. pp. 15–29. Szegedi Tudományegyetem, Informatikai Intézet, Szeged, Magyarország (2022)
- Yang, Z.Gy., Dodé, R., Ferenczi, G., Héja, E., Jelencsik-Mátyus, K., Kőrös, A., Laki, L.J., Ligeti-Nagy, N., Vadász, N., Váradi, T.: Jönnek a nagyok! BERT-Large, GPT-2 és GPT-3 nyelvmodellek magyar nyelvre [The giants are coming! BERT-Large, GPT-2, and GPT-3 language models for Hungarian]. In: XIX. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2023). pp. 247–262. Szegedi Tudományegyetem, Informatikai Intézet, Szeged, Hungary (2023)
- Yang, Z.Gy., Váradi, T.: Training language models with low resources: RoBERTa, BART and ELECTRA experimental models for Hungarian. In: Proceedings of 12th IEEE International Conference on Cognitive Infocommunications (CogInfoCom 2021). pp. 279–285. IEEE, Online (2021)
- Ács, J., Halmaj, J.: Hunaccent: Small Footprint Diacritic Restoration for Social Media. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). pp. 3526–3529. Portoroz, Slovenia (2016)





# Középkori nyelvek feldolgozása ingyenes és kereskedelmi generatív nyelvmodellekkel

Pethő Gergely<sup>1,2</sup>, Swaroop Krishna<sup>3</sup>

<sup>1</sup>Debreceni Egyetem, Egy Egészség Intézet

<sup>2</sup>Regensburgi Egyetem, Romanisztikai Intézet

<sup>3</sup>Debreceni Egyetem, Informatikai Kar

**Kivonat** A középkori szövegek számítógépes feldolgozása hagyományosan számos nehézségbe ütközik, amelyek főleg két okra vezethetők vissza. Az egyik a szükséges erőforrások hiánya akár szoftvertermékek (morfológiai és szintaktikai elemzők), akár adatok (elektronikus szótárak, kézzel annotált tanítóadatok) tekintetében. A másik a nyelvi sztenderd hiánya, ami kiterjed mind a szabályozott és következetes helyesírás hiányára, mind pedig „egy” adott nyelv számottevő szinkrón és diakrón változatosságára. Kutatásunkban azt vizsgáltuk, hogy a napjainkban divatos „kis” és „közepes” méretű ingyenes, saját hardveren futtatható generatív nyelvmodellek, amelyeket többnyelvű modern nyelvi adatokon tanítottak elő, mennyire alkalmasak történelmi nyelvek feldolgozására a nagy kereskedelmi modellekkel összevetve. Ehhez Albucasis sebészetről szóló tankönyvének arab eredetije, valamint annak latin, ófrancia és óokcitan nyelvű középkori változatai alapján e modellekkel készített gépi fordításokat értékeltünk ki a mű filológiai igényű angol fordítását referenciaként használva. Az eredmények segítenek annak megítélésében, hogy középkori nyelvek feldolgozásakor milyen generatív nyelvmodellek használatával érdemes próbálkozni a számítási erőforrásokat és a költségeket is figyelembe véve.

**Kulcsszavak:** generatív nyelvmodel, LLM, SLM, nyelvtörténet, alacsony erőforrású nyelvek

## 1. Bevezetés

A történelmi szövegek feldolgozásában a megszokott nyelvtchnológiai megoldások alkalmazása komoly nehézségekbe ütközik. A hagyományos szótár- és szabályalapú eszközök fejlesztésének és alkalmazásának fő akadályát a korai nyelvéllapotok „szabályozatlansága”, a nyelvi sztenderd hiánya jelenti. Ez kiterjed mind a helyesírási szabályok, illetve általánosan követett szokások hiányára, mind pedig arra, hogy a nyelvet egységesítő kulturális termékek (könyvnyomtatás, közoktatás, sajtó stb.) hiányában a szinkrón nyelvéllapotokon belül jelentősen nagyobbak voltak a területi vagy akár egyéni eltérések, mint az újkortól kezdve. A szinkrón állapoton belüli változatosság mellett a diakrón eltérések is számottevőek, így még az egyazon nyelvtörténelmi korszakban keletkezett írások nyelvi

jellemzői is drasztikus eltéréseket mutathatnak. Mindezen tényezők kizárják a hagyományos számítógépes nyelvfeldolgozási megoldások átfogó alkalmazását.

A gépi tanuláson, ezen belül különösen az újabb neurális nyelvmodelleken alapuló eszközök rendelkeznek olyan jellemzőkkel, amelyek alkalmassá teszik ezeket az ilyen értelemben „problémás” szövegek feldolgozására. Működésük nagyjából illeszkedő minták illesztésén nyugszik, így például egy generatív nyelvmodell olyan folytatást generál (azaz következő tokent, illetve iteratív módon ilyenek sorozatát) egy adott prompthoz (azaz bemenetként kapott tokensorozathoz), amely hasonlít azokhoz a tokensorozatokhoz, amelyek a tanítókorpuszában a prompthoz hasonló, azonban vele rendszerint nem pontosan egyező tokensorozatok után következtek. Ennek köszönhetően a neurális nyelvmodellek jól ellenállnak a feldolgozandó szövegben megtalálható zajnak, írásmódbeli („helyesírási”), morfológiai, szóhasználatbeli anomáliáknak. Ez a képesség, amely a neurális nyelvmodelleket alkalmassá teszi a sztenderdtől eltérő nyelvváltozatokban íródott nyelvi termékek, így például a közösségi médiában funkcionálisan írástudatlan nyelvhasználók által előállított szövegek értelmezésére, segítségükre lehet a régi szövegek feldolgozásában is.

Ugyanakkor a korai történelmi nyelvállapotokkal kapcsolatban nehézséget jelent, hogy egyrészt kevés olyan nyelvmodell áll rendelkezésre, amely kifejezetten ilyen nyelvű szövegeken lett tanítva, másrészt a tanításra rendelkezésre álló szövegek mennyisége eleve erősen korlátozott még a gazdag nyelvméltanyaggal rendelkező nyelvek esetében is. Az összes ómagyar szövegemlék (MGTSZ korpusz) terjedelme például mindössze 3 millió szövegszó, míg a középkori francia korpusz (BFM) mintegy 6 millió szövegszót foglal magában. Ez legalább 2-3 nagyságrenddel kevesebb egy használható neurális nyelvmodell tanításához szükséges adatmennyiségnél. A középkori nyelvek feldolgozására képes neurális nyelvmodellek fejlesztése így a jelenlegi architektúrák és tanítási elvek (azaz rekurrens vagy transzformerhálózatok gradiensmódszerrel, maszkolással tanítással) mellett a közismert skálázási szabályokat figyelembe véve (Kaplan és mtsai, 2020; lásd még Hoffmann és mtsai, 2024) gyakorlatilag reménytelen.

Tanulmányunkban azt járjuk körül, hogy a döntően modern nyelvi szövegeken tanított nyelvmodellek alkalmasak-e középkori nyelvek feldolgozására. Más szóval azt vizsgáljuk, hogy történelmi szövegek mennyire sikeresen modellezhetőek eltorzított, zajos modern nyelvi szövegekként. E célkitűzéssel kapcsolatban felmerül ugyanakkor az a módszertani kérdés, hogy miként lehet számszerűen mérni és összehasonlítani az egyes modellek használhatóságát, tekintve, hogy nem állnak rendelkezésre (tudomásunk szerint) ilyen benchmarkok. Ezek hiányában a többnyelvű generatív nyelvmodellek egy jól ismert képességét aknáztuk ki nyelvtudásuk tesztelésére: az adott nyelvből angolra fordítottattunk velük. Bár vizsgálatunk eredményei közvetlenül csak a modellek fordítási képességéről adnak tájékoztatást, a generatív nyelvmodellek univerzális alkalmazhatóságával kapcsolatos tapasztalatok alapján abból indulhatunk ki, hogy arra is következtetni engednek, megfelelő promptolás vagy finomhangolás mellett adott modell más célokra, például morfológiai, szintaktikai címkézésre, bizonyos jelentéskategóriákba eső szavak (pl. tulajdonnevek) felismerésére stb. használhatóvá tehető-e.

Vizsgálatunk egy olyan, viszonylag nagy terjedelmű szövegen alapul, amelynek arab nyelvű eredetije a 10. század végén íródott, ennek a 13. században készült el latin nyelvű fordítása, majd feltehetőleg ez utóbbi fordítás alapján két vernakuláris fordítás ófrancia és óokcitán nyelvre. Ennek köszönhetően egyazon szöveg négy különböző középkori változata alapján készült gépi fordítások minőségét tudjuk összehasonlítani. Így lehetővé válik, hogy nemcsak a vizsgált nyelvmodellek fordítási képességét vessük össze, hanem azt is, hogy különböző középkori nyelvek egymáshoz képest mennyire sikeresen dolgozhatók fel velük.

## 2. Kapcsolódó irodalom

A nagy generatív nyelvmodelleknek a középkori, illetve általánosabban a történelmi nyelvek és nyelvállapotok feldolgozására való használata feltáratlan terület, nem tudunk átfogó kutatásról e kérdéskörrel kapcsolatban.

A klasszikus arab nyelv nagy generatív nyelvmodellekkel történő feldolgozására nincs példa a szakirodalomban. Az e nyelvvel foglalkozó legújabb szakirodalom csak kisebb, elsősorban modern arab köznyelvi anyagra előtanított encodernyelvmodellek alkalmazásáról számol be, amelyek tanítását kiegészítették korlátozott mennyiségű klasszikus arab szöveggel is (Malhas és Elsayed 2022, ElKoumy és Sarhan 2024), illetve találunk példát kifejezetten klasszikus arabra tanított BERT-modellre is (Inoue és mtsai, 2021).

Ókori nyelvek, többek között a latin gépi tanulási eszközökkel történő feldolgozásáról közölt a közelmúltban átfogó áttekintést Sommerschild és mtsai (2023). Generatív nyelvmodellek alkalmazására nem találunk példát. Volk és mtsai (2024) azt vizsgálja egy igen kis terjedelmű, mindössze 3000 szövegszós tesztanyagban, hogy kora újkori latin és német nyelvű írások fordítására mennyire alkalmas a GPT-4 a Google Fordítóval és más gépi fordító eszközökkel összevetve. Stüssi (2023) és Stüssi és Ströbel (2024) alapos, szisztematikus munka, amely különböző ChatGPT-verziók használhatóságát vizsgálja 16. századi latin szövegek tokenjeinek szófaji címkézésére.

Torres Aguilar (2022) túlnyomórészt modern anyagon tanított többnyelvű encodermodeleket használ középkori, többek között latin és ófrancia nyelvű szövegekben tulajdonnevek felismerésére.

Her és Kruschwitz (2024) és Ondrejová és Šuppa (2024) azt a kérdést vizsgálja, hogy modern nyelvek alacsony erőforrású nyelvjárásait (a német nyelv bajor, illetve a szlovák nyelv šariš dialektusát) mennyire sikeresen képesek feldolgozni generatív nyelvmodellek.

A jelen tanulmány közvetlen előzménye két konferencia-előadás, amelyek keretében ismertettük eljárásunkat és az ófranciára és óokcitánra vonatkozó, e tanulmányban is idézett eredményeinket (Pethő és mtsai, 2024b), illetve az arabra és latinra vonatkozókat (Pethő és mtsai, 2024a).

### 3. Anyagok

Vizsgálatunk alapjául a középkori Európában leginkább Albucasis néven ismert 10. századi andalúziai arab orvos 30 kötetes, *Kitáb at-taszríf* című orvosi enciklopédiájának utolsó, egyben legnagyobb terjedelmű, *A sebészetről és az eszközökről* című kötete szolgált. Albucasis *Sebészete* (ahogy a továbbiakban hivatkozunk rá) az egyik legfontosabb orvosi tankönyv volt a második évezred első évszázadaiban, jelentősége és hatása révén számos másolatban és több fordításban fennmaradt. Vizsgálatunk során azt elemeztük, hogy a *Sebészet* eredetijének és három különböző középkori európai fordításának különböző generatív nyelvmódellekkel készült angol nyelvű gépi fordításai mennyire jól közelítenek az arab mű kritikai kiadásában (Spink és Lewis, 1973) közölt, tudományos igénnyel készült angol nyelvű referenciafordításhoz. A *Sebészet* három könyvön belül 200 fejezetből áll. A terjedelem szövegszavakban mérve nyelvenként eltérő, wc -w szerint: arab 59873, latin 75582, francia 77120, okcitan 99528, angol 91283. A tokenszám egy találmásra kiválasztott tokenizáló (a Google Gemma 2 9b-é) szerint: arab 126002, latin 145723, francia 122661, okcitan 154918, angol 110222.

Albucasis tankönyve klasszikus arab nyelven íródott. Ez a 9. század során nyelvten- és szótárírók által kanonizált, onnantól kezdve az iszlám szent nyelvként változatlan formában fennmaradt nyelv mind szintaxisában, mind morfológiájában igen közel áll a 19. században kialakult modern arab köznyelvhez, amely összarab műveltségi nyelvként funkcionál napjainkig.

Az arab eredeti mellett a mű 1200 körül keletkezett latin, 13. századi ófrancia és 14. századi óokcitan nyelvű fordítását dolgoztuk fel. Albucasis más teljes középkori fordítása nem ismert. Ugyan a mű fordításainak szöveg hagyományát még nem tárta fel a kutatás, a szakirodalom (pl. Trotter, 1999, Green, 2011) feltételezi, hogy mind az ófrancia, mind az óokcitan változat nem közvetlenül az arab eredetén, hanem a latin fordításon alapul.

A latin fordítás több mint 20 másolatban középkori kéziratként máig fennmaradt (ennek részleteiről lásd Green, 2011), kritikai kiadása nem készült. Latin elektronikus szöveggént egy 1532-es nyomtatvány alapján a würzburgi egyetemen Irina Galynina által készített XML-dokumentumot használtunk. Az ófrancia fordítás kritikai kiadása Trotter (2005), ezen alapul a francia középkori korpusz (BFM) számára Alexei Lavrentiev által készített XML-változat, amelyet használtunk. Az óokcitan fordításnak több kritikai kiadása is készült, ezek közül a legpontosabb a szakma véleménye szerint Elsheikh (1992). Az ez alapján P.T. Ricketts által készített, majd Dominique Billy által TEI formátumra hozott XML-t használtuk a feldolgozás során. Az arab eredeti már említett kritikai kiadása alapján szintén Würzburgban Samer Alsaj és Azzam Hasan által készített XML-változattal dolgoztunk.

Amint már említettük, referenciafordításként az arab mű kritikai kiadásában (Spink és Lewis, 1973) szereplő angol fordítást használtuk. E változatokon kívül tudunk még az arab eredetinek egy 18. századi tudományos latin, egy 19. század végi modern francia, valamint egy, az 1980-as években készült filológiai orosz fordításáról, azonban ezeket nem használtuk.

## 4. Módszer

### 4.1. Előfeldolgozás

A négy középkori nyelv esetében az említett XML-dokumentumokból indultunk ki. A referenciafordításként használt Spink és Lewis-féle angol nyelvű fordításnak nem rendelkezünk digitalizált változatával, ezért ezt magunk készítettük el: A kritikai kiadás PDF-változatából a PyMuPDF (<https://github.com/pymupdf/PyMuPDF>) szoftver segítségével kinyertük az angol nyelvű oldalak meglévő OCR-szövegét, automatikusan eltávolítottuk a lábjegyzeteket, oldalszámokat, végül részben automatizáltan, részben kézzel olyan XML-szerkezetet alakítottunk ki a szövegben, amely pontosan tükrözte az arab változatnak megfelelő fejezetegységeket. Ugyanezen fejezetek az okcitan kéziratban (és az XML-ben) címmel és fejezetszámmal jelölve voltak, azonban a dokumentum nem volt ezek mentén XML-elemekre tagolva, így ezeket hozzá kellett adni. A francia XML ugyan már tartalmazott fejezetelemeket, ezek számozása és határai azonban eltértek a másik három nyelvi változat egységes szerkezetétől, ezért ezeket kézzel hozzáigazítottuk az utóbbiakhoz. Végeredményként azonos szerkezetű, azonos módon tagolt, fejezetszinten illesztett XML-eket kaptunk.

Az előfeldolgozás következő lépése a mondatokra tagolás volt. A meglévő XML-dokumentumok mondatelemeket nem tartalmaztak. A bekezdéselemeket a négy indoeurópai nyelv esetében az XML-ben szereplő központosítás alapján a SpaCy angol, illetve (a három másik nyelvben) többnyelvű modelljével bontottuk automatikusan mondatokra. Az ófrancia kritikai kiadásban annyira hosszú, esetenként oldalnyi egységek voltak mondatvégi ponttal jelölve, hogy itt a SpaCy tagolása nem volt elegendő, így az utóbbival kapott „mondatokat” tovább bontottuk a kettőspontok és pontosvesszők mentén. Az arab kritikai kiadás és XML nem tartalmazott mondatvégi központosítást, így ott ugyanez nem volt járható. Különböző megoldásokkal próbálkoztunk, többek között azzal, hogy LLM-ekkel illesszük a Spink és Lewis-féle, már mondatokra tagolt fordításhoz az arab szöveget, illetve tagoltassuk mondategységekre a bekezdéseket, de végül kézi feldolgozás mellett döntöttünk. Az XML-t arab anyanyelvű munkatársunk, Esra' Abdelzاهر bontotta a klasszikus arab hagyomány értelmében mondatnak minősülő egységekre. Amint utólag kiderült, ez olyan sajátos egységekhez vezetett, amelyekben a mondat- és tagmondathatárok sok esetben markánsan eltértek az indoeurópai nyelvek megfelelő mondatainak tagolásától, így azokra sokszor nem illeszkedtek pontosan.

A mondattagolás nyomán az angolban 3486, az arabban 2343, a latinban 4055, a franciában 3621, az okcitanban 3097 mondategységet kaptunk.

### 4.2. Mondatszintű illesztés

A négy történelmi szöveget mondatszinten automatikusan az angol referenciafordításhoz illesztettük fejezetenként haladva. Ehhez a SentAlign alkalmazást használtuk (Steingrímsson és mtsai, 2023). Mivel nem lehettünk biztosak abban, hogy a SentAlign által használt többnyelvű mondatbeágyazás-modell elég jól tud

a négy történelmi nyelven ahhoz, hogy sikeresen illeszteni tudja a szövegeket, előbb a történelmi szövegeket egy tetszőleges nagy kereskedelmi nyelvmodellel (történetesen kényelmi okokból arab: Gemini Pro; latin, okcitan: GPT-4o; francia: Claude Sonnet) szigorúan mondategységenként lefordítottuk angolra, majd az így kapott, előzetes ellenőrzésünk alapján általunk jó minőségűnek ítélt gépi fordítást illesztettük a szintén angol referenciához. A SentAlign futása mintegy 3 percet vett igénybe nyelvpáronként. Ugyan kiváló minőségű illesztéseket hozott létre, mindemellett minden nyelvpár esetében illesztetlenül hagyott olyan mondatokat, amelyek ténylegesen valamelyik oldalon 3-nál vagy 4-nél több mondategységet tartalmazó 1:n vagy m:n biszegmenst alkottak. Ezért minden nyelvpár esetében az illesztéshez használt angol gépi fordítást és a referenciafordítást összevetve ellenőriztük az összes illesztetlen mondategységet, és szükség esetén kézzel javítottuk az illesztést.

Az illesztés nyomán az arab-angol párra 2173, a latinra 2881, a franciára 2517, az okcitanra 2806 biszegmenst kaptunk.

### 4.3. A gépi fordításhoz használt modellek

A 1. táblázatban felsorolt modelleket értékeltük ki.

Modell neve	Paraméterek száma (mrd.)	Ingyenes?
Aya 23	8 és 35	+
Aya Expanse	8	+
Claude 3.5 Sonnet	ismeretlen	-
Command R	35	+
Gemini 1.5 Flash 8b, Flash, Pro	8, illetve ismeretlen	-
Gemma	9	+
Gemma 2	9 és 27	+
Granite 3 Dense	2 és 8	+
GPT-4o (ChatGPT)	ismeretlen	-
Llama 3	8 és 70	+
Llama 3.1	8	+
Llama 3.2	3	+
Mistral	7	+
Mistral-Nemo	12	+
Mistral-Small	22	+
Mixtral	8 x 7	+
Nemotron-Mini	4	+
Phi-3	3,8 és 14	+
Phi-3.5	3,8	+
Qwen2	7	+
Qwen2.5	7, 14 és 32	+

1. táblázat. Kiértékelt modellek

Mindhárom fontos kereskedelmi modell nagy változata (Claude 3.5 Sonnet, Gemini 1.5 Pro, ChatGPT GPT-4o) mellett számos különböző méretű modellt vizsgáltunk, amelyek az Ollama alkalmazással (<https://ollama.com/>) könnyedén telepíthetők és futtathatók GPU-val (grafikus processzorral) felszerelt személyi számítógépeken. Az általunk tesztelt modellek mai mércével mérve többnyire a kis-közepes méretkategóriába tartoznak, a legkisebbek (2-4 milliárd paraméter) pedig kifejezetten ún. kis nyelvmodellnek (SLM) minősülnek.

A ChatGPT és a Claude esetében a forrásnyelvi szövegrészeket az ingyenes, böngészőben elérhető felületen adtuk át a modelleknek, azokat kézzel a csevegőmezőbe másolva, majd a generált fordítást a böngészőből egy szöveges dokumentumba másoltuk. A Geminit a Google API-n keresztül, ingyenes Google Cloud kreditekkel fizetve használtunk. Minden más modellt az Ollama alkalmazással futtattunk részben lokális asztali számítógépeken, részben a HUN-REN felhőben rendelkezésünkre álló két V100 GPU-s virtuális gépen.

A 70 milliárd paraméteres Llama 3 kivételével minden ingyenes modellt az Ollama modellkönyvtárban alapértelmezett 4 bites kvantált változatban (Q4\_0) használtuk, ezen belül az utasításkövetésre hangolt (instruct) változatban. Néhány találmányra kiválasztott modell esetében kiértékeljük próbaképpen a 8 bites kvantált és a teljes 16 bites lebegőpontos (fp16) változatot is, de miután semmilyen eltérést nem tapasztaltunk a fordítások minőségében a 4 bites kvantált modellekhez képest, az utóbbiakkal vittük végig a vizsgálatot.

A kiértékelte modellek kiválasztása tekintetében döntő szempontnak tekintettük, hogy ezek futtatásához ne legyen szükség szuperszámítógépes környezetre, hanem egy viszonylag olcsó, néhány száz eurós kereskedelmi grafikus kártyán is működjének. Ezek körében tipikus RAM-felszereltségek a következő modellek futtatására elegendőek: 8 GB memóriájú grafikus kártyákon a 8 milliárdig terjedő paraméterszámú modellek, 12 GB GPU-memóriával a 14 milliárdig terjedőek, 24 GB-tal (két darab 12 GB-os grafikus kártyával felszerelt PC-n) a 32 milliárdig terjedőek futtathatók. A 70 milliárd paraméteres Llama 3 modell csak legfeljebb 2 bites kvantált változatban fut az általunk vizsgált legnagyobb grafikus memóriával (32 GB) rendelkező konfiguráción, így azt teszteltük. (Számunkra tökéletesen lényegtelen, hogy e modell 4 bites kvantált változata milyen teljesítményt nyújtott volna, ugyanis annak használatához bő 40 GB GPU-memóriára lenne szükség, ami lokális hardveren gyakorlatilag elérhetetlen.) Az ennél nagyobb modellek, pl. a 72b-s Qwen2 legkisebb kvantált változata sem futott 32 GB-on, így a 70b-nél nagyobb modelleket egyáltalán nem vizsgáltuk. Ugyan az Ollama úgy is képes futtatni nyelvmodelleket, hogy a modell nem fér el teljes mértékben a GPU-RAM-ban, és ilyen esetekben részben CPU-n, részben GPU-n futtatja a modellt, annak áteresztőképessége olyan drasztikusan csökken, hogy gyakorlatilag nem használhatóak. Albucasis *Sebészet*-ének egy teljes fordítása 100%-ban GPU-n modelltől és hardvertől függően tipikusan 1-4 órát vesz igénybe.

A modellek kiértékelésére általánosan használt sztenderd szemantikai, matematikai stb. benchmarkokon (pl. *MMLU*, Hendrycks és mtsai, 2021a; *GPQA*, Rein és mtsai, 2023; *HellaSwag*, Zellers és mtsai, 2019; *Winogrande*, Sakaguchi és mtsai, 2021; *MATH*, Hendrycks és mtsai, 2021b stb. stb.) az az általános ta-

pasztalet és gyakorlat (vö. pl. Zhao és mtsai, 2024, Yang és mtsai, 2024), hogy a hasonló méretű modellek egymással tudnak versenyezni, és a jóval nagyobb modellek (pl. 14 milliárd szemben a 8 paraméterrel) mért teljesítménye világosan felülmúlja a kisebbekét. Előzetesen azt vártuk, hogy ez a középkori nyelvek feldolgozására is érvényes lesz.

Az ingyenesen használható modellek egy részét kifejezetten többnyelvűként hirdetik a készítőik, ilyen az Aya (23 nyelv), a Qwen 2 (29), a Qwen 2.5 (konkrét szám nélkül, többek között arab és francia), a Llama 3 család (30), a Mixtral (5, többek között francia, olasz, spanyol), a Command R („több mint 10”). A többi modell feltehetőleg erőteljesen angol központú tanítóanyag alapján. Feltételeztük, hogy a dedikáltan többnyelvű modellek jelentősen jobban teljesítenek a többihez képest.

Amint látni fogjuk, az említett előzetes várakozások egyikét sem igazolták végül a mérések.

#### 4.4. Promptolás

Mind a négy nyelvhez egy-egy angol nyelvű kezdő promptban összefoglaltuk a feladatot: Albucasis középkori sebészeti tankönyvének, illetve latin stb. nyelvű fordításának mondatait kell lefordítani angolra. A modellnek a megadott szöveg fordításával kell válaszolnia, semmi mást ne fűzzön hozzá. Az ófrancia és óokcitán fordítás kezdő promptja ezenkívül egy 4-shot példát is tartalmazott, tehát négy-négy ófrancia, illetve óokcitán mondatot, valamint mindegyik mondat után annak egy tetszőleges kereskedelmi LLM-mel (történetesen a GPT-4o-val) készített jó minőségű angol fordítását. A kezdő promptot követően a fordítandó mondat egységeket egyenként adtuk át az Ollama Python-könyvtárával a nyelvmodellnek, válaszként egysoros fordítást vártunk. Ha nem egyetlen sorral válaszolt, a kontextust újra átadtuk legfeljebb három alkalommal, majd ha még ezután sem kaptunk egysoros választ, a kapott válasz első sorát tekintettük a modell által adott fordításnak. Ilyen esetekben, valamint akkor, ha a csevegés előzményei (tehát a felhasználó és a modell „hozzászólásai” páronként egymás után) hosszúsága meghaladta az 1000 tokent az adott nyelvmodell tokenizálójának szerint, lenulláztuk a kontextust, és újból a kezdő prompttal, majd a soron következő fordítandó mondatral indítottuk a beszélgetést.

A Gemini modellek promptolása ettől az Ollamától eltérő API miatt különbözött, ott az instrukciókat rendszerpromptként adtuk át, kontextusként ezt követte a megelőző öt felhasználó-modell hozzájáruláspár, majd ezt zárta a következő fordítandó mondat.

A ChatGPT-nek és a Claude-nak egy a fentiekhez hasonló bevezető instrukció után kb. 3000-4000 karakternyi részt adtuk át egyszerre, ami tipikusan egy-egy fejezetnek felelt meg. Soronkénti fordítást kértünk, utólag a kimenet sorainak számát ellenőriztük, és ha nem egyezett a bemenetével, újra átadtuk ugyanazokat a sorokat fordításra. A ChatGPT esetében két, a Claude esetében minden egyes ilyen szakasz fordítása után új chatet kezdtünk. A ChatGPT és Claude esetében eltekintettünk az API használatától, mert ez pénzbe került volna, és a Google-lel ellentétben ezek a cégek nem adtak használható mennyiségű ingyen kreditet.



#### 4.5. Kiértékelés

A három nagy kereskedelmi nyelvmoddellel minden középkori szöveg (arab, latin, ófrancia, óokcitan) minden mondatához egy-egy angol fordítást generáltunk, majd az illesztett biszegmensek minőségét összesítve értékeltük ki.

Az óokcitanhoz minden egyéb modellel 10-10 gépi fordítást generáltunk a teljes szövegre, majd minden ilyen fordítást külön-külön kiértékeltek, hogy megállapítsuk, mekkora szórás tapasztalható az eredményekben. Azt találtuk, hogy modellenként a 10 fordítás között nem volt tapasztalható érdemi eltérés, pl. a BLEU-értékek között mindössze 0,2-0,3 pontnyi különbség jelentkezett, míg a különböző modellek fordításai között szignifikáns és nagy mértékű eltéréseket találtunk. Következésképpen nem lett volna haszna a további nyelvekhez is 10-10 fordítást generálnunk, ezért azokból csak 3-3 teljes fordítás készült.

A modellek által előállított kimenetek utófeldolgozására szándékosan nem tettünk kísérletet sem. Megfigyeltünk ugyan például olyan anomáliákat, hogy egyes modellek kéretlenül emojikat, díszítő elemeket, szögletes zárójelbe tett kommentárokat és hasonlókat tettek a válaszukba, és ezek egy részét könnyedén eltávolíthattuk volna. Ettől eltekintettünk annak érdekében, hogy az értékelés során kapott pontszámok azt is tükrözzék, ha egy modell rendszeresen oda nem tartozó szemetet tett a fordításba. A kapott pontszámok így nemcsak a modell fordítási képességét, hanem egyszersmind utasításkövetését is jellemzik.

A gépi fordítások és a referenciafordítás mondategységei alkotta biszegmenseket számos különböző fordításminőség-metrika alapján értékeltük. A SacreBLEU (Post, 2018) szoftverrel BLEU pontszámot, emellett egyéb szoftverekkel a fordítások NIST (Doddington, 2002), METEOR (Banerjee és Lavie, 2005), ROUGE (Lin, 2004), valamint BLEURT (Sellam és mtsai, 2020, Pu és mtsai, 2021) és BERTScore (Zhang és mtsai, 2020) pontszámát számoltattuk ki. A két utóbbi BERT (illetve tágabban encodermoddell) alapú, a szemantikai egyezést számszerűsíteni próbáló mérőszám, míg a többi szóunigramoknak és -bigramoknak a gépi fordítás (hipotézis) és a referencia közötti egyezését, valamint relatív elhelyezkedését méri. A nagyszámú metrika mellett azért döntöttünk, mert látni akartuk, hogy azonos módon rangsorolják-e a különböző modellek kimenetét.

A vizsgálatunkhoz felhasznált valamennyi anyagot és szoftvert nyilvános repozitóriumban tettük elérhetővé: [https://gitlab.hadw-bw.de/gpetho/albuc\\_translate](https://gitlab.hadw-bw.de/gpetho/albuc_translate). Ez az anyag a jövőben új modellek kiadása után könnyedén felhasználható lesz azoknak az általunk vizsgált négy középkori nyelven történő benchmarkolására.

### 5. Eredmények

A kiértékelések eredményét az 1. ábra összegzi. Minden kék vonal átlagosan 20-30 óra, minden más színű vonal kb. 5-10 óra GPU-idő árán előállított gépi fordítást jellemez. A három általunk kiszámított ROUGE pontszám közül csak a bigramok egyezését mérő ROUGE-2-ét közöljük, mivel a ROUGE-1 és a ROUGE-L diagramjai egymással majdnem pontosan egyeztek, valamint a modellek közötti

különbségek ezeken kevésbé egyértelműen rajzolódtak ki. A különböző metrikákkal kapott számokból egyező tendenciák olvashatóak le.

A 4 milliárd paraméterig terjedő tipikus SLM-ek közül a Granite3 és a Llama 3.2 látványosan felülmúlja a náluk jóval nagyobb Nemotron-Mini és Phi-3 modelleket, de ezzel együtt is gyengék a kis-közepes modellekhez mérten.

A 7-9 milliárd paraméter közötti modellek teszik ki a teljes mezőny kb. felét. Itt négy olyan párt is találunk, amelyeket egy modell és frissített változata alkotnak. Míg a Llama 3 és a Llama 3.1, illetve az Aya és az Aya Expansé teljesítménye között minimális eltérést tapasztalunk, a Gemma 2 a Gemmával, illetve a Qwen2.5 a Qwen2-vel szemben drasztikusan jobb lett mind a négy nyelvben, pedig ezek kiadása között mindössze 3-4 hónap telt el. Feltűnő a Gemini 1.5 Flash-8b teljesítménye, amely ugyan a Google által a Gemma modellek elnevezésében követett gyakorlattal összhangban valójában szinte biztosan nem 8, hanem 9 milliárd paramétert tartalmaz, de ezzel együtt is nemcsak a saját méretkategóriájából emelkedik ki látványosan, hanem a kétszer, háromszor ekkora ingyenes modelleket is felülmúlja a nyelvek többségében.

A 10 milliárd paraméter feletti kategóriában a legfeltűnőbb a hatalmas Llama 3 70b-nek még a tizedakkora modellekkel képest is igen gyenge szereplése. Az egyetlen vizsgált mixture-of-experts modell, a Mixtral 8x7b teljesítménye is kiábrándító. A nagy Phi-3 modell – SLM testvéreihez hasonlóan – gyakorlatilag használhatatlan, a kis-közepes méretű modellekhez képest is jóval gyengébb teljesítményt nyújtott. A 12 GB-os GPU-n kényelmesen futó, hasonló méretű Mistral-Nemo és Qwen2.5 14b nagyjából fej fej mellett, viszonylag jól teljesít, bár utóbbi következetesen minimálisan jobb az előbbinél, kivéve az arab feldolgozásában, ahol jelentősen jobb. A 20 és 40 milliárd paraméter közötti modellek általában minimálisan jobb eredményeket adtak az előbbiekhöz képest.

A három nagy kereskedelmi nyelvmodell összességében hasonló teljesítményt nyújtott. Az arab és az ófrancia fordításában a Gemini, a latinban és az óokcitánban a GPT-4o teljesített legjobban. A két magas erőforrású nyelv tekintetében a legjobb ingyenes modellek pontszámai megközelítik, esetenként (vö. GPT-4o arab) felül is múlják a kereskedelmi LLM-ek eredményeit. Ugyanakkor az adott nyelvben legsikeresebb LLM látványosan pontosabban fordított, mint a legsikeresebb ingyenes modell (arab: Qwen2.5 32b és Gemini 1.5; latin: Mistral-Small és GPT-4o). Az alacsony erőforrású nyelvek tekintetében minden kereskedelmi modell messze felülmúlta a legsikeresebb ingyenes modellt (Qwen2.5 32b).

Fontos hangsúlyozni, hogy a referenciam fordítás az arab eredeti alapján készült, így törvényszerűen azt tükrözi a legpontosabban. A latin fordítás viszonylag pontos, bár a fordító egyes arab kifejezéseket, amelyeknek nem volt kézenfekvő latin megfelelője, esetleg amelyeket nem értett, meghagyott latin betűkre átírt arab szavakként (vö. Trotter, 1999, 360–361), valamint azonosíthatóak kihagyások és félreértések is (i.m. 364). Az óokcitán és az ófrancia ezeket a rontásokat átvette, azonban ettől eltekintve az óokcitán szöveg jóval pontosabban megfeleltethető az arab eredetinek, mint az ófrancia. Az utóbbi fordítója feltűnően sok helyen pontatlanul dolgozott, sok részt kihagyott, leegyszerűsített. Mindebből következik, hogy ideális fordításminőség mellett az arab alapján készült fordításnak kell



1. ábra: Gépi fordítások kiértékelésének eredményei

legközelebb állnia a referenciához, ezt kell követnie a latinnak, az óokcitánnak, majd jelentősen lemaradva az ófranciának. Az elméletileg lehetséges optimális fordítások közötti viszonyt legpontosabban a Gemini BLEU eredményei tükrözik, illetve a sikeresebb modellek ROUGE értékei.

Az eredmények ugyanakkor semmit nem árulnak el arról, hogy a gépi fordítások abszolút értelemben mennyire jók, így például egy 15-ös BLEU-t mutató latinból készült fordítás mennyire pontos és érthető. Ugyan szisztematikus kvalitatív elemzést nem végeztünk, szűrőpróbaszerűen nyelvenként 100-200 biszegmenst ellenőriztünk, és azt találtuk, hogy a kereskedelmi LLM-ek fordításmínősége minden középkori nyelvre jól használható, nem marad el jelentősen egy emberi fordítótól elvárható minőségtől. A 10 alatti BLEU-t mutató fordítások ellenben összességében használhatatlanok voltak.

## 6. Összegzés

Eredményeink azt mutatják, hogy középkori latin és arab nyelvű szövegek számítógépes nyelvészeti feldolgozásában jól használhatóak a megfizethető GPU-val felszerelt asztali számítógépen is futtatható közepes méretű nyelvmodellek, így ezek feltehetőleg jól finomhangolhatóak lennének például címkézési feladatokra. Ugyanakkor tekintettel arra, hogy a Google által adott 300 dollár ingyen kreditből a történelmi korpuszokhoz képest igen nagy mennyiségű kimenet generálható a Gemini 1.5 Flash modellel, ennek használata észszerű alternatívája lehet az ingyenes modelleknek nyelvtörténeti NLP-alkalmazások fejlesztésekor.

Az alacsony erőforrású középkori nyelvek feldolgozása terén ugyanakkor az ingyenes modellek nem tudnak versenyezni a kereskedelmi LLM-ekkel. A 10 milliárd paraméternél kisebb modellek egyértelműen nem alkalmasak ezek értelmezésére. Az ugyanakkor nem derül ki egyértelműen az értékekből, hogy az ezeknél világosan jobban fordító közepes méretű Qwen2.5 és Mistral modellek megfelelő finomhangolással alkalmassá tehetők-e például óokcitán szövegek jó minőségű szófaji címkézésére, amire előzetes kísérleteink alapján a GPT-4o zero-shot alapon, finomhangolás nélkül is képes.

## Köszönetnyilvánítás

Köszönjük Esra' Abdelzaher intenzív közreműködését az arab szövegváltozat előfeldolgozásában, ezen belül különösen az arab XML általunk használt módosított verziójának kialakításában. Christina Hodeib szintén hasznos megjegyzésekkel segítette az arab változat feldolgozását.

Hálásak vagyunk a Nyelvtudományi Kutatóközpont Lexikai Tudásreprezentáció Kutatócsoportjának, hogy díjmentesen engedélyezte számunkra a rendelkezésére álló GPU-s számítási kapacitás használatát e kutatás keretében.

A *Lexikai tudásreprezentáció* és a *Harmadik generációs szekvenálási adatok bioinformatikai elemzése* projektek nevében köszönetet mondunk a HUN-REN Cloud (lásd: Héder és mtsai, 2022; <https://science-cloud.hu/>) használatáért, ami hozzájárult a publikált eredmények eléréséhez.

## Hivatkozások

- Banerjee, S., Lavie, A.: METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. pp. 65–72. ACL, Ann Arbor, Michigan (Jun 2005), <https://www.aclweb.org/anthology/W05-0909>
- Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: Proceedings of HLT '02. p. 138–145. Morgan Kaufmann, San Francisco (2002)
- ElKoumy, M., Sarhan, A.: Improving ranking-based question answering with weak supervision for low-resource Qur'anic texts. *Artificial Intelligence Review* 58(1), 17 (2024)
- Elsheikh, M.S. (szerk.): *Abū'l Qāsim Halaf Ibn Abbās az-Zahrāwī, La Chirurgia. Versione occitanica della prima metà del Trecento.* nill, Firenze (1992)
- Green, M.H.: Moving from philology to social history: The circulation and uses of Albucasis's Latin surgery in the Middle Ages. In: Glaze, F.E., Nance, B. (szerk.) *Between text and patient: the medical enterprise in Medieval & Early Modern Europe*, pp. 331–372. SISMELE, Edizioni del Galluzzo, Firenze (2011)
- Héder, M., Rigó, E., Medgyesi, D., Lovas, R., Tenczer, S., Török, F., Farkas, A., Emődi, M., Kadlecik, J., Mező, G., Pintér, Á., Kacsuk, P.: The past, present and future of the ELKH cloud. *Információs Társadalom* 22(2), 128 (aug 2022), <https://doi.org/10.22503/2Finftars.xxii.2022.2.8>
- Hendrycks, D., és mtsai: Measuring massive multitask language understanding (2021a), <https://arxiv.org/abs/2009.03300>
- Hendrycks, D., és mtsai: Measuring mathematical problem solving with the MATH dataset (2021b), <https://arxiv.org/abs/2103.03874>
- Her, W., Kruschwitz, U.: Investigating neural machine translation for low-resource languages: Using Bavarian as a case study. In: Melero, M., Sakti, S., Soria, C. (szerk.) *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024*. pp. 155–167. ELRA and ICCL, Torino, Italia (May 2024), <https://aclanthology.org/2024.sigul-1.20>
- Hoffmann, J., és mtsai: Training compute-optimal large language models. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. pp. 30016–30030. NIPS '22, Curran Associates, Red Hook (2024), <https://dl.acm.org/doi/10.5555/3600270.3602446>
- Inoue, G., és mtsai: The interplay of variant, size, and task type in Arabic pre-trained language models. In: Habash, N., és mtsai (szerk.) *Proceedings of the Sixth Arabic Natural Language Processing Workshop*. pp. 92–104. Association for Computational Linguistics, Kyiv (Apr 2021), <https://aclanthology.org/2021.wanlp-1.10>
- Kaplan, J., és mtsai: Scaling laws for neural language models. *CoRR abs/2001.08361* (2020), <https://arxiv.org/abs/2001.08361>
- Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: *Text Summarization Branches Out*. pp. 74–81. Association for Computational Lin-

- guistics, Barcelona, Spain (Jul 2004), <https://www.aclweb.org/anthology/W04-1013>
- Malhas, R., Elsayed, T.: Arabic machine reading comprehension on the holy Qur'an using CL-AraBERT. *Information Processing & Management* 59(6), 103068 (2022)
- Ondrejová, V., Šuppa, M.: Can LLMs handle low-resource dialects? A case study on translation and common sense reasoning in Šariš. In: Scherrer, Y., Jauhainen, T., Ljubešić, N., és mtsai (szerk.) *Proceedings of the Eleventh Workshop on NLP for Similar Languages, Varieties, and Dialects (VarDial 2024)*. pp. 130–139. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024), <https://aclanthology.org/2024.vardial-1.11>
- Pethő, G., Abdelzaher, E., Krishna, S.: Evaluating generative language models on historical texts: A comparative study of Arabic, Latin and two low-resource medieval languages. In: *Workshop on Natural language and AI: New perspectives for linguistic studies at ÖLT 2024, Innsbruck*. Innsbruck (Dec 2024a)
- Pethő, G., Menon, R., Schöffel, M., Tittel, S.: Comparing the ability of generative language models to understand medieval languages. In: *2nd International Conference on Data & Digital Humanities: Generative Artificial Intelligence for Text and Multimodal Data*. Braga (Dec 2024b)
- Post, M.: A call for clarity in reporting BLEU scores. In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. pp. 186–191. Association for Computational Linguistics, Belgium, Brussels (Oct 2018), <https://www.aclweb.org/anthology/W18-6319>
- Pu, A., Chung, H.W., Parikh, A.P., Gehrmann, S., Sellam, T.: Learning compact metrics for MT. In: *Proceedings of EMNLP* (2021)
- Rein, D., és mtsai: GPQA: A graduate-level Google-proof Q&A benchmark (2023), <https://arxiv.org/abs/2311.12022>
- Sakaguchi, K., Bras, R.L., Bhagavatula, C., Choi, Y.: WinoGrande: an adversarial winograd schema challenge at scale. *Commun. ACM* 64(9), 99–106 (Aug 2021), <https://doi.org/10.1145/3474381>
- Sellam, T., Das, D., Parikh, A.P.: BLEURT: Learning robust metrics for text generation. In: *Proceedings of ACL* (2020)
- Sommerschild, T., Assael, Y., Pavlopoulos, J., és mtsai: Machine learning for ancient languages: A survey. *Computational Linguistics* pp. 703–747 (Sep 2023), <https://aclanthology.org/2023.c1-3.5>
- Spink, M.S., Lewis, G. (szerk.): *Albucasis on Surgery and Instruments: A Definitive Edition of the Arabic Text with English Translation and Commentary*. Publications of the Wellcome Institute of the History of Medicine; 12, The Wellcome Institute of the History of Medicine, London (1973)
- Steingrímsson, S., Loftsson, H., Way, A.: SentAlign: Accurate and Scalable Sentence Alignment. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. ACL, Singapore (Dec 2023)
- Stüssi, E.: *Part-of-Speech Tagging on 16th-Century Latin*. Institut für Computerlinguistik, Universität Zürich (2023)

- Stüssi, E., Ströbel, P.: Part-of-speech tagging of 16th-century Latin with GPT. In: Bizzoni, Y., Degaetano-Ortlieb, S., Kazantseva, A., Szpakowicz, S. (szerk.) Proceedings of the 8th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature (LaTeCH-CLfL 2024). pp. 196–206. ACL, St. Julians, Malta (Mar 2024), <https://aclanthology.org/2024.latechclfl-1.18>
- Torres Aguilar, S.: Multilingual named entity recognition for medieval charters using stacked embeddings and BERT-based models. In: Sprugnoli, R., Passarotti, M. (szerk.) Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages. pp. 119–128. European Language Resources Association, Marseille, France (Jun 2022), <https://aclanthology.org/2022.lt4hala-1.17>
- Trotter, D.A.: Arabic surgery in Eastern France and in the Midi: The Old French and Occitan versions of the Chirurige d’ Albucasis. Forum for Modern Language Studies XXXV(4), 358–371 (1999)
- Trotter, D. (szerk.): Albucasis, Traité de Chirurgie: Edition de la traduction en ancien français de la Chirurgie d’Abu’l Qasim Halaf Ibn Abbas al-Zahrawi du manuscrit BNF, français 1318. Niemeyer (2005)
- Volk, M., Fischer, D.P., Fischer, L., és mtsai: LLM-based machine translation and summarization for Latin. In: Sprugnoli, R., Passarotti, M. (szerk.) Proceedings of the Third Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA) @ LREC-COLING-2024. pp. 122–128. ELRA and ICCL, Torino, Italia (May 2024), <https://aclanthology.org/2024.lt4hala-1.15>
- Yang, A., és mtsai: Qwen2 technical report (2024), <https://arxiv.org/abs/2407.10671>
- Zellers, R., és mtsai: HellaSwag: Can a machine really finish your sentence? (2019), <https://arxiv.org/abs/1905.07830>
- Zhang, T., Kishore, V., Wu, F., és mtsai: BERTScore: Evaluating text generation with BERT. In: International Conference on Learning Representations (2020), <https://openreview.net/forum?id=SkeHuCVFDr>
- Zhao, J., és mtsai: LoRA Land: 310 fine-tuned LLMs that rival GPT-4, A technical report (2024), <https://arxiv.org/abs/2405.00732>





# PULI Llumix modell

## Egy folytatólagosan előtanított nagy nyelvi modell

Yang Zijian Győző<sup>1</sup>, Dodé Réka<sup>1</sup>, Ferenczi Gergő<sup>1</sup>, Hatvani Péter<sup>1,2</sup>, Héja Enikő<sup>1</sup>, Lengyel Mariann<sup>1</sup>, Ligeti-Nagy Noémi<sup>1</sup>, Madarász Gábor<sup>1</sup>, Sárossy Bence<sup>1</sup>, Varga Kristóf<sup>1</sup>, Varga Tamás<sup>3</sup>, Váradi Tamás<sup>1</sup>

<sup>1</sup>HUN-REN Nyelvtudományi Kutatóközpont

vezetéknev.(középsőnév).keresztnev@nytud.hun-ren.hu

<sup>2</sup>Pázmány Péter Katolikus Egyetem, Bölcsész- és Társadalomtudományi Kar,  
Nyelvtudományi Doktori Iskola

<sup>3</sup>Pázmány Péter Katolikus Egyetem, Bölcsész- és Társadalomtudományi Kar  
tamasvarga76@gmail.com

**Kivonat** Az utóbbi hónapokban a nagyméretű nyelvi modellek fejlesztése felgyorsult, ahogy egyre több technológiai vállalat törekszik különböző nyelvi feldolgozási feladatok megoldására alkalmas modellek létrehozására. Ebben a cikkben bemutatunk egy magyar nyelvre optimalizált, Llama-2 alapú modellt, amely folytatólagos előtanítást és utasításkövető finomhangolást alkalmaz a magyar nyelvi sajátosságok figyelembevételével. A modell finomhangolásához egy 66 000 angol és 15 000 magyar promptot tartalmazó adatbázist használtunk, így a modell képes lett magyar nyelvű utasítások pontos követésére is. Az így létrejött modell különböző hazai benchmarkokon – köztük a HuCOLA, HuSST és HuRTE teszteken – kimagasló teljesítményt nyújtott, az *zero shot* tesztek során a HuCOLA teszten 66,98%-os, a HuSST teszten 70,06%-os, míg a HuRTE teszten 74,54%-os pontosságot ért el, ami felülmúlja a korábbi modelleket.

A kvantitatív eredmények mellett kvalitatív elemzéseket is végeztünk a modell teljesítményének mélyebb megértése érdekében. A modell magas szintű kontextuskezelő képességét például egy hosszú szöveggörnyezetben végzett keresési feladattal értékeltük, míg a stílusérzékenységet különböző nyelvi regiszterekben tett próbákkal vizsgáltuk. Eredményeink alátámasztják, hogy a transfertanulás révén a modell jelentős tudást képes átvenni más nyelvekből is, javítva a magyar nyelvű teljesítményét.

**Kulcsszavak:** PULI modellek, nagy nyelvi modell, utasításkövető modell, Llama-2, előtanítás, folytatólagos előtanítás, finomhangolás

## 1. Bevezetés

Az elmúlt hónapokban a nagyméretű nyelvi modellek rendkívül nagy figyelmet kaptak. Minőségük napról napra fejlődik és egyre több helyen elkezdték őket integrálni a munkafolyamatokba. A nagyvállalatok meglátták a bennük rejlő lehetőséget és kialakult egy verseny a nagy nyelvmodellek tanításában. Céljuk, hogy

olyan alapmodelleket hozzanak létre, amelyek jó alapot adhatnak egy ChatGPT-hez hasonló alkalmazás létrehozására. A versenyben komoly konkurensek léteznek (Google, Microsoft, Mistral AI<sup>1</sup>, AI-PRO<sup>2</sup>), de az egyik kiemelkedő vállalat a Meta<sup>3</sup>, amely a Llama modelljeit szabad felhasználású licenccel publikálja. Ez lehetőséget biztosít egy újfajta modelltanítási irányhoz, hogy egy jól előtanított nyelvmodellt adaptáljunk egy doménspecifikus tudásra, vagy egy másik nyelvre. Az így folytatólagosan előtanított nyelvmodellek képesek a korábbi tudásukat transzferálni az új nyelvre és az így készült új modell az új nyelv elsajátítása mellett képes megőrizni a korábbi feladatmegoldó képességeit is. Ezzel a módszerrel az új nyelven akár kevesebb adattal is tanítható egy olyan modell, amelynek jobb a minősége, mint a több adaton, de nulláról tanított modelleknek.

Magyar nyelvre vonatkozóan eddig csak nulláról betanított nagyméretű nyelvi modellek állnak rendelkezésre. Ezek közé tartoznak a PULI modell család tagjai, mint például az egynyelvű, 6,7 milliárd paraméteres PULI 3SX (Yang és mtsai, 2023a), valamint a háromnyelvű (magyar-angol-kínai) 7,7 milliárd paraméteres PULI Trio (Yang és mtsai, 2023b). Továbbá elérhető még az angol-magyar kétnyelvű, 6,7 milliárd paraméteres HILANCO-GPTX modell is<sup>4</sup>.

Kutatásunkban egy 7 milliárdos Llama 2 modellt tanítottunk tovább magyar nyelvre, majd finomhangoltuk utasításkövetésre. Az így elkészült modell szignifikánsan felülmúlta az eddigi PULI modelleket.

Jelenlegi utasításkövető modellünk tesztelhető a demóoldalunkon<sup>5</sup>. A folytatólagosan előtanított nagyméretű nyelvi modellünk elérhető a Hugging Face oldalunkon<sup>6</sup>.

## 2. Kapcsolódó irodalom

A generatív nagy nyelvi modellek előtti nyelvmodellekkel – mint BERT, RoBERTa – is voltak már folytatólagos előtanítási kísérletek. Azokat utótanításnak hívták (*post-training*), és nagyrészt doménadaptálási céllal készültek (Xu és mtsai, 2019; Ke és mtsai, 2022, 2023).

Az utóbbi hónapokban a folytatólagos előtanítás új értelmet nyert az olyan modelleknek köszönhetően, mint a Llama modellek (Touvron és mtsai, 2023a,b), amelyek magas minőséget képviselnek és a megengedő licencüknek köszönhetően rendkívül elterjedtek mind a kutatók, mind a vállalatok körében. Ezek a modellek annyira magas minőségűek, hogy könnyen lehet belőlük akár finomhangolással, akár további előtanítással nagyon jó minőségű célalkalmazásokat tanítani.

A mi modellünk megjelenésének egyidőben jelentek meg a SambaNova Systems<sup>7</sup> által tanított SambaLingo modellek (Csaki és mtsai, 2024). Gyakorlatilag

<sup>1</sup> <https://mistral.ai>

<sup>2</sup> <https://ai-pro.org>

<sup>3</sup> <https://about.meta.com>

<sup>4</sup> <https://hilanco.github.io>

<sup>5</sup> <https://puli.nytud.hu>

<sup>6</sup> <https://huggingface.co/NYTK>

<sup>7</sup> <https://sambanova.ai>

ugyanúgy folytatólagos előtanítással tovább tanították a Llama 2 modelleket más nyelvekre, köztük magyar nyelvre is. Ők két mérettel is kísérleteztek, a 7 és a 70 milliárd paraméteres modellekkel. A magyar nyelvű kutatásukban több mint 60 milliárd magyar szavas korpuszsal tanították tovább és az angol-központúság problémájára reflektálva a szótár 10%-át lecserélték olyan tokenekre, amelyek magyar szóelemeket is tartalmaznak. Továbbá az effektív tanulás érdekében a tanítóadat 75%-a magyar és 25%-a angol (Csaki és mtsai, 2023).

Alves és mtsai (2024) folytatólagos előtanítással hoztak létre jobb minőségű gépi fordító alapmodellt. Kutatásukban egy Llama 2 modellt tovább előtanítottak egynyelvű és párhuzamos korpuszon, majd azután végeztek utasításkövető finomhangolást gépi fordításra.

### 3. Korpusz

Kutatásunk során a Together AI által készített LLaMA-2-7B-32K<sup>8</sup> modellt tanítottuk tovább magyar nyelvre. Ehhez a feladathoz a PULI Trio modell tanításához használt korpusz (Yang és mtsai, 2023b) hosszú dokumentumait használtuk fel. Csak azokat a dokumentumokat tartottuk meg, amelyek meghaladják az 5000 szót. A korpusz főbb jellemzői az 1. táblázatban találhatóak.

	Dokumentum	Szó	Átlagos dokumentum hossza átlag / medián (szó)
PULI hosszú	763,704	7,902,519,115	10 823,38 / 7 149
Long Context QA	88,957	1,009,562,704	11 348.88 / 11 274
BookSum	9,600	42,339,698	4,410.39 / 3 265.5

1. táblázat. Az előtanításra használt korpuszok főbb tulajdonságai

A tanítás során, hogy a modell ne felejtse el az angol tudását, beépítettük az eredeti finomhangolási korpuszokat<sup>9</sup>, amelyeket a Together AI használt a LLaMA-2-7B-32K modell betanításához:

- *Long Context QA*: *Natural Questions* korpuszból (Kwiatkowski és mtsai, 2019) származó több szövegrészből (*Multi-passage QA*) álló kérdés-válasz alkörpusz.
- *BookSum* (Kryscinski és mtsai, 2022): Könyvösszefoglaló korpusz, amelyben hosszú szövegek és ahhoz tartozó szövegösszefoglalók találhatóak.

Az 1. táblázatban láthatók a korpuszok statisztikái.

Az utasításkövető finomhangoláshoz ugyanazt a korpuszt használtuk, mint Yang és mtsai (2024a) a kutatásukban, annyi módosítással, hogy nem kevertünk bele kínai promptokat. A korpusz az alábbi főbb tulajdonságokkal rendelkezik:

<sup>8</sup> <https://huggingface.co/togethercomputer/LLaMA-2-7B-32K>

<sup>9</sup> <https://huggingface.co/datasets/togethercomputer/Long-Data-Collections>

- 15 064 magyar prompt: Vegyesen benne van a magyar nyelvre fordított Stanford Alpaca (Yang és mtsai, 2024b), szintetikusan előállított promptok, HULU benchmarkokból generált promptok, felhasználók által feltett kérdések és nyilvános adatbázisokból gyűjtött kérdés válaszok.
- 66 771 angol prompt: angol Stanford Alpaca (Taori és mtsai, 2023) és Dolly (Conover és mtsai, 2023) korpusz.

## 4. Kísérletek és mérések

### 4.1. Folytatólagos előtanítás

A kutatásunk fő célja, hogy a Llama-2 modellt folytatólagos előtanítással továbbtanítsuk és adaptáljuk a magyar nyelvre. Az eredeti Llama-2 modell bemeneti kontextushossza 4096 token volt. Gyakorlati tapasztalataink alapján ez a kontextusablak gyakran nem elegendő, bizonyos feladattípusok igénylik a hosszabb bemeneti kontextusablakot. Ilyen feladat lehet a hosszú szöveg összefoglalása, a dokumentumból információ kinyerése, a RAG (*retrieval augmented generation*) rendszerek (Lewis és mtsai, 2020) vagy a hosszabb párbeszéd rendszerek. Dodé és Yang (2024) kutatásában is a Llama 2 modellt használták, és megállapították, hogy a Llama 2 szótára angolközpontú, ezért a tokenizálás megtöbbszörözi a szöveg méretét. Lemértük a magyar tanítóanyagunkon mi is, és a következő értékeket kaptuk:

- Tokenek száma: 25 027 297 913
- Átlagos dokumentum hossza (tokenszám) a tokenizálás után (átlag / medián): 32 770,94 / 21 133

A tokenizált szöveg hossza több mint a háromszorosa az eredeti szövegnek, ez megegyezik a Dodé és Yang (2024) kutatásában megfigyelt értékekkel. Ez azt jelenti, hogy ha egy hosszabb dokumentumot adunk a modellnek, már nem fér bele a négyezer token hosszú kontextusablakba. Ezért esett a választásunk az alap Llama 2 modell helyett a 32 768 tokenre növelt kontextushosszra tanított Llama-2-7B-32K<sup>10</sup> modellre. Ezt a modellt a Together AI tanította az OpenChatKit implementációval<sup>11</sup>. A modell kontextushosszát pozíció-interpoláció (Chen és mtsai, 2023) segítségével növelték meg. A LLaMA-2-7B-32K modell továbbtanításához szintén ezt az implementációt alkalmaztuk. Az első fél epoch során csak a magyar korpuszt használtuk. Fél epoch után végeztünk némi felületes tesztet, amelynek során azt tapasztaltuk, hogy a modell elkezdte „elfelejteni” angol nyelvi ismereteit. Ennek elkerülése érdekében a második fél epoch alatt hozzákevertük azokat a finomhangolási korpuszokat (Long Context QA és BookSum), amelyeket az eredeti LLaMA-2-7B-32K modellhez használtak. A tanításhoz a következő hiperparamétereket alkalmaztuk:  $2e-5$  tanulási ráta, 8 batch méret GPU-ként, fp16. Ehhez a feladathoz 8 darab NVIDIA A100 (80GB) GPU-t használtunk. A

<sup>10</sup> <https://huggingface.co/togethercomputer/LLaMA-2-7B-32K>

<sup>11</sup> <https://github.com/togethercomputer/OpenChatKit>

tanítás körülbelül két hónapig tartott, és 100 000 lépésnél állítottuk le. Az új, folytatólagosan előtanított modellünk neve: **PULI LlumiX**.

Az előtanítás után a modellünket utasításkövetésre finomhangoltuk. A finomhangoláshoz a Stanford Alpaca implementációját (Taori és mtsai, 2023) alkalmaztuk, ugyanolyan beállításokkal, mint Yang és mtsai (2023b) a kutatásukban. A finomhangolás után különböző kvantitatív és kvalitatív kiértékelést végeztünk a modellen. Utasításkövetésre finomhangolt modellünk neve: **PULI LlumiX Instruct**.

A modellünk létrejöttének idején publikálta a SambaNova Systems a SambaLingo magyar modelljeit, amelyek gyakorlatilag ugyanazon az elven készültek, mint a mi modellünk. Ők is folytatólagosan tovább előtanították a Llama-2 modellt a magyar nyelvre. Ezért a kiértékelésünk során a 7 milliárdos SambaLingo-Hungarian-Base és a 7 milliárdos SambaLingo-Hungarian-Chat modelleket is megvizsgáltuk. Továbbá a korábbi PULI modellek közül a PULI Trio és a ParancsPULI modelleket, mivel ezek rendelkeznek hasonló többnyelvű háttértudással, valamint ugyanazon a finomhangoló anyagon tanult a ParancsPULI, mint a mi modellünk.

## 4.2. Kiértékelési kísérletek

Mivel még nincsen hivatalos generatív modelleket kiértékelő korpusz, a kvantitatív kiértékeléshez három magyar benchmarkot választottunk, a HuCOLA, HuSST és HuRTE korpuszokat (Ligeti-Nagy és mtsai, 2022). Azért ezt a három korpuszt választottuk, mert a HuCOLA ad egy képet arról, hogy a modell mennyire sajátította el a magyar nyelvi tulajdonságokat, a HuSST egy népszerű szentimentelemző osztályozási feladat, a HuRTE pedig egy következtetési feladat, ami komplexebb logikai következtetést igényel. A HuCOPA egy többválasztós feladat, amely komolyabb prompt kísérleteket tenne szükségessé, ami megnehezíti a kiértékelést és torzíthatja az eredményt. A HuWNLI és a HuCB (Ligeti-Nagy és mtsai, 2023) is következtetési feladatok, így ezeket már nem vettük bele a kiértékelésbe.

Fontosnak tartjuk, hogy részletesebben kifejtsük a promptokkal való kísérleteinket. Sajnos nem egyértelmű, hogy egy-egy ilyen nagy nyelvi modell kiértékeléshez milyen módon használjunk promptokat. Ez vonatkozik mind a nyers, mind az utasításkövető modellekre. Ezeknek a nagy nyelvi modelleknek a kimenetei nem determinisztikusak, a paraméterbeállításokkal növelhetjük ezt a képességet, azonban azt tapasztaltuk, hogy ilyenkor gyakran romlik a kimenet minősége. Yang és mtsai (2023b) alacsonyabb hőmérsékletet alkalmaztak, 0,4 és 0,6 közötti, míg Csaki és mtsai (2024) kutatásában magasabb, 0,6 és 0,8 közötti értékeket állítottak be. A *top-p* értéket csak Csaki és mtsai (2024) állították át 0,9-re, a többi paramétert, mint *top-k*, alapértelmezettnek hagyták. A kutatásunkban a középutat választottuk, csak a hőmérséklet értékét változtattuk 0,6-ra. Ez olyan érték, amely már változatosabban ad válaszokat, azonban a determinisztikusság alacsonyabb. Ezért érdemes többször is lefuttatni a méréseket, hogy pontosabb képet kapjunk.

A kvantitatív mérések során kétféle kiértékelést végeztünk. Az első mérésünk a nyers nyelvmodellekre irányult. Ehhez a *few-shot* kísérletet alkalmaztuk. Azonban egy kis módosítást vittünk véghez. Yang és mtsai (2023b, 2024a) kutatásaiban fixen az első néhány promptot vették a tanítóanyagból. Ez egy szerencsés választás esetén magas eredményt hozhat, azonban egy kevésbé szerencsés választás esetén a modell gyengén teljesíthet. Ehelyett Zhu és mtsai (2023) kísérlete alapján véletlenszerűen választottunk ki promptokat a tanítóanyagból minden egyes tesztszegmensre. Hogy megállapítsuk az optimális promptmennyiséget, többféle mennyiségű prompttal kísérleteztünk.

Következő kísérletünk a nyers modellek kiértékelése volt, amihez a *few-shot* módszert választottuk. A *few-shot* mérésekhez a HuCOLA és a HuSST esetében a következő promptsablont használtuk:

- [szöveg] = [konvertált címke]
- 1. Példa: Az Angliáról való könyv tetszik. = rossz
- 2. Példa: Könnyű, aranyos és felejthető. = semleges

A fenti példában az figyelhető meg, hogy a címkék helyett egy konvertált szövegalapú címkét alkalmaztunk. Megfigyelésünk az volt, hogy az eredeti szám alapú címke (0, 1, 2) kevésbé segít a nyelvmodellnek a feladatmegoldásban, és jobb eredményt tudunk elérni, ha szöveges címkét adunk meg. A HuCOLA esetében a *helyes/helytelen* és a *jó/rossz* címkékkal kísérleteztünk, és az utóbbival tudtunk magasabb eredményt elérni. A HuSST esetében a konvertált címkék a következők voltak: *negatív, semleges, pozitív*. A HuRTE esetében kétféle megoldással is kísérleteztünk. Az első, amikor egy [SEP] tokenel választottuk el a premisszát a hipotézistől (hasonló megközelítés, mint amit Laki és Yang (2023) alkalmaztak a kutatásukban), valamint hasonlóan a HuCOLA esetén kísérleteztünk a *helyes/helytelen* és a *jó/rossz* címkékkal, és az utóbbival értünk el jobb eredményt. Az első változat promptsablona a következő volt:

- [premissza] [SEP] [hipotézis] = [konvertált címke]
- Példa: Ciprus, legyen bár megosztott vagy sem, május 1-jén csatlakozik az EU-hoz. [SEP] Ciprust május 1-jén osztották két részre. = rossz

A második változatban Yang és Ligeti-Nagy (2023) kutatását vettük alapul, ahol a premisszát és a hipotézist szövegesen kötötték össze, ezzel segítve a modellnek a szöveg értelmezésében. A második változat promptsablona a következő volt:

- [premissza] Ebből az következik, hogy [hipotézis] = [konvertált címke]
- Példa: Az eladásból befolyt pénz Hepburn családjához letétbe kerül. Ebből az következik, hogy a bevétel Hepburn családjáé. = jó

A nyers modellek kiértékelése után következett az utasításkövető modellek kiértékelése. Ehhez a feladathoz a *zero-shot* módszerét választottuk. A *zero-shot* kísérletnél is többféle promptot próbáltunk ki. Itt a prompt sablon kötött volt, a ParancsPULI és a mi utasításkövető PULI Llumix Instruct modellünk a magyarra fordított Stanford Alpaca sablonját alkalmazza, ami megegyezik a Yang és mtsai (2023b) által publikált cikkben leírt sablonnal. A SambaLingo modelljei

egy *chat* sablont alkalmaznak (Csaki és mtsai, 2024). A három benchmarkra különböző promptokkal kísérleteztünk, majd amelyik a legjobbnak bizonyult, azt tartottuk meg és mindegyik modellre 3-3 mérést végeztünk, majd a három mérés átlagát láthatjuk az 5.1. fejezet 3. táblázatában.

A ParancsPULI és az utasításkövető PULI Llumix Instruct modell esetében az „Utasítás” mezőbe tettük a kérdést megfogalmazó promptot, és az „Bemenet” mezőbe a kiértékelendő szöveget. A SambaLingo esetében közvetlenül egymás alatt volt a kérdés és a kiértékelendő szöveg.

A HuCOLA esetében a következő promptokat próbáltuk ki, a vastagon szedett szöveg teljesített a legjobban:

- **Az [alábbi/következő] szöveg a magyar [nyelvtani/grammatikai] követelményeknek megfelelően jó/helyes/megfelel vagy nem jó/helyes/felel meg?**
- Helyes-e az [alábbi/következő]?
- Az [alábbi/következő] magyar mondat helyes?
- Az [alábbi/következő] magyar mondat [grammatikailag/nyelvtanilag] helyes vagy nem helyes?

A HuSST esetében a következő promptokkal kísérleteztünk, a vastagon szedett szöveg teljesített a legjobban:

- **Az [alábbi/következő] mondat pozitív, semleges, vagy negatív hangvételű?**
- Mi az [alábbi/következő] mondat szentimentje?
- Mi az [alábbi/következő] mondat szentimentje (pozitív, semleges, negatív)?
- Mi az [alábbi/következő] mondat szentimentje? Pozitív, semleges vagy negatív?

A SambaLingo esetében hozzá kellett tenni azt a mondatot, hogy „Röviden válaszolj!”, mivel enélkül hajlamos volt hosszan válaszolni és a válasza belefoglalta mind a három címkét (például megismételte a kérdést), ami megnehezítette a kiértékelést.

A HuRTE esetében kétféle kísérlet elegendő volt. Az egyik az a *few-shot* 2. változat volt, míg a másik felhasználta a korpusz mezőit, az utóbbi teljesített jobban, amit kiemeltünk vastagon:

- Helyes az alábbi következtetés?  
[premissza] Ebből az következik, hogy [hipotézis]
- **Következik-e a hipotézis a premisszából?**  
premissza: [premissza]  
hipotézis: [hipotézis]

## 5. Eredmények és kiértékelések

### 5.1. Kvantitatív kiértékelések

Célunk nem a benchmarkon való versenyen való részvétel volt, ezért a kiértékeléshez az egyszerűség és a Yang és mtsai (2023b) kutatásával való összehasonlíthatóság kedvéért, valamint a korpuszokban lévő címkék nem egyenletes eloszlása

miatt, a kiegyensúlyozott pontosság (*balanced accuracy*) (Brodersen és mtsai, 2010) metrikát alkalmaztuk.

Az első feladat az volt, hogy megkeressük az optimális promptmennyiséget. A 2. táblázatban láthatóak a kísérletek, hogy megtaláljuk ezt az optimális számot. A kísérletben látható, hogy a mi esetünkben a 20 promptpélda volt a legkedvezőbb a modellünk számára. Így a továbbiakban *20-shot* kísérleteket végeztünk a többi modellel is.

	HuCOLA	HuSST	HuRTE
5-shot	48,49	64,80	58,43
10-shot	49,69	66,01	59,30
15-shot	49,56	66,76	59,88
<b>20-shot</b>	<b>55,84</b>	<b>68,53</b>	<b>61,49</b>
25-shot	52,58	65,21	61,16

2. táblázat. PULI Llumix modell kiértékelése különböző mennyiségű promptokkal

A 3. táblázatban láthatóak a *few-shot* és *zero-shot* eredmények. A véletlenszerű kiválasztás módszere már ad egyfajta átlagot, és néhány kiegészítő mérés alapján azt tapasztaltuk, hogy nem volt jelentős eltérés, ezért az eredményeinkben az elsőre mért értékeket jelenítettük meg (lásd 3. táblázat). A 3. táblázatban a HuRTE mindkét változatának az eredményét láthatjuk a következő formában: 1. változat / 2. változat.

A *few-shot* mérések (lásd 3. táblázat) egyértelműen azt mutatják, hogy a folytatólagosan előtanított modellünk jobban teljesít a többi modellel, de még a SambaLingo modellel is. Azonban egy esetet leszámítva a Sambanova modelljei felülmúlják a korábbi PULI modelleket, ezzel alátámasztva azt, hogy a folytatólagos tanítással jobb eredményt lehet elérni. Az egyetlen érték, amiben alulmaradt a Sambanova, az az utasításkövető HuCOLA kísérlet. Ennek két okát látjuk, az első, hogy a mi utasításkövető finomhangoló anyagunk tartalmaz 1000-1000 darab HuCOLA, HuSST és HuRTE anyagot, ezzel előnyt biztosít a mi modelljeink számára ezekben a feladatokban. Azonban így is aluteljesít a ParancsPULI a legtöbb esetben. Ebből látszik a transzfer tanulás ereje. A másik okát a SambaLingo modell finomhangolási anyagában kerestük, ami nagyrészt angol nyelvű anyagból áll össze, ezzel úgy érezzük, hogy a magyar nyelvről alkotott tudása romlik.

Továbbá az is látszik, hogy a HuRTE kísérletben a szöveges átalakítás segített a modellnek, ezzel alátámasztva Yang és Ligeti-Nagy (2023) kísérletét.

## 5.2. Kvalitatív kiértékelés

A kvantitatív kiértékelések után végeztünk kvalitatív méréseket is. Egy nagy nyelvi modell esetében különösen fontos ez a rész, hiszen az eddig használt benchmarkok nem egészen arra valók, hogy a generatív modelleket kiértékeljük. Egy alapot adnak, de amíg nincsen magyar nyelvre ilyen jellegű korpusz, addig a



	HuCOLA	HuSST	HuRTE
	20-shot		
PULI Trio	53,47	54,96	53,34 / 54,30
<b>PULI LlumiX</b>	<b>55,84</b>	<b>68,53</b>	<b>56,97 / 61,49</b>
SambaLingo Hungarian Base	53,59	66,76	54,99 / 60,05
	zero-shot (átlag)		
ParancsPULI	51,97	54,50	52,05
<b>Utasításkövető PULI LlumiX Instruct</b>	<b>66,98</b>	<b>70,06</b>	<b>74,54</b>
SambaLingo-Hungarian-Chat	50,83	58,24	69,64

3. táblázat. A PULI LlumiX modell kiértékelése és összehasonlítása HuLU benchmarkokon

kvalitatív mérésekkel kapunk teljesebb képet a modellekről. A kvalitatív méréseink első sorban az utasításkövető modellekre irányultak, hiszen ezekben az esetekben a legfontosabb a jelentősége, hogy milyen gyakorlati tudásokat voltak képesek elsajátítani.

Első kvalitatív kiértékelésünk egy úgynevezett „Tű a szénakazalban” (*Needle in a haystack*) kísérlet volt. Az ötletet az Arize AI kísérletéből<sup>12</sup> vettük, ezt implementáltuk újra a saját feladatunkra. A kísérlet egyfajta stressz-teszt, aminek a célja a nagy nyelvi modellek teljesítményének kiértékelése különböző kontextusméretek mellett. A teszt során egy adott, célzott információt („tű”) ágyaznak be egy nagyobb, összetettebb magyar szöveggörnyezetbe („szénakazal”). A cél az, hogy felmérjék egy modell azon képességét, hogy képes-e az adott információt azonosítani és felhasználni egy hatalmas adatmennyiség közepette.

A kísérlethez „A Trónok harca” könyvet használtuk fel. Ebbe véletlenszerűen betettünk egy bele nem illő mondatot, ami a mi esetünkben a következő volt:

- Ezen a napon ünnepelte [város neve] város a [0-100]. évfordulóját.

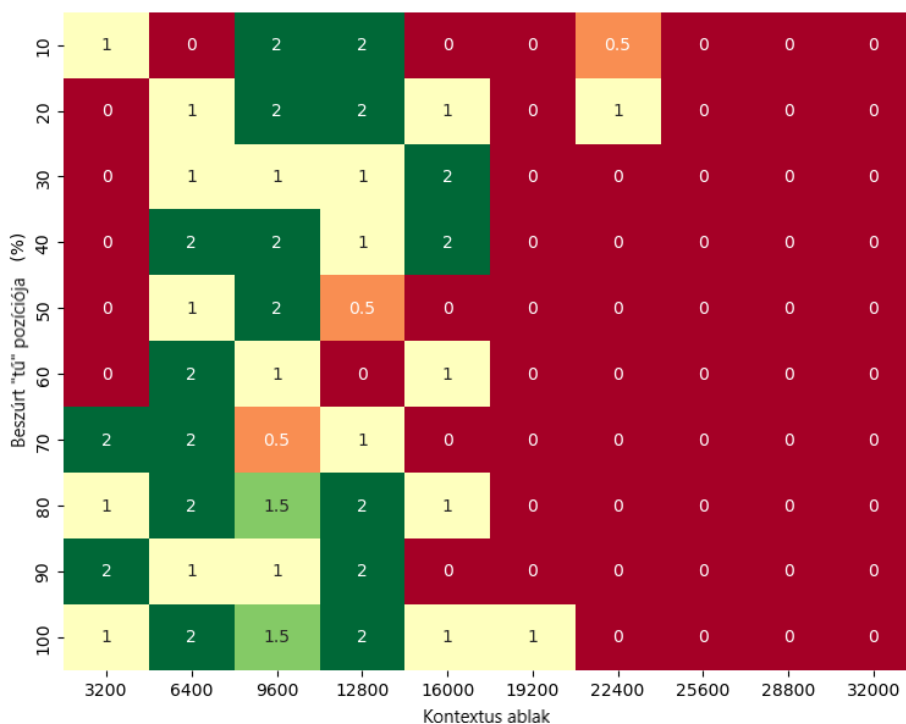
A prompt, amivel rákérdeztünk az adott mondat tartalmára, a következő volt:

- Kizárólag a következő szöveg alapján, hanyadik évfordulóját ünnepelte [város neve] város? Csak egy számot adj vissza!

A teszt során a bemeneti kontextusablakot tíz részre osztottuk. Minden tizednek a 10, 20, 30 stb. százalékába helyeztük véletlenszerűen az adott mondatunkat. A kapott válasz alapján a 1. ábrán látható az utasításkövető PULI LlumiX Instruct modellünk eredménye. Az eredeti kísérletben a GPT-4 modellt kiértékelve azt lehetett látni, hogy a modell a bemeneti kontextusablak hátsó részében gyengébben teljesíti ezt a feladatot. De egy Claude 2.1 modell esetén ez a teljesítménycsökkenés még drasztikusabb. A mi modellünknel is ezt a jelenséget várjuk, hiszen a finomhangoló promptok, amikkel dolgoztunk, átlagosan 330,51 tokent tartalmaztak; a leghosszabb prompt 9 999 token hosszú (Llama-2 tokenizálóját használtuk). Ez alapján valóban az elvárásainknak megfelelően (lásd 1. ábra) a

<sup>12</sup> <https://arize.com/blog-course/the-needle-in-a-haystack-test-evaluating-the-performance-of-llm-rag-systems>

16 000 tokenhossz után már egyáltalán nem találja meg a helyes választ, valamint a 12 000 tokenhosszig teljesít egész stabilan. Bár érdekes jelenség az első oszlop, ami további elemzést igényel.



1. ábra: Tű a szénakazalban kísérlet

A második kvalitatív mérésünk során azt vizsgáltuk, a modell válaszainak stílusa mennyire felel meg a prompt által implicált helyzetnek. A stílus megfelelőségét elsősorban a tegező és magázó igealakok, a mondathossz, valamint a válaszban használt szókészlet alapján vizsgáltuk. A nyelvtani helyességet az elemzés során nem vettük figyelembe. A példák a 4. táblázatban láthatóak. Az első promptrra adott válasz stilisztikai szempontból nagy mértékben megfelel az elvártnak. Udvarias nyelvezetet, hosszú mondatokat, magázó formát használ, és alkalmazza az üzleti nyelvezetben gyakran használt bizonyos fordulatokat is (Minden erőfeszítésünkkel azon dolgozunk. . .). Talán egyedül az *elszomorít* és a *megkérnénk* igék nem illenek bele a stílusba.

A táblázat második példájában informális stílus használata az elvárt, amelynek a modell válasza nagyrészt megfelel. Tegező igeformákat, és az utolsó előtti kivételével rövid, egyszerű mondatokat használ. A használt szavak az *összpontosítani* igeen kívül megfelelnek a gyerek nyelvezetének.

	<p><b>Prompt:</b></p> <p>1. Írj egy levelet egy hoteligazgató nevében, amelyben bocsánatot kér egy vendégtől, mert koszos volt a vendég szobája!</p> <p><b>Válasz:</b></p> <p>Tisztelt Vendégünk! Ezúton szeretnénk elnézést kérni a nem megfelelő szobatisztaság miatt. Tudjuk, hogy mennyire fontos a kényelem és a tisztaság, és elszomorít bennünket, hogy nem tudtuk megfelelő szinten biztosítani szobáját. Minden erőfeszítésünkkel azon dolgozunk, hogy szállodánkat a legkényelmesebb és legtisztább helyé tegyük, és elnézést kérünk, hogy ezúttal nem sikerült megfelelnünk elvárásainak. Megkérnénk, hogy jelezze, ha további segítségre van szüksége, vagy ha bármi mást tehetünk a kellemetlenségekért. Hálásak vagyunk, hogy nálunk szállt meg, és reméljük, hogy a jövőben pozitív tapasztalatokat szerez majd szállodánkban. Köszönjük, hogy felhívta a figyelmünket a problémára, és reméljük, hogy hamarosan újra vendégül láthatjuk. Üdvözlettel: [Hoteligazgató nevében]</p>
	<p><b>Prompt:</b></p> <p>2. Kérj egy kisgyerek nevében ajándékot a Jézuskától!</p> <p><b>Válasz:</b></p> <p>Kedves Jézuska! Én egy 6 éves kislány vagyok, a nevem Bence. Szeretnék egy LEGO készletet karácsonyra. Tudom, hogy nagyon sok mindent kell még a Jézuskának a karácsonyi ajándékokra összpontosítania, de nagyon szeretném, ha az enyém lenne a LEGO City 60. születésnapra torta készlet. Köszönöm szépen! Bence</p>
	<p><b>Prompt:</b></p> <p>3. Milyen szavakkal köszönt egy kislány egy ismeretlen idős férfit?</p> <p><b>Válasz:</b></p> <p>A kislányok általában kedvesen, szeretettel és tisztelettel köszöntik az ismeretlen idős férfiakat. Így például azt mondhatják: „Jó napot!”, „Szia!”, „Üdvözlöm!” vagy „Köszönöm, hogy segített!”</p>
	<p>4. Milyen szavakkal köszönt egy kislány egy ismeretlen idős nőt?</p> <p><b>Válasz:</b></p> <p>A kislány lehet, hogy így köszönti az ismeretlen idős nőt: "Üdvözlöm, hölgyem! Hogy van ma?" vagy "Szia! Hogy hívnak?" vagy "Hello, asszonyom! Hogy érzi magát?" vagy "Üdvözlöm, hölgyem! Hogy szolgál az egészsége?"</p>

## 4. táblázat. Stilisztikai példák

A stilisztikai kérdéseknél megvizsgáltuk a női-férfi és az életkorból fakadó nyelvhasználati jellemzőket is. A példákban a modell a kislányként és kislányként meghatározott beszélő számára gyakran tegező formát javasol, ami nem helyes. A javasolt magázó formák egy része (pl.: *Üdvözlöm*) azonban a beszélő életkorának nem felel meg. A *Hello, asszonyom!* köszöntés pedig sem az udvariassági, sem pedig az életkori elvárásoknak nem felel meg.

Végül harmadik kvalitatív kiértékelésünk során 117, magyar kultúrával, történelemmel, nyelvvel kapcsolatos kérdésre adott válaszok tényyszerűségét hasonlítottuk össze. A tényyszerűséget először egy hármas skálán értékeltük:

- 0: a promptrra adott válasz teljesen helytelen
- 1: a promptrra adott válasz részben helyes, de tartalmaz ténybeli tévedéseket, vagy helyes, de nem a legrelevánsabb választ adta

- 2: a válasz tényszerű és pontos, nem tartalmaz ténybeli tévedéseket

Fontos megjegyezni, hogy nem vettük figyelembe a válasz stilisztikai jellemzőit, a nyelvi helyességét, a formai megfelelőségét, csak azt vizsgáltuk, hogy a kérdésre helyesen válaszolnak-e a modellek. Az eredményeket az 5. táblázatban foglaltuk össze.

Témakör	Kérdések száma	PULI	LlumiX	Instruct	ParancsPuli	Sambalingo
Egyéb	19		61%		47%	66%
Földrajz	17		82%		85%	71%
Irodalom	26		67%		63%	65%
Művészet	14		54%		61%	43%
Nyelv	22		70%		48%	45%
Történelem	16		72%		41%	75%
	114		<b>67%</b>		57%	61%

5. táblázat. A modellek kiértékelésének eredménye a kulturális tárgyi tudást mérő adathalmazon.

A PULI LlumiX Instruct modell jó teljesítményt nyújtott a kulturális tárgyi tudást mérő kérdések megválaszolásában, és a legmagasabb átlagos pontosságot érte el (**67%**). Különösen jól szerepelt az *Irodalom* kategóriában, ahol a többi modellt meghaladva **67%-os** pontossággal válaszolt a kérdésekre, szemben a *ParancsPuli* 63%-os és a *Sambalingo* 65%-os eredményével. A *Nyelv* témakörben a PULI LlumiX Instruct modell 70%-os pontossága jóval felülmúlja a másik két modell alacsony, 48% és 45%-os pontosságát.

Fontos azonban megemlíteni, hogy bizonyos témakörökben más modellek is felülmúlták a PULI LlumiX Instruct modellt. A *Földrajz* kategóriában például a *ParancsPuli* érte el a legjobb eredményt **85%-kal**, amely kicsit meghaladja a PULI LlumiX Instruct modell 82%-os teljesítményét, míg a *Sambalingo* itt 71%-kal szerepelt. A *Történelem* kategóriában a *Sambalingo* bizonyult a legjobbnak, 75%-os pontossággal, ami kicsit felülmúlja a PULI LlumiX Instruct modell 72%-os eredményét. Az *Egyéb* kategóriában a *Sambalingo* vezet, **66%-kal**, míg a PULI LlumiX Instruct modell 61%-ot ért el ezen a területen.

## 6. Összegzés

Kutatásunk során kifejlesztettünk egy magyar nyelvre adaptált, Llama-2 alapú, folytatólagosan előtanított és utasításkövető finomhangolással továbbfejlesztett nagyméretű nyelvi modellt. A modell különböző hazai benchmarkokon, mint a HuCOLA, HuSST és HuRTE, kimagasló pontosságot ért el, felülmúlva a korábbi modelleket, és erősségeit kvalitatív elemzések is igazolták. Eredményeink rávilágítanak arra, hogy a transzferTanulás jelentősen növeli a modell nyelvi teljesítményét, lehetőséget nyújtva a magyar nyelvű nyelvtechnológiai feladatok pontosabb megoldására.

## Hivatkozások

- Alves, D.M., Pombal, J., Guerreiro, N.M., Martins, P.H., Alves, J., Farajian, A., Peters, B., Rei, R., Fernandes, P., Agrawal, S., Colombo, P., de Souza, J.G.C., Martins, A.F.T.: Tower: An open multilingual large language model for translation-related tasks (2024)
- Brodersen, K.H., Ong, C.S., Stephan, K.E., Buhmann, J.M.: The balanced accuracy and its posterior distribution. In: 2010 20th International Conference on Pattern Recognition. pp. 3121–3124 (2010)
- Chen, S., Wong, S., Chen, L., Tian, Y.: Extending context window of large language models via positional interpolation (2023), <https://arxiv.org/abs/2306.15595>
- Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., Xin, R.: Free Dolly: Introducing the World’s First Truly Open Instruction-Tuned LLM (2023), <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm>
- Csaki, Z., Li, B., Li, J., Xu, Q., Pawakapan, P., Zhang, L., Du, Y., Zhao, H., Hu, C., Thakker, U.: Sambalingo: Teaching large language models new languages (2024)
- Csaki, Z., Pawakapan, P., Thakker, U., Xu, Q.: Efficiently adapting pretrained language models to new languages (2023), <https://arxiv.org/abs/2311.05741>
- Dodé, R., Yang, Z.Gy.: Kulcsszógenerálás magyar nyelvű, hosszú szövegekből nagy nyelvi modellekkel. In: Berend, G., Gosztolya, G., Vincze, V. (szerk.) XX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 257–267. Szegedi Tudományegyetem, online (2024)
- Ke, Z., Shao, Y., Lin, H., Konishi, T., Kim, G., Liu, B.: Continual pre-training of language models. In: The Eleventh International Conference on Learning Representations (2023), [https://openreview.net/forum?id=m\\_GDIItaI3o](https://openreview.net/forum?id=m_GDIItaI3o)
- Ke, Z., Shao, Y., Lin, H., Xu, H., Shu, L., Liu, B.: Adapting a language model while preserving its general knowledge. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (szerk.) Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 10177–10188. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022), <https://aclanthology.org/2022.emnlp-main.693>
- Krystinski, W., Rajani, N., Agarwal, D., Xiong, C., Radev, D.: BOOKSUM: A collection of datasets for long-form narrative summarization. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (szerk.) Findings of the Association for Computational Linguistics: EMNLP 2022. pp. 6536–6558. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022), <https://aclanthology.org/2022.findings-emnlp.488>
- Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.W., Dai, A.M., Uszkoreit, J., Le, Q., Petrov, S.: Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics 7, 452–466 (2019), <https://aclanthology.org/Q19-1026>

- Laki, L.J., Yang, Z.Gy.: Sentiment analysis with neural models for hungarian. *Acta Polytechnica Hungarica* 20(5), 109–128 (2023)
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive nlp tasks. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20*, Curran Associates Inc., Red Hook, NY, USA (2020)
- Ligeti-Nagy, N., Ferenczi, G., Héja, E., Jelencsik-Mátyus, K., Laki, L.J., Vadász, N., Yang, Z.Gy., Váradi, T.: HuLU: magyar nyelvű benchmark adatbázis kiépítése a neurális nyelvmodellek kiértékelése céljából. In: *XVIII. Magyar Számítógépes Nyelvészeti Konferencia*. p. 431–446. JATEPress, Szeged (2022)
- Ligeti-Nagy, N., Héja, E., Laki, L.J., Takács, D., Yang, Z.Gy., Váradi, T.: Hát te mekkorát nőttél! - A HuLU első életévé új adatbázisokkal és webszolgáltatással. In: *XIX. Magyar Számítógépes Nyelvészeti Konferencia*. p. 217–230. JATEPress, Szeged (2023)
- Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., Hashimoto, T.B.: Stanford Alpaca: An Instruction-following LLaMA model (2023), [https://github.com/tatsu-lab/stanford\\_alpaca](https://github.com/tatsu-lab/stanford_alpaca)
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971* (2023a)
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T.: Llama 2: Open Foundation and Fine-Tuned Chat Models (2023b)
- Xu, H., Liu, B., Shu, L., Yu, P.: BERT post-training for review reading comprehension and aspect-based sentiment analysis. In: Burstein, J., Doran, C., Solorio, T. (szerk.) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. pp. 2324–2335. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), <https://aclanthology.org/N19-1242>
- Yang, Z.Gy., Dodé, R., Ferenczi, G., Hatvani, P., Héja, E., Madarász, G., Ligeti-Nagy, N., Sáróssy, B., Szaniszló, Zs., Váradi, T., Verebélyi, T., Prószéky, G.: The first instruct-following large language models for hungarian. In: *2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS) Proceedings*. pp. 247–252. University of Debrecen, Debrecen, Hungary (2024a)

- Yang, Z.Gy., Dodé, R., Ferenczi, G., Héja, E., Jelencsik-Mátyus, K., Kőrös, A., Laki, L.J., Ligeti-Nagy, N., Vadász, N., Váradi, T.: Jönnék a nagyok! BERT-Large, GPT-2 és GPT-3 nyelvmodellek magyar nyelvre. In: XIX. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2023). pp. 247–262. Szegedi Tudományegyetem, Informatikai Intézet, Szeged, Hungary (2023a)
- Yang, Z.Gy., Laki, L.J., Váradi, T., Prószéky, G.: Mono- and multilingual gpt-3 models for hungarian. In: Text, Speech, and Dialogue. pp. 94–104. Lecture Notes in Computer Science, Springer Nature Switzerland, Plzeň, Czech Republic (2023b)
- Yang, Z.Gy., Ligeti-Nagy, N.: Improve performance of fine-tuning language models with prompting. *Infocommunications Journal Special Issue on Applied Informatics*, 62–68 (2023), <https://doi.org/10.36244/ICJ.2023.5.10>
- Yang, Z.Gy., Szlávik, S., Ligeti-Nagy, N.: Magyar nyelvű utasításkövető korpusz építése Stanford Alpaca promptok fordításával és lokalizálásával [Building a Hungarian instruct-following corpus by translating and localizing Stanford Alpaca prompts]. In: XX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 243–255. Szegedi Tudományegyetem, Szeged (2024b)
- Zhu, W., Liu, H., Dong, Q., Xu, J., Huang, S., Kong, L., Chen, J., Li, L.: Multilingual machine translation with large language models: Empirical results and analysis (2023)





# KORPUSZ, NYELVI ELEMZÉS



# A HuSpaCy és e-magyar elemzőláncok teljesítményének átfogó összehasonlítása országgyűlési szövegeken: a tokenizálástól a függőségi elemzésig

Skrabák Boglárka<sup>1</sup>, Ligeti-Nagy Noémi<sup>2</sup>

<sup>1</sup>ELTE Informatikai Kar

<sup>2</sup>HUN-REN Nyelvtudományi Kutatóközpont

skrabakbogi@gmail.com

ligeti-nagy.noemi@nytud.hun-ren.hu

**Kivonat** A magyar nyelv számítógépes feldolgozásához megbízható elemzőrendszerekre van szükség, amelyek különböző nyelvi szinteken képesek pontos kimenetet biztosítani. Jelen tanulmány célja, hogy részletes összehasonlítást nyújtson két kiemelkedő magyar nyelvi elemzőrendszer, a HuSpaCy és az e-magyar teljesítményéről több szempontból, ideértve a tokenizálást, a szófaji címkézést, a morfoszintaktikai elemzést és a névelem-felismerést. Az elemzéseinket a magyar országgyűlési jegyzőkönyvek egy szövegrészén végeztük, amely lehetőséget nyújtott a rendszerek tesztelésére formailag és nyelvileg változatos szövegeken. Eredményeink segítséget nyújthatnak a nyelvi feldolgozó eszközök felhasználóinak abban, hogy a specifikus nyelvi alkalmazási igényeiknek legmegfelelőbb rendszert válasszák. A tanulmány rámutat az elemzők erősségeire és hiányosságaira, melyek alapján továbbfejleszthetők a magyar nyelvre irányuló nyelvtechnológiai megoldások.

**Kulcsszavak:** nyelvfeldolgozás, magyar nyelvi elemző, HuSpaCy, e-magyar, tokenizálás, morfoszintaktikai elemzés, névelem-felismerés

## 1. Bevezetés

A nyelvi elemzőeszközök terén a HuSpaCy (Orosz és mtsai, 2022, 2023) és az e-magyar (Váradí és mtsai, 2017; Indig és mtsai, 2019; Simon és mtsai, 2020) rendszerek kiemelkednek, mint a magyar nyelv feldolgozására alkalmas nyílt forráskódú megoldások. A HuSpaCy egy korszerű, könnyen használható, gyors elemzőrendszer, amely a SpaCy keretrendszerre épülve (Honnibal és mtsai, 2020) képes a magyar nyelv morfoszintaktikai feldolgozására. Az e-magyar rendszer egy komplex, többkomponensű elemzői *pipeline*, amely a magyar nyelv specifikus sajátosságait is figyelembe veszi.

A magyar nyelvre optimalizált ipari nyelvfeldolgozó eszközök száma korlátozott, és csak néhány átfogó rendszer érhető el a fentiekén kívül. Ezek közé tartozik a magyarlanc (Zsibrita és mtsai, 2013), amely egy Java-alapú, ipari alkalmazásokhoz tervezett eszköz. Bár számos fontos nyelvfeldolgozási funkciót nyújt –

mint a tokenizálás, mondathatárok felismerése, szófaji címkézés, lemmatizálás és függőségi elemzés –, hiányossága a névelem-felismerés és a szóbeágyazások támogatásának hiánya.

A magyar nyelvspecifikus rendszerek mellett nemzetközi eszközök, mint a Stanza (Qi és mtsai, 2020) és az UDPipe (Straka, 2018), szintén relevánsak a nyelvfeldolgozás területén. Ezek több nyelvet támogatnak, és képesek nyers szövegek morfológiai és szintaktikai elemzésére. Ugyanakkor ezek az eszközök sem nyújtanak névelem-felismerési lehetőséget, és teljesítményüket jelentősen korlátozza a nyilvánosan elérhető annotált korpuszok kis mérete.

Jelen tanulmány célja, hogy összehasonlítsa a fent említett két elemzőrendszert, a HuSpaCy-t és az e-magyart, különböző feldolgozási szinteken, beleértve a tokenizálást, a szófaji címkézést, a morfoszintaktikai elemzést és az egyéb nyelvi feldolgozási szinteket. Vizsgálatunk fő motivációja egy most készülő korpusz,<sup>1</sup> melyet részletes elemzéssel kell ellátnunk. A megfelelő elemző kiválasztásához elengedhetetlen egy alapos, módszeres összehasonlítás. Az összehasonlítás eredményei ezen felül hozzájárulhatnak a magyar nyelvtechnológiai alkalmazások fejlesztéséhez és optimalizálásához, valamint segíthetnek az alkalmazók számára a legmegfelelőbb elemzőrendszer kiválasztásában.

## 2. Kapcsolódó irodalom

A **HuSpaCy** egy ipari felhasználásra szánt, nagy hatékonyságú eszközlánc, amely a spaCy keretrendszerre építve biztosít alapvető nyelvfeldolgozási funkciókat, mint a tokenizálás, szófaji címkézés, lemmatizálás, dependenciaelemzés és tulajdonnévfelismerés (NER) (Orosz és mtsai, 2022, 2023). Célja a gyors és pontos elemzés erőforrástakarékos környezetben. A szerzők mérései alapján kimagasló pontosságot ér el a tokenizálásban és a mondathatárok felismerésében. Előbbiben szinte minden rendszerrel összehasonlítva a legjobb eredményt mutatta (99,89%-os F1-érték). A mondathatárok felismerése terén is kiemelkedő (97,66%), bár itt az e-magyar valamivel jobbnak bizonyult. A szófaji címkézésben a HuSpaCy különösen jól teljesít, az Universal Dependencies (UD) adatain tanított modell 94,7%-os pontosságot ér el, míg a Szeged Korpusz (Csendes és mtsai, 2004) adataival továbbfejlesztett modell 96,58%-ot. A függőségi elemzésben a HuSpaCy eredményei nem érik el a Stanza által elért legmagasabb pontszámokat. Lemmatizációs modulja az egyik legpontosabb a vizsgált rendszerek között. A Szeged Korpusz teljes adatkészletén tanított modell 95,53%-os pontosságot ért el, ami meghaladja az e-magyar teljesítményét (94,94%) és jelentősen jobb a UDPipe (88,5%) eredményeinél. A névelem-felismerés terén azonban nem éri el a BERT-alapú modellek pontosságát.

Az **e-magyar** rendszer célja a magyar nyelv elemzésére alkalmas eszközök integrálása egy moduláris, kutatásorientált keretbe. A legújabb, emtsv verzió (Indig és mtsai, 2019) teljes modularitást kínál, amely lehetővé teszi, hogy különböző modulok (pl. tokenizálás, morfológiai elemzés) különálló egységekként

<sup>1</sup> Részletek a tanulmány végső változatában.

működjenek, és az elemzőlánc bármely pontján kimenetet biztosítsanak. A szerzők által közölt teszteredmények alapján az emtsv hasonlóan teljesít más magyar nyelvi elemző rendszerekhez. Az elemzési lánc az **emToken** modul által végzett tokenizálással és mondathatárok meghatározásával kezdődik. Ezt követi a morfológiai elemzés az **emMorph** (Novák, 2014; Novák és mtsai, 2016) és az **emLem** modulokkal. Az egyértelműsítést a POS címkéző modul (**emTag**, Orosz és Novák, 2013) végzi. A morfológiai elemzés után a mondattani elemzés két irányban folytatódik: az **emDep** modul függőségi struktúrákat tár fel, míg az **emCons** modul az összetevők hierarchiáját állapítja meg. A szövegek további feldolgozása során az **emChunk** főnévi csoportokat azonosít, míg az **emNer** modul tulajdonneveket és speciális kifejezéseket jelöl. A rendszer tartalmaz kiegészítő modulokat is, mint például az **emZero**, amely zérónévmásokat illeszt be, és az **emTerm**, amely több szóból álló kifejezéseket annotál.

### 3. Módszer

Az összehasonlító elemzés alapjául a 2022. 06. 20-ai országgyűlési jegyzőkönyv szövege szolgált. A szöveg 2498 sorból, 290 606 karakterből áll, és nyelvezetét, formáját tekintve változatos. Az igényesen megfogalmazott, retorikus élőbeszéd mellett tartalmaz természetes élőbeszédből eredő kifejezéseket, mondatstruktúrákat is, a lejegyzésből adódó helyesírási hibákat, valamint a jegyzőkönyvi formából eredő nem nyelvi elemeket is (például felszólalók nevei, időpontok, sorszámok és paragrafusok jelzése, telefonszám, postacím, szövegtagoló karakterek). Ezek mind nagyszerű lehetőséget nyújtottak arra, hogy nem tipikus, de írott szövegben mégis előforduló nyelvi helyzetekben is teszteljük az elemzők teljesítményét. Választásunkat az is indokolja, hogy kifejezetten a készülő korpuszunk számára keressük a legmegfelelőbb elemzőláncot, így indokoltnak tűnik egy általunk gyűjtött, és később felhasználható szövegen összevetni őket, nem pedig meglévő, *gold standard*-nek tartott korpuszokon, amelyek jellemzően egyszerre csak egy-egy elemzési szint validálását tennék lehetővé. Így biztosítjuk, hogy olyan szövegen teszteljük a két elemzőt, amit azok korábban biztosan nem láttak.

Érdemes külön kiemelni a *whitespace* karakterek szerepét. A dokumentumban a szóközök mellett a leggyakoribb *whitespace* karakter a sortörés („\n”), ugyanis a bemeneti szöveg bekezdéseinek tagolását általában kettő vagy több egymást követő sortörés valósítja meg. Ezek nagymértékben befolyásolják az elemzési eredményeket a tokenizálás és a dependencia szintjén; kezelésük az elkészült összehasonlító program sarkalatos pontja. Általánosságban elmondható, hogy a szövegben nagy mennyiségben és sokféle formában előforduló központosítási, numerikus és *whitespace* karakterek nagyban megnehezítik a megfelelő összehasonlítás elkészítését mind a HuSpaCy és az e-magyar, mind pedig az őket összehasonlító alkalmazás szintjén.

Ez utóbbit a `Launcher_HuSpaCy_emagyar` alkalmazás<sup>2</sup> teszi lehetővé. Ez egy Python nyelven írt programcsomag, amely konzolos alkalmazásként, többféle

<sup>2</sup> <https://github.com/nytud/thesis-works.git>

módban indítható, és a működése során tetszőleges `txt` kiterjesztésű szövegen lefuttatja az elemzőket – vagy csak az egyiket, vagy mindkettőt. Az elemzési eredményeket fájlban eltárolja, valamint opcionálisan a terminálra is kiírja. Mindemellett lekérhető belőle a különböző elemzési szinteken kapott eredmények összehasonlítása egy egyszerű, de az adott elemzési szempont sajátosságaihoz illeszkedő táblázatos formában.

Az alkalmazás indítása a `launcher.py` program segítségével lehetséges. Indításkor parancssori argumentumként meg lehet adni az elemzendő fájl elérési útvonalát, a kívánt elemzőt (`-huspacy` vagy `-emagyar` kapcsoló), illetve az összehasonlítás támogatására szolgáló kapcsolók valamelyikét. Ezek lehetnek arra vonatkozó parancsok, hogy az összesített elemzés megjelenjen-e a terminálablakon is (`-oute` az emagyar, `-outh` a HuSpaCy számára), valamint a nyelvi szint összehasonlítási eredményének megjelenítésére vonatkozó kapcsolók (`-tok`: tokenizálás, `-morph`: morfológia, `-lem`: lemmatizálás, `-pos`: szófaji címkézés, `-dep`: dependencia, `-ner`: névelem-felismerés). Az összehasonlítási kapcsolók értelem-szerűen csak akkor lépnek életbe, ha mindkét elemzőt futtatjuk, egyéb esetben nincsen semmilyen hatásuk.

Az alkalmazás (továbbiakban: Launcher) túlmutat az elemzők lefuttatásán: egy egész környezetet valósít meg, amely minél több szempontból támogatja az elemzők nyelvészeti teljesítményének összehasonlítását. Ezek a szempontok a következő kategóriákba sorolhatók: futtatás, logikai egységesítés, különbségek kezelése, teljesítmény javítása, illetve egyszerű teljesítménymetrika.

### 3.1. Futtatás

A HuSpaCy egy SpaCy-könyvtár formájában működik<sup>3</sup>, így kényelmesen használható Python-scripteken belül. Jelen elemzésünkben a `hu_core_news_lg` HuSpaCy-modellt használjuk. Az e-magyar jelenlegi változata Docker-alkalmazásként működik<sup>4</sup>, így a felhasználónak kell kezelnie az ezzel járó műveleteket (pl. konténer indítása, futtatása). A Launcher egységesíti a két megközelítést, így a felhasználónak most már csak egyetlen Python-programot kell elindítania. Ez a program a `huspacy` könyvtár felhasználásával elvégzi a HuSpaCy lefuttatását, valamint a `docker` programkönyvtár<sup>5</sup> felhasználásával nyit egy konténert, lefuttatja az e-magyart, majd a konténerben keletkezett elemzési eredményt átemeli a konténerből a lokális tárhelyre, végül a konténert leállítja és törli.

### 3.2. Logikai egységesítés

Az összehasonlításhoz elengedhetetlen, hogy a két elemző kimenetét egy egységes logika szerint lehessen értelmezni. Formailag mindkét elemző lehetővé teszi egy-egy elemzési táblázat létrehozását, amelynek soraiban a kapott tokenek,

<sup>3</sup> <https://github.com/huspacy/huspacy>, <https://spacy.io/>

<sup>4</sup> <https://github.com/nytud/emtsv>

<sup>5</sup> <https://docker-py.readthedocs.io/en/stable/>

oszlopaiban pedig az adott tokenre vonatkozó, elemzési szempontok szerinti ki-  
menet látható. A megjelenítendő oszlopokat az alapján választottuk ki a HuS-  
paCy eszközkészletéből, hogy melyek azok a nyelvi szintek, amelyekben az elemzés  
megfeleltethető az e-magyar valamelyik elemzési szempontjának (token, lemma,  
morfológiai elemzések több típusban, szófaji címke, függőségi címke és fej).

A táblázatos forma létrehozása megkövetelt egyfajta elvi szintű egységesítést  
is, ugyanis a két elemző tokenizálási szemlélete is eltér, illetve a gyakorlatban  
létrejött eredmény is különbözhet több esetben (ez utóbbiról részletesebben ld.  
Eredmények szekcióban). A szemléletbeli eltérés kritikus pontja a *whitespace* ka-  
rakterek kezelése. Az e-magyar tokenizálása viszonylag intuitív: a tokenek azok az  
egységek, amelyeket az elemzőrendszer egy szónak vagy egy központoszási nyelvi  
formának ítél. *Whitespace* karakterek alapvetően nem kerülnek a tokenek közé,  
csak azokban az esetekben, amikor azok is a token szerves részét képezik (pl.  
szóközzel tagolt számok esetén; „1 441-4000” egy tokennek számít). Ezzel szem-  
ben a HuSpaCy a *newline* karaktert tartalmazó *whitespace*-karakter-csoportokat  
tokenként tünteti fel, sőt, elemzést is próbál társítani hozzájuk, amelyeknek az  
esetek többségében természetesen nincsen értelme. Ezek a *whitespace* bemenetek  
szerencsére informatikailag kiszűrhetők, ezt a Launcher meg is valósítja.

A függőségi elemzés összehasonlításánál nagy különbséget okoz az, hogy nem  
ugyanazzal a címkehalmazzal dolgozik a két elemző, így ezek között konverzió  
szükséges. Az alkalmazás komponensei között így megtalálható a `dep_converter`  
fájlban egy egyszerű, szótár jellegű konverter, amely erre tesz kísérletet. Mivel a  
HuSpaCy által használt Universal Dependencies <sup>6</sup> címkehalmaz jóval bővebb az  
e-magyar által használt halmaznál (Vincze és mtsai, 2010), a konverzió a HuS-  
paCy címkéiről történik az e-magyar címkéire. A nem egyező számosság miatt a  
megfeleltetésben alapvetően történik információvesztés, és emellett néhány cím-  
ke esetén éppen az e-magyar dolgozik részletesebb megnevezésekkel, így nem is  
teljesen egyértelmű a konverzió még egy irányban sem. Mindenesetre a konver-  
ter – bár elkerülhetetlenül magában hordoz egyfajta pontatlanságot – legalább  
alapvető szinten közös nevezőre tudja hozni a címkéket, így már értelmezhető  
közöttük összehasonlítás, egyenlőségvizsgálat.

A formai különbség mellett a dependencia elvi szinten is eltér az elem-  
zőkben. Az e-magyar függőségi elemzése minden mondatban beszámolja a to-  
keneket 1-től a mondat hosszáig. A dependenciafa gyökerének választott elem a  
0 fejcímkét kapja, a többi token pedig annak az elemnek az indexét, ami az ő  
feje a függőségi fában. 0 címkét kap még tipikusan a mondatzáró írásjel, illetve  
néhány nemtipikus esetben egy-egy olyan elem, amelyet az elemzés sehova sem  
tudott megfelelően bekötni (ld. Eredmények szekció). Így az e-magyar tisztán  
indexekkel dolgozik, amelyek közül néhányhoz (a 0-ás indexekhez) nem is tarto-  
zik token. Ehhez képest a HuSpaCy dependenciafejként rögtön azt a tokenet adja  
vissza, amely fejként szolgál, ROOT esetében pedig önmagát. Így a Launcher a  
HuSpaCy logikájához igazodva visszavezeti az e-magyar indexeit is tokenekre,  
hogy lehessen tényleges egyenlőségvizsgálatot végezni a dependenciafejekre is.

<sup>6</sup> <https://universaldependencies.org/u/dep/index.html>

A névelemfelismerés esetében is szükség van egy minimális egységesítésre: az e-magyar által használt „1-” és „E-” címkéket a HuSpaCy „B-”, illetve „I-” címkéjére konvertáljuk.

A tokenizálásban azokat az eseteket is tudnunk kell kezelni, amikor a két elemző különböző tokenizálási eredményt ad. Ez általában elcsúszást eredményez az összehasonlítási táblázat soraiban. A Launcher ezt heurisztika segítségével szintén megoldja.<sup>7</sup>

### 3.3. Teljesítményjavítás

A morfológiai elemzéshez az e-magyarban 2 formátum áll rendelkezésre: az xpostag és az upostag (utóbbi jelenti a Universal Dependencies stílusú annotációt). Az elemzőláncban a morfológiai részt az EmMorph komponens valósítja meg. A HuSpaCy-ben szintén megtalálható egy integráción keresztül az EmMorph komponens, így a HuSpaCy is képes azt a részletes és szemléletes címkehalmozatot használni, amelyet az e-magyar xpostag. A HuSpaCy először lefuttatja az EmMorph-komponens elemzését, majd ezt átkonvertálja UD-formátumra is (Vadász és Simon, 2019). Ezután lefuttatja a SpaCy-jellegű, saját morfológiai elemzését is, amely alaphoz UD-formátumú. Az integrációs feladat nehézsége abban rejlik, hogy az EmMorphból közvetlenül és konverzióval kapott eredmények listák a lehetséges elemzésekről, az integrált komponens ugyanis az egyértelműsítési logikát (tehát a valóban helyes elemzés kiválasztását) nem tartalmazza. Ezt kell megvalósítania az integráló kódrészletnek. Ehhez összeveti a lista elemeit a saját elemzésével, és az annak leginkább megfelelő EmMorph-elemzéseket adja vissza. Az integráció bizonyos szintű hibát visz a rendszerbe, ugyanis a kiválasztás alapjául szolgáló metrika sok esetben nem a helyes választást adja meg, illetve bizonyos tokenekre az integrált komponens sem fut le megfelelően (None eredményt kapunk). További érdekesség, hogy az eredeti integrációs változat használatával az elemzés több esetben össze is omlik, ugyanis nem tudja megfelelően kezelni a felső- és túlzófokú mellékneveket. Egy párhuzamosan futó projekt keretében ezt az integrációt informatikailag is javítottuk. Jelen összehasonlítást lefuttattuk az eredeti verzió<sup>8</sup> hibamentesített változatával<sup>9</sup> és a javított változattal is. Jelen tanulmányban a két változat eredményeit egyaránt közöljük, a javítási folyamat informatikai részleteit és a jövőben várhatóan elkészülő végleges verzióját egy külön tanulmányban mutatjuk be.

A Launcher lefuttatása után kapott eredményeket részletesen kiértékeljük, az ebből levont következtetéseket nyelvi szintenként közöljük.

<sup>7</sup> Ennek részletes leírását a Launcher dokumentációja tartalmazza.

<sup>8</sup> <https://github.com/huspace/huspace/blob/master/docs/recipes/emmorph.md>

<sup>9</sup> A továbbiakban ezt a hibamentesített változatot hívjuk eredeti változatnak, ugyanis a program logikája a hibajavítás mellett is alapvetően változatlan maradt.



## 4. Eredmények

### 4.1. Tokenizálás

A tokenszám alapvetően különbözik: a HuSpaCy 46607 tokent talált, míg az e-magyar 46452-t. Az eltérést kategorizálni lehet az alábbi típusokba:

- zárójelben („(,”) vagy „[,]”) lévő mondat utolsó szava utáni pontot az e-magyar a szó részének veszi (nem külön token a pont), a HuSpaCy pedig nem (153 előfordulás)
- „(1)” : HuSpaCy szerint 3 token, e-magyar szerint 2: „(” és „1)” (7 előfordulás)
- hármas kötőjeles szó: e-magyar szerint a kötőjelek és a szavak is külön tokenek (5 token), HuSpaCy szerint pedig az egész 1 db token (5 esetért felel)
- szóközzel tagolt számok: HuSpaCy külön tokennek veszi őket szóköz szerint, az e-magyar viszont egyben kezeli a számokat, 1 db tokent készít (5 esetért felel)
- sorszám (pl. „13.”, „2022.”): e-magyar szerint külön token a pont, HuSpaCy szerint nem (3 esetért felel)

A maradék elcsúszásért 1 vagy 2 előfordulásban egy-egy speciális token felel. Ezek szinte mind atipikus kifejezések, például rövidítések, speciális sorszámok, központosási csoportok, helyesírási hibás alakok közvetlen környezetében található kifejezések. Nem látható egyértelmű tendencia arra nézve, hogy az egyik elemző ezeket inkább egy tokennek, míg a másik többnek venné, vegyesen előfordul mindkét elemzőnél mindkét hozzáállás. Ha ezeket az eseteket is külön-külön számoljuk, összesen 15 szempont van, ahol a két elemző eltér.

Az eltérő eseteket manuálisan is áttekintettük és osztályoztuk aszerint, hogy biztosan hibás-e az egyik elemző, vagy pedig vitatható, hogy melyik elemzőnek adjunk igazat<sup>10</sup>. Az eset előfordulási számával való súlyozás nélkül a HuSpaCy 6 esetben egyértelműen jobb az e-magyarnál, az e-magyar 5 esetben a HuSpaCynél, és 4 db véleményes eset van. Így a HuSpaCy jobb az összes eset 40%-ában és az egyértelmű esetek 54,54%-ában. Súlyozással (188 hibahellyel és a fent közölt előfordulási számokkal) számolva a HuSpaCy 169 db esetben jobb, az e-magyar 11 esetben, illetve 8 véleményes esetről beszélünk. A HuSpaCy itt egyértelműen jobb, az összes eset 89,9%-ában győz az e-magyar elemzéséhez képest, az egyértelmű eseteket tekintve pedig ez az arány 93,9%-os. A zárójelben lévő mondatokon tehát az e-magyar szinte konzekvensen hibázik, és mivel a tesztszöveg ilyen esetekben bővelkedett, a 153 előfordulás a többi eltérési esethez képest kiugróan magas. Azonban súlyozás nélkül is látható, hogy a HuSpaCy jobb teljesítményt nyújt, mint az e-magyar.

### 4.2. Morfológia

Ahogy a Teljesítményjavítás részben is láthattuk, a morfológiai elemzéshez többféle kimenettel is dolgozhatunk. Jelen elemzésünkben az EmMorph komponens által használt címkehalmazzal dolgozunk, amely szemléletes és részletes

<sup>10</sup> Ez az elemzés, illetve minden további elemzés részletes adatai a GitHub repository-ban lesznek elérhetőek.

elemzést nyújt a tokenek morfológiai szerkezetéről. Így az összehasonlítás során az EmMorph-beli elemzési eredményeket hasonlítottuk össze, amelyeket a két elemző kiválaszt mint helyes morfológiai elemzést. Az eredeti Huspacy-EmMorph integráció felhasználásával 46635 tokenen futtatva azt kaptuk, hogy 18796 esetben különbözött az elemzés (40,3%), speciális karakterek nélkül 11091 esetben (23,8%), és ebből azok az esetek, amikor a Huspacy-EmMorph None eredménnyel tért vissza (az integráció valamely részén történt probléma miatt) 4987-en voltak. A javított integrációval ezek a számok így változtak: 13883 különbség (29,8%), ebből a speciális karaktereket kiszűrve 6178 db (13,24%). (A None eredményű esetek száma nem változott.)

A speciális karakterek felelnek a HuSpaCy-EmMorphban a None kimenetek jelentős részéért, így ezeket leszűrve sokkal informatívabb képet kapunk arról, hogy milyen típusú szavakra nem működik megfelelően a HuSpaCy. Ezekről elmondható, hogy többnyire jól meghatározható típusú szavak: vezetéknév, szervezetek nevei, helyesírási hibás alakok, számokat tartalmazó kifejezések. A helyes, csak valamiért az elemző számára szokatlan szavak közül néhány példa: *alacsonyapdlós*, *ambiciózus*, *Covid* szót tartalmazó kötőjeles szóösszetételek stb.

Mivel a None kimenetek száma jelentős és az eltérések száma is viszonylag nagy, a HuSpaCy-EmMorph elemző több bizonytalanságot mutat az e-magyaránál. Ennek valamekkora része valószínűleg az integráció informatikai nehézségein alapszik, így a HuSpaCy-EmMorph a két eszköz közül jelenleg a kevésbé biztonságos választás.

### 4.3. Lemmatizálás

A lemmatizálás során is használhatjuk a HuSpaCy-EmMorph integrációt, amely elkészíti a lemmatizált alakot. Elemzésünkben az e-magyar saját lemmatizálóját, a HuSpaCy-EmMorph lemmatizálási eredményét, valamint a HuSpaCy SpaCy-féle lemmatizálóját használjuk. Ezzel 3 címke között vizsgálunk egyenlőséget.

Az eredeti integrációt használva 14826 különbséget kapunk (31,8%), speciális karakterek nélkül 7121-et (15,2%), amelyből 4987 None kimenet. A javított integrációval a különbségek száma 14398-ra csökken (30,9%), speciális karakterek nélkül számolva 6693-ra (14,4%), amelyből – a morfológiához hasonlóan – 4987 None kimenetű alak. Megfigyelhető tendencia, hogy az e-magyar és a HuSpaCy-EmMorph meglehetősen gyakran köznevesíti azokat a tulajdonnévi lemmákat, amelyekben köznevet ismer fel (így például a keresztneveket nem, a minisztériumok neveit viszont igen), és ezt nem feltétlenül egymással összhangban teszik. Viszont az e-magyar a csupa nagybetűs vezeté- és keresztneveket képes visszaalakítani megfelelő tulajdonnévi formára (pl. *BRENNER*-ből *Brenner*), míg a HuSpaCy ezeket meghagyja eredeti formájukban. Azonban az egyik megfigyelési szűrés során fény derült az e-magyar egy súlyosabb hibájára: a *Novák* vezetéknév lemmája 8 alkalommal *Nova* az e-magyar elemzésében.

További érdekesség, hogy a *-va/-ve* ragot a HuSpaCy meglehetősen nehezen tudja kezelni. Egy eset kivételével az összes olyan esetben, ahol hibázik – azaz vagy tévesen leválaszt egy *-va/-ve* szóvégződést, vagy pedig nem választ le egy *-va/-ve* toldalékot –, a lemma részének értékeli a *-va/-ve* toldalékot, abban az

egy kivételes esetben pedig rosszul képez lemmát a szóból (*vezényelve* – *vezényelw*). A *-va/-ve* végződésű szavakat leszűrve a ténylegesen ragozott alakokra 125 előfordulást kapunk (tegyük fel, hogy most csak a toldalék jelenlétét figyeljük, nem pedig ennek a helyességét). Ebből az *illetve* szónál konzekvensen egyik elemző sem választja le a toldalékot, és 9 további szó esetében szintén mindkét elemző a lemmán hagyja a *-va/-ve* ragot (ezeknél viszont nem minden esetben mondható el a konzekvencia, vagy túl kevés az előfordulás, hogy konzekvenciát tudjunk megállapítani). A HuSpaCy hasonlóan a lemma részének értékeli gyakran a *-hat/-het* képzőt is.

Mindhárom elemzőeszköz számára problémás a létige és annak különböző módokon képzett és ragozott alakjai (629 token, a viszonylag távolról származtatható alakokat is beleszámítva, pl. *lehetőség*). A HuSpaCy a múlt idejű és a jelen idejű alak különböző változataira mond *van* lemmát (320 eset), a jövő idejű létigéből származtatható igei és igenévi alakokat *lesz* lemmával jelöli (amennyiben felismeri benne a létigei tövet: *lehet, lenne, lennie* stb.; 201 előfordulás). Az e-magyar a *lesz* alakot a *lenne, lennie* stb. szóalakokra javasolja (100 eset). Az e-magyarban *van* a válasza a *van, volt* és *lehet* alakjaira (436 eset). Egy *volt* előfordulásnál az e-magyar *volt* lemmát mond, a HuSpaCy *van*-t.

A None kimenetek nagy száma miatt kijelenthetjük, hogy a HuSpaCy-EmMorph rossz teljesítményt nyújt. A fent kiemelt eseteket tekintve az a konklúzió vonható le, hogy az e-magyar pontosabban dagozik. Amikor azonban hibázik, akkor a hibája súlyosabb, mint a HuSpaCy-é, amely bizonytalanság esetén többnyire változatlanul hagyja a szövegbeli alakot.

#### 4.4. Szófaji címkézés

Összesen 4-féle mód van szófaji címkézésre: a HuSpaCy *pos* és *tag* elemzése, amely a SpaCy-ből ered, és egyszerű / részletezett címkézést hajt végre, valamint a HuSpaCy-EmMorph-ből eredő *pos*-címké és az e-magyar saját címkéi.

Az eredeti integrációval itt 13540 különbségről indulunk (29%), speciális karakterek nélkül 5835-ről (12,5%), amelyekből None kimenetű eset továbbra is 4987 (10,7%). A javított változat itt nem változtat a számokon.

Érdekes jelenség, hogy a két elemző között nincsen 100%-os egyetértés a PUNCT címkék kiosztásában. Az e-magyar az aposztrófot, valamint a megkülönböztetetten alsó és felső idézőjelet felismeri PUNCT elemként, a HuSpaCy viszont mindenféle más címkét ad ezeknek, PUNCT címkét egyszer sem. Olyan tokenek is előfordulnak, amelyeket a HuSpaCy PUNCT címkével látott el, az e-magyar pedig az ADJ, NOUN vagy NUM címkék valamelyikével: *);, §, 15:00* stb. Mindkét elemzőnek vannak nehézségei a központosási karakterek megállapításában, viszont inkább az e-magyar kerül ki győztesként ebből az összehasonlításból.

Az eredeti bemenet első 929 tokenes részében kézzel is áttekintettük azt a 16 tokent, amelyeknél az elemzők különböző, de értelmes eredményt adtak (tehát azoktól az esetektől eltekintettünk, amelyeknek a HuSpaCy-EmMorph None kimenetet ad, hiszen ezek mindenképpen hamis eredményt adnak az összehasonlításnál). A 4 eszköz közül a HuSpaCy *tag* nyújtja a leggyengébb teljesítményt, a többi három nagyjából hasonlóan jól teljesít.

#### 4.5. Függőségi elemzés

A függőségi elemzés kérdésében mindkét elemző elég bizonytalan. A kezdeti tesztek lefuttatása után úgy döntöttünk, a szöveg első néhány bekezdésén készítjük el az összehasonlítási elemzést, ugyanis ez egy mindössze 929 tokenes, belátható és manuálisan ellenőrizhető méretű bemenet. Ez a szövegrészlet nem tartalmazza azt a mondatot, amely az e-magyar elemzésében súlyos elvi hibát okoz.

Az elvi hiba az, hogy az e-magyar nem állapít meg ROOT címkét a következő mondatban: „(Szórványos taps a Momentum soraiban.)”. (A mondat ténylegesen zárójelben van az eredeti szövegben.) Az a token, amelynek ezt a címkét fel kellene vennie (*taps*), APPEND címkét kap. A mondat szinte minden tekintetben normális, egy tipikus tokenizálási hiba van benne: a zárójelben lévő mondat utolsó szava és a mondatvégi írásjel egy tokenné kategorizálása. Azonban van egy minimálpárja és 94 hasonló szerkezetű („(Taps a [...] soraiban.)”) rokon jellegű mondata a szövegben, és mindegyik kap ROOT címkét.

Elvi hibája a HuSPaCy-nek is van; erről esett már néhány szó a különbségek kiegyelésénél. Mivel a dependenciafej választása a teljes tokenkészletből dolgozik, előfordul, hogy egy *whitespace*-csoportot kap fejként valamelyik token.<sup>11</sup>

A dependenciacímke 289 esetben (31,1%) különbözik a két elemző esetében (itt megjegyezzük, hogy ennek bizonyos részét a konverzióból adódó hibák is okozhatják). Ebből azon esetek száma, ahol a fej sem egyezik, 222 db (23,9%). A fej összesen 437 helyen (47%) nem egyezik a két elemzőnél. Az adatokra ránézve itt is láthatjuk, hogy a speciális karakterek meglehetősen kérdéses esetek, így itt is kiszűrhetjük őket. Ekkor a dependenciacímke 286 (30,1%) esetben különbözik, a dependenciafej 306 esetben (32,9%), mindkettő egyszerre pedig 221 esetben (32,8%) tér el.

Érdekes eset az *és* kötőszó is. A 16 db előforduló *és* token mindegyike esetében különbözik a fej, és 15 esetében meg is állapítható, hogy az e-magyar az első lehetőséget, a HuSpaCy pedig a második lehetőséget választja. Ebben a tekintetben tehát a HuSpaCy működik az UD elveknek megfelelően<sup>12</sup>.

<sup>11</sup> Ezt azért nem lehet megoldani ugyanolyan eredmény szintű szűréssel, mint az eddigieket, mert itt „bemenetet” kellene szűrnünk, azt a halmazt, amiből a HuSpaCy dolgozik a fejek választásánál. Kisebb technikai nehézség, hogy amikor a fejként kapott *whitespace*-csoport tartalmaz sortörést, akkor ez ténylegesen sortörésként (nem pedig „\n” karakterként) jelenik meg, így elcsúsztatja a táblázatot. Ezt is megoldja a Launcher olyan módon, hogy az ilyen esetekben a valódi fej helyett „HEAD IS WHITESPACE!” üzenettel helyreállítja a táblázatot. Mivel a bemeneti szöveg gyakorlatilag bármilyen mennyiségű és összetételű *whitespace*-csoportot tartalmazhat, egységesen kezeljük a sortörés tartalmazásának ténye alapján az összes lehetőséget. (A szöközők és tabulátorok nem okoznak ilyen elcsúszást, így ezeket eredeti formájukban hagyjuk, így is megfelelő információt hordoznak.) Ezzel veszítünk annyi információt, hogy pontosan milyen összetételű a csoport, de mivel egyébként is értelmetlen a token, és nyilván hibás, hogy fejként szerepel, ezzel a gyakorlatban nem veszítünk érdemi információt. A rövidített bemeneti fájlban 8 ilyen eset szerepel (0,9%).

<sup>12</sup> A Universal Dependencies útmutatása alapján a mellérendelések első elemét tekintjük a szülő csomópontnak: „Coordinate structures are in principle symmetri-

Összességében tehát azt látjuk, hogy az e-magyar van, hogy eltér a Universal Dependencies formátumtól, amelyre az elemzőt használó jogosan számíthatna. Függőségi elemzésre inkább a HuSpaCy használata alkalmas, kiszámíthatósága és bővebb címkehalmaza miatt.

#### 4.6. Névelem-felismerés

A HuSpaCy egyértelműen sokkal több névelemet talál meg egy adott szövegben, mint az e-magyar. Ennek feloldásához az alap országgyűlési fájl első néhány bekezdését (929 tokenes rész) manuálisan is annotáltuk, és ehhez viszonyítva is megnéztük az elemzők teljesítményét.

A névelemek felismerését többféleképpen is megközelíthetjük, és az alább felsoroltak közül a Launcher mindegyiket megvalósítja. Az első megközelítés az, hogy minden egyes tokenről meg tudjuk mondani, hogy névelem-e, ehhez használhatjuk az IOB-notációt, amely mindkét elemzőben megtalálható. Az eredmények a teljes fájlra futtatva: 46635 tokenből 1402 esetben (3%) van eltérés. A HuSpaCy 44099 db tokent azonosított nem-névelemként, az e-magyar pedig 44689 db-ot, ebből az 590-es különbségből látható, hogy 10000-es nagyságrendű hosszú bemenet esetén körülbelül 100-as nagyságrendű eltérés van az elemzők között. A kézzel annotált, rövidített tesztfájlon 929 tokenből 18 különbséget látunk (1,9%), a HuSpaCy 10-zel több névelemet talált.

A másik megközelítés arra fókuszál, hogy egy adott tulajdonnév milyen címkét (címkéket) kap a teljes szövegen belül. A Launcher által előállított táblázatban elsőként a HuSpaCy és e-magyar által megítélt címkék halmazainak egyenlőségét látjuk, majd a közösen megtalált névelemet, illetve rendre a HuSpaCy és az e-magyar által megítélt címkék halmazát. Természetesen vannak névelemek, amelyeket a másik elemző nem feltétlenül talált meg, ezeket a táblázat után listába gyűjtve láthatjuk (ezúttal az egyenlőségvizsgálat és a másik elemző halmaza nélkül). Ez a típusú megközelítés 283 közösen megtalált névelemet jelez az eredeti bemeneten, valamint 288 csak a HuSpaCy által megtalált névelemet és 28 csak az e-magyar által megtalált névelemet. (Fontos megjegyezni, hogy itt egy token teljesen megegyező előfordulásait egynek számoljuk.) A 284 db közösen megtalált névelem esetén a címkehalmaza 51 esetben tér el (18%).

A címkéket összefoglalva az látszik, hogy az e-magyar esetében az ORG megállapítása a legnehezebb. Az e-magyar által ORG-nak ítélt szavak szinte mind-egyikéhez a HuSpaCy talált más értelmezést is. Az önmagukban álló vagy toldalékolt országnevekben konzekvens eltérés figyelhető meg: a HuSpaCy mindig ORG-nak, az e-magyar mindig LOC-nak tekinti őket. Az e-magyar sok esetben nyilvánvalóan téved (pl. személyneveknél ezt a szöveg vagy *gold standard* címkézés nélkül is meg lehet állapítani), így erősen valószínű, hogy a HuSpaCy teljesítményét ebben a komponensben jobbnak mondhatjuk. Az 4.6 táblázatban láthatunk egy összehasonlítást a rövid fájlrészleten, manuális elemzéssel.

---

cal, but the first conjunction is by convention treated as the parent (or “technical head”) of all subsequent coordinated clauses via the conj relation.” (<https://>

névelem	kézi	HuSpaCy	e-magyar	névelem	kézi	HuSpaCy	e-magyar
Országgyűlés	ORG	ORG	ORG	Szűcs Lajos	PER	PER	PER
Hiszékeny Dezső	PER	PER	PER	Ház	ORG	nincs	ORG
Országgyűlés(ről)	ORG	ORG	nincs	Országgyűlés	ORG	ORG	ORG
Fidesz	ORG	ORG	ORG	Országgyűlés	ORG	MISC	ORG
Ungár Péter	PER	PER	PER	LMP	ORG	ORG	nincs
UNGÁR PÉTER	PER	ORG	nincs	LMP	ORG	ORG	nincs
Országgyűlés	ORG	PER	ORG	EU(-integráció)	ORG	nincs	nincs
Magyarország	LOC	LOC	LOC	Magyarország(on)	LOC	LOC	LOC
Thomas Piketty	PER	PER	PER	Magyarország	LOC	LOC	LOC
Európai Unió(tól)	ORG	ORG	ORG	Magyarország	LOC	ORG	LOC
Európai Unió	ORG	ORG	ORG	Európai Unió(t)	ORG	ORG	ORG
Egyesült Államok(kal)	LOC	ORG	LOC	Európai Unió	ORG	ORG	ORG
Audi(nak)	ORG	ORG	ORG	Tanács	ORG	ORG	nincs
Rétvári Bence	PER	PER	nincs	Audi	ORG	ORG	ORG
Magyarország(nak)	LOC	ORG	LOC	Audi(nak)	ORG	ORG	nincs
Audi(nak)	ORG	ORG	ORG	Ház(nak)	ORG	ORG	ORG
Bundestag(nak)	ORG	ORG	ORG				

1. táblázat. NER-összehasonlítás. Az első és ötödik oszlop a humán elemzés során megtalált névelemeket, a „kézi” oszlop a humán címkét, a HuSpaCy és e-magyar oszlopok pedig az elemzők által adott címkét tartalmazzák, illetve a „nincs” szót, ha az adott elemző az adott névelemet nem jelölte meg.

A manuális elemzés 35 db néveleméből teljes egyezés 19 db (54,3%). A HuSpaCy 25 alkalommal címkézett jól (71,43%), 2 alkalommal (5,71%) nem találta meg a névelemet, 7 alkalommal rossz címkét adott (20%). Az e-magyar 26 alkalommal teljesített jól (74,29%), 8 névelemet nem talált meg (22,86%), nem volt olyan, hogy rossz címkét adott volna (0%).

Összességében megállapíthatjuk, hogy a HuSpaCy jóval nagyobb fedéssel dolgozik, viszont az e-magyar típuscímkéi pontosabbak.

## 5. Összegzés

Az összehasonlítás eredményei alapján megállapítható, hogy a HuSpaCy és az e-magyar elemzők különböző erősségekkel és gyengeségekkel rendelkeznek, így a választás a konkrét alkalmazási céltól függ. A HuSpaCy kiváló teljesítményt nyújt tokenizálásban, névelem-felismerésben és dependenciacímkézésben, különösen ipari környezetben, ahol gyorsasága és precizitása előnyt jelenthet. Ugyanakkor a morfológiai és lemmatizálási teljesítménye, különösen az EmMorph integráció során tapasztalt problémák miatt, alulmúlta az e-magyart. Az e-magyar elemző pontosabb morfológiai és lemmatizálási eredményeket nyújtott. Összehasonlításunk természetesen nem teljeskörű; egy adott stílusrétegbe tartozó szöveg

[universaldependencies.org/u/dep/conj.html](https://universaldependencies.org/u/dep/conj.html)), a kötőszót pedig mindig az utolsó elemhez kötjük (<https://universaldependencies.org/u/dep/cc.html>).

egy példányán vetettük össze a két elemzőt, korlátozottan kvantitatív módon. Teljesebb, pontosabb és nagyobb volumenű összehasonlítást tenne lehetővé egy olyan szöveg, amely megfelelően nagy méretű, és minden nyelvi szinten *gold standard* annotációval van ellátva. Ennek előállítása még várat magára. További feladataink közé tartozik a HuSpaCy-EmMorph integráció javítása, hogy az egyébként kiegyensúlyozott teljesítményt nyújtó HuSpaCy a lemmatizálás és a morfológia terén az e-magyaréhoz hasonló pontossággal tudjon dolgozni.

## Köszönetnyilvánítás

A kutatást az MTA „Tudomány a Magyar Nyelvért Nemzeti Program” támogatta.

## Hivatkozások

- Csendes, D., Csirik, J., Gyimóthy, T.: The Szeged Corpus: A POS tagged and syntactically annotated Hungarian natural language corpus. In: Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora (LINC 2004) at The 20th International Conference on Computational Linguistics (COLING 2004). pp. 19–23 (2004)
- Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: spaCy: Industrial-strength Natural Language Processing in Python (2020)
- Indig, B., Sass, B., Simon, E., Mittelholcz, I., Kundráth, P., Vadász, N.: emtsv – Egy formátum mind felett. In: Magyar Számítógépes Nyelvészeti Konferencia. Szeged (2019), in Hungarian
- Novák, A.: A New Form of Humor – Mapping Constraint-Based Computational Morphologies to a Finite-State Representation. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014). European Language Resources Association (ELRA), Paris (2014)
- Novák, A., Siklósi, B., Oravecz, Cs.: A new integrated open-source morphological analyzer for hungarian. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), Paris (2016)
- Orosz, Gy., Novák, A.: PurePos 2.0: A Hybrid Tool for Morphological Disambiguation. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013). pp. 539–545 (2013)
- Orosz, Gy., Szabó, G., Berkecz, P., Szántó, Zs., Farkas, R.: Advancing Hungarian Text Processing with HuSpaCy: Efficient and Accurate NLP Pipelines. In: Ekštejn, Kamil and Pártl, František and Konopík, Miloslav (szerk.) Text, Speech, and Dialogue. pp. 58–69. Springer Nature Switzerland, Cham (2023)
- Orosz, Gy., Szántó, Zs., Berkecz, P., Szabó, G., Farkas, R.: HuSpaCy: an industrial-strength Hungarian natural language processing toolkit. In: XVIII. Magyar Számítógépes Nyelvészeti Konferencia. pp. 59–73 (2022)

- Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A Python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020)
- Simon, E., Indig, B., Kalivoda, A., Mittelholcz, I., Sass, B., Vadász, N.: Újabb fejlemények az e-magyar háza táján. In: Berend, G., Gosztolya, G., Vincze, V. (szerk.) XVI. Magyar Számítógépes Nyelvészeti Konferencia. pp. 29–42. Szegedi Tudományegyetem Informatikai Tanszékcsoport, Szeged (2020)
- Straka, M.: UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. pp. 197–207. Association for Computational Linguistics, Brussels, Belgium (October 2018), <https://www.aclweb.org/anthology/K18-2020>
- Vadász, N., Simon, E.: Konverterek magyar morfológiai címkekészletek között. In: XV. Magyar Számítógépes Nyelvészeti Konferencia. pp. 99–111. Szegedi Tudományegyetem TTIK, Informatikai Intézet, Szeged (2019)
- Vincze, V., Szauter, D., Almási, A., Móra, Gy., Alexin, Z., Csirik, J.: Hungarian Dependency Treebank. In: Proceedings of LREC 2010. ELRA, Valletta, Malta (May 2010)
- Váradi, T., Simon, E., Sass, B., Gerőcs, M., Mittelholtz, I., Novák, A., Indig, B., Prószéky, G., Vincze, V.: Az e-magyar digitális nyelvfeldolgozó rendszer. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2017). pp. 49–60. Szegedi Tudományegyetem Informatikai Tanszékcsoport, Szeged (2017)
- Zsibrita, J., Vincze, V., Farkas, R.: magyarlanc: A toolkit for morphological and dependency parsing of Hungarian. In: Proceedings of RANLP. pp. 763–771 (2013)



# HuAMR: A Hungarian AMR Parser and Dataset

Botond Barta<sup>1</sup>, Endre Hamerlik<sup>1</sup>, Milán Konor Nyist<sup>2</sup>, Judit Ács<sup>1</sup>  
{botondbarta,hamerlik.endre,acsjudit}@sztaki.hu,  
nyist.milan78@gmail.com

<sup>1</sup>HUN-REN Institute for Computer Science and Control  
<sup>2</sup>Eötvös Loránd University

**Abstract.** We present **HuAMR**, the first Abstract Meaning Representation (AMR) dataset and a suite of large language model-based AMR parsers for Hungarian, targeting the scarcity of semantic resources for non-English languages. To create HuAMR, we employed Llama-3.1-70B to automatically generate silver-standard AMR annotations, which we then refined manually to ensure quality. Building on this dataset, we investigate how different model architectures — mT5 Large and Llama-3.2-1B — and fine-tuning strategies affect AMR parsing performance. While incorporating silver-standard AMRs from Llama-3.1-70B into the training data of smaller models does not consistently boost overall scores, our results show that these techniques effectively enhance parsing accuracy on Hungarian news data (the domain of HuAMR). We evaluate our parsers using Smatch scores and confirm the potential of HuAMR and our parsers for advancing semantic parsing research.

## 1 Introduction

Large Language Models (LLMs) have revolutionized the field of natural language processing, achieving state-of-the-art results across tasks like translation, summarization, and question answering (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023). These models have demonstrated remarkable capabilities in understanding and generating human-like text, opening up a wide range of research and application possibilities.

Despite these advancements, LLMs often exhibit issues related to factual accuracy and consistency, sometimes generating outputs that are incorrect or misleading (Ji et al., 2023; Maynez et al., 2020). This limitation is particularly evident in tasks requiring deep semantic understanding and faithful representation of information (Cao et al., 2022).

A critical challenge underlying these issues is the representation of meaning in the generated text. LLMs primarily rely on statistical patterns in data, which can lead to a lack of deep semantic comprehension (Bender et al., 2021). Abstract Meaning Representation (AMR) (Banarescu et al., 2013) offers a solution through a structured, graph-based formalism that represents the core meaning of sentences. This representation enables more accurate language understanding and generation.

AMR has been successfully applied in various contexts to improve semantic interpretation. For instance, Zhu et al. (2020) explored ways to integrate AMR-derived graph information into Transformer-based architectures, demonstrating how semantic graphs can enhance model performance on parsing tasks. Dou et al. (2022) further showed that leveraging AMR can improve factual consistency in summarization, underscoring the potential of AMR to enhance both the quality and accuracy of generated text.

Despite these advancements, most AMR resources and tools are available only for English, limiting the broader applicability of these techniques (Biloshmi et al., 2020). There is a critical need to develop AMR tools for a wider range of languages to enhance the semantic capabilities of LLMs globally.

In this work, we address this gap by translating the gold standard AMR (AMR 3.0 Knight et al. (2020)) dataset into Hungarian, creating a valuable resource for this language. We present and publish a suite of language models, ranging from small (1B parameters) to large (70B parameters), capable of generating AMRs for Hungarian texts<sup>1</sup>. Our main contributions are:

- **Creation of a Hungarian AMR dataset:** We translate the AMR 3.0 dataset into Hungarian, providing the first AMR resource for this language. We also publish a synthetic AMR dataset (HuAMR) generated with a Llama-3.1-70B model for further coverage.
- **Development of high-performance Hungarian AMR parsers:** We develop and release a series of AMR parsers for Hungarian texts, via finetuning mT5 Large and Meta Llama 3.2 models. These models enable accurate AMR graph generation for Hungarian, a significant advancement for low-resource language processing.
- **Extensive evaluation of modeling techniques:** We conduct a thorough analysis of different model architectures, examining the impact of additional silver training data on AMR parsing performance. Our evaluation provides valuable insights into optimizing cross-lingual AMR parsers for non-English languages.
- **Evaluation of data augmentation strategies:** We leverage larger models to generate silver-standard AMR annotations for Hungarian texts, which are then incorporated into the training data for smaller models. This approach enhances the performance of smaller models, demonstrating an effective strategy for low-resource settings.

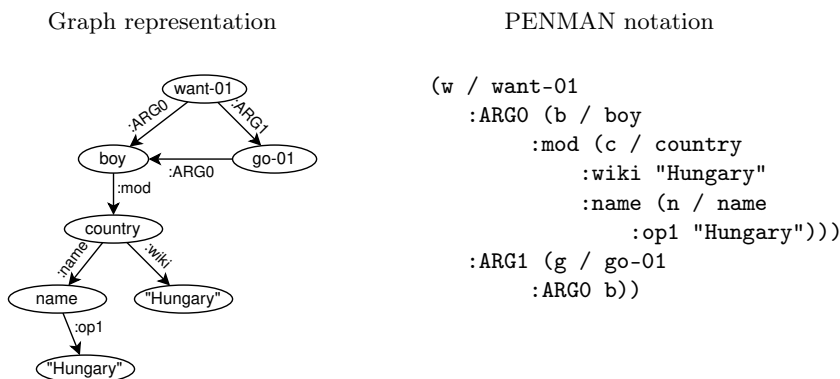
These efforts pave the way for extending AMR parsing to multiple languages and contribute toward the development of multilingual AMR parsers.

## 2 Abstract meaning representation

AMR is a semantic framework that represents the meaning of a sentence using a graph structure, abstracted from specific lexical and syntactic forms. It captures key relationships, such as who performs an action, what the action is, and

<sup>1</sup> Resources available at <https://github.com/botondbarta/HuAMR>

who or what it affects. In an AMR graph, nodes represent concepts like verbs, entities, and PropBank framesets (Kingsbury and Palmer, 2002), while edges denote semantic roles and relationships between these concepts. In cross-lingual setting the nodes and edges are always labeled in English regardless of the input language, and the entities are always lemmatized. AMR graphs are composed of logical triples that describe relationships between sources and targets. These triples provide an abstract, computationally readable representation of meaning, which can be expressed in both graph and PENMAN notation, allowing AMR to be applied in various natural language processing tasks like machine translation and information extraction. An example for the sentence *"The Hungarian boy wants to go"* is shown in Figure 1 using graph representation and PENMAN notation. The sentence is represented with variables and concepts like `want-01` and `go-01`, linked to the subject (*"the boy"*) and the action. Modifiers like the nationality "Hungarian" are captured using the `:mod` role, while the `:wiki` role links the modifier to its corresponding Wikipedia page, "Hungary". The official and more detailed description of AMR is available in the AMR specification<sup>2</sup>.



Depth-first serialization after the preprocessing steps described in Section 4.3.

```
( w / want-01 :ARGO ( b / boy :mod ( c / country :name ( n / name :op1
"Hungary" ) ) ) :ARG1 ( g / go-01 :ARGO b ) )
```

Fig. 1: AMR notations for the sentence "The Hungarian boy wants to go".

### 3 Related Work

Vanroy and Van de Cruys (2024) explored cross-lingual AMR parsing for English, Dutch, and Spanish using both multilingual and monolingual configura-

<sup>2</sup> <https://github.com/amrisi/amr-guidelines/blob/master/amr.md>

tions of BART-large. Their findings indicate that monolingual models consistently outperform multilingual ones, highlighting the limitations of multilingual approaches in this context.

Instruction fine-tuned language models show strong generalization to unseen tasks (Chung et al., 2024; Ouyang et al., 2022). However, standard parsing tasks like AMR are usually excluded from such instruction datasets and models. Lee et al. (2023) experimented with FLAN-T5 instruction fine-tuned models showing significant improvement in AMR parsing over earlier BART (Lewis et al., 2020) based models. Their work included a two step training approach: first a full fine-tuning and then a parameter efficient fine-tuning using Low Rank Adaptation (LoRA) (Hu et al., 2021).

Regan et al. (2024) created MASSIVE-AMR, a cross-lingual AMR dataset in the domain of question answering (QA) consisting of 1,685 utterances which are localized in 52 languages, Hungarian included. However the dataset contains some problems, as entities are inconsistently translated or left untranslated, and sometimes appear in inflected forms, making the parsing process difficult. It also lacks punctuation marks at the end of sentences, which are crucial as they often indicate parts of the AMR.

More recently, Kang et al. (2024) experimented with cross-lingual AMR parsing across 13 languages using a meta-learning model. They found that fine-tuning a model on a small number of examples from previously unseen languages failed to improve parser performance, underscoring the challenges of cross-lingual generalization in AMR parsing.

## 4 Data

### 4.1 Gold AMR graph - silver translation

To create Hungarian training data from the AMR 3.0 dataset we translated each sentence into Hungarian using DeepL<sup>3</sup>, keeping the AMRs unchanged. We refer to this translated dataset as AMR<sup>trans</sup>. We evaluated the translation quality with a reference-free evaluation metric, COMET (Rei et al., 2020). The COMET score of the translation is  $85.2 \pm 0.7$ .

### 4.2 Silver Training Data

Following the methodology of Damonte and Cohen (2018) we used the Europarl parallel corpus (Koehn, 2005) to generate silver training data. We used an English AMR parser<sup>4</sup> to produce silver quality AMR graphs for the English sentences. These AMR graphs then served as target representations for their corresponding Hungarian translations.

<sup>3</sup> <https://www.deepl.com/>

<sup>4</sup> [https://github.com/bjasob/amrlib-models/releases/parse\\_xfm\\_bart\\_large-v0\\_1\\_0](https://github.com/bjasob/amrlib-models/releases/parse_xfm_bart_large-v0_1_0)

We also experimented with creating another silver training dataset by first fine-tuning an adapter on the Llama-3.1-70B<sup>5</sup> model using the AMR<sup>trans</sup> dataset. The exact training process for this model is described in Section 5.1. With this adapted model, we generated AMR graphs for 50k samples drawn from the Hunsum-2 corpus (Barta et al., 2024) using the first sentence of each selected article’s lead. We named this dataset *HuAMR*.

To ensure that the generated silver AMR graphs are high-quality, we validated them against PropBank frame argument descriptions verifying that the argument structure of the frames in the graphs matched the arguments specified in PropBank. Many of the generated graphs included unnecessary ‘and’ operators. These operators were often introduced as the first node in the graph, but the model frequently failed to end the AMR properly, resulting in incomplete structures. This problem broke the logical flow of the AMR, making the graph inconsistent with the meaning of the sentence. Therefore, we validated the graphs to ensure that whenever an ‘and’ is used, it has at least two operands. With these validations we discarded 6189 instances from the HuAMR dataset. Of the remaining data, 3811 has been reserved for testing and the remaining 40k are available for training.

We compared the three different datasets by the 15 most frequent top nodes. These statistics are shown in Table 1. Notably, the ‘and’ concept appears as the most frequent node across all three datasets. Other frequently appearing nodes include ‘say-01’ and ‘contrast-01’ which are prevalent in AMR 3.0 and HuAMR, but different nodes like ‘obligate-01’ and ‘recommend-01’ appear more frequently in Europarl.

It is important to note that HuAMR consists exclusively of news data, whereas only a small fraction of AMR 3.0 is derived from news articles. This difference in data composition may explain the variations in node frequency, as news data typically includes a distinct set of linguistic structures and terminology compared to other types of content.

### 4.3 Preprocessing

Before training we remove the wiki tags from the AMR graphs. The graphs are then serialized using a depth-first approach, replacing newlines with spaces and compressing multiple spaces into a single one. To ensure consistent tokenization, a space is inserted before and after each parenthesis, as some tokenizers may otherwise interpret consecutive parentheses as a single token. An example serialization process is shown in Figure 1.

## 5 Experiments

### 5.1 Model for silver data generation

We fine-tuned a Llama-3.1-70B-Instruct model using LoRA under a 4-bit quantization setup to enable efficient training on limited computational resources as

<sup>5</sup> <https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct>

<b>AMR 3.0</b>	<b>#</b>	<b>HuAMR</b>	<b>#</b>	<b>Europarl</b>	<b>#</b>
and	7.0k	and	4.7k	and	57.7k
say-01	3.0k	say-01	4.6k	contrast-01	37.4k
contrast-01	3.0k	contrast-01	2.2k	say-01	34.5k
multi-sentence	1.7k	possible-01	1.9k	obligate-01	24.0k
possible-01	1.7k	state-01	0.7k	cause-01	23.5k
cause-01	1.6k	cause-01	0.7k	possible-01	19.4k
state-01	1.5k	have-concession	0.6k	recommend-01	15.8k
have-concession	0.9k	win-01	0.5k	multi-sentence	14.0k
think-01	0.9k	announce-01	0.4k	need-01	11.2k
person	0.7k	find-01	0.4k	like-02	10.0k
have-03	0.6k	report-01	0.3k	have-concession	8.8k
have-condition	0.6k	show-01	0.3k	believe-01	7.4k
date-entity	0.5k	write-01	0.3k	think-01	6.7k
know-01	0.5k	start-01	0.3k	important-01	5.4k
have-degree	0.4k	decide-01	0.2k	have-03	4.7k

Table 1: The top-15 most frequent AMR nodes in AMR 3.0, HuAMR and Europarl.

it reduces the memory and compute requirements while preserving much of the model’s capabilities, which makes it possible to train such a large model. LoRA allows us to fine-tune the model by introducing a small number of trainable parameters in low-rank matrices, significantly reducing computational overhead compared to full fine-tuning. In our setup, we applied LoRA to all projection layers of the transformer architecture. The LoRA specific hyperparameters included a rank of 8, an alpha value of 8 with a learning rate of  $5e-5$ . The task type was set to `CASUAL_LM`.

## 5.2 Fine-tuning with additional silver data

We explored the effect of varying the amount of silver data on model performance by training multiple models. Our experiments were carried out using two distinct architectures: the mT5-large model (Xue et al., 2021) that follows a sequence-to-sequence approach and the Llama-3.2-1B-Instruct model, reflecting the recent trend toward decoder-only architectures. Both models are within the 1.2-1.3 billion parameter range.

For training, we used subsets from the training set of both the HuAMR corpus and the silver Europarl corpus, testing different amounts of silver data in each experiment. From the HuAMR corpus, we experimented with four data sizes: 10k, 20k, 30k, and 40k samples. These subsets were combined with the AMR<sup>trans</sup> dataset to assess how increasing portions of silver data affected performance.

We also trained models using data from the silver Europarl corpus with 40k, 160k and 520k additional sentences. This allowed us to investigate how the

models respond to different amounts of data from a parallel corpus created by a high-quality parser.

## 6 Results

### 6.1 Silver data

We evaluated the performance of our models on the test set of AMR<sup>trans</sup> and HuAMR using Smatch scores (Cai and Knight, 2013). The Smatch score measures the semantic similarity between two AMR graphs by calculating the overlap of matched triples. For these calculations we used the Smatchpp (Opitz, 2023) library. The scores are shown in Table 2.

Model	Additional Data		Smatch F <sub>1</sub>	
	HuAMR	Europarl	AMR <sup>trans</sup>	HuAMR
mT5 Large				
mT5 Large	-	-	<b>72.90</b>	69.32
mT5 Large	10k	-	72.28	73.65
mT5 Large	20k	-	72.59	75.99
mT5 Large	30k	-	71.98	75.26
mT5 Large	40k	-	72.22	<b>76.11</b>
Llama 3.2 1B				
Llama 3.2 1B	-	-	67.43	54.44
Llama 3.2 1B	10k	-	68.35	67.65
Llama 3.2 1B	20k	-	68.41	70.10
Llama 3.2 1B	30k	-	69.44	71.98
Llama 3.2 1B	40k	-	<b>69.59</b>	<b>73.03</b>
Llama 3.2 1B	-	40k	68.67	60.05
Llama 3.2 1B	-	160k	<b>70.70</b>	<b>62.92</b>
Llama 3.2 1B	-	520k	66.16	59.08

Table 2: Smatch F<sub>1</sub> scores for Llama 3.2 1B and mT5 Large models with increasing training data sizes from silver datasets HuAMR and Europarl. A dash (‘-’) indicates no additional data. The best score for each model is **bolded**, and the overall best score is both **bolded and underlined**.

Table 2 reveals several key trends:

- **Performance Superiority of mT5 Large:** Across all configurations of additional silver data from HuAMR and Europarl, mT5 Large consistently outperforms Llama 3.2 1B in Smatch scores.

- **Saturation on AMR<sup>trans</sup> Evaluation:** Both models exhibit saturated performance on the AMR<sup>trans</sup> dataset; additional silver data yields minimal improvements in Smatch scores. This suggests that the models may have already reached their maximum potential for this particular dataset, and additional data may not provide sufficient new information to enhance their parsing accuracy.
- **Llama 3.2 1B’s Limitations with Europarl Data:** Incorporating up to 520k additional Europarl training instances benefits Llama 3.2 1B, but this model does not achieve parity with mT5 Large’s performance.

Building on the observation of performance saturation on the AMR<sup>trans</sup> test set despite increasing amounts of silver data, we conducted a control experiment to investigate whether the mT5 Large model is already saturated on the training set of AMR<sup>trans</sup>. We trained the models using varying amounts of both AMR<sup>trans</sup> and HuAMR training data while evaluating performance on the AMR<sup>trans</sup> test set. The results, shown in Figure 2, reveal that introducing a small amount of HuAMR data (100–2000 data points) provides a slight positive shift in performance. However, when scaling the AMR<sup>trans</sup> data from 4000 to 40k data points, models trained on AMR<sup>trans</sup> data obtain a consistent performance advantage of roughly 5% in Smatch F<sub>1</sub> scores over those augmented with HuAMR. These trends confirm that the models are not overfitted or saturated when trained solely on 40k AMR<sup>trans</sup> data points.

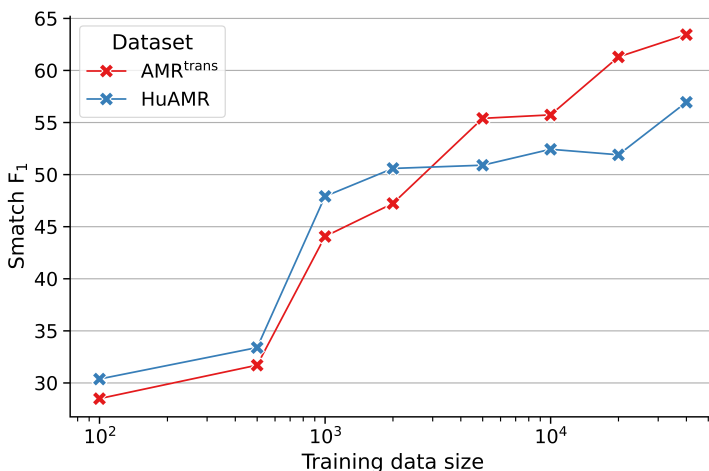


Fig. 2: Smatch F<sub>1</sub> score as a function of the training data size.



## 7 Conclusion

This study presents the first Abstract Meaning Representation (AMR) resource for Hungarian by translating the gold-standard AMR 3.0 dataset (Knight et al., 2020). Utilizing this resource, a suite of AMR parsers for Hungarian texts was developed, leveraging models ranging from 1B to 70B parameters. The models demonstrate accurate AMR graph generation capabilities for Hungarian, significantly advancing semantic parsing for this underrepresented language.

We conducted an extensive evaluation of modeling techniques to examine the impact of additional silver training data and model size on AMR parsing performance. The results indicate that the mT5 Large model consistently outperforms the Llama 3.2 1B model across all configurations of additional silver data from HuAMR and Europarl. Despite substantial additions of silver data, both models showed limited improvements on the AMR<sup>trans</sup> test set.

Our control experiment indicates that increasing the quantity of gold-standard data leads to continued improvements in Smatch  $F_1$  scores. Models trained on 40k AMR<sup>trans</sup> data points achieved approximately 5% higher scores than those trained on the silver standard HuAMR.

These findings suggest that the models did not overfit nor saturate when trained solely on the full AMR<sup>trans</sup> dataset. The limited impact of additional silver data implies that the performance saturation is attributable to the lower quality of silver-standard annotations. The results highlight the critical role of high-quality training data and adequate model capacity in achieving optimal performance in semantic parsing tasks.

## Acknowledgements

This study was supported by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory, Hungary.

## References

- Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., Schneider, N.: Abstract meaning representation for sembanking. In: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse. pp. 178–186 (2013)
- Barta, B., Lakatos, D., Nagy, A., Nyist, M.K., Ács, J.: From news to summaries: Building a Hungarian corpus for extractive and abstractive summarization. In: Calzolari, N., Kan, M.Y., Hoste, V., Lenci, A., Sakti, S., Xue, N. (eds.) Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024). pp. 7503–7509. ELRA and ICCL, Torino, Italia (May 2024), <https://aclanthology.org/2024.lrec-main.662>

- Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* pp. 610–623 (2021)
- Blloshmi, R., Tripodi, R., Navigli, R.: Xl-amr: Enabling cross-lingual amr parsing with transfer learning techniques. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. pp. 2487–2500 (2020)
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Nee-lakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33, 1877–1901 (2020)
- Cai, S., Knight, K.: Smatch: an evaluation metric for semantic feature structures. In: Schuetze, H., Fung, P., Poesio, M. (eds.) *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. pp. 748–752. Association for Computational Linguistics, Sofia, Bulgaria (Aug 2013), <https://aclanthology.org/P13-2131>
- Cao, S., Zhang, L., Narayan, S., Cheung, J.C.K., Bansal, M.: Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. *Transactions of the Association for Computational Linguistics* 10, 417–433 (2022)
- Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al.: Scaling instruction-finetuned language models. *Journal of Machine Learning Research* 25(70), 1–53 (2024)
- Damonte, M., Cohen, S.B.: Cross-lingual Abstract Meaning Representation parsing. In: Walker, M., Ji, H., Stent, A. (eds.) *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pp. 1146–1155. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018), <https://aclanthology.org/N18-1104>
- Dou, Z.Y., Liu, P., Neubig, G., Faltings, B.: Gsum: A general framework for guided neural abstractive summarization. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. pp. 4831–4844 (2022)
- Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021)
- Ji, Z., Lee, N., Fries, J., Yu, T., Sachan, M.: A survey on hallucination in natural language generation. *arXiv preprint arXiv:2301.12017* (2023)
- Kang, J., Coavoux, M., Lopez, C., Schwab, D.: Should cross-lingual amr parsing go meta? an empirical assessment of meta-learning and joint learning amr parsing. *arXiv preprint arXiv:2410.03357* (2024)
- Kingsbury, P.R., Palmer, M.: From treebank to propbank. In: *LREC*. pp. 1989–1993 (2002)
- Knight, K., Badarau, B., Baranescu, L., Bonial, C., Bardocz, M., Griffitt, K., Hermjakob, U., Marcu, D., Palmer, M., O’Gorman, T., Schneider, N.:

- Abstract Meaning Representation (AMR) Annotation Release 3.0 (2020), <https://catalog.ldc.upenn.edu/LDC2020T02>
- Koehn, P.: Europarl: A parallel corpus for statistical machine translation. In: Proceedings of Machine Translation Summit X: Papers. pp. 79–86. Phuket, Thailand (Sep 13-15 2005), <https://aclanthology.org/2005.mtsummit-papers.11>
- Lee, Y.S., Astudillo, R.F., Florian, R., Naseem, T., Roukos, S.: Amr parsing with instruction fine-tuned pre-trained language models. arXiv preprint arXiv:2304.12272 (2023)
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7871–7880. Association for Computational Linguistics, Online (Jul 2020), <https://aclanthology.org/2020.acl-main.703>
- Maynez, J., Narayan, S., Bohnet, B., McDonald, R.: On faithfulness and factuality in abstractive summarization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 1906–1919 (2020)
- OpenAI: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
- Opitz, J.: SMATCH++: Standardized and extended evaluation of semantic graphs. In: Vlachos, A., Augenstein, I. (eds.) Findings of the Association for Computational Linguistics: EACL 2023. pp. 1595–1607. Association for Computational Linguistics, Dubrovnik, Croatia (May 2023), <https://aclanthology.org/2023.findings-eacl.118>
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in neural information processing systems 35, 27730–27744 (2022)
- Regan, M., Wein, S., Baker, G., Monti, E.: MASSIVE multilingual Abstract Meaning Representation: A dataset and baselines for hallucination detection. In: Bollegala, D., Shwartz, V. (eds.) Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (\*SEM 2024). pp. 1–17. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024), <https://aclanthology.org/2024.starsem-1.1/>
- Rei, R., Stewart, C., Farinha, A.C., Lavie, A.: COMET: A neural framework for MT evaluation. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 2685–2702. Association for Computational Linguistics, Online (Nov 2020), <https://aclanthology.org/2020.emnlp-main.213>
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
- Vanroy, B., Van de Cruys, T.: Less is enough: Less-resourced multilingual amr parsing. In: Proceedings of the 20th Joint ACL-ISO Workshop on Interoperable Semantic Annotation@ LREC-COLING 2024. pp. 82–92 (2024)

- Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., Raffel, C.: mT5: A massively multilingual pre-trained text-to-text transformer. In: Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y. (eds.) Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 483–498. Association for Computational Linguistics, Online (Jun 2021), <https://aclanthology.org/2021.naacl-main.41>
- Zhu, J., Ji, D., Liu, H.: Incorporating graph information in transformer-based amr parsing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. pp. 6981–6992 (2020)

# HunSimpleNews: Az első autentikus magyar nyelvű szövegekből álló szövegegyszerűsítési korpusz

Prótár Noémi<sup>1</sup>, Nemeskey Dávid Márk<sup>1,2</sup>

<sup>1</sup>Eötvös Loránd Tudományegyetem

<sup>2</sup>Digitális Örökség Nemzeti Laboratórium

protarnoemi@student.elte.hu

nemeskey.david@btk.elte.hu

**Kivonat** A tanulmány az első autentikus magyar nyelvű szövegekből álló szövegegyszerűsítési korpuszt mutatja be. A korpusz 2832 darab könnyen érthető (egyszerűsített) és standard nyelvi szövegpárból áll, melyet a PannonRTV honlapjáról gyűjtöttünk. A tanulmányban bemutatjuk a korpusz összeállításának és minőségellenőrzésének folyamatát, kitérünk a korpusz két domainje közötti kvantitatív különbségekre és hasonlóságokra, végül a korpusz alapján felmérjük az általunk elérhető, magyarul tudó nagy nyelvmodellek szövegegyszerűsítési képességét in-context tanítási környezetben. Cikkünkben kimutatjuk, hogy az egyszerűsített szövegek és a standard nyelvi szövegek között lényeges különbségek vannak, mind az általuk használt szavak varianciájában, mind pedig a mondatok hosszában. Eredményeinkből láthatóvá válik, hogy a korpusz használata one-shot és few-shot tanítási környezetben is javítja a modellek egyszerűsített kimenetét a zero-shot eredményeinkhez képest.

**Kulcsszavak:** szövegegyszerűsítés, korpusz, könnyen érthető kommunikáció, párhuzamos korpusz

## 1. Bevezetés

A szövegegyszerűsítés<sup>1</sup> (text simplification) általános célja az, hogy különböző, standard nyelven íródott szövegeket érthetővé tegyen nem prototipikus befogadók számára. Prototipikus befogadók alatt a cikkben a felnőtt, ép, anyanyelvi beszélőket értjük, így tehát nem prototipikus befogadók például a gyerekek, a nyelvtanulók, funkcionális analfabéták, illetve olyan testi vagy értelmi fogyatékos személyek, akiknek fogyatékosága hátráltatja őket egy adott, standard nyelvű diskurzus megértésében (demensek, afáziások, nyelvelsajátítás előtti halláskárosodásuk volt, stb.) (Maaß, 2020). Az tehát, hogy szövegeket bizonyos befogadók

<sup>1</sup> A gyógypedagógiai szakirodalom az egyszerűsített szövegeket általában magyarul a *könnyen érthető szöveg* vagy *könnyen érthető kommunikáció* terminussal illeti. Mivel azonban az informatikai hagyomány ezen szövegek előállítására *text simplification*-ként hivatkozik, így ez a tanulmány is konzekvensen a *szövegegyszerűsítés* terminust használja, az egyértelműség és a kurrens diskurzusba való beilleszkedés miatt.

számára egyszerűsítünk, nagyban megkönnyíti, vagy alapjában lehetővé teszi azt, hogy mindennapi diskurzusokban részt vehessenek, valamint azt is, hogy a megfelelő információk eljussanak hozzájuk, és azok önállóan feldolgozhatóvá váljanak számukra.

Kiemelendő, hogy a szövegegyszerűsítés nem ekvivalens sem a szövegösszefoglalással, sem pedig a különböző szövegek közérthetővé tételével. Ugyan az egyszerűsített szövegekben lehetnek olyan részletek, amelyeket összefoglalunk a könnyebb megérthetőség érdekében, a szövegösszefoglalásnak a célja a szövegek hosszának csökkentése a felesleges információk elhagyásával (Shardlow (2014), Alva-Manchego és mtsai (2020)). A szövegegyszerűsítés ezzel szemben a szöveg bonyolultságának redukciójára törekszik: amennyiben ez azt igényli, hogy bizonyos fogalmakat megmagyarázzon, akkor ezzel gyakran él – így tehát az egyszerűsített szövegek akár hosszabbak is lehetnek, mint a forrásszövegeik. A közérthető szövegek pedig prototipikus befogadók számára tesznek érthetővé olyan szövegeket, amelyek szakspecifikusak – a könnyen érthető, azaz jelen tanulmányban *egyszerűsített*ként jellemzett szövegek ezzel szemben standard nyelvi, hétköznapi témákat érintő szövegeket tesznek érthetővé nem prototipikus befogadók számára.

A legtöbb szöveg leegyszerűsítése manuális módszerekkel történik – ez azonban időigényes és nagy humán erőforrást kíván. Így a szövegegyszerűsítés témaköre a nyelvtechnológiai jellegű kutatásokban is előtérbe került az utóbbi években. Jelen cikk az első olyan, magyar nyelvű hírszövegpárokból álló szövegegyszerűsítési korpuszt mutatja be, amely autentikus, tehát nem fordított, hanem eredetileg is magyarul írt adatokból áll. A korpusz kutatási célra szabadon elérhető a Hugging Face Hubon<sup>2</sup>. Cikkünkben először áttekintjük a szövegegyszerűsítés eddigi és jelenlegi állását (2), ezután bemutatjuk a korpusz összeállítását (3), majd ismertetjük a korpuszon végzett, a két szövegdomaint összehasonlító vizsgálatok eredményeit (4), végül bemutatjuk a korpusz használatával, one- és few-shot tanítási környezetben végzett vizsgálataink eredményét (5). A cikket egy rövid összefoglalás zárja (6).

## 2. Kapcsolódó irodalom

### 2.1. A szövegegyszerűsítés megközelítései

A nyelvtechnológiai jellegű, szövegegyszerűsítéssel foglalkozó kutatások legnagyobb része teljes szövegek egyszerűsítése helyett gyakran csak mondatok egyszerűsítésével foglalkozik. Ennek főképp az az oka, hogy az elérhető, nagyobb mennyiségű adatok száma minden nyelven kevés, illetve az egyszerűsített szövegek gyakran nem standard nyelvi szövegekből jöttek létre, hanem azokkal párhuzamosan, vagy tőlük függetlenül – így a szövegek nem felelnek meg egymásnak teljesen. Emellett megemlítendő, hogy a szövegeket általában automatikusan párosítják össze – ez mondatok esetében könnyebb, hiszen mondatok automatikus párosítására, hasonlóságuk mérésére viszonylag régóta vannak elérhető, jól

<sup>2</sup> <https://huggingface.co/datasets/ELTE-DH/HunSimpleNews>

használható rendszerek, (hosszabb) szövegek esetében azonban ez nem minden esetben magától értetődő.

Így tehát már az első modern, neurális hálót használó kutatások is a mondatok egyszerűsítését tartották céljuknak angol nyelven (Nisioi és mtsai, 2017). Ezt több nyelvre követték hasonló korpuszok és próbálkozások: a MultiSimV2 korpuszában (Ryan és mtsai, 2023) például összesen tizenhárom nyelv szerepel (arab, baszk, brazil portugál, dán, angol, francia, német, olasz, japán, orosz, szlovén, spanyol és urdu), melyek nagyrészt mondat szintű egyszerűsítésekből állnak. Kiemelendő, hogy több olyan nyelv is van, amelyre csak mondat szintű egyszerűsítések érhetőek el: ilyen például az arab (Khallaf és mtsai, 2022), a dán (Klerke és Søgaard, 2012), a szlovén (Gorenc és Robnik-Šikonja, 2022) és az urdu (Qasmi és mtsai, 2020). A dokumentumalapú párhuzamos korpuszok esetében pedig sokszor problémát okoz a korpusz nem nyilvános mivolta, vagy az adathalmaz kis mérete: az olasz Teacher és Terence korpuszok például mindössze 18 és 32 szövegpárból állnak (Brunato és mtsai, 2015).

Magyar nyelvre eddig egy párhuzamos korpusz jött létre (Prótár és Nemeskey, 2023), ez azonban egy fordított korpusz, melynek alapját az angol Simple English Wikipediából automatikus eszközökkel összeállított anyag adta (Zhu és mtsai, 2010). Ez a korpusz, éppúgy, mint a legtöbb a nemzetközi trendekben, szintén mondatpárokból áll. A szakirodalom azonban gyakran nem vet számot azzal, hogy a szövegek egyszerűsítése nem oldódik meg a mondatok izolált egyszerűsítésével: ugyan több cikk is foglalkozik azzal, hogy milyen műveleteket kell elvégezni egy mondaton belül ahhoz, hogy azok megfelelően egyszerűek legyenek (pl. Alva-Manchego és mtsai (2017)), és habár valóban az (elemi) mondatok a megnyilatkozások feldolgozásának alapvető egységei (Tátrai, 2017), ezek a metódusok azonban nem számolnak a szövegkonstruálás mondatstruktúráján felüli alapvető elemeivel, például az információk sorrendjének a meghatározásával, azok elosztásával, a kontextualizálás folyamatával stb. Nem számol emellett azzal sem, hogy a szövegértelmezés nem kompozicionális jellegű, azaz nem egyenlő csupán a mondatok értelmének egymás után tett vagy összeadott halmazával (vö. Tolcsvai Nagy (2006)) – így tehát a mondategyszerűsítés nem tehető egyenlővé a szövegegyszerűsítéssel, és szövegek egyszerűsítése nem megvalósítható a szöveg mondatainak egymás utáni leegyszerűsítésével. Emellett nem hagyhatjuk figyelmen kívül azt sem, hogy az emberi szövegegyszerűsítés sem mondat szinten, a mondatok egymás után való egyszerűsítésével, hanem szövegszinten zajlik, így tehát hasznosnak tűnik az automatikus egyszerűsítési kísérleteket is a szövegszintű egyszerűsítés felé közelíteni. Jelen tanulmány a hosszabb, kohezívebb szövegek egyszerűsítésének irányába kívánja tehát megtenni az első lépéseket magyar nyelven.

## 2.2. Szövegegyszerűsítés nagy nyelvi modellek segítségével

A korai automatikus szövegegyszerűsítési módszertanok főleg lexikai, azaz szó szintű egyszerűsítéssel foglalkoztak (pl. Bott és mtsai (2012), Keskisärkkä (2012)) – azonban a transformer architektúra (Vaswani és mtsai, 2017) megjelenése a szövegegyszerűsítésre is hatással volt: az első ezen alapuló mondategyszerűsítő

modellt Nisioi és mtsai (2017) mutatták be. A nagy nyelvi modellek megjelenése további előrelépést jelentett a szövegegyszerűsítésben: Feng és mtsai (2023) kutatásai szerint a nagy nyelvi modellek teljesítménye meghaladja a mondat-egyszerűsítés kurrens modelljeinek legjobbjait, sőt, a teljesítményük az emberi egyszerűsítéshez is mérhető. Ezzel ellentétben Padovani és mtsai (2024) eredményei alapján egy megfelelően finomhangolt BERT-modell (Devlin és mtsai, 2019) jelentősen jobban teljesít az automatikus metrikák tekintetében, mint az általuk használt ChatGPT-3.5 ((Brown és mtsai, 2020) alapján). Jelen cikk szintén ehhez az emergens kutatási módszertanhoz kíván kapcsolódni, azonban nem a mondat-, hanem a szövegegyszerűsítés terén. Nem csupán a szöveghosszúság és az ezzel járó komplexitás oldaláról kínál újdonságot, hanem a magyar nyelvű szövegegyszerűsítéssel is: ugyan többnyelvű szövegegyszerűsítésre készültek már benchmarkok (Ryan és mtsai, 2023), ezek a magyar nyelvet nem tartalmazták.

### 3. A korpusz összeállítása

A korpusz anyaga a PannonRTV szabadkai híroldal cikkeiből származik. A kérésünkre a portál üzemeltetői valamint a cikkek írója tájékoztatott minket, hogy a hírportál könnyen érthető szövegeit egy személy írja a könnyen érthető iránylevek alapján (ld. pl. (Europe, 2017)).<sup>3</sup> Ezen a híroldalon kívül nincs tudomásunk olyan magyar nyelvű forrásról, amely megfelelő mennyiségben és minőségben állítana elő párhuzamos, vagy párhuzamosítható, műfaji szempontból viszonylag egységes standard nyelvi–könnyen érthető szövegpárokat magyar nyelven. Magyarul ezen kívül nagyon kevés, és nagyon különböző műfajú könnyen érthető szöveg jelent meg, amelyeknek standard nyelvi párja is van: ilyen például a Nógrádi Gergely és Bodonyi Panni által írt János vitéz-átdolgozás, mely Petőfi művének első három énekét ülteti át egyszerűen érthető és könnyen érthető formába. (Petőfi és mtsai, 2022)

A korpusz összeállításához először a Scrapy (Kouzis-Loukas, 2016) segítségével legyűjtöttük az oldalon megjelentetett cikkeket, az első egyszerűsített cikk megjelenésének időpontjáig. Mivel a híroldalon megjelenő egyszerűsített cikkek nincsenek semmilyen módon összekötve a standard nyelvi párjukkal, így az összes standard nyelvi hírszöveget le kellett menteni, hogy azokat összepárosíthassuk a könnyen érthető szövegekkel. A legyűjtések során összesen 4364 darab egyszerűsített hírszöveget, és több, mint 66 500 standard nyelvi szöveget mentettünk le. A gyűjtés két fázisban, 2023 novemberében és 2024 augusztusában zajlott, utóbbi alkalommal csak azokat a szövegeket mentettük le, amelyeket korábban nem.

A HTML-fájlok lementése után a szövegekből kinyertük a releváns adatokat, melyek a következők voltak: a cikk címe, leadje, szövege, a hozzátartozó címkék; a

<sup>3</sup> A korlátozott szerzői kör természetesen bizonyos mértékig befolyásolja a korpusz vizsgálhatóságát és felhasználhatóságát, azonban mivel semmilyen más, még csak közelítő méretű szöveganyag sem elérhető magyar nyelven könnyen érthető szövegekből, így ezeket használtuk fel.



cikk URL-je, közlésének dátuma, valamint a cikk kategóriája. Az utóbbi az oldalon megjelenő tizenkét magyar nyelvű kategóriát tartalmazta (politika, világ, régió, gazdaság, társadalom, természetvédelem, önkormányzatok, kultúra, kékfény, sport, színes, KÉK-hírek), amelybe a cikkek megjelenésükkor a szerkesztők vagy az újságírók által besorolásra kerülnek. Az adatokat egy JSONL-struktúrában tároltuk el.

A korábban említett standard és egyszerűsített szövegek közötti hivatkozások hiánya miatt a szövegek párosítására szükség volt, azonban ez a nagyszámú szövegmennyiség miatt nem valósulhatott meg manuálisan. A szövegek hasonlóságát Doc2Vec (Řehůřek és Sojka, 2010) és Sentence-BERT (Reimers és Gurevych, 2019) beágyazásokkal vizsgáltuk, egy olyan további kikötést téve, hogy a standard nyelvi szöveg az egyszerűsített szöveghez képest 0–3 nappal előbb kellett, hogy keletkezzen. A szövegbeágyazásokat lemmatizált és nem lemmatizált adatokon is elvégeztük, majd ezeket összevetettük, ahol pedig a két eljárás különböző eredményt adott, manuálisan döntöttünk a helyes párosításról. Végül a kétfajta beágyazási módszerrel keletkezett eredményeinket is összevetettük, és ugyanígy jártunk el.

Így összesen 3629 lehetséges párt kaptunk.

Az elvégzett párosításokat az ELTE Digitális Bölcsészet Tanszékének négy annotátora ellenőrizte manuálisan.

Az annotáció alatt három csoportba sorolhatták a lehetséges párokat:

- Az összepárosított két szöveg **valóban pár**: a két szöveg ugyanarról szól, és az egyszerűsített szöveg nem tartalmaz olyan kiegészítő információkat, amelyek nincsenek benne a standard nyelvi szövegben. Ebből összesen **2833** darab volt a korpuszban.
- Az összepárosított két szöveg **nem pár**, azaz a két szöveg nem ugyanarról szól. Ebből összesen **175** darab volt a korpuszban.<sup>4</sup>
- Az összepárosított két szöveg **ugyanarról szól, de az egyszerűsített szöveg tartalmaz olyan kiegészítő információkat, amelyek nincsenek benne a standard nyelvi szövegben**. Ebből összesen **621** volt a korpuszban.

Jelen cikkben a párként kategorizált cikkpárokkal dolgoztunk. A plusz információt tartalmazó cikkek esetleges javítása további munka tárgyát képezheti.

#### 4. Korlátok és lehetőségek – tanulságok a korpusz szerkezetének vizsgálatáról

A korpuszon a few-shot tanítás előtt elvégeztünk néhány, a korpusz szerkezetét érintő vizsgálatot. A vizsgálatokkal célunk a korpusz használati lehetőségeinek és

<sup>4</sup> A nem párként elkönyvelt szövegek egyértelműen nem voltak párok – ez például a következő, helytelenül összepárosított, de az annotátorok által nem párnak ítélt szövegek címeiből is látszik:

Tisztújító közgyűlést tart májusban a VMSZ – 1 dinárral drágább az üzemanyag; Szerbia legnagyobb vidámparkja várja majd a Palicson majálisozókat – Elektromos autókat fognak gyártani Szerbiában

korlátainak bemutatása volt, valamint az egyszerűsített és standard nyelvi szövegek jellemzőinek összevetése. A korpusz vizsgálata előtt megemlítendő, hogy a nyelvi anyag egy forrásból származik, így a magyar nyelvű egyszerűsített szövegek általános tulajdonságaival kapcsolatban csak nagyon óvatos megállapítások tehetők. Nem lehet figyelmen kívül hagyni azt sem, hogy mindkét korpuszkomponens a hírszövegek műfajába tartozik – ezáltal a nyelvi anyaggal kapcsolatos konklúziók ezen műfajhoz tartozó szövegekre lesznek leginkább alkalmazhatók. A fent felsorolt nehézségek ellenére a szövegekorpusszal kapcsolatos vizsgálatokat elengedhetetlennek tartjuk: mivel a magyar nyelvű szövegegyszerűsítéssel kapcsolatban kevés vizsgálat készült, valamint kevés egyszerűsített szöveganyag lelhető fel magyarul, így szükségesnek tartjuk azt, hogy a vizsgálatok reflektáljanak a már meglévő szöveganyagra, nyelvhasználatra, és ne kizárólag a priori megközelítéssel viszonyuljanak a szövegegyszerűsítés problematikájához.

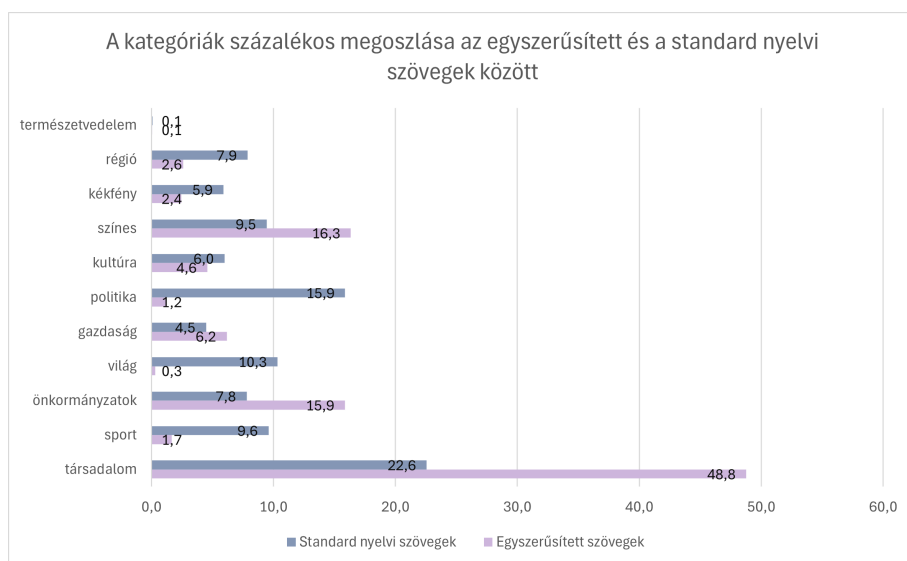
Ahogy az korábban említésre került, a cikkek a cikkírók vagy az oldal szerkesztői által kategóriákba lettek rendezve. A párosítás után az egyszerűsített cikkek „megörökölték” az adatszettben a standard nyelvi párjuk kategóriáját,<sup>5</sup> így vizsgálhatóvá vált az, hogy azok milyen kategóriájú cikkekből lettek előállítva, valamint az is, hogy az egyszerűsített szövegek kategóriái mennyiben képezik le az összes hírszöveg kategóriájának megoszlását, azaz hogy vannak-e preferált vagy elhanyagolt témakörök az egyszerűsített hírszövegekben.

Az 1. ábra ezt a megoszlást mutatja be, azaz az egyszerűsített szövegek és a standard nyelvi szövegek kategóriáinak százalékos megoszlását a saját korpuszon belül. A standard nyelvi szövegek esetében az összes, az egyszerűsített cikkek keletkezésének időszakában közzétett cikket használtuk a százalékos adatok meghatározásához.

Jól látható, hogy a megoszlások erősen különböznek a két csoport esetében: ugyan mindkét csoport esetében a *társadalom* kategória volt a leggyakrabban használt, azonban az egyszerűsített szövegek esetében ebbe a kategóriába a szövegeknek közel a fele sorolódott, míg a standard nyelvi szövegek esetében csupán körülbelül a negyedét tették ki az ide kategorizált elemek. A többi szöveg esetében pedig még drasztikusabb a különbség: a *politika*, *sport* vagy a *világ* kategóriák esetében erősen alulreprezentáltak az egyszerűsített szövegek a standard nyelvekhez képest, míg az *önkormányzatok* és *színes* kategóriák az egyszerűsített szövegek korpuszában kaptak nagyobb szerepet. Ebből látható, hogy az egyszerűsített szövegek írói alapvetően más, valószínűleg a befogadóhoz közelebb álló, számára egyszerűbben beazonosítható témákat részesítenek előnyben.

A korpuszméreték és a különböző lemmák számossága alapján az 1. táblázatból jól látszik, hogy az egyszerűsített korpusz jelentősen kisebb szótárral operál: míg a nagysága körülbelül a 80%-a a standard nyelvinek, a különböző lemmák száma nagyjából feleannyi. Úgy tűnik tehát, hogy az egyszerűsített szövegek a repetitívásra törekednek a változatosság helyett: a könnyebb érthetőség miatt a szinonimák helyett ugyanazokat a szavakat használják ugyanazon dolog

<sup>5</sup> Az öröklésre azért volt szükség, mert az egyszerűsített szövegeknek külön egységes kategóriájuk volt az oldalon belül; feltételeztük azonban, hogy mivel a két szöveg alaptémája megegyezik, így a kategóriabesorolásuk is megegyezne.



1. ábra: A kategóriák megoszlásának százalékos összehasonlítása az egyszerűsített szövegek között és az ugyanebben az időszakban keletkezett összes többi, standard nyelvi hírszöveg esetében.

	Standard	Egyszerűsített
Összes lemma	574 808	487 728
Lemmatípusok száma	35 972	18 778
Hapaxok száma	18 005	8227
Egy lemma átlagos előfordulása	15,98	25,98
Egy lemma átlagos előfordulása hapaxok nélkül	30,99	45,45

1. táblázat. A két korpuszdomain lemmatizált adatainak megoszlása.

vagy fogalom megnevezésére. A nagy mennyiségű hapax, illetve a különböző tulajdonnevek előfordulásai miatt az átlagos előfordulások száma csupán a fenti adatokkal együtt értelmezhető, önmagában nem – azonban ezek az adatok is arra mutatnak, hogy az egyszerűsített korpuszban a megnyilatkozó sokkal gyakrabban használja ugyanazokat a szavakat, mint a standard nyelvi korpuszban.

A 2. táblázatból kiolvasható, hogy a két korpusz nemcsak lemmaszinten, de mondatszinten is különbözik: a standard nyelvi hírszövegek esetében a mondatok általában több szóból állnak (tehát hosszabbak). A leadek mondatai tendenciózusan rövidebbnek tűnnek mindkét korpuszban, mint a szövegtörzs mondatai – ennek oka valószínűleg az, hogy a lead funkciója a hírszöveg összefoglalása, nem pedig az egyes aspektusok kifejtése. A táblázat adataiból leszűrhető, hogy az egyszerűsített szövegek mondatai alapvetően rövidebbek. Mivel a mondatok száma is kevesebb, így feltehető az is, hogy bizonyos információkat vagy informá-

	Szavak Mondatok Átlagos mondat hossz		
Standard nyelvi (csak szövegtest)	498 923	29 308	17,02
Egyszerűsített (csak szövegtest)	444 963	37 619	11,83
Standard nyelvi (szöveg + lead)	557 512	36 066	15,46
Egyszerűsített (szöveg + lead)	472 912	43 287	10,93

2. táblázat. A mondatok és a bennük lévő szavak aránya a két korpuszdomainben.

ciórészleteket elhagy az egyszerűsített szöveg. Az, hogy ez mikor történik meg, és a standard nyelvi szövegekhez képest milyen információk kerülnek elhagyásra vagy megmagyarázásra a szövegben, további, kvalitatív vizsgálatok tárgyát képezi.

## 5. A korpusz segítségével elért eredmények

A korpusz segítségével elsődlegesen négy olyan magyarul tudó utasítás-finomhangolt modellt teszteltünk, amelyek viszonylag könnyen hozzáférhetők: a zárt súlyú GPT-4o minit (Achiam és mtsai, 2024) API-n keresztül<sup>6</sup>, a Llama 3.3 70 milliárd paraméteres, 4 bitre kvantált és a Llama 3.1 8 milliárd paraméteres változatait (Dubey és mtsai, 2024), valamint a Qwen 2.5 7 milliárd paraméteres változatát (Yang és mtsai, 2024; Qwen, 2024)) pedig saját GPU-n futtatva. A nyílt modellek elfutnak egy A100 40GB-s kártyán (a Llama 70B kettőn) 20-30 ezres kontextushosszal, a GPT-4o mini pedig megfizethető árazású. A négy fő modell mellett összehasonlításképpen kimértünk egy kis modellt, a Llama 3.2 3 milliárdos változatát, illetve a GPT-4o mini két alternatíváját, a GPT-4o-t és a Claude 3.5 Haikut (Anthropic, 2024) is. Ezek eredményei a korábbi Llama 3.1 70B-s méréseinkkel együtt a B függelékben találhatók.

A modelleket a SARI- (Xu és mtsai, 2016) és a BLEU-pont (Papineni és mtsai, 2002) alapján vetettük össze. A SARI pontszámot a Huggingface *evaluate* könyvtárával számoltattuk ki (von Werra és mtsai, 2020), míg a BLEU-pontszámokat a *sacreBLEU* könyvtár segítségével (Post, 2018). Mind a BLEU-, mind pedig a SARI-pontszám általánosan használt az egyszerűsített szövegek megfelelőségének mérésére: az előbbi n-gram-egyezéseket mér, míg az utóbbi a standard nyelvi szöveget és a referenciaszöveget veti össze a generált szöveggel, annak fényében, hogy a generált szöveg mennyire jól:

- ad hozzá a standard nyelvi szöveghez releváns információkat,
- törli ki a standard nyelvi szövegből a nem fontos elemeket,
- hagyja meg azt a szöveget érintetlenül, amit érintetlenül kell hagynia.

Mindkét pontszám esetében 0 és 100 közötti értéket vehet fel az egyszerűsített szöveg minősítése, ahol a 100 jelenti azt, hogy a referenciaszöveg és a generált szöveg teljes egészében megegyezik. Mindkét pontszám esetében a szövegek címét,

<sup>6</sup> 2024 november 20-án futtattuk.

leadjét és törzsszövegét egybevetve, egy egységként kezeltük őket. A mondatokra való lebontást és egyesével való értékelést nem tartottuk sem megvalósíthatónak, sem célravezetőnek, hiszen a szövegek értelmezése – ahogy amellet korábban is érveltünk – nem egyenlő az egyes mondatok értelmezésével. Természetesen ezen lebontás elkerülése miatt az eredményeink csak korlátozottan összevethetőek a többi, az automatikus szövegegyesítését mondatszinten vizsgáló cikk eredményeivel.

Minden modell esetében ugyanazt a promptot használtuk, amely a zero-shot és a one- vagy few-shot megközelítések esetében a példák megadásában, valamint a szeparátorkarakterek felsorolásában különbözött. A promptban az egyszerűsítést adtuk meg a modelleknek feladatként, valamint a 4. részben bemutatott eredmények alapján összefoglaltuk az egyszerűsített szövegek néhány jellemzőjét: a túl hosszú mondatok lerövidítésére kértük, a bonyolult fogalmak elmagyarázására, valamint arra, hogy ugyanazt a fogalmat végig ugyanazzal a szóval illesse a szövegben, és ne használjon szinonimákat. A promptok megtalálhatók a függelékben (A).

A különböző vizsgálati esetekhez a cikkeket az írók által megadott kategóriánként szegmentáltuk, valamint a standard nyelvi és az egyszerűsített szövegek között karakteralapú Levenshtein-távolságot számoltunk a Levenshtein csomaggal (Bachmann, 2024). A távolság alapján ezután három egyforma számosságú csoportot hoztunk létre: az alacsony távolságúakba a távolságmérika szerinti első harmadba, a közepesekbe a középsőbe, míg a felsőbe a felső harmadba tartozó szövegek kerültek. Minél nagyobb távolság volt a két szöveg között, feltételezésünk szerint annál több változtatást (információ-hozzáadást vagy -elhagyást) kellett eszközölni a két szövegverzió között.

A vizsgálati esetek a következők voltak:

1. zero-shot környezet (0-shot);
2. one-shot környezet, ahol a példa cikk-kategóriája nem egyezett meg a teszteset kategóriájával (1-shot);
3. one-shot környezet, ahol a példa cikk-kategóriája megegyezett a teszteset kategóriájával (1-shot + kat.);
4. three-shot környezet, ahol a példa cikk-kategóriája nem egyezett meg a teszteset kategóriájával, és a Levenshtein-távolság alapján randomizálva kerültek elemek be a példák közé (3-shot);
5. three-shot környezet, ahol a példa cikk-kategóriája nem egyezett meg a teszteset kategóriájával, és a Levenshtein-távolság alapján mindhárom csoportból egy-egy elem került be a példák közé (3-shot + Levenshtein);
6. three-shot környezet, ahol a példa cikk-kategóriája megegyezett a teszteset kategóriájával, és a Levenshtein-távolság alapján mindhárom csoportból egy-egy elem került be a példák közé (3-shot + kat. + Levenshtein);
7. three-shot környezet, ahol a példa cikk-kategóriája megegyezett a teszteset kategóriájával, és a Levenshtein-távolság alapján randomizálva kerültek elemek be a példák közé (3-shot + kat);

Minden modellt összesen ötször futtattunk le minden adatsorra, majd az eredményeket átlagoltuk. Ez alól kivétel a Claude és a GPT-4o; itt pénzügyi

okokból az elsőt csak egyszer futtattuk végig minden vizsgálati esetre, a másodikat pedig csak arra a két esetre, amin a GPT-4o mini legjobb és legrosszabb eredményeit érte el.

Modell / vizsgálati eset	BLEU	SARI	Perplexitás
GPT-4o mini			
1. (0-shot)	10,48 ± 6,52	42,49 ± 5,69	15,80 ± 5,76
2. (1-shot)	12,33 ± 6,96	43,99 ± 5,71	15,14 ± 5,39
3. (1-shot + kat.)	12,43 ± 6,88	43,74 ± 5,23	15,14 ± 5,31
4. (3-shot)	12,39 ± 6,75	43,89 ± 5,27	15,31 ± 5,44
5. (3-shot + Levenshtein)	<b>13,38 ± 7,40</b>	<b>44,58 ± 5,58</b>	14,85 ± 5,27
6. (3-shot + kat. + Levenshtein)	13,28 ± 7,17	44,36 ± 5,58	<b>14,58 ± 4,58</b>
7. (3-shot + kat)	12,48 ± 7,11	44,05 ± 5,78	15,06 ± 5,26
Llama 3.3 70B, kvantált			
1. (0-shot)	6,42 ± 4,19	38,59 ± 6,71	14,79 ± 5,36
2. (1-shot)	8,51 ± 5,22	40,62 ± 6,19	13,50 ± 4,68
3. (1-shot + kat.)	8,57 ± 5,06	40,88 ± 6,58	13,40 ± 5,01
4. (3-shot)	8,72 ± 5,21	41,02 ± 6,57	13,48 ± 5,03
5. (3-shot + Levenshtein)	<b>10,65 ± 5,98</b>	42,41 ± 5,91	<b>11,72 ± 4,06</b>
6. (3-shot + kat. + Levenshtein)	10,60 ± 6,05	<b>42,43 ± 6,19</b>	11,66 ± 3,91
7. (3-shot + kat)	8,45 ± 5,52	40,61 ± 6,88	13,57 ± 4,98
Llama 3.1 8B			
1. (0-shot)	6,32 ± 4,26	38,17 ± 6,22	25,11 ± 13,84
2. (1-shot)	6,65 ± 4,81	38,58 ± 6,29	23,57 ± 12,41
3. (1-shot + kat.)	6,51 ± 4,58	38,52 ± 6,08	23,39 ± 11,08
4. (3-shot)	6,58 ± 4,89	38,53 ± 6,17	23,86 ± 12,42
5. (3-shot + Levenshtein)	<b>7,65 ± 5,63</b>	39,27 ± 5,95	<b>19,97 ± 9,26</b>
6. (3-shot + kat. + Levenshtein)	7,52 ± 4,88	<b>39,49 ± 5,61</b>	20,32 ± 9,47
7. (3-shot + kat)	6,94 ± 5,11	38,93 ± 6,18	24,58 ± 12,06
Qwen2.5 7B			
1. (0-shot)	5,64 ± 4,62	37,53 ± 6,28	65,54 ± 45,84
2. (1-shot)	8,37 ± 6,20	39,63 ± 5,50	45,36 ± 29,16
3. (1-shot + kat.)	8,55 ± 6,29	39,69 ± 5,89	44,45 ± 29,07
4. (3-shot)	8,41 ± 6,08	39,93 ± 5,63	47,64 ± 30,27
5. (3-shot + Levenshtein)	9,95 ± 6,84	40,34 ± 5,65	38,32 ± 25,25
6. (3-shot + kat. + Levenshtein)	<b>10,17 ± 6,61</b>	<b>40,65 ± 5,44</b>	<b>37,05 ± 22,17</b>
7. (3-shot + kat)	8,53 ± 5,70	39,75 ± 5,54	44,33 ± 29,79

3. táblázat. A BLEU- és SARI-pontok, valamint a perplexitás megoszlása modellenként. A legjobb értékek minden oszlopban, minden modell esetében félkövérrel vannak jelölve. A jelölések követik a vizsgálati esetek leírásánál feltüntetettet: a + *kat.* jelölés a kategóriaegyeztetést jelenti, a + *Levenshtein* pedig a három Levenshtein-csoport figyelembe vételét veszik. A jelek hiánya az adott tulajdonság randomizált vagy figyelembe nem vett voltára utal.

A 3., a 4., az 5., a 6. valamint a 7. táblázatból jól látható, hogy minden modell esetében a 0-shot módszertan hozta a legalacsonyabb átlag- és mediánértéket is: ebből következtethetünk arra, hogy az összeállított korpusz használata kevés és one-shot környezetben is segít az egyszerűsítésben. Továbbá 0-shot esetében több modell néha, a Claude viszont mindig hozzáfűzött egy bevezető mondatot a kimenethez (pl. „Íme az egyszerűsített szöveg:”). Az in-context tanulás ezt is korrigálta.

Úgy tűnik, hogy 3-shot esetben minden modellnél azok a példák voltak a legsikeresebbek, ahol a Levenshtein-hasonlóság szerinti csoportosítást figyelembe vettük – azaz a modellek jobban teljesítettek akkor, ha a standard nyelvi és az egyszerűsített szöveg közötti átalakításokra variábilis példákat láttak.

A modellek teljesítménye nagyobb szórást mutat: a legjobban a GPT-4o-mini teljesít a SARI-pontszámában, még a nála sokkal drágább GPT-4o-t is megelőzi kb. 2 ponttal (utóbbi  $41,29 \pm 5,89$  SARI pontot ér el 0-shoton, és  $42,70 \pm 5,85$  pontot 3-shoton különböző Levenshtein-távolságokkal). A 3.3-as Llama 70B verziója kvantálva a nagyobb OpenAI modellhez hasonló teljesítményt nyújt. A sort a kisebb modellek: a Llama 3.1 8B és a Qwen2.5 7B zárják további 2-3 pont hátránnyal.

A BLEU-pontszámok ezzel szemben lényegesen alacsonyabbak minden modell esetében. Ennek oka valószínűleg a magyar nyelv ragozó mivolta, illetve a feladat maga: a szövegegyszerűsítés megköveteli mind a mondatszerkezet, mind a szavak egyszerűbbre cserélését, ami megtöri a hosszabb n-gramokat. Ez utóbbi hatást erősíti, hogy minden cikkhez csak egy referencia van; alternatív egyszerűsítésekkel a pontszám valószínűleg magasabb lenne.

Megemlítendő, hogy a BLEU és a SARI ugyan gyakran korrelálnak, azonban nem mindig – egymástól erősen eltérő, magasabb és alacsonyabb eredmények több nyelv több korpuszán végzett méréseken is megfigyelhetők (Ryan és mtsai, 2023).

A 3. táblázatban feltüntettük a válaszok a PULI-GPT-3SX modell szerinti perplexitását is (Yang és mtsai, 2023). Ez a metrika a fenti kettővel szemben elsősorban nem az egyszerűsítés sikerességével, hanem a válasz nyelvi helyességével és „magyarságával” áll korrelációban. A referenciaválaszok átlagos perplexitása 11,09; ehhez legközelebb a magyar nyelvet hivatalosan nem támogató Llama 3.3 70B került, legtávolabb pedig a Qwen. Ez utóbbi nem meglepő annak fényében, hogy a nyelvtudása erősen hullámzó: nagyjából helyes mondatok váltakoznak teljesen értelmetlenekkel<sup>7</sup>. Emellett a szövegben gyakran találtunk kínai (olykor angol) szöveget, akár egy magyar mondat közepén is, melyekben leggyakrabban a szöveg egyszerűsítésére vonatkozó utasítások voltak találhatóak.<sup>8</sup> Ebből kifolyólag magyar nyelvmodellezésre nem ajánlható. A Qwenhez hasonló, vagy még rosszabb eredményt ért el a Llama 3.2 3B, amely egy egészen meglepő, 600

<sup>7</sup> *Egy fiatal Szabadkán hőse a napon megszomba volt. vagy*

*A házak falára terepolták a táblákat, hogy könnyebbé tessék a fájdalom utasításának olvasását.*

<sup>8</sup> *Az üléseken is meg酌取材料, 简写文本并用简单词汇表达: — Holnap a válságstábla ülése lesz.*

fölötti perplexitással rendelkező szöveget is generált, mely pár értelmes magyar szó után legnagyobb részben értelmetlen betűsorokból állt.<sup>9</sup>

Egy adott modellhez tartozó perplexitásokat összehasonlítva kitűnik, hogy a prompt szerepe nagyon fontos nem csak az egyszerűsítés, hanem a szövegminőség miatt is. Minden modell a legrosszabb perplexitást a 0-shot esetben érte el. A kategória egyeztetése érdemben nem befolyásolta a perplexitást, viszont a Levenshtein-távolság-alapú válogatás akár 15–20 százalékkal is javított rajta. A modellek közül a magyar nyelvet hivatalosan is támogató GPT-4o mini perplexitását befolyásolta a tesztkörnyezet a legkevésbé, azonban itt is közel egy pont különbséget lehet elérni egy jobb prompittal.

A perplexitás azonban fenntartásokkal kezelendő: szűrőpróbaszerűen összehasonlítva a GPT-4o mini és a kvantált 70B-s Llama által generált szövegeket, világosan látszik, hogy az utóbbi vét nyelvtani hibákat, amik a GPT-re nem jellemzőek.

Összességében elmondható, hogy a modellek között az előre sejthető sorrend alakult ki: a GPT-4o mini mind nyelvi, mind szövegegyszerűsítési szempontból a legjobb az általunk tesztelt modellek közül. A Llama 3.3 70B jó prompittal megközelíti, de a SARI tekintetében kis mértékben elmarad tőle. A sort a Llama 3.1 8B és a Qwen 2.5 7B zárja, utóbbi nehezen értékelhető nyelvi teljesítménnyel.

## 6. Összefoglalás

A cikk az első olyan magyar nyelvű szövegegyszerűsítési korpuszt mutatta be, amely autentikus, tehát eredetileg is magyarul írt adatokból álló szövegpárokat tartalmaz. A korpusz a PannonRTV híroldal szöveganyagából áll, melyek automatikus módszerekkel lettek összepárosítva. A korpusz kutatási célra szabadon hozzáférhető. A korpuszon először a korpusz két domainje közötti vizsgálatot végeztük el: itt az eredményekből egyértelművé vált, hogy az egyszerűsített szövegek átlagosan lényegesen rövidebb mondatokat használnak. Kimutattuk emellett, hogy az egyszerűsített szövegek ebben a korpuszban egy sokkal kisebb, majdnem feleakkora szótárral operálnak a szövegek megkonstruálásakor. Végül a korpuszt négy nagy nyelvi modell segítségével zero- és few-shot tesztkörnyezetekben próbáltuk ki, melynek során megállapítottuk, hogy a korpusz használata elősegíti a megfelelő egyszerűsített szövegek automatikus létrehozását. Ezek alapján érdemesnek tűnik további vizsgálatokat folytatni a magyar nyelvű szövegegyszerűsítéssel kapcsolatban, akár modellek finomhangolásával, akár az adatszett további bővítésével kapcsolatban.

---

<sup>9</sup> *Kannor mel on deix filtrborh vál Bone Bern dadFSka datas izsak Proapers vie toplyír Kft be menyel EstatyAll. eszet cleverno regalk Bslsn ln dansro ban Neros ours mamm st gef vistaVdre Communications Arg Kanas fant wasm Fcoris*



## Hivatkozások

- Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., ...: GPT-4 technical report (2024), <https://arxiv.org/abs/2303.08774>
- Alva-Manchego, F., Bingel, J., Paetzold, G., Scarton, C., Specia, L.: Learning how to simplify from explicit labeling of complex-simplified text pairs. In: Kondrak, G., Watanabe, T. (szerk.) Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 295–305. Asian Federation of Natural Language Processing, Taipei, Taiwan (Nov 2017), <https://aclanthology.org/I17-1030>
- Alva-Manchego, F., Scarton, C., Specia, L.: Data-driven sentence simplification: Survey and benchmark. Computational Linguistics 46(1), 135–187 (2020), <https://aclanthology.org/2020.cl-1.4>
- Anthropic: The Claude 3 model family: Opus, Sonnet, Haiku (2024), [https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model\\_Card\\_Claude\\_3.pdf](https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf)
- Bachmann, M.: Levenshtein. <https://rapidfuzz.github.io/Levenshtein/> (2024), python package
- Bott, S., Rello, L., Drndarevic, B., Saggion, H.: Can spanish be simpler? lexis: Lexical simplification for spanish. In: International Conference on Computational Linguistics (2012), <https://api.semanticscholar.org/CorpusID:7939405>
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020), <https://arxiv.org/abs/2005.14165>
- Brunato, D., Dell’Orletta, F., Venturi, G., Montemagni, S.: Design and annotation of the first Italian corpus for text simplification. In: Meyers, A., Rehbein, I., Zinsmeister, H. (szerk.) Proceedings of the 9th Linguistic Annotation Workshop. pp. 31–41. Association for Computational Linguistics, Denver, Colorado, USA (Jun 2015), <https://aclanthology.org/W15-1604>
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (szerk.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), <https://aclanthology.org/N19-1423>
- Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., ...: The llama 3 herd of models (2024), <https://arxiv.org/abs/2407.21783>
- Europe, I.: Információ mindenkinek - a könnyen érthető kommunikáció európai alapelvei (2017), [https://www.inclusion-europe.eu/wp-content/uploads/2017/06/HU\\_Information\\_for\\_all.pdf](https://www.inclusion-europe.eu/wp-content/uploads/2017/06/HU_Information_for_all.pdf)

- Feng, Y., Qiang, J., Li, Y., Yuan, Y., Zhu, Y.: Sentence simplification via large language models (2023), <https://arxiv.org/abs/2302.11957>
- Gorenc, S., Robnik-Šikonja, M.: Slovene text simplification dataset SloTS (2022), <http://hdl.handle.net/11356/1682>, slovenian language resource repository CLARIN.SI
- Keskisärkkä, R.: Automatic Text Simplification via Synonym Replacement. Szakdolgozat, Linköping University (2012), <https://api.semanticscholar.org/CorpusID:16139324>
- Khallaf, N., Sharoff, S., Soliman, R.: Towards Arabic sentence simplification via classification and generative approaches. In: Bouamor, H., Al-Khalifa, H., Darwish, K., Rambow, O., Bougares, F., Abdelali, A., Tomeh, N., Khalifa, S., Zaghouni, W. (szerk.) Proceedings of the Seventh Arabic Natural Language Processing Workshop (WANLP). pp. 43–52. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Hybrid) (Dec 2022), <https://aclanthology.org/2022.wanlp-1.5>
- Klerke, S., Søgaaard, A.: DSIm, a Danish parallel corpus for text simplification. In: Calzolari, N., Choukri, K., Declerck, T., Doğan, M.U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S. (szerk.) Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12). pp. 4015–4018. European Language Resources Association (ELRA), Istanbul, Turkey (May 2012), [http://www.lrec-conf.org/proceedings/lrec2012/pdf/270\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2012/pdf/270_Paper.pdf)
- Kouzis-Loukas, D.: Learning Scrapy. Packt Publishing Ltd (2016)
- Maaß, C.: Easy Language - Plain Language - Easy Language Plus: Balancing Comprehensibility and Acceptability. Frank & Timme GmbH (January 2020)
- Nisioi, S., Štajner, S., Ponzetto, S.P., Dinu, L.P.: Exploring neural text simplification models. In: Barzilay, R., Kan, M.Y. (szerk.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 85–91. Association for Computational Linguistics, Vancouver, Canada (Jul 2017), <https://aclanthology.org/P17-2014>
- Padovani, F., Marchesi, C., Pasqua, E., Galletti, M., Nardi, D.: Automatic text simplification: A comparative study in italian for children with language disorders. In: Proceedings of the 13th Workshop on Natural Language Processing for Computer Assisted Language Learning (NLP4CALL 2024). vol. 211, pp. 176–186. Rennes, France (2024)
- Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. p. 311–318. ACL '02, Association for Computational Linguistics, USA (2002), <https://doi.org/10.3115/1073083.1073135>
- Petőfi, S., Nógrádi, G., Bodonyi, P.: János vitéz 1-3. ének. Egyszerűen érthetően / könnyen érthetően. Petőfi Irodalmi Múzeum és Kortárs Irodalmi Központ, Budapest (2022)
- Post, M.: A call for clarity in reporting BLEU scores. In: Proceedings of the Third Conference on Machine Translation: Research Papers. pp. 186–

191. Association for Computational Linguistics, Belgium, Brussels (Oct 2018), <https://www.aclweb.org/anthology/W18-6319>
- Prótár, N., Nemeskey, D.M.: huPWKP: A Hungarian text simplification corpus. In: Mitkov, R., Angelova, G. (szerk.) Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing. pp. 898–907. INCOMA Ltd., Shoumen, Bulgaria, Varna, Bulgaria (Sep 2023), <https://aclanthology.org/2023.ranlp-1.97>
- Qasmi, N.H., Zia, H.B., Athar, A., Raza, A.A.: SimplifyUR: Unsupervised lexical text simplification for Urdu. In: Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. (szerk.) Proceedings of the Twelfth Language Resources and Evaluation Conference. pp. 3484–3489. European Language Resources Association, Marseille, France (May 2020), <https://aclanthology.org/2020.lrec-1.428>
- Qwen, T.: Qwen2.5: A party of foundation models (September 2024), <https://qwenlm.github.io/blog/qwen2.5/>
- Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), <http://is.muni.cz/publication/884893/en>
- Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (11 2019), <http://arxiv.org/abs/1908.10084>
- Ryan, M., Naous, T., Xu, W.: Revisiting non-English text simplification: A unified multilingual benchmark. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 4898–4927. Association for Computational Linguistics, Toronto, Canada (Jul 2023), <https://aclanthology.org/2023.acl-long.269>
- Shardlow, M.: A survey of automated text simplification. International Journal of Advanced Computer Science and Applications 4 (01 2014)
- Tolcsvai Nagy, G.: Szövegtan. In: Kiefer, F. (szerk.) Magyar nyelv, pp. 149–174. Akadémiai Kiadó, Budapest (2006)
- Tátrai, S.: Pragmatika. In: Tolcsvai Nagy, G. (szerk.) Nyelvtan, chap. Pragmatika, pp. 899–1058. Osiris Kiadó, Budapest, Hungary (2017)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR abs/1706.03762 (2017), <http://arxiv.org/abs/1706.03762>
- von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., Gallouédec, Q.: Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl> (2020)
- Xu, W., Napoles, C., Pavlick, E., Chen, Q., Callison-Burch, C.: Optimizing statistical machine translation for text simplification. Transactions of the Association for Computational Linguistics 4, 401–415 (2016), <https://aclanthology.org/Q16-1029>

- Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., ...: Qwen2 technical report. arXiv preprint arXiv:2407.10671 (2024)
- Yang, Z.G., Dodé, R., Ferenczi, G., Héja, E., Jelencsik-Mátyus, K., Kőrös, Á., Laki, L.J., Ligeti-Nagy, N., Vadász, N., Váradi, T.: Jönnek a nagyok! BERT-large, GPT-2 és GPT-3 nyelvmodellek magyar nyelvre. In: XIX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 247–262. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2023)
- Zhu, Z., Bernhard, D., Gurevych, I.: A monolingual tree-based translation model for sentence simplification. In: Huang, C.R., Jurafsky, D. (szerk.) Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010). pp. 1353–1361. Coling 2010 Organizing Committee, Beijing, China (Aug 2010), <https://aclanthology.org/C10-1152>

## A. Promptok

### Zero-shot prompt

You are an assistant, specializing in text simplification. Your task is to simplify news for mentally disabled people.

\*\*\*RULES FOR SIMPLIFICATION\*\*\*

- \*\*\*Only generate text in Hungarian.\*\*\* Do not generate text in any other language.

- Simplify the standard-language text in the \*\*\*TASK\*\*\* section. Only use information that is found in this text. Do not make up any information or facts that is not in this text.

- Explain the concepts that you deem too hard.

- If you think that a sentence is too long, shorten it.

- Use simple, well-known words. Use the same word for the same concept throughout the simplified text. Do \*\*\*not\*\*\* use synonyms for the same concept to make the text less repetitive.

- If you think that some information in the standard-language text is not necessary for the reader to understand the text, you may leave it out of the simplified text.

\*\*\*TASK\*\*\* task

### Few-shot prompt

You are an assistant, specializing in text simplification. Your task is to simplify news for mentally disabled people.

\*\*\*RULES FOR SIMPLIFICATION\*\*\*

- \*\*\*Only generate text in Hungarian.\*\*\* Do not generate text in any other language.

- Simplify the standard-language text in the \*\*\*TASK\*\*\* section. Only use information that is found in this text. Do not make up any information or facts that is not in this text.

- Explain the concepts that you deem too hard.

- If you think that a sentence is too long, shorten it.

- Use simple, well-known words. Use the same word for the same concept throughout the simplified text. Do \*\*\*not\*\*\* use synonyms for the same concept to make the text less repetitive.

- If you think that some information in the standard-language text is not necessary for the reader to understand the text, you may leave it out of the simplified text.

- Use the examples from the \*\*\*EXAMPLES FOR SIMPLIFICATION\*\*\* section to gain information about what a good simplification is. Make sure to generate text that is similar to these examples in grammar and simplicity.

- The examples are separated by '#####'.

\*\*\*EXAMPLES FOR SIMPLIFICATION\*\*\* examples

\*\*\*TASK\*\*\* task

## B. További modellek

A lenti táblázat tartalmazza néhány további modell eredményét. A számokat összevetve a 3 táblázattal, a legfontosabb észrevétel, hogy a GPT-4o mini jelentősen felülmúlja mind a GPT-4o-t, mind az Claude 3.5 Haikut, annak ellenére, hogy ezen a feladaton előbbi  $15\times$ , utóbbi több mint  $6\times$  drágább. A másik fontos tanulság, hogy egy alverzió különbség is sokat tud jelenteni: a Llama 3.3 70B kvantált változatának SARI pontszáma majdnem egy teljes ponttal, perplexitása 1,5-tel javult a 3.1-es verzióhoz képest.

Modell / vizsgálati eset	BLEU	SARI	Perplexitás
Claude 3.5 Haiku			
1. (0-shot)	$6,28 \pm 3,87$	$38,90 \pm 6,14$	$15,11 \pm 4,67$
2. (1-shot)	$8,20 \pm 5,08$	$41,04 \pm 6,29$	$14,10 \pm 4,30$
3. (1-shot + kat.)	$7,98 \pm 5,12$	$40,83 \pm 6,62$	$14,02 \pm 4,17$
4. (3-shot)	$7,96 \pm 5,14$	$40,74 \pm 6,22$	$14,11 \pm 4,17$
5. (3-shot + Levenshtein)	$10,58 \pm 6,02$	$42,81 \pm 5,81$	<b><math>12,63 \pm 3,69</math></b>
6. (3-shot + kat. + Levenshtein)	<b><math>10,86 \pm 6,15</math></b>	<b><math>42,87 \pm 5,96</math></b>	$12,72 \pm 3,73$
7. (3-shot + kat.)	$8,15 \pm 5,07$	$40,95 \pm 6,35$	$13,89 \pm 4,09$
GPT-4o			
1. (0-shot)	$8,82 \pm 5,12$	$41,29 \pm 5,89$	$15,93 \pm 5,72$
5. (3-shot + Levenshtein)	$10,24 \pm 5,75$	$42,70 \pm 5,85$	$14,98 \pm 5,25$
LLama 3.1 70B, kvantált			
1. (0-shot)	$6,36 \pm 4,85$	$38,28 \pm 6,85$	$18,45 \pm 7,74$
2. (1-shot)	$8,13 \pm 5,37$	$40,21 \pm 6,23$	$15,40 \pm 6,39$
3. (1-shot + kat.)	$8,82 \pm 5,44$	$40,76 \pm 6,37$	$15,34 \pm 6,27$
4. (3-shot)	$8,31 \pm 5,44$	$40,11 \pm 6,24$	$15,59 \pm 6,57$
5. (3-shot + Levenshtein)	$10,55 \pm 6,96$	$41,50 \pm 5,88$	<b><math>13,10 \pm 4,52</math></b>
6. (3-shot + kat. + Levenshtein)	<b><math>10,76 \pm 6,60</math></b>	<b><math>41,80 \pm 6,52</math></b>	$13,21 \pm 5,44$
7. (3-shot, + kat.)	$8,60 \pm 5,73$	$40,54 \pm 6,40$	$15,36 \pm 5,56$
Llama 3.2 3B			
1. (0-shot)	$5,83 \pm 4,34$	$37,12 \pm 5,70$	$59,32 \pm 52,30$
2. (1-shot)	$6,73 \pm 5,25$	$38,21 \pm 5,92$	$38,86 \pm 38,57$
3. (1-shot + kat.)	$6,41 \pm 4,88$	$37,62 \pm 5,84$	$41,57 \pm 41,65$
4. (3-shot)	<b><math>7,13 \pm 5,67</math></b>	$38,24 \pm 6,39$	$43,28 \pm 40,85$
5. (3-shot + Levenshtein)	$7,02 \pm 5,67$	<b><math>38,49 \pm 6,24</math></b>	<b><math>40,47 \pm 44,63</math></b>
6. (3-shot + kat. + Levenshtein)	$6,97 \pm 5,56$	$38,47 \pm 6,10$	$40,84 \pm 45,33$
7. (3-shot + kat.)	$6,45 \pm 4,88$	$37,90 \pm 6,38$	$39,04 \pm 33,18$

4. táblázat. A BLEU- és SARI-pontok, valamint a perplexitás megoszlása modellenként. A legjobb értékek minden oszlopban, minden modell esetében félkövérrel vannak jelölve. A jelölések követik a vizsgálati esetek leírásánál feltüntetettet: a *+ kat.* jelölés a kategóriaegyeztetést jelenti, a *+ Levenshtein* pedig a három Levenshtein-csoport figyelembe vételét veszik. A jelek hiánya az adott tulajdonság randomizált vagy figyelembe nem vett voltára utal.

## C. Részletes SARI-, BLEU- és Perplexitás-adatok

A jelölések követik a vizsgálati esetek felsorolásánál feltüntetettet.

Modell	Vizsgálati eset	Mean $\pm$ Stdev	Min	Max
Claude 3.5 Haiku	1. (0-shot)	6,28 $\pm$ 3,87	0,52	27,68
	2. (1-shot)	8,20 $\pm$ 5,08	0,65	25,44
	3. (1-shot + kat.)	7,98 $\pm$ 5,12	0,56	34,55
	4. (3-shot)	7,96 $\pm$ 5,14	0,62	27,89
	5. (3-shot + Levenshtein)	10,58 $\pm$ 6,02	0,54	29,02
	6. (3-shot + kat. + Levenshtein)	10,86 $\pm$ 6,15	0,84	29,68
	7. (3-shot + kat.)	8,15 $\pm$ 5,07	0,49	28,10
GPT-4o	1. (0-shot)	8,82 $\pm$ 5,12	1,61	35,62
	5. (3-shot + Levenshtein)	10,24 $\pm$ 5,75	0,74	27,52
GPT-4o mini	1. (0-shot)	10,48 $\pm$ 6,52	0,90	35,25
	2. (1-shot)	12,33 $\pm$ 6,96	1,13	34,81
	3. (1-shot + kat.)	12,43 $\pm$ 6,88	0,89	37,32
	4. (3-shot)	12,39 $\pm$ 6,75	0,98	36,28
	6. (3-shot + kat. + Levenshtein)	13,28 $\pm$ 7,17	1,23	41,17
	5. (3-shot + Levenshtein)	13,38 $\pm$ 7,40	1,67	35,54
	7. (3-shot + kat.)	12,48 $\pm$ 7,11	1,14	34,41
Llama 3.3 70B, kvantált	1. (0-shot)	6,42 $\pm$ 4,19	0,65	23,46
	2. (1-shot)	8,51 $\pm$ 5,22	0,22	29,83
	3. (1-shot + kat.)	8,57 $\pm$ 5,06	0,64	27,29
	4. (3-shot)	8,72 $\pm$ 5,21	0,43	34,70
	5. (3-shot + Levenshtein)	10,65 $\pm$ 5,98	0,35	33,65
	6. (3-shot + kat. + Levenshtein)	10,60 $\pm$ 6,05	0,86	30,84
	7. (3-shot + kat.)	8,45 $\pm$ 5,52	0,55	30,50
LLama 3.1 70B, kvantált	1. (0-shot)	6,36 $\pm$ 4,85	0,06	31,69
	2. (1-shot)	8,13 $\pm$ 5,37	0,58	29,24
	3. (1-shot + kat.)	8,82 $\pm$ 5,44	0,09	27,42
	4. (3-shot)	8,31 $\pm$ 5,44	0,63	27,52
	5. (3-shot + Levenshtein)	10,55 $\pm$ 6,96	0,33	36,96
	6. (3-shot + kat. + Levenshtein)	10,76 $\pm$ 6,60	0,01	36,17
	7. (3-shot + kat.)	8,60 $\pm$ 5,73	0,10	30,53
Llama 3.1 8B	1. (0-shot)	6,32 $\pm$ 4,26	0,68	22,12
	2. (1-shot)	6,65 $\pm$ 4,81	0,05	24,29
	3. (1-shot + kat.)	6,51 $\pm$ 4,58	0,17	26,41
	4. (3-shot)	6,58 $\pm$ 4,89	0,26	27,25
	5. (3-shot + Levenshtein)	7,65 $\pm$ 5,63	0,07	28,68
	6. (3-shot + kat. + Levenshtein)	7,52 $\pm$ 4,88	0,49	26,63
	7. (3-shot + kat.)	6,94 $\pm$ 5,11	0,26	27,89
Llama 3.2 3B	1. (0-shot)	5,83 $\pm$ 4,34	0,13	25,30
	2. (1-shot)	6,73 $\pm$ 5,25	0,35	35,72
	3. (1-shot + kat.)	6,41 $\pm$ 4,88	0,52	28,76
	4. (3-shot)	7,13 $\pm$ 5,67	0,07	36,07
	5. (3-shot + Levenshtein)	7,02 $\pm$ 5,67	0,07	34,50
	6. (3-shot + kat. + Levenshtein)	6,97 $\pm$ 5,56	0,01	29,48
	7. (3-shot + kat.)	6,45 $\pm$ 4,88	0,23	27,17
Qwen2.5 7B	1. (0-shot)	5,64 $\pm$ 4,62	0,01	25,48
	2. (1-shot)	8,37 $\pm$ 6,20	0,00	32,42
	3. (1-shot + kat.)	8,55 $\pm$ 6,29	0,00	32,07
	4. (3-shot)	8,41 $\pm$ 6,08	0,87	33,52
	5. (3-shot + Levenshtein)	9,95 $\pm$ 6,84	0,80	44,54
	6. (3-shot + kat. + Levenshtein)	10,17 $\pm$ 6,61	0,94	32,46
	7. (3-shot + kat.)	8,53 $\pm$ 5,70	0,11	27,12

5. táblázat. BLEU eredmények



Modell	Vizsgálati eset	Mean $\pm$ Stdev	Min	Max
Claude 3.5 Haiku	1. (0-shot)	38,90 $\pm$ 6,14	24,16	58,61
	2. (1-shot)	41,04 $\pm$ 6,29	23,52	63,53
	3. (1-shot + kat.)	40,83 $\pm$ 6,62	21,91	62,41
	4. (3-shot)	40,74 $\pm$ 6,22	26,25	62,41
	5. (3-shot + Levenshtein)	42,81 $\pm$ 5,81	26,08	60,06
	6. (3-shot + kat. + Levenshtein)	42,87 $\pm$ 5,96	26,23	63,70
	7. (3-shot + kat.)	40,95 $\pm$ 6,35	23,97	60,89
GPT-4o	1. (0-shot)	41,29 $\pm$ 5,89	22,67	57,12
	5. (3-shot + Levenshtein)	42,70 $\pm$ 5,85	24,08	59,16
GPT-4o mini	1. (0-shot)	42,49 $\pm$ 5,69	25,08	64,70
	2. (1-shot)	43,99 $\pm$ 5,71	28,92	63,74
	3. (1-shot + kat.)	43,74 $\pm$ 5,23	25,24	59,97
	4. (3-shot)	43,89 $\pm$ 5,27	25,52	61,21
	5. (3-shot + Levenshtein)	44,58 $\pm$ 5,58	30,00	65,02
	6. (3-shot + kat. + Levenshtein)	44,36 $\pm$ 5,58	30,04	70,66
	7. (3-shot + kat.)	44,05 $\pm$ 5,78	24,54	63,85
Llama 3.3 70B, kvantált	1. (0-shot)	38,59 $\pm$ 6,71	18,26	60,95
	2. (1-shot)	40,62 $\pm$ 6,19	19,99	60,78
	3. (1-shot + kat.)	40,88 $\pm$ 6,58	16,11	62,63
	4. (3-shot)	41,02 $\pm$ 6,57	18,48	57,80
	5. (3-shot + Levenshtein)	42,41 $\pm$ 5,91	23,42	58,73
	6. (3-shot + kat. + Levenshtein)	42,43 $\pm$ 6,19	25,00	68,95
	7. (3-shot + kat.)	40,61 $\pm$ 6,88	19,84	58,53
LLama 3.1 70B, kvantált	1. (0-shot)	38,28 $\pm$ 6,85	19,41	59,78
	2. (1-shot)	40,21 $\pm$ 6,23	23,24	60,14
	3. (1-shot + kat.)	40,76 $\pm$ 6,37	21,65	59,36
	4. (3-shot)	40,11 $\pm$ 6,24	19,20	61,36
	5. (3-shot + Levenshtein)	41,50 $\pm$ 5,88	24,90	59,54
	6. (3-shot + kat. + Levenshtein)	41,80 $\pm$ 6,52	22,34	61,70
	7. (3-shot, + kat.)	40,54 $\pm$ 6,40	21,00	58,29
Llama 3.1 8B	1. (0-shot)	38,17 $\pm$ 6,22	15,08	58,73
	2. (1-shot)	38,58 $\pm$ 6,29	17,01	63,09
	3. (1-shot + kat.)	38,52 $\pm$ 6,08	17,92	55,43
	4. (3-shot)	38,53 $\pm$ 6,17	15,96	60,53
	5. (3-shot + Levenshtein)	39,27 $\pm$ 5,95	25,12	59,03
	6. (3-shot + kat. + Levenshtein)	39,49 $\pm$ 5,61	25,42	52,00
	7. (3-shot + kat.)	38,93 $\pm$ 6,18	20,84	56,70
Llama 3.2 3B	1. (0-shot)	37,12 $\pm$ 5,70	19,60	52,43
	2. (1-shot)	38,21 $\pm$ 5,92	21,73	58,40
	3. (1-shot + kat.)	37,62 $\pm$ 5,84	20,71	53,04
	4. (3-shot)	38,24 $\pm$ 6,39	19,89	57,00
	5. (3-shot + Levenshtein)	38,49 $\pm$ 6,24	18,38	52,61
	6. (3-shot + kat. + Levenshtein)	38,47 $\pm$ 6,10	22,42	59,56
	7. (3-shot + kat.)	37,90 $\pm$ 6,38	17,94	54,83
Qwen2.5 7B	1. (0-shot)	37,53 $\pm$ 6,28	17,87	56,65
	2. (1-shot)	39,63 $\pm$ 5,50	23,39	55,35
	3. (1-shot + kat.)	39,69 $\pm$ 5,89	17,17	57,12
	4. (3-shot)	39,93 $\pm$ 5,63	22,94	57,18
	5. (3-shot + Levenshtein)	40,34 $\pm$ 5,65	23,61	57,14
	6. (3-shot + kat. + Levenshtein)	40,65 $\pm$ 5,44	26,24	63,97
	7. (3-shot + kat.)	39,75 $\pm$ 5,54	17,89	55,87

6. táblázat. SARI eredmények

	Átlag	Minimum	Maximum	
Claude 3.5 Haiku	1. (0-shot)	15,11 ± 4,67	7,45	33,78
	2. (1-shot)	14,10 ± 4,30	7,49	31,66
	3. (1-shot + kat.)	14,02 ± 4,17	7,40	29,37
	4. (3-shot)	14,11 ± 4,17	6,88	27,03
	5. (3-shot + Levenshtein)	12,63 ± 3,69	6,02	23,13
	6. (3-shot + kat. + Levenshtein)	12,72 ± 3,73	5,88	25,40
	7. (3-shot + kat.)	13,89 ± 4,09	6,02	26,73
GPT-4o	1. (0-shot)	15,93 ± 5,72	7,52	47,48
	5. (3-shot + Levenshtein)	14,98 ± 5,25	5,87	57,49
GPT-4o mini	1. (0-shot)	15,80 ± 5,76	6,64	47,00
	2. (1-shot)	15,14 ± 5,39	6,91	34,35
	3. (1-shot + kat.)	15,14 ± 5,31	5,95	64,81
	4. (3-shot)	15,31 ± 5,44	7,02	39,08
	5. (3-shot + Levenshtein)	14,85 ± 5,27	7,02	42,52
	6. (3-shot + kat. + Levenshtein)	14,58 ± 4,58	7,07	34,15
	7. (3-shot + kat.)	15,06 ± 5,26	7,22	37,36
Llama 3.3 70B, kvantált	1. (0-shot)	14,79 ± 5,36	6,60	44,09
	2. (1-shot)	13,50 ± 4,68	6,07	37,93
	3. (1-shot + kat.)	13,40 ± 5,01	5,81	33,86
	4. (3-shot)	13,48 ± 5,03	6,13	37,76
	5. (3-shot + Levenshtein)	11,72 ± 4,06	5,25	38,82
	6. (3-shot + kat. + Levenshtein)	11,66 ± 3,91	5,04	26,39
	7. (3-shot + kat.)	13,57 ± 4,98	5,90	35,79
LLama 3.1 70B, kvantált	1. (0-shot)	18,45 ± 7,74	7,10	50,28
	2. (1-shot)	15,40 ± 6,39	6,00	67,82
	3. (1-shot + kat.)	15,34 ± 6,27	5,97	42,57
	4. (3-shot + Levenshtein)	13,10 ± 4,52	4,54	27,85
	5. (3-shot)	15,59 ± 6,57	6,57	41,24
	6. (3-shot + kat. + Levenshtein)	13,21 ± 5,44	5,54	38,29
	7. (3-shot + kat.)	15,36 ± 5,56	7,19	44,61
Llama 3.1 8B	1. (0-shot)	25,11 ± 13,84	9,13	163,04
	2. (1-shot)	23,57 ± 12,41	5,69	165,13
	3. (1-shot + kat.)	23,39 ± 11,08	8,77	73,61
	4. (3-shot)	23,86 ± 12,42	6,07	118,37
	5. (3-shot + Levenshtein)	19,97 ± 9,26	6,13	88,03
	6. (3-shot + kat. + Levenshtein)	20,32 ± 9,47	5,67	59,32
	7. (3-shot + kat.)	24,58 ± 12,06	8,99	102,85
Llama 3.2 3B	1. (0-shot)	59,32 ± 52,30	5,00	577,32
	2. (1-shot)	38,86 ± 38,57	3,60	299,56
	3. (1-shot + kat.)	41,57 ± 41,65	4,76	351,74
	4. (3-shot)	43,28 ± 40,85	9,96	367,61
	5. (3-shot + Levenshtein)	40,47 ± 44,63	3,83	600,01
	6. (3-shot + kat. + Levenshtein)	40,84 ± 45,33	1,46	734,61
	7. (3-shot + kat.)	39,04 ± 33,18	9,48	160,63
Qwen2.5 7B	1. (0-shot)	65,54 ± 45,84	6,93	236,78
	2. (1-shot)	45,36 ± 29,16	10,32	514,96
	3. (1-shot + kat.)	44,45 ± 29,07	10,19	163,75
	4. (3-shot)	47,64 ± 30,27	8,57	154,69
	5. (3-shot + Levenshtein)	38,32 ± 25,25	1,23	161,58
	6. (3-shot + kat. + Levenshtein)	37,05 ± 22,17	2,66	133,32
	7. (3-shot + kat.)	44,33 ± 29,79	8,78	137,05

7. táblázat. A perplexitás eredményei

# A méret a lényeg? Morfológiailag annotált korpuszok összehasonlító kiértékelése

Dömötör Andrea<sup>1</sup>, Indig Balázs<sup>1,2</sup>, Nemeskey Dávid Márk<sup>1</sup>

<sup>1</sup>Digitális Örökség Nemzeti Laboratórium

<sup>2</sup>ELTE Informatikai Kar

{domotor.andrea,nemeskey.david}@btk.elte.hu

indig.balazs@inf.elte.hu

**Kivonat** A korpuszok egyik legjellemzőbb tulajdonsága a méretük, második szempontként pedig az annotációk minőségét szokás említeni. Minőség alatt elsősorban konzisztenciát értünk: azt mérjük, hogyan teljesít a korpusz egy részén betanított modell a korpusz egy másik részén. Cikkünkben különböző méretű morfológiailag annotált korpuszokon (ELTE DH gold standard korpusz, NYTK-NerKor, Szeged Treebank) vizsgáltuk, hogy mekkora az *elég nagy* korpusz. Mivel a vizsgált korpuszok standard címkekészleteket követnek, kézenfekvőnek tűnt a kombinálásukkal is kísérletezni, így ugyanis a nagyobb méret mellett nagyobb műfaji változottságot is elérhetünk. Eredményeink szerint nem a korpuszok mérete a volt döntő szempont a teljesítményben, a különböző korpuszok vegyítése pedig még rontott is az eredményeken az annotációs sémák eltérő értelmezései miatt.

**Kulcsszavak:** morfológia, korpuszannotáció, korpuszkiértékelés

## 1. Bevezetés

Már szinte közhely, hogy a gépi tanulásra épülő természetesnyelv-feldolgozó algoritmusoknak nagy mennyiségű tanítóadatra van szükségük. De mit jelent pontosan a *nagy mennyiség*? Ez nyilván nagyban függ a feladattól. Kutatásunkban a szótövesítés és a morfológiai címkézés feladatát választottuk a kérdés megvizsgálásához, mert ehhez több különböző méretű gold standard korpusz áll rendelkezésre. A korpuszok teljesítményének összehasonlításával elsősorban arra kerestük a választ, hogy mekkora korpusz elég ahhoz, hogy elfogadható eredményeket kapjunk a kérdéses feladatokon. Azaz alapvetően a méret és a teljesítmény összefüggését vizsgáltuk, nem volt célunk a korpuszok minőségének értékelése.

A korpuszméret kapcsán felmerül a különböző korpuszok együttes használatának lehetősége is. Ez korábban a névelem-felismerésben jól működő módszernek bizonyult (Simon és mtsai, 2022), így a morfológiai elemzés tanításához is megpróbáltuk kombinálni az azonos címkekészletet használó korpuszokat. Mivel ezek a címkekészletek (emMorph és UD) erősen szabványosítottak, így elvileg az ezeket használó korpuszok annotációi kompatibilisek egymással. Nem szabad azonban figyelmen kívül hagyni, hogy mind a szótövesítés, mind a morfológiai

elemzés jóval szubjektívebb feladat a névelem-annotálásnál. Emiatt erősen kérdéses, hogy az elvileg kompatibilis korpuszok vajon valóban kompatibilisek-e (és egyáltalán van-e rá szükség, hogy azok legyenek).

## 2. Kapcsolódó munkák

A mieinkhez hasonló kérdéseket vizsgált Straka és Straková (2017) a UDPipe 1.1-es verziójának kiértékelése kapcsán. Mivel számos nyelvre többféle UD korpusz is létezik, ezekkel külön-külön és vegyítve is végeztek tanításokat. Általánosságban arra jutottak, hogy a modellek jobb eredményt értek el, ha a tanításhoz csak egy korpuszt használtak fel, a különböző korpuszok vegyítése rontott a teljesítményen. A kisebb korpuszok esetén részletesebb kísérleteket is végeztek a tanítóadatok más korpuszokból való dúsításának lehetőségeiről. Az eredményekben csak azokat a korpuszokat jelenítették meg, ahol a más korpusz adataival való bővítés jobb teljesítményt eredményezett a függőségi annotációban. Ez összesen 12 korpuszt jelent ógörög, cseh, angol, francia, olasz, latin, szlovén és svéd nyelveken. Ebből a POS-tagelés 6, a morfológiai jegyek meghatározása 4, a szótövesítés pedig 7 esetben volt sikeresebb a bővített korpuszok esetén. A korpuszméret más forrásból való növelése tehát még a kis korpuszoknál sem bizonyult egyértelműen jó ötletnek. A szerzők ezt a különböző korpuszok annotációinak inkonzisztenciájával magyarázzák (*"the Universal Dependencies are yet not so universal as everyone would like"*).

A UD korpuszok annotációinak eltéréseit vizsgálja Wisniewski és Yvon (2019), elsősorban angol és francia nyelvű treebankeken (mivel ezekből elég sok van). A korpuszok közötti eltérések detektálására Boyd és mtsai (2008) módszerét használták, mely szerint ha két azonos szekvencia máshogy van annotálva, akkor valamelyik szekvencia vélhetően inkonzisztens. Inkonzisztenciák természetesen előfordulhatnak (és elő is fordulnak) egy korpuszon belül is, Wisniewski és Yvon (2019) eredményei szerint azonban az eltérő annotációk aránya minden esetben nagyobb volt, ha a korpuszt egy másik korpuszal hasonlították össze, mint ha saját magával. A korpuszok közötti eltérések jellemzésére egy másik kísérletet is végeztek a szerzők. Ebben egy bináris osztályozót tanítottak be, amelynek az volt a feladata, hogy eldöntse egy mondatról, hogy két korpusz közül melyikhez tartozik. Intuitív feltételezés szerint minél nagyobb lesz ennek az osztályozónak a hibaaránya, annál hasonlóbb a két korpusz. Az osztályozót betanították csak szavakra, csak POS-tagekre és szó + POS-tag kombinációkra is. A legsikeresebb osztályozást ez utóbbinál érték el, ami arra utal, hogy az azonos szavak (vagy szószekvenciák) eltérő annotációi jól jellemzik a korpuszokat, azaz a korpuszok közötti eltérések szisztematikusak.

A korpuszméret és a vele betanított modell sikerességének kapcsolatához érdemes még megemlíteni Martin és mtsai (2020) munkáját, akik CamemBERT nevű BERT alapú francia modelljüket (és összehasonlításképpen más modelleket) tanították be és értékelték ki négy különböző korpuszon. Ezek közül a legnagyobb 389363 tokent tartalmazott, a szófaji címkézésben a legjobb eredmé-

nyeket azonban a nála jóval kisebb, 68615 tokenből álló Sequoia korpuszal érték el a szerzők.

Egyrészt elmondható tehát, hogy az annotációs sémák eltérése a különböző korpuszokban más nyelveknél is ismert probléma. Másrészt feltehető az is, hogy a morfológiai címkéző modellek eredményességének nem feltétlenül az óriási korpuszméret a kulcsa.

### 3. Felhasznált korpuszok és eszközök

Kísérleteinkhez 3 különböző méretű kézzel annotált korpuszt használtunk fel. Ezek közül a legnagyobb a HuSpacy tanítóanyagául is szolgáló Szeged Treebank (Vincze és mtsai, 2010). Ennek teljes mérete 1 362 505 token. Az eredeti annotáció nagyrészt automatikusan lett a Universal Dependencies szabványra konvertálva<sup>1</sup>, a UD annotációk csak a korpusz egy kis részén (42 032 token) kaptak kézi ellenőrzést.<sup>2</sup>

A második legnagyobb felhasznált korpusz az NYTK-NerKor (Simon és Vadász, 2021)<sup>3</sup>, amely összesen 1 017 340 tokent tartalmaz. A korpuszhoz gold standard morfológiai annotáció készült az emMorph címkekészletével, majd ez az emtsv emmorph2ud2 (Vadász és Simon, 2019) eszközével lett UD szabványra konvertálva. A konvertált UD annotációk automatikus és kézi javításokon is át estek.

A legkisebb vizsgált korpusz az ELTE DH gold standard korpusza (K. Molnár és Dömötör, 2023)<sup>4</sup>, amely 496 060 tokenből áll, és az NYTK-NerKorhoz hasonló módszertannal készült. Mindkét korpuszt az emtsv (Indig és mtsai, 2019) morfológiai elemző moduljaival előelemezték, majd az elemző által adott annotációkat javították kézzel az annotátorok. Az eredeti, kézzel ellenőrzött annotációk ennek megfelelően itt is az emMorph címkekészletét követik, majd ezek is az emmorph2ud2 eszközzel lettek UD annotációval kiegészítve. Ez utóbbi annotációs réteg nem kapott kézi ellenőrzést.

Mindhárom korpusz műfajilag heterogén, tartalmaznak egymással átfedő és egyedi szövegtípusokat is, így a korpuszok kombinálásával nem csak nagyobb méretet, hanem nagyobb műfaji változatosságot is elérhetünk. A korpuszokban található műfajokat az 1. táblázatban foglaltuk össze.

<sup>1</sup> <https://github.com/huspace/huspace-resources/tree/master/data/processed/szeged-corpus>

<sup>2</sup> [https://github.com/UniversalDependencies/UD\\_Hungarian-Szeged/](https://github.com/UniversalDependencies/UD_Hungarian-Szeged/)

<sup>3</sup> <https://github.com/nytud/NYTK-NerKor/>

<sup>4</sup> <https://github.com/ELTE-DH/gold-standard>

	ELTE DH	NYTK-NerKor	Szeged Treebank
Szépirodalom	✓	✓	✓
Tudományos-ismeretterjesztő (cikkek)	✓	(wikipedia) ✓	
Blog	✓		
Jogi	✓	✓	✓
Hírek		✓	✓
Web		✓	
Diákok fogalmazásai			✓
Informatikai			✓

1. táblázat. A korpuszok műfajai

A korpuszok teszteléséhez elsősorban a HuSpaCy (Orosz és mtsai, 2023) lemmatizáló és morfológiai elemző moduljait használtuk, mivel azonban a három vizsgált korpuszból kettő eredetileg emMorph címkekészlettel készült, így ezek összehasonlításához a PurePos (Orosz és Novák, 2013) tanításával is végeztünk kísérleteket. Ez utóbbinál lehetőség van előelemzést is végeztetni a szabályalapú emMorph (Novák és mtsai, 2016) modul segítségével.

A korpuszok train-dev-test felosztásához a HuSpaCy eredeti (Szeged Treebankból származó) tanítóanyagának arányait használtuk. A vágásnál szempont volt, hogy minden alkorpuszból nagyjából azonos arányban kerüljön szöveg a train, dev és test halmazokba, illetve hogy a bekerülő szövegek teljes mondatok legyenek. A korpuszokat először külön-külön használtuk tanításra és tesztelésre, majd megpróbáltuk kettesével kombinálni őket.

A HuSpaCy eredményei összehasonlíthatók a spaCy (Honnibal és mtsai, 2020) más nyelvű modelljeivel. A hivatalos weboldalon<sup>5</sup> 24 nyelv összesen 82 modelljének eredményei érhetők el. Ebből 16 nyelv esetén van adat mind a UD POS-tag, a morfológiai jegyek és a lemmatizálás kiértékelésére. A modellek átlagos teljesítménye az egyes feladatokon a 2. táblázatban látható.

POS	Morph	Lemma
0,966	0,944	0,940

2. táblázat. A spaCy különböző nyelvű modelljeinek átlagos accuracy értékei

A legjobb eredményt a spanyol, a katalán és a német modellek érték el, ezek mindhárom feladaton az átlagosnál jobban teljesítettek. A leggyengébbek (mindhárom feladaton átlag alatti) teljesítményt a finn, a litván és a görög modellek adták. Ezen kívül még az olasz és a norvég modellek eredményei mondhatók kiegyensúlyozottnak: mindkettő átlagos eredményt nyújt POS-tagelésben, a má-

<sup>5</sup> <https://spacy.io/models>

sik két feladaton pedig átlagnál jobbat. A többi modell sikeressége eltérő volt az egyes feladatoknál.

A 3. táblázat az előbb említett modellek eredményeit és a modell alapjául szolgáló korpuszok méretét mutatja. Az egyes nyelveknél a legjobb eredményt elérő modellt vettük figyelembe. A korpuszméret a modell tanításához és kiértékeléséhez használt teljes korpusz tokenszámát jelenti.

Nyelv	Korpuszméret	POS	Morph	Lemma
<b>német</b>	900 000	0,99	0,97	0,98
<b>spanyol</b>	500 000	0,99	0,99	0,97
<b>katalán</b>	500 000	0,99	0,99	0,98
<b>norvég</b>	311 277	0,97	0,96	0,97
<b>olasz</b>	278 429	0,97	0,97	0,98
<b>finn</b>	201 948	0,96	0,92	0,86
<b>litván</b>	70 047	0,95	0,89	0,86
<b>görög</b>	61 673	0,96	0,91	0,90

3. táblázat. A spaCy legjobb és legrosszabb eredményt elért modelljei

Itt az látszik, hogy a gyengébb teljesítményű modellek tanítókorpusza valóban jellemzően kisebb, egy bizonyos korpuszméret fölött (a 278 ezres olasz korpusztól kezdve) azonban már nincs lényeges különbség az eredmények között. Bár az eredmények pontos értelmezéséhez nyilvánvalóan figyelembe kellene venni az egyes nyelvek (morfológiai gazdagság) és korpuszok (műfaji sokszínűség) különbségeit, az általános tendencia mégis arra utal, hogy nem a méret a lényeg.

## 4. Eredmények

### 4.1. HuSpaCy

A 4. táblázatban a HuSpaCy eredményei láthatók a különböző korpuszokon és azok kombinációin tanítva. A szófaj meghatározásában (POS), a NerKor érte el a legjobb eredményt. A lemmatizálásnál nagyjából a mérettel arányosan változott a teljesítmény az egyes korpuszoknál. A morfológiai jegyek azonosításában (Feats) a Szeged Treebank jelentősen alulteljesít a másik két korpuszhoz képest. A legnagyobb eltérés a ritka jegyeknél (**Reflex**, **Aspect**, **Number [psed]**), a melléknevek fokozásánál (**Degree**) és a számjegyeknél (**NumType**) mutatkozott. Összességében azonban elmondható, hogy mindhárom korpusz eredménye eléri, sőt meghaladja a spaCy más nyelvű modelljeinek előző fejezetben ismertetett átlagos teljesítményét, és közelítik a hasonló korpuszmérettel bíró német, spanyol és katalán modellek eredményeit.

A táblázat alsó részében azt láthatjuk, hogy a különböző korpuszok kombinálása szinte minden esetben lényegesen rontott az eredményeken; egyedül az

ELTE–Szeged párosítás lemma eredménye nem rosszabb mindkét korpusz önálló teljesítményénél. A legrosszabb eredményt éppen a két nagyobb korpusz, a NerKor és a Szeged Treebank párosítása adta.

Korpusz	train	dev	test	POS	Lemma	Feats
<b>ELTE DH</b>	485 525	5250	5285	0,982	0,975	0,977
<b>NerKor</b>	997 002	10 167	10 148	0,986	0,982	0,979
<b>Szeged</b>	1 340 639	11 418	10 448	0,983	0,987	0,969
<b>ELTE DH + Szeged</b>	1 826 164	16 668	15 733	0,976	0,979	0,954
<b>NerKor + Szeged</b>	2 337 641	21 585	20 596	0,914	0,918	0,897

4. táblázat. A HuSpaCy eredményei különböző korpuszokon tanítva

Az eddigiek szerint tehát a kisebb korpuszok nem teljesítenek (sokkal) rosszabbul a nagyobbaknál, így azt is mértük, mi az a minimális korpuszméret, amivel még értékelhető eredményt lehet elérni. Ehhez a legkisebb (ELTE-DH) korpusz train és dev alkorpuszait vágtuk fokozatosan kisebb szeletekre (75-50-25-10-5%). A tesztalmazon nem változtattunk. Ennek eredményei az 5. táblázatban láthatók.

train	dev	POS	Lemma	Feats
485 525	5250	0,982	0,975	0,977
351 723	4101	0,977	0,970	0,975
237 287	3019	0,975	0,967	0,972
122 254	1426	0,976	0,962	0,973
47 971	877	0,965	0,930	0,958
23 540	610	0,233	0,640	0,079

5. táblázat. A HuSpaCy teljesítménye kisméretű korpuszokon

A POS-tageknél a korpusz 75%-ra (350 ezer token) csökkentése 0.5% visszaesést eredményez, a következő vágásoknál (50 és 25%) azonban nem tapasztalunk ekkora változást. A 25%-os (120 ezres) korpusz mind a POS, mind a feats esetében egy kicsivel még jobb eredményt is ad, mint az 50%-os (237 ezres) korpusz. A lemmák pontossága 25%-ig folyamatos csökkenést mutat. Nagyobb romlást mindhárom feladatnál a 10%-os (48 ezres) korpusznál látunk először, a még ennél is kisebb tanítókörpusz esetén pedig már értékelhetetlen eredményt kapunk. (A 64%-os lemma accuracy értelmezéséhez megjegyezzük, hogy a tesztkorpusz szavainak 63,4%-ára igaz, hogy a szótó megegyezik a szóalakkal.)



## 4.2. PurePos

Hasonló kísérleteket végeztünk a PurePos tanításával is a két emMorph annotációt tartalmazó korpusszal (ELTE DH és NerKor). Az elemzőt először az emMorph modul kikapcsolásával tanítottuk, azaz előelemzés nélkül, csak a korpuszokból kellett megtanulnia a címkekészletet. Itt is először külön-külön, majd együtt is tanítottunk a két korpusszal. A tanítási kísérleteket elvégeztük az ELTE DH korpusz kisebbre vágott változataival is. Ezek eredményei a 6. és 7. táblázatban láthatók. A 6. táblázatban azért szerepel külön oszlopban a UD és az emMorph lemma, mert a NerKorban kétféle lemma érték is tartozik a szavakhoz: az eredeti (emMorph) lemmákat a UD-re konvertálás során a UD sémájára igazították a korpusz készítői. Kísérleteinkben ezért mindkét lemmával elvégeztük a tanítást.

Korpusz	train	test	Tag	Lemma (UD)	Lemma (emMorph)
<b>ELTE DH</b>	485 525	10 535	0,948	0,925	0,925
<b>NYTK-NerKor</b>	997 002	20 315	0,948	0,940	0,925
<b>ELTE + NerKor</b>	1 482 527	30 850	0,942	0,923	0,919

6. táblázat. A PurePos eredményei különböző korpuszokon tanítva emMorph nélkül

A két korpusz eredményeit összehasonlítva azt látjuk, hogy a címkézési feladatnál különböző méretük ellenére is azonos teljesítményt nyújtott a két korpusz. A lemmatizálásnál a NerKor UD lemmái könnyebben tanulhatónak bizonyultak, mint az eredeti lemmák, ez utóbbiak szintén azonos eredményt mutatnak az ELTE DH korpusszal. Itt is igaz, hogy a két korpusz kombinálása nemhogy nem javított, de még rontott is az eredményeken.

Train méret	Tag accuracy	Lemma accuracy
485 525	0,948	0,925
351 723	0,947	0,927
237 287	0,946	0,927
122 254	0,946	0,938
47 971	0,925	0,935
23 540	0,894	0,912
11 951	0,889	0,908
5 863	0,836	0,872

7. táblázat. A PurePos teljesítménye kisméretű korpuszokon emMorph nélkül

A 7. táblázatban látható eredmények szerint a PurePos a HuSpaCy-nél kevésbé érzékeny a korpuszméretre. (Ami nem meglepő, hiszen a PurePosnak ki-

fejezett célja volt, hogy kis korpuszokon is képes legyen viszonylag megbízható eredményt adni.) A lemmatizálásban még egészen kicsi, alig 12 ezer token méretű tanítókorpusszal is elérte a 90%-ot. Érdekes, hogy a 120 és a 48 ezres korpuszoknál a lemmatizálás eredményessége átmenetileg még javult is, ez valószínűleg a részkorpuszok sajátosságai miatt lehet. A tagelésnél itt is a 48 ezres korpusznál látjuk az első nagyobb visszaesést, de ennek az eredményei is sokáig 90% közelében maradnak.

A 8. és 9. táblázatban ugyanezeket a tanításokat látjuk, de ezúttal használtuk a beépített emMorph előelemző modult, azaz a korpuszokból betanított modellnek már csak az egyértelműsítés volt a feladata. Ehhez referenciának érdemes megvizsgálni, hogy a szavak mekkora része eleve egyértelmű. Ez legkönnyebben az ELTE DH korpusz xml változatán volt mérhető, ez ugyanis tartalmazza az összes alternatív emtsv elemzést. Eszerint a szavak közel felénél (45,7%) mind a szótó, mind az elemzés egyértelmű, így nem meglepőek az előzőeknél még erősebb eredmények. A legjobban az ELTE DH korpusza teljesített. A NerKor alulmaradása valószínűleg annak tudható be, hogy a korpusz készítői vélhetően többször adtak meg saját elemzést az emtsv által felajánlottak helyett.

Korpusz	train	test	Tag Lemma (UD)	Lemma (emMorph)
<b>ELTE DH</b>	485 525	10 535	0,963	0,982
<b>NYTK-NerKor</b>	997 002	20 315	0,936	0,948
<b>ELTE + NerKor</b>	1 482 527	30 850	0,942	0,958

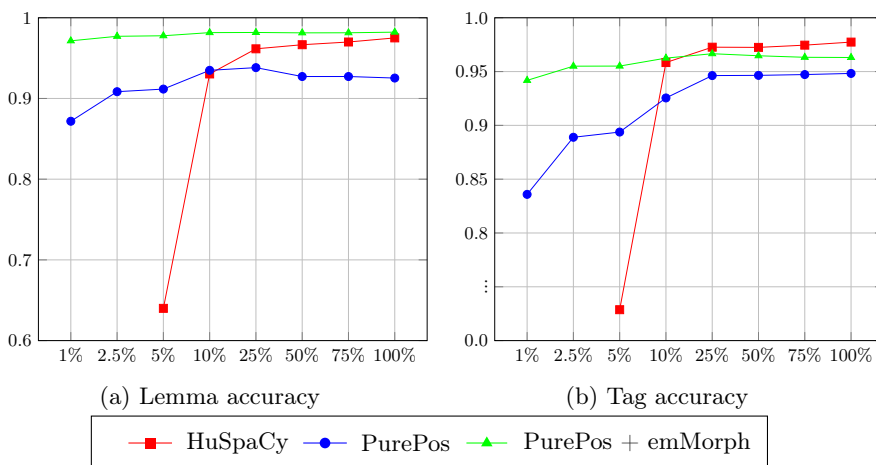
8. táblázat. A PurePos eredményei különböző korpuszokon emMorph előelemzéssel

A korpuszméret csökkentése itt – nem meglepő módon – még kevésbé befolyásolta az eredményeket: a korpusz csupán 10%-át (48 ezer tokent) használva is ugyanolyan eredményeket kaptunk, mint a teljes korpuszon tanítva.

Train méret	Tag accuracy	Lemma accuracy
485 525	0,963	0,982
351 723	0,963	0,981
237 287	0,965	0,981
122 254	0,967	0,982
47 971	0,963	0,982
23 540	0,955	0,978
11 951	0,955	0,977
5863	0,942	0,971

9. táblázat. A PurePos teljesítménye kisméretű korpuszokon emMorph előelemzéssel

Az 1. ábrán még jobban látszik a neurális (HuSpaCy) és a HMM (PurePos) rendszerek eltérő karakterisztikája. A HuSpaCy eredménye mind a szótövesítésben, mind a tagelésben meredeken zuhan a 10%-osra redukált korpuszméret alatt, míg a PurePos teljesítménye jóval visszafogottabb ütemben csökken. (A HuSpaCy esetén az (1b) ábrán a feats eredményeket jelenítettük meg, mert nehézségben ez a leginkább összevethető az emMorph címkézéssel.) Az emMorph előelemzővel az esés mindkét feladaton még lassabb, a szótövesítés pontossága pedig végig a HuSpaCy-é felett marad.



1. ábra. A lemma- és POS-tag pontosság alakulása a korpuszméret függvényében

## 5. Diskusszió

Az eredmények tükrében visszatérhetünk az eredeti kérdéshez: mennyire számít a méret? Úgy tűnik, hogy a neurális háló alapú rendszereknél, amilyen a HuSpaCy, számít, de ott sem mindenképp felett. A kisebb korpuszok nemcsak versenyképesnek bizonyultak a legnagyobb mérettel bíró Szeged Treebankkel szemben, de bizonyos feladatokban meg is előzték. Azt mondhatjuk tehát, hogy nagyjából félmillió tokenről (ami az ELTE DH korpusz teljes mérete) kezdve már nem feltétlenül számít a méret. Az ennél kisebb részkorpuszokon azonban egyértelmű volt a teljesítménycsökkenés, különösen a lemmatizálásnál. A POS-tagek és a morfológiai jegyek pontossága viszont meglepően kis korpuszméretnél (120 ezer token) is a "vállalható" tartományban maradt.

Összességében tehát úgy tűnik, hogy mindhárom vizsgált korpusz mérete megfelelő a HuSpaCy tanításához, a PurePosnak pedig még kisebb is elég lenne, különösen ha a címkézéslet lehetővé teszi az emMorph előelemzésének használatát. Ez azt is megmutatja, hogy a neurális hálók és a mesterséges intelligencia

térhódítása mellett is lehet helye a szabályalapú moduloknak: az emMorph és a PurePos éppen a HuSpaCy számára leginkább kihívást jelentő lemmatizálásban volt különösen sikeres a kisebb korpuszokon. Az, hogy a tagelési feladatban viszont (bár abban is jó) nem éri el a lemmatizáló szintjét amiatt lehet, hogy a szófaj kérdése sokszor a szakértő emberi annotátorok számára sem egyértelmű. Mind a NerKorban, mind az ELTE DH korpuszban gyakori tévesztés volt a melléknevek és főnevek keverése, ami a nyelvészeknek is "szürke zónának" számít.

További érdekes és hasznos tanulság, hogy bármennyire is csábító gondolat a nagyobb méret elérése érdekében együtt használni a meglévő korpuszokat, ez sajnos minden esetben rossz ötletnek bizonyult. Úgy tűnik, hogy a lemmatizálás és morfológiai elemzés túlságosan sokrétű feladat ahhoz, hogy bármely két műhely egységesen tudjon annotálni, még akkor is, ha azonos sémát próbálnak követni. A korpuszok közötti eltérések részletes vizsgálata egy külön tanulmányt érdemelne, itt csak néhány különbséget említünk meg, amelyek vagy már a korpuszok leírásából kiderülnek, vagy az általunk végzett mérések eredményeiből nagyon szembetűnőek voltak.

A NerKor például sajátos elemzést alkalmaz a többtagú tulajdonnevekre. Ezek belső tagjai nem kapnak esetragot, csak sima [/N] taggel annotálják őket, a szótó pedig azonos a szóalakkal. Ez a UD konverzióba nem került át, így csak a NerKor és az ELTE DH kompatibilitását érinti (amennyiben az emMorph annotációkat használjuk). További különbség a két kisebb korpusz között, hogy a NerKor megkülönbözteti a melléknévi igeneveket a melléknevektől, előbbiekre a [/V] [\_ImpfPtcp/Adj], [/V] [\_PerfPtcp/Adj] és [/V] [\_ModPtcp/Adj] címkéket használja, míg az ELTE DH korpuszban ez a különbség csak a részletes elemzésben jelenik meg, az egyszerű címke minden esetben [/Adj]. Ez érinti a UD változatot is, a NerKor a melléknévi igeneveknél használ egy extra **VerbForm** feature-t, ami se az ELTE DH, se a Szeged Korpuszban nem szerepel, ezekben a melléknévi igenevek annotációja megegyezik a sima melléknevekével. Egyébként pont ezek az eltérések okozzák a NerKor gyengébb teljesítményét a PurePos+emMorph tanításában, az emMorph ugyanis nem ad sima [/N] illetve igenévi címkéket, így ezek okozták a legtöbb tévesztést ennél a modellnél.

Az ELTE DH korpusz sajátossága, hogy a ragozott névmások (pl. *róla*) esetén a szótóban különbséget tesz a személyek (*ő*) és a nem személyek (*az*) között, míg a másik két korpusz nem használja az *az* szótóvet ezekben az esetekben. A NerKor dokumentációja továbbá megjegyzi, hogy a Szeged Treebank nem mindig következetes a névmások szótóvesítésében, és az sem könnyítette meg a szabályok egyértelmű meghatározását, hogy néhány ragozott alakra nincs példa a korpuszban.

Végül meg kell említeni a segédigék kérdéskörét, amit szintén eltérően kezelnek a korpuszok. A Szeged Treebankben **AUX** címkét kap a múlt idejű feltételes mód (*volna*) és a jövő idő (*fog*) segédigéje, valamint a kopula. Az emMorph címkészletben azonban nincs segédige kategória, így az emmorph2ud2 konvertáló eszközbe sem került bele. A NerKor a sima [/V] címkével különbözteti meg a segédigét a többi igétől, ezt azonban csak a *volna* esetében használják (ami érthető, hiszen a jövő időt jelző *fog*-nak és a kopuláknak van ragozása). Az ELTE DH

korpusz UD változatában viszont (mivel az az emmorph2ud2 eszközzel készült) egyáltalán nem szerepel az AUX címke.

## 6. Összegzés

Összefoglalva, a morfológiai annotálás tanítása a tapasztalataink szerint nem feltétlenül igényel óriási tanítókörpuszt. A méretnél lényegesebb szempont az annotációk konzisztenciája. A korpuszok kombinálásával kapott eredmények azt mutatták, hogy már az annotációs sémák kis eltérései is komoly nehézséget tudnak okozni a tanuló algoritmusoknak.

Mindebből jogosan következne az igény a különböző korpuszok annotációs sémáinak egységesítésére, különösen egy olyan nemzetközi szabvány esetén, mint a Universal Dependencies. Ugyanakkor azt is láttuk, hogy nincs feltétlenül szükség a meglévőknél nagyobb annotált korpuszokra, így az is elfogadható irány lehet, ha hagyjuk, hogy az egyes korpuszok megőrizzék az "egyénségüket", aminek az emMorph elemzési réteg jó tere lehet.

## Köszönetnyilvánítás

A kutatást a Digitális Örökség Nemzeti Laboratórium támogatta.

## Hivatkozások

- Boyd, A., Dickinson, M., Meurers, D.: On detecting errors in dependency treebanks. *Research on Language and Computation* 6, 113–137 (10 2008)
- Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A.: *spaCy: Industrial-strength Natural Language Processing in Python* (2020)
- Indig, B., Sass, B., Simon, E., Mittelholcz, I., Kundraóth, P., Vadász, N.: emtsv – Egy formátum mind felett. In: XV. Magyar Számítógépes Nyelvészeti Konferencia. pp. 235–247. Szegedi Tudományegyetem Informatikai Intézet, Szeged (2019)
- K. Molnár, E., Dömötör, A.: Gondolatok a gondola-tokról. Morfológiai annotációt javító módszerek tesztelése gold standard korpuszon. In: Berend, G., Gosztolya, G., Vincze, V. (szerk.) XIX. Magyar Számítógépes Nyelvészeti Konferencia. pp. 341–356. Szeged (2023)
- Martin, L., Muller, B., Ortiz Suárez, P.J., Dupont, Y., Romary, L., de la Clergerie, É., Seddah, D., Sagot, B.: CamemBERT: a tasty French language model. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (szerk.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. pp. 7203–7219. Association for Computational Linguistics, Online (Jul 2020), <https://aclanthology.org/2020.acl-main.645/>
- Novák, A., Siklósi, B., Oravecz, Cs.: A New Integrated Open-source Morphological Analyzer for Hungarian. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pp. 1315–1322. Portorož (2016)

- Orosz, Gy., Szabó, G., Berkecz, P., Szántó, Zs., Farkas, R.: Advancing Hungarian Text Processing with HuSpaCy: Efficient and Accurate NLP Pipelines. In: Ekštejn, K., Pártl, F., Konopík, M. (szerk.) Text, Speech, and Dialogue. pp. 58–69. Springer Nature Switzerland, Cham (2023)
- Orosz, Gy., Novák, A.: PurePos 2.0: a hybrid tool for morphological disambiguation. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013). pp. 539–545. INCOMA Ltd. Shoumen, Hissar, Bulgaria (2013)
- Simon, E., Vadász, N.: Introducing NYTK-NerKor, A Gold Standard Hungarian Named Entity Annotated Corpus. In: Ekštejn, K., Pártl, F., Konopík, M. (szerk.) Text, Speech, and Dialogue - 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6-9, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12848, pp. 222–234. Springer (2021)
- Simon, E., Vadász, N., Lévai, D., Nemeskey, D.M., Orosz, Gy., Szántó, Zs.: Az NYTK-NerKor több szempontú kiértékelése. In: Berend, G., Gosztolya, G., Vincze, V. (szerk.) XXVIII. Magyar Számítógépes Nyelvészeti Konferencia. pp. 403–416. Szegedi Tudományegyetem TTIK, Informatikai Intézet, Szeged (2022)
- Straka, M., Straková, J.: Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In: Hajič, J., Zeman, D. (szerk.) Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. pp. 88–99. Association for Computational Linguistics, Vancouver, Canada (Aug 2017), <https://aclanthology.org/K17-3009/>
- Vadász, N., Simon, E.: Konverterek magyar morfológiai címkekészletek között. In: Berend, G., Gosztolya, G., Vincze, V. (szerk.) XV. Magyar Számítógépes Nyelvészeti Konferencia. pp. 99–111. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2019)
- Vincze, V., Szauter, D., Almási, A., Móra, Gy., Alexin, Z., Csirik, J.: Hungarian Dependency Treebank. In: Proceedings of LREC 2010. ELRA, Valletta, Malta (May 2010)
- Wisniewski, G., Yvon, F.: How Bad are PoS Tagger in Cross-Corpora Settings? Evaluating Annotation Divergence in the UD Project. In: Burstein, J., Doran, C., Solorio, T. (szerk.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 218–227. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), <https://aclanthology.org/N19-1019/>

# POSZTER, LAPTOPOS BEMUTATÓ





# Zárt és nyílt súlyú LLM-ek teljesítményértékelése RAG-alapú felsőoktatási chatbot alkalmazás példáján

Csáki Csaba<sup>1</sup> és Vándor Péter<sup>1</sup>

<sup>1</sup> Budapesti Corvinus Egyetem, Fővám tér 8,  
H-1093 Budapest, Magyarország  
peter.vandor@stud.uni-corvinus.hu  
csaki.csaba@uni-corvinus.hu

**Kivonat:** A nagy nyelvi modellek (LLM) terjedésével egyre népszerűbbek a célzott saját (custom) chatbotok. Ezek fejlesztése sokszor a Retrieval Augmented Generation (RAG) megoldásra épül. Azonban mind a fejlesztés és tesztelés, mind az alkalmazás szempontjából fontos döntés, melyik LLM-t használjuk a saját chatbot mögött: a választott alternatíva nem közömbös sem teljesítmény, sem költségek tekintetében. A kutatás keretében egy felsőoktatási célú chatbot esetén vizsgáltuk a legnépszerűbb zárt és nyílt súlyú LLM-ek API hívásokkal történő használatának minőségi és költség jellemzőit. A chatbotba az oktatók tölthetik fel tananyagaikat, a hallgatók kérdéseket tehetnek fel a modellnek, végül az oktatók megtekinthetik a tanulók kérdéseit és statisztikáit, így támogatva a személyre szabott tanulást. A vizsgálat rávilágított, hogy a költségek jelentős részét teszik ki a saját tudásbázis kiépítése és tesztelése. A lokális tudásbázis felépítése során az egyes modellek nagyon eltérő minőséget eredményeznek. A teszt és felhasználói kérdésekre adott válaszokban a költségek rövid távon nem jelentősek (a fejlesztéshez képest), ugyanakkor a nyílt súlyú modellek is jó eredményt tudnak biztosítani, megnyitva a költségek csökkentését hosszú távon.

**Kulcsszavak:** nyíltkérdés-megválaszolás, RAG, nagy nyelvi modell, LLM, generatív válaszolás, retriever-reader architektúra

## 1 Bevezetés: a felsőoktatás generatív MI kihívása

A nagy nyelvi modellek (LLM) és az azokra épülő különböző mesterséges intelligencia (MI) megoldások egyik legnépszerűbb alkalmazási területe a chatbotok. E chatbotok alkalmazása azonban számos kérdést vet fel, melyek egy része technikai problémákból (pl. hallucináció), más része minőségi hiányosságokból (pl. hibás, nem adekvát vagy terjedős válasz) ered, emellett felvetődnek adatvédelmi kockázatok is (pl. a belső adatok feltöltése egy külső chatbotba). Az általános generatív modellekre épülő botok nem minden esetben adnak pontos választ egy-egy felhasználói kérdésre, különösen, ha azok egy specifikus szakterületre vagy tartalomra irányulnak, így ilyen esetekben gyakrabban adnak helytelen választ vagy akár hallucinálhatnak. Ugyanakkor konkrét alkalmazások esetén a felhasználó számára fontos lehet a pontos és akár ellenőrizhető válasz (amihez forrás megjelölés is kapcsolódhat) (Berkecz és mtsai, 2024). E problémák el-lensúlyozására egyre többet figyelembe vett lehetséges megoldás a saját (ún. custom

vagy lokális) botok építése. A lehetőség előnye, hogy a tartalom közvetlenül az üzleti igényhez igazítható, és az üzemeltető nagyobb kontrollt gyakorolhat a tartalom és a lekérdezés felett (Gao és mtsai, 2023). Ugyanakkor kihívás, hogy a chatbotot ki kell fejleszteni – melyhez fejlesztési környezet és szaktudás kell –, illetve jól kell megválasztani a bot mögött működő nyelvi modelleket.

Egy saját bot kialakítása azonban nem csak fejlesztői kihívás, de tesztelni is kell a tartalmat, hisz ez az egyik legfőbb célja a bot önálló fejlesztésének, és figyelembe kell venni a fejlesztés és üzemeltetés költségeit is. Számos döntés kapcsolódik egy saját bot kialakításához, de ezek közül a legfontosabb, hogy a custom bot építése és használata során alkalmazott LLM milyen teljesítményre képes, és milyen várható költségekkel jár. A költségek tekintetében két lényegi megközelítés érhető el a piacon: az ún. nyílt súlyú modellek, melyek ingyenesen használhatók (akár netes eléréssel, akár lokális futtatással kapacitástól függően), míg a zárt modellek használatáért fizetni kell, ami lehet havi előfizetés (személyre szabottan) vagy használat (azaz token mennyiség) függő.

Technikai megoldás szempontjából egyre népszerűbbek az ún. Retrieval Augmented Generation (RAG) megoldások (Lewis és mtsai, 2020). Egy RAG-alapú rendszer jellemzően öt fő részből áll, egy adatbeviteli funkcióból (mely a szakterület vagy célspecifikus információkat gyűjti és vektorizálja), egy lokális tudásbázisból (mely a szakterületi anyagokból készült embedded vektorokat eltárolja), egy hozzá kapcsolódó lekérdezőből és kontextus azonosítóból (mely kettő együtt megkeresi a felhasználói kérdések alapján a választ és a releváns kontextust, majd azt jellemzően prompttá kiegészítve továbbadja a nyelvi modellnek), egy (generatív) LLM-ből (mely a kontextushoz illő végső válaszokat nyelvtanilag helyesen legenerálja). Kell még (vég)felhasználói interfész is (mely minimálisan egy prompt-felületet, de más funkciókat is nyújthat).

Egy custom chatbot kialakítása jól meghatározható területek és lekérdezések esetén lehet sikeres, melyre jó példa a felsőoktatás, ahol egy-egy tantárgy tananyaga általában jól körülhatárolható. Lehetnek elérhető tankönyvek és jegyzetek, vagy az oktató használhat cikkeket, előadás anyagokat és más forrásokat, melyek az adott tárgyban elvárt ismereteket, dedikált tudást tartalmaznak. Egy felsőoktatási kontextusban azonban a jelenlegi piaci körülmények között nem alternatíva az egyéni előfizetés (jellemzően a hallgatói létszám miatt, de a várható alacsony kihasználtság okán is), így tipikusan a tokenes (API) elérést vagy az ingyenes LLM-eket érdemes megvizsgálni. Természetesen a költségek mellett törekedni kell a minél pontosabb válaszadásra is.

Az e tanulmányban bemutatott kutatás fő célja annak vizsgálata, hogy egy felsőoktatásra tervezett, cél-orientált és tantárgyra szabható custom chatbot fejlesztése és üzemeltetése esetén milyen ár-érték arányt tudnak biztosítani a zárt modellek, és vajon a nyílt modellek fel tudják-e venni velük a versenyt.

## 2 RAG-alapú egyedi, testre szabott chatbot

### 2.1 A RAG megoldás és technikai komponensei

Egy cél-orientált, lokális chatbot lényege, hogy képes egy fókuszált területre vonatkozóan kérdéseket helyesen megválaszolni, és az eredményt kontextusban adott nyelven

megformált szöveggént visszaadni. Az egyik legnépszerűbb és gyorsan felkapottá vált technológia ilyen jellegű kérdés-felelet (Q&A) bot építésére a Retrieval Augmented Generation (Gao és mtsai, 2023). A RAG megközelítéssel csökkenthetők a költségek, saját tartalmakat lehet kereshetővé tenni és úgy csökkenthető a hallucináció jelensége, hogy a saját tartalmakat nem kell feltölteni a nyelvi modell szolgáltatójának felhőjébe (Shuster és mtsai, 2021). Ilyen RAG megoldással kereshetővé tehető saját anyagok, elsősorban például PDF-ben vagy szövegesen tárolt dokumentumok. Mivel a megoldás lokálisan üzemeltethető, így a funkciók is kontrollálhatók és a keresési adatok (promptok és megtalált válaszok leírása és statisztikái) is elérhetőek. Ezek a jellemzők aztán felhasználhatók mind a RAG továbbfejlesztésére, mind a tartalom javítására, mind a kérdezők promptolási technikáinak képzésére.

A bot kiépítése során számos döntést kell hozni, melyek a fenti modulok implementálására vonatkoznak amennyiben a lehetséges technikai megoldások közül választani kell. A két legfontosabb döntés a bejövő anyagok feldolgozását végző modell kiválasztása, illetve a lekérdezésre adott végső választ legeneráló LLM megválasztása.

## 2.2 A felsőoktatási chatbot mint teszt környezet és a használt technológiák

Az LLM-ek tesztelését és eredményeik összehasonlítását egy olyan felsőoktatási chatbot fejlesztése során végeztük el mely lehetővé teszi PDF fájlok feltöltését és támogatja a hallgatókat az adott tananyagra vonatkozó kérdéseik megválaszolásában. A felsőoktatási chatbot implementációja során fontos szempont volt, hogy a hallgatók a rendszer által feldolgozott tananyagok alapján pontos, hallucinációktól mentes válaszokat kapjanak kérdéseikre. További célkitűzés volt, hogy részletes statisztikákat biztosítson az oktatók számára, hogy a tananyagokat és az oktatást teszteredmények alapján továbbfejleszthessék. A rendszer ezért a RAG megoldásra épül: az oktató feltöltheti a tantárgyi anyagokat, valamint megtekintheti a tanulók kérdéseit és a statisztikákat, így támogatva a személyre szabható tanulást. A hallgatói oldal egy prompt felületből és tudás-tesztelési funkcióból áll.

A bot létrehozása során két fő részre bontottuk projektet: (1) a PDF anyagok betöltésének, feldolgozásának, vektorizálásának kezdeti lépésére, amelyből a lokális tudásbázis keletkezett; (2) a kérdés-válasz RAG módszer szerinti implementációjára és a megoldás tesztelésére. Mindkét lépésnél figyelembe vettük a költségeket is és ennek megfelelően határoztuk meg az optimális megoldást.

A teljes RAG folyamat teszteléséhez egy angol nyelvű, 649 oldalas tankönyvet használtunk (Laudon, 2021) PDF formátumban, amelyben 393 oldal szkennelve, képként szerepel. A könyv bemutatja, hogyan használják a vállalatok az információs rendszereket üzleti céljaik elérésére és problémák megoldására.

### 2.2.1 Adatbevitel és tárolás

Minden egyes oldal, amin ábra található nem csak az ábrát, grafikont tartalmazza képként, hanem az oldalon található többi szöveget. Ennek következtében az első technológiai kihívás a képként tárolt oldalak transzformálásánál jelentkezett. További problémát okozott, hogy a könyvben gyakoriak a magyarázó ábrák, diagramok és minden fejezet 1-2 szövegdobozban kiemelt esettanulmányt is tartalmaz, amelyek félbeszakították a törzsszöveget. Ebből következően nem lehetett alkalmazni az alapértelmezett,

mindenhol használt szöveg összefűzési módszert, ahol az oldal alján véget érő szöveghez a következő oldal tetején található szöveget kötjük, mivel akár a következő oldal közepén vagy 2-3 oldallal később folytatódhatott a félbehagyott törzsszöveg. Ezután szemantikailag koherens kisebb részekre kellett tovább osztani a feldolgozott szöveget, mivel a teljes könyv folyamatos elküldése jelentősen megdrágította volna a válaszadást.

A PDF dokumentumok betöltésére és pontos feldolgozására már régóta születtek megoldások, amelyek alapvetően megpróbálták valamilyen determinisztikus algoritmus alapján felismerni az egyes szövegblokkok sorrendjét, a formázási kiemelések alapján kitalálni a címeket és szintjüket Markdown formátumban, valamint képek esetén OCR technikát alkalmazva felismerni a karaktereket. Előbbire példa a PyMuPDF (McKie és Liu, 2016), Camelot (Camelot Developers, 2021), Tabula (Ariga, 2016), PdfMiner (Shinyama, 2019), míg utóbbira a legjobban elterjedt Tesseract rendszer (Smith, 2007). A PDF elsősorban azonban nyomtatási, megjelenítési formátum és emiatt a szemantikai egység erősen csorbul. A szöveg értelmezése nélkül sok esetben nem lehet megállapítani a sorrendet, kapcsolódást, valamint a diagramokról sem tudnak összefoglaló leírást készíteni ezek a hagyományos szoftverek. A legújabb LLM-k közül több már rendelkezik vision képességgel, amely kiválóan használható OCR kiváltására és annál magasabb szintű PDF feldolgozásra. A kutatás során 6 zárt (o1-preview, GPT-4-turbo, GPT-4o, GPT-4o-mini (OpenAI, 2024), Gemini PRO 1.5 (Gemini Team, 2024), Claude Sonnet 3.5 (Anthropic, 2024a)) és 7 nyílt modell (Mixtral-8x22B-Instruct (Jiang és mtsai, 2024; Mistral AI team, 2024), Qwen2-72B VL (Yang és mtsai, 2024), Meta-Llama 3.1 405B (Llama Team, 2024), Llama 3.2 11B Vision, Llama 3.2 90B Vision (Meta, 2024), Mini CPM-Llama3 (Pondhouse Data, 2024), Google Gemma 2-27B (Gemma Team, 2024)) használtunk úgy, hogy az egyes feladatokhoz ezekből választottunk.

Tesztjeink alapján mind a hagyományos PDF betöltőknél (PyMuPDF, PyMuPDF4llm), mind a Tesseract OCR rendszerénél jobb eredmény érhető el vision LLM alkalmazásával (amit a GPT-4o, Google Gemini PRO 1.5, Claude Sonnet 3.5, Qwen2 72B, Llama 3.2 11B Vision, Llama 3.2 90B Vision, Mini CPM-Llama3 tud). A nyelvi modellek pontosabbak a karakterfelismerésben főleg, ha diagramon, folyamatábrán szerepel a felirat, jobban követik a dokumentum struktúráját, képesek részletes leírást készíteni az ábrákról és alapvető kimeneti formátumuk a JSON struktúra, amely sokkal alkalmasabb további szoftveres feldolgozásra mint a Markdown. A tesztek során kipróbáltuk, hogy a PyMuPDF-el vagy Tesseracttal már felismertetett szöveget hozzáadjuk a prompthoz: a nagyobb modellek ezt teljesen figyelmen kívül hagyták, de a kis modelleket megzavarta és önismétlésbe kezdtek. A költségeket és feldolgozási időt figyelembe véve ezért a hagyományos rendszerek teljes elhagyása mellett döntöttünk a folyamatból. Az LLM-mel történő feldolgozás gyorsítását pedig aszinkron hívásokkal értük el ügyelve a szolgáltató (pl. OpenAI, Anthropic) által meghatározott lekérdezési korlátozásokra (rate limit).

A több oldalra szétszórt fejezetek problémáját felismerve a PDF oldalait egy általunk létrehozott JSON szerkezetbe töltöttük be, amely oldalanként tárolta a fejléct, lábléct, külön az esetlegesen felismert oldalszámot, valamint kép, táblázat és törzsszöveg típusú blokkokban az oldalon megtalálható szövegrészeket, ábrák leírását. Minden szövegblokk tartalmazott címet is, ahová opcionálisan a kép, táblázat vagy a kezdődő fejezet címét írta be az LLM kérésünkre. Ha egy szövegblokk nem rendelkezett címmel, de valamelyik előző oldalakról származó szöveg folytatása volt, akkor egy speciális cím

beírására kértük a nyelvi modellt. Ezután az előző oldal végéről blokkonként visszafelé haladó algoritmussal összekötöttük az aktuális oldal elejétől indulva ezeket a speciális címmel ellátott JSON részeket. Mivel ezt szekvenciálisan oldalanként elvégeztük, ezért a több oldalra szétszórt fejezeteket láncolt listába tudtuk szervezni és sikeresen összekötni. Például a 468. oldal alján kezdődő fejezethez hozzákapcsoltuk a 470. oldal alján folytatódó bekezdéseket és végül a 471. oldal közepén található befejező részt.

Az LLM-k a PDF feldolgozás során sok esetben kényszeresen befejezték a félbehagyott mondatokat. Erről a szokásról a hőmérséklet 0-ra állításával és specifikusan megfogalmazott mondatokkal szoktattuk le őket. Például: „You must extract all text from the page image exactly as it appears in the image. Including the text at the top of the page, which in most cases is a continuation of the previous page, hence the lack of a title”. A Google Gemini és a Claude Sonnet modellek esetén az oldalak tömeges feldolgozásakor újabb problémába ütköztünk, mivel ezek a modellek több oldal után hibával tértek vissza arra hivatkozva, hogy nem szöveges tartalmak szó szerinti visszaadására találták ki őket (Anthropic Privacy Center, 2024). Ennek valószínű oka az ellenük indított sajtóperektől való félelem lehetett. Ezeket az LLM-eket sajnos nem lehet OCR feladatokra használni. Az OpenAI GPT-4o pedig 1 oldal esetén tartalomszűrő (content filter) hibával tért vissza, ami azt jelenti, hogy veszélyesnek ítélte meg az arcfelismerő szoftverekről szóló tankönyvi esettanulmányt, amit a feldolgozandó PDF oldal tartalmazott. A prompt módosításával nem sikerült megkerülni a problémát ezért tartalék LLM (Llama 3.1 90B) bevetése mellett döntöttünk, amely képes volt feldolgozni az oldalt.

### 2.2.2 Releváns szövegrészek azonosítása felhasználói kérdéshez

A releváns szövegrészek keresése szempontjából a fejezetek kiváló kiindulási pontot biztosítanak, mert a megfelelően szerkesztett könyvek, cikkek egy fejezetben ugyanahhoz a témakörhöz kapcsolódó gondolatokat fogalmazzák meg, viszont továbbra is túl hosszúak általában kivéve a képeket és táblázatokat. Az előbbiekhöz az LLM által készített leírások hossza átlagosan 500–600 karakter, így alatta marad a tapasztalati alapon meghatározott 1000 karakteres határnak. A táblázatokat pedig nehéz lenne szétvágni szemantikusan, de a tapasztalatok alapján nem haladják meg ezek sem az előbb említett határt. Az 1000 karakternél hosszabb szöveges fejezeteket először rekurzív módon a bekezdések végét jelző dupla sortöréseknél, mondatok végén igyekeztünk szétvágni a jól ismert RecursiveCharacterTextSplitter módszerrel, de több esetben is egybetartozó szövegrészeket vágott ketté. Ezután egymás után következő 1 mondatban különböző 3 mondatos blokkok távolságát ( $1 - \cos$  hasonlóság) vizsgáltuk, és a távolságok 95%-nál nagyobb értékek esetén vágunk, de ebben az esetben teljesen kiegyensúlyozatlan lett az egyes eredményként kapott szövegrészek hossza. Végül ágens, azaz LLM-k segítségével végzett darabolás mellett döntöttünk, mert az LLM értelmezte a szöveget, és az egyes gondolati összefüggések végén határozta meg a vágási pontokat (itt a GPT-4o-t és a 4o-mini-t alkalmaztuk). Ezt erősítette, hogy sok esetben bekezdések végén darabolta a szöveget pedig a promptban ilyen irányú instrukció nem szerepelt, hanem csak a szöveg szemantikai egybetartozására vonatkozó utasításokkal láttuk el.

A legtöbb helyes válasz elérése szempontjából célszerű a legjobban illeszkedő, legnagyobb relevanciával rendelkező dokumentumrészeket átadni az LLM-nek, hogy választ ezekre a szövegrészekre alapozza és mást ne vegyen figyelembe, ne hallucináljon. Így jelentősen lehet csökkenteni a kitalált tények mennyiségét, mivel ha nem

határozható meg az átadott szövegrészek alapján a válasz, akkor a promptban arra utasítottuk az LLM-t, hogy térjen vissza a kontextus alapján nem adható válasz szöveggel. Kétféle egymástól teljesen eltérő keresési módszert alkalmaztunk, hogy megtaláljuk az összes kapcsolódó dokumentumrészt. A FAISS vektoradatbázisba betöltöttük a vektorokká alakított dokumentum darabokat és a felhasználó eredeti kérdését, valamint LLM-mel generált 3 db hasonló kérdést is vektorizáltunk (Ma és mtsai, 2023), majd koszinusz távolság alapján megkerestük a jelentéstartalom alapján közel álló szövegrészeket és a relevancia pontszámát is meghatároztuk. Ugyanezt végeztük el a BM25 módszerrel is, ami nem vektorizál, hanem szavakat, kifejezéseket keres és a kulcsszavak, valamint a dokumentumok hosszának figyelembevételével állapítja meg azok relevancia pontszámát (Robertson és Zaragoza, 2009). Az algoritmus különösen alkalmas olyan alkalmazásokhoz, amelyeknél a kulcsszavak megjelenése fontos szerepet játszik a releváns találatok kiválasztásában. A BM25 súlyozza a dokumentumban előforduló szavak gyakoriságát, ugyanakkor normalizálja a hosszabb dokumentumok értékelését, így elkerüli a hosszabb szövegek túlértékelését.

Ezután normalizáltuk a kapott relevancia pontszámokat és a FAISS, BM25 találati listákat a normalizált pontszámok alapján egyesítettük külön az eredeti felhasználói kérdésre és külön a 3db generált hasonló kérdésre. Így 4 relevancia listát kaptunk, amelyeket a Reciprocal Rank Fusion módszerrel újra rendeztünk (re-ranking) és összefűztünk. Ennek a végső listának a tartalmát kapta meg kontextusként a válaszadásra kiszemelt LLM a felhasználó eredeti kérdése mellé.

Kipróbáltuk még a hasonló kérdések generálása helyett, hogy a dokumentum fejezeteiről egy bekezdés hosszúságú összefoglalókat generálunk LLM segítségével és a hasonlósági kereséssel vektoradatbázisban megtalált releváns szövegrészekhez ezeket csatoljuk (Anthropic, 2024b). A célja ennek elsősorban az volt, hogy az adott szövegrész kapcsolódását lehessen látni a fejezet egészéhez, miközben nem növeljük jelentősen a bemeneti tokenek számát.

### 2.2.3 Felhasználói kérdések (promptok) tesztelése nyelvi modelleken

A kérdés-válasz rész tesztelése a felhasználóktól várható jellemző kérdések segítségével történt mely során az adott tantárgy tesztkérdéseit használtuk: 69 kérdéssel teszteltünk és értékeltünk 8 kiválasztott nyílt és zárt LLM-t (GPT-4-turbo, GPT-4o, GPT-4o-mini, Claude Sonnet 3.5, Mixtral-8x22B-Instruct, Qwen2-72B-Instruct, Meta-Llama-3.1-405B-Instruct, Google Gemma-2-27b-it), hogy a pontosság és költségek szempontjából optimális megoldást találjunk. Az értékelés során pontos egyezést kerestünk a feleletválasztós és igaz/hamis kérdésekre, míg a fennmaradó kérdéseket a GPT-4o-val értékeltük, összehasonlítva a várt és a kapott válaszokat 0-1 skálán. A végső eredményeknél a manuális ellenőrzés tapasztalatai szerint csak a 0,8 feletti értéket fogadtuk el, mert ezek mindig helyes válasznak bizonyultak a tesztesetekben.

## 3 A tesztek leírása és a vizsgált nagy nyelvi modellek költségei

A RAG implementálása után a kutatás célja annak kiderítése volt, hogyan lehet a költségeket csökkenteni egyrészt a tananyag feltöltési folyamat optimalizálásával (beleértve a keletkező tudásbázis építését és tesztelését LLM-mel), másrészt a hallgatói

kérdések megválaszolása során alkalmazott nagy nyelvi modell költséghatékony megválasztásával (mivel ma már elérhetők zárt-súlyú modellek különböző árú fizetős előfizetésekkel, de vannak nyílt-súlyú, azaz ingyenes megoldások is, melyek futtathatók lokálisan vagy a felhőben).

Az 1. táblázat mutatja az általunk tesztelt összes (12) modell jelenlegi árazását bemeneti és kimeneti tokenek számának függvényében 1 millió tokenre vetítve. A könyv 1 oldala feldolgozási árának meghatározásához figyelembe vettük az általunk készített, részletes instrukciókat tartalmazó OCR feldolgozási lépést leíró prompt hosszát (1096 token) a bemeneti tokenek árával, valamint kiszámoltuk a könyv oldalairól 200 dpi felbontással készített teljes oldalas képek (1654px x 2339px) bemeneti költségeit is. A kimeneti költségeket több oldal hosszának átlagolásával (891 token) határoztuk meg. A bemeneten megadott képek árazása felbontásfüggő és eltérő az egyes szolgáltatóknál, többen tokenekre konvertálják, míg az OpenAI, Google költségszámítást biztosít. Például GPT-4o 0,0028 USD-t kalkulál 1 oldal képére, míg a Claude  $\frac{(\text{szélesség} \times \text{magasság})}{750}$

képpel számolva 5158 tokenre konvertálja át a képet, amelyből a bemeneti tokenek árával felszorozva megkapjuk a képfeldolgozás költségét. Az ingyenes modelleket a Together.ai szolgáltatónál vettük igénybe, amely optimális ár-érték arányt nyújt a nyílt-súlyú modellek futtatásához. Az összes modellt a Mixtral 8x22B-től kezdve ennél a szolgáltatónál teszteltük. Ezekben az esetekben a kép költségét a  $\min\left(2, \max\left(\frac{\text{magasság}}{560}, 1\right)\right) \times \min\left(2, \max\left(\frac{\text{szélesség}}{560}, 1\right)\right) \times 1601$  kell számolni, ami 6404 plusz bemeneti tokent eredményez. Mivel 1 oldal költsége viszonylag kicsi, ezért kiszámoltuk a teljes 649 oldalas könyv feldolgozási összköltségét, ahol az OCR költségekhez még hozzáadtuk az ágens alapú feldaraboló prompt hosszát és annak kimenetét.

Az OCR lépést csak olyan modellekkel lehet megvalósítani, amelyek rendelkeznek vision képességekkel (lásd 1. táblázat 4. oszlop). A többi modellt csak a kérdés-válasz tesztelésnél lehetett felhasználni (de a teljes könyv költségnél az OCR helyett a Together.ai oldal árát vettük figyelembe, ezért dőlt betűvel jeleztük, hogy ez az érték csak elméleti). Látható, hogy a legdrágább megoldásnak a vision modellek közül a GPT 4 Turbo (52,51 USD) bizonyult, amelynek árát az OpenAI szándékosan magasán tartja, hogy inkább a GPT-4o felé terelje a felhasználókat a saját erőforrásainak védelme érdekében. A Claude Sonnet 3.5 (41,7 USD) lett a 2. helyezett, míg az OpenAI GPT-4o (18,71 USD) követi a 3. helyen. Mindkét modell 100%-os eredményt ért el az OCR tesztjeink során, ezért az ár alapján érdemesebb a GPT-4o-t választani.

A tokenek árazása tapasztalataink alapján erős korrelációt mutat a modellek méretével. Ebből következően a 8-13 milliárd paraméteres tartományban mozgó LLM-ek a legolcsóbbak, sőt a táblázatban nem szereplő MiniCPM Llama3 8B modellt lokális GPU hardveren futtattuk, így ingyenesnek tekinthető, ha a GPU beruházási költségeit nem vesszük figyelembe. Azonban hiába kerülnek jóval kevesebbe ezek a modellek, sajnos az eredményeik is sokkal pontatlanabbak, mint a nagyobb modellek esetén. Például a MiniCPM 59,28%-ban volt pontos, a Llama 3.2 11B szavak egyezése szempontjából ugyan 98%-ot ért el, de a JSON kimenetben megduplázta az egyik bekezdést egy további szövegblokkot hozzáadva az oldal tartalmához.

A várható költségekről előzetes becslést végeztünk a teljes választott könyv és 1000 kérdés-válasz párra (1. táblázat utolsó 2 oszlop). A kérdés-válasz párok esetén több kérdés bementét és kimenetét átlagoltuk a számításhoz. A kérdéshez hozzávettük a RAG módszer által megtalált releváns szövegdarabok hosszát is. Az 1000 kérdésre

vetített költség alapján a legdrágább az o1-preview lett, amelynek erőssége a bonyolultabb matematikai-logikai következtetések, így nem érdemes kontextus alapú kérdés megválaszolásra használni, hanem a GPT-4o sokkal költséghatékonyabb és gyorsabb opció helyette. A 2. helyen a GPT 4 turbo végzett, amit a Claude Sonnet 3.5, majd a Llama 3.1 405B követ.

1. táblázat: PDF feldolgozás és Q&amp;A költségei zárt és nyílt modellek esetén

(Minden költség USD)	Bemeneti token USD/1M	Kimeneti token USD/1M	1 oldal OCR költség	Kérdés- válasz költség	Teljes könyv 649 oldal	1000 kérdés- válasz
o1-preview	15	60	<i>nincs</i>	0,0311	94,32	31,08
GPT-4-turbo	10	30	0,0405	0,0189	52,51	18,93
GPT-4o	2,5	10	0,0144	0,0052	18,71	5,18
GPT-4o-mini	0,15	0,6	0,0062	0,0003	8,08	0,3108
Gemini PRO 1.5	1,25	5	0,0062	0,0026	7,99	2,59
Claude Sonnet 3.5	3	15	0,0321	0,0068	41,70	6,753
Mixtral-8x22B-Instr.	1,2	1,2	<i>nincs</i>	0,0018	13,07	1,842
Qwen2-72B VL	1,2	1,2	0,0101	0,0018	13,07	1,842
Meta-Llama 3.1 405B	3,5	3,5	<i>nincs</i>	0,0054	38,12	5,3725
Llama 3.2 11B Vision	0,18	0,18	0,0015	0,0003	1,96	0,2763
Llama 3.2 90B Vision	1,2	1,2	0,0101	0,0018	13,07	1,842
Google Gemma 2-27B	0,8	0,8	<i>nincs</i>	0,0012	8,71	1,228

A feldolgozás sebességét aszinkron hívások segítségével gyorsítottuk fel. Ez gyakorlatilag azt jelenti, hogy időben egyszerre küldjük el a feldolgozandó oldalakat az LLM API-nak, amelyek saját erőforrásaik védelme érdekében korlátozzák az egyidőben elküldhető API hívások számát. Általában a korlátok ötféleképpen mérhetők: percenkénti lekérdezések (RPM), napi lekérdezések (RPD), percenkénti tokenek (TPM), napi tokenek (TPD) és percenkénti képek (IPM). A felhasználás függvényében az elsőként elért limit alapján kerül alkalmazásra a korlátozás. Ezen kívül az API hívások használatára előre befizetett összegtől és a használt nyelvi modelltől is függ a korlátozások mértéke. A 2. szint (Tier 2) legtöbb esetben 50 USD befizetésével érhető el és néhány nap várakozási időt is megkövetelnek a szolgáltatók a szint eléréséhez. Az Anthropic Claude Sonnet 1000 RPM-t, 80 000 TPM-t és 2,5 millió TPD-t, míg az OpenAI GPT-4o 5000 RPM-t és 450000 TPM-t, Together.ai modellfüggetlenül pedig 1800 RPM-t és 250000 TPM-t engedélyez. Látható, hogy a Claude esetén a legszigorúbbak a korlátozások. A feldolgozás gyakorlati tapasztalatai alapján mind a Claude, mind a Together.ai esetén mesterséges várakoztatást (sleep) kellett alkalmaznunk ahhoz, hogy a korlátozást elkerüljük. A kódba automatikus háromszori újra próbálkozást építettünk be arra az esetre, ha az API hívás rate limit hibával tért vissza.

Összesen 69 hivatalos tesztadatbankból származó vizsgakérdést teszteltünk automatizáltan a választott tankönyv 11. fejezetéhez kapcsolódóan. A tesztadatbank tartalmazta a kérdésekhez tartozó hivatalos válaszokat is. A tesztadatbázis feleletválasztós (43 db), igaz/hamis (16 db) és esszé (10 db) kérdést tartalmazott, amelyeket pontos



minta alapú (pattern matching) egyezéssel, illetve a GPT-4o modell használatával értékeltünk ki. A GPT-4o az esszé kérdések helyességét 0-tól 1-ig terjedő skálán állapította meg kérésünkre (prompt), ahol a határt manuális ellenőrzés után 0,8-nál állapítottuk meg.

## 4 Eredmények

### 4.1 A tudásbázis-építés tapasztalatai

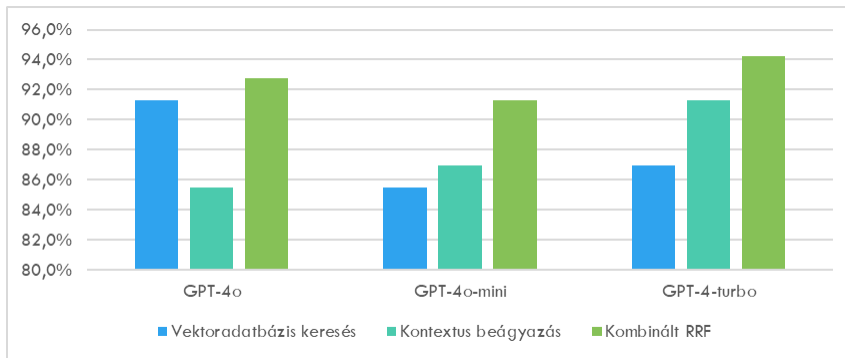
A tesztelt hét modell és egy OCR szoftver közül 100%-os pontossággal a GPT-4o és a Claude Sonnet 3.5 végzett az élen. Az utóbbi viszont kiesett egy, a tömeges feldolgozás során előkerült említett probléma miatt: a Claude Sonnet több oldal beküldése után a saját használati feltételeire hivatkozva megtagadja az oldalak OCR feldolgozását, mivel készítői szerint nem arra tervezték, hogy tartalmakat szó szerint visszaadjon („regurgitating”). A következő a nyílt Qwen2 72B lett 99,87%-kal, majd ezt követte az ugyancsak nyílt Llama 3.2 90B 98,96%-kal. Ez világosan mutatja, hogy az OCR feladatok területén a nyílt modellek felzárkóztak a zárt LLM-ekhez. A Llama 3.2 11B is ilyen eredményt ért el, de megduplázta az egyik bekezdést a JSON kimenetben, ami jelentősen rontja a későbbi használhatóságot. A Tesseract látszólag jó eredményt ért el, de a többi hasonló eredményhez képest sokkal több szót rosszul ismert fel vagy feleslegesen tett bele a dokumentumba, ami leértékeli az elért eredményt. A kis modellek érték el a legrosszabb eredményt. A lokálisan futtatott MiniCPM csak 59%-ban adott helyes eredményt.

Költségeket tekintve a kis modellek bizonyultak a legolcsóbbnak, de pontosságuk igen gyenge. A GPT-4o esetén 18,71 USD a teljes 649 oldalas könyv feldolgozása, míg a Qwen2 72B és a Llama 3.2 90B felhasználásával csak 13,07 USD. Ez az árkülönbség azonban nem jelentős, ha a pontosságot is figyelembe vesszük, mivel a GPT-4o 100%-ot ért el, míg az utóbbi 2 modell kevesebbet. A Google Gemini PRO 1.5 pontossága is hasonló a két nyílt modelléhez, és az ára jóval alacsonyabb, 7,99 USD, azonban a Claude Sonnethez hasonló hiba merült fel a tömeges OCR végrehajtása során, amelyet a Google tiltott recitationnak nevez.

A fentiek alapján a chatbot megvalósításánál választásunk a GPT-4o-ra esett, tartalomszűrő hiba esetén a Llama 3.2 90B tartalék LLM bevetésével.

### 4.2 Keresési módszerek és rangsorolás

Három módszert hasonlítottunk össze a 69 kérdésen a választott GPT-4 három változatán (lásd 1 ábra): vektoradatbázis keresés, kontextus beágyazás és kombinált megközelítés. Ezek közül a kombinált RRF (Cormack és mtsai, 2009) módszer teljesített a legjobban, különösen a GPT-4-turbo modell, amely 94,2%-os pontosságot ért el teljes egyezés és 98,55%-ot a 80%-os egyezés esetén.



1. ábra: A 3 keresési módszer tesztelése: GPT-4o, GPT-4o mini és GPT-4 turbo (saját munka)

### 4.3 Kérdés-válasz teszt eredmények

Az előző teszt eredményeképpen bebizonyosodott, hogy a legjobban teljesítő módszer a vektor adatbázis és BM25 kereséssel kombinált RRF algoritmus. Ezért a korábban kiválasztott 8 LLM-et egységesen teszteltük mind a 69 tesztkérdésre a kombinált RRF módszer felhasználásával. Csak egy kisebb modellt választottunk (Google Gemma 2 27B), mivel előzetes tapasztalataink alapján a kis paraméterszámmal rendelkező modellek nem teljesítettek jól. Mint a 2. táblázatban látható, ez itt is beigazolódott, mivel a Google Gemma eredménye (86,96%) az utolsó helyre volt elegendő, de azért nem sokkal maradt el a csúcsmodellek eredményétől. A legjobb eredményeket a paraméterek számának függvényében megvizsgálva észrevehető, hogy az bár alapvetően befolyásolja a teljesítményt, a jó eredmény nem csak ezen múlik. A Qwen2 72B nyílt modell (mely a közepes méretű LLM-ek táborához tartozik) végzett a 2. helyen (95,65%) a GPT-4-turbo (98,55%) mögött (mely egyes források szerint 8x222 milliárd paraméterrel dolgozik). Ehhez képest a 405 milliárd paraméteres Llama 3.1 már picivel rosszabbul teljesített.

A mintázat alapú egyezés a feleletválasztós és az igen/nem kérdések esetén volt alkalmazható. Ekkor a megfelelő betűjelet vagy az igen/nem értékeket kerestük a válaszban, és ezt hasonlítottuk össze a hivatalos eredménnyel. Az esszé kérdések így eleve kiestek ebből a körből, és meglepő módon a kérdésben egyértelműen nagybetűvel megjelölt válaszlehetőségek betűjelét sem mindig írták bele az LLM-ek válaszukba. A legtöbb esetben így is helyes volt a válasz, azonban ez már csak a GPT-4o-val elvégzett kiértékelés során derült ki. A modellt arra kértük megfelelő prompt, az eredeti kérdés, a kapott válasz és a hivatalos válasz megadásával, hogy hasonlítsa össze az eredményt a hivatalossal, és 0-1-ig terjedő skálán pontozza. A GPT-4o minden esetben 1 tizedesre kerekített pontszámokat (pl. 0,9, 0,8) adott, és az egyes különböző pontszámokhoz tartozó válasz darabszámokat külön is kigyűjtöttük.

2. táblázat: A tesztkérdések kiértékelése 4 zárt és 4 nyílt LLM-el tesztelve

FAISS és BM25 keresés RRF-el rangsorolva	Zárt				Nyílt			
	GPT-4-turbo	GPT-4o	GPT-4o-mini	Claude Sonnet 3.5	Qwen2-72B-Instruct	Meta-Llama-3.1-405B-Instruct	Mixtral-8x22B-Instruct	Google Gemma-2-27b-it
Minden kérdés	69	69	69	69	69	69	69	69
Mintázat alapú egyezés	54	51	51	37	43	50	38	38
GPT kiértékelt eredmények	15	18	18	32	26	19	31	31
Összes helyes eredmény (1.0 + mintázat)	65	64	63	62	62	61	59	56
Eredmény (%)	94,20	92,75	91,30	89,86	89,86	88,41	85,51	81,16
Eredmény (0.9)	2	0	0	0	2	1	2	1
Eredmény (0.8)	1	1	2	1	2	3	1	3
Eredmény (0.7)	0	1	1	0	0	0	1	2
Eredmény (0.5)	0	0	0	0	1	1	1	0
Eredmény (0.0)	1	3	2	6	2	3	5	7
Futásidő (sec)	127	127	127	815	375	821	489	635
<b>Eredmény (%) 0.9, 0.8 tartalmazva</b>	<b>98,55</b>	<b>94,20</b>	<b>94,20</b>	<b>91,30</b>	<b>95,65</b>	<b>94,20</b>	<b>89,86</b>	<b>86,96</b>
Minta összes (kiv. esszé %)	78,26 (91,53)	73,91 (86,44)	73,91 (86,44)	53,62 (62,71)	62,32 (72,88)	72,46 (84,75)	55,07 (64,41)	55,07 (64,41)

Bár csak a mintázat alapú egyezéseket figyelembe véve (táblázat utolsó sora) több modell rosszul teljesített, a GPT-4o-val végzett 1 pontot kapó kiértékeléseket is figyelembe véve már minden modell 81% felett teljesített (táblázat Eredmény % sora). Megvizsgálva a nyelvi modell által végzett automatikus kiértékelés pontszámait, megállapítottuk, hogy a 0,8 és 0,9 pontszámmal rendelkező válaszok is elfogadhatók jó megoldásnak. A legtöbb esetben azért vont le a GPT-4o 1 tizedet, mert a válasz terjengősebb volt, és olyan állítások is bekerültek, amelyek nem feltétlenül voltak szükségesek. Ellenben a 0,7 és 0,5 pontszámok esetében a válasz nem volt jó és több esetben rossz következtetéseket is tartalmazott. A 0 esetén leginkább a promptban megfogalmazott kérdésünknek tett eleget a vizsgált LLM, mert azt válaszolta, hogy a kapott kontextus alapján nem tudja a megoldást.

A 2. ábra a 2. táblázat 3 legfontosabb eredmény sorát ábrázolja. A kék vonal a mintázat alapú egyezés százalékos eredményeit mutatja, a világoszöld a mintázat mellett 1 pontot kapó eredményeket, míg a zöld a végső eredményt, amely a 0,8 és 0,9 pontszámokat kapó válaszokat is tartalmazza. Ez utóbbi alapján rendeztük csökkenő sorrendbe az egyes modelleket. A legjobb eredményt a GPT-4-turbo érte el 98,55%-kal, ami azért érdekes, mert a GPT-4o fejlettebb modellnek számít. A 2. helyen egy nyílt modellt találunk, a Qwen2 72B-t 95,65%-kal, amelynek az előzőekben ismertetett token költsége is nagyon alacsony.



2. ábra: A legjobb, RRF alapú módszer tesztelése a 8 választott LLM rendszeren (saját munka)

## 5 Tanulságok összefoglalása

A felsőoktatási, RAG módszerekkel testre szabott chatbot esetében részletes költség- és teljesítményelemzést végeztünk többféle méretű nyílt és zárt súlyú LLM összehasonlításával. A 649 oldalas, képeket is tartalmazó PDF feldolgozási folyamatban 100%-os karakterfelismerési pontosság legkedvezőbben 18,71\$-ért érhető el (GPT-4o). A GPT-4o-mini vagy open-source modellek használatával ez az összeg tovább csökkenthető. Az OCR jellegű felhasználásban a nyílt modellekre mindig támaszkodhatunk, mivel a szabály alapú, standard feldolgozóknál pontosabb, strukturáltabb eredményt adnak és zárt társaik a tartalmi szűrések miatt néhány esetben nem végzik el a feladatot. A tartalomszűrő intézkedés miatt a 13,07\$-os nyílt súlyú Llama 3.2 Vision 90B a tartalék modell a TogetherAI szolgáltatást használva, a teszt alapján 98,96%-os pontosságot ér el.

A kérdés-válasz feladatban a legjobb RAG módszert 8 LLM-re teszteltük 69 hivatalos tesztkérdéssel. Ezek közül a GPT-4-turbo 98,55%-ban helyes válaszokat adott, a legjobb pontosságot elérve. Az eredmények az mutatják, hogy a nyílt súlyú rendszerek elérhetik a SOTA modellek szintjét a RAG alapú kérdések megválaszolásában (megfelelően megválasztott tudástárolási megoldások és lekérdezések alkalmazásával), mivel a Qwen2-72B 95,65%-os pontossága felülmúlta a GPT-4o (94,2%) és Claude Sonnet 3.5 (91,3%) megoldását. A Llama 3.1 405B egy szinten teljesített a legújabb OpenAI modellekkel. Az 1000 kérdésnél fellépő költségeket figyelembe véve a GPT-4o-mini a legjobb költséghatékony választás, míg a Qwen2-72B a nagyobb zárt modellek költségei alatt biztosít jobb teljesítményt. Mivel zárt modellek esetén a kimeneti token drágább, így érdemes arra optimalizálni.

## Köszönetnyilvánítás

Az EKOP-24-2-003 azonosítószámú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási, Fejlesztési és Innovációs Alapból nyújtott támogatásával a 2024/2025 tanévre meghirdetett Egyetemi Kutatói Ösztöndíj Programjának a finanszírozásában valósult meg.

A chatbot fejlesztését és szerver szolgáltatását a Webra Kft. ([webra.hu](http://webra.hu)) támogatta.

## Bibliográfia

- Anthropic. (2024a). Claude 3.5 Sonnet Model Card Addendum. <https://paperswithcode.com/paper/claude-3-5-sonnet-model-card-addendum>
- Ariga M. 2016. Tabula. <https://github.com/chezou/tabula-py>. Letöltve: 2024.12.23
- Berkecz, P., Zombori, T., Banga, G., Szabó, G., Szántó, Zs., Novák, A. & Farkas, R. SHunQA: egy nyíltkérdés-megválaszoló rendszer. XX. Magyar Számítógépes Nyelvészeti Konferencia online kiadás, Magyarország, Szegedi Tudományegyetem (2024) pp. 73-84., 12 p.
- Camelot Developers. 2021. Camelot. <https://github.com/camelot-dev/camelot>. Letöltve: 2024.12.23
- Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009*. <https://doi.org/10.1145/1571941.1572114>
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997. [arXiv.org. https://arxiv.org/abs/2312.10997](https://arxiv.org/abs/2312.10997)
- Gemini Team Google. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. [arXiv.org. https://doi.org/10.48550/arxiv.2403.05530](https://doi.org/10.48550/arxiv.2403.05530)
- Gemma Team, Google DeepMind. (2024). Gemma 2: Improving Open Language Models at a Practical Size. [arXiv.org. https://arxiv.org/abs/2408.00118v3](https://arxiv.org/abs/2408.00118v3)
- Introducing Contextual Retrieval. (2024b). Anthropic.com. <https://www.anthropic.com/news/contextual-retrieval>
- Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., Casas, D.D., Hanna, E.B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L.R., Saulnier, L., Lachaux, M., Stock, P., Subramanian, S., Yang, S., ... Sayed, W.E. (2024). Mixtral of Experts. [arXiv, abs/2401.04088. arXiv.org. https://arxiv.org/abs/2401.04088](https://arxiv.org/abs/2401.04088)
- Laudon, K. és Laudon, J. (2021). *Management Information Systems: Managing the Digital Firm*. 17th (Global) edition. Pearson Education. Harlow UK.
- Lewis, P. S. H., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Neural Information Processing Systems*, 33, 9459–9474.
- Llama Team. (2024). The Llama 3 Herd of Models. [arXiv.org. https://arxiv.org/abs/2407.21783v2](https://arxiv.org/abs/2407.21783v2)
- LLM Document Extraction: How to use AI to get structured data from legacy documents. (2021). Pondhouse-Data.com. <https://www.pondhouse-data.com/blog/document-extraction-with-llms>
- Ma, X., Gong, Y., He, P., Zhao, H., & Duan, N. (2023). Query Rewriting in Retrieval-Augmented Large Language Models. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. <https://doi.org/10.18653/v1/2023.emnlp-main.322>

- McKie, J. X.; és Liu, R. 2016. PyMuPDF. <https://github.com/pymupdf/PyMuPDF>. Letöltve: 2024.12.23
- Meta. (2024). Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta.com. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>
- Mistral AI team. (2024). Cheaper, Better, Faster, Stronger. Mistral.ai. <https://mistral.ai/news/mixtral-8x22b/>
- OpenAI. (2024). GPT-4o System Card. *arXiv.org*. <https://arxiv.org/abs/2410.21276>
- Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389. <https://doi.org/10.1561/1500000019>
- Shinyama Y. 2019. PDFMiner. <https://github.com/euske/pdfminer>. Letöltve: 2024.12.23
- Shuster, K., Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). Retrieval Augmentation Reduces Hallucination in Conversation. *Conference on Empirical Methods in Natural Language Processing*.
- Smith, R. (2007). An overview of the Tesseract OCR engine. *Proceedings of the International Conference on Document Analysis and Recognition*, 629–633. <https://doi.org/10.1109/icdar.2007.4376991>
- Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., ... Fan, Z. (2024). Qwen2 Technical Report. *arXiv.org*. <https://arxiv.org/abs/2407.10671v4>
- Why am I receiving an “Output blocked by content filtering policy” error? Anthropic Privacy Center. (2024). Anthropic.com. <https://privacy.anthropic.com/en/articles/10023638-why-am-i-receiving-an-output-blocked-by-content-filtering-policy-error>

## Magyar szerkeztár demó

Sass Bálint<sup>1</sup>, Indig Balázs<sup>2</sup>, Kalivoda Ágnes<sup>1</sup>, Lagos Cortes Mátyás<sup>3</sup>, Lipp Veronika<sup>1</sup>, Makrai Márton<sup>4</sup>, Pethő Gergely<sup>5</sup>, Simon László<sup>1</sup>, Vadász Noémi<sup>6</sup>

<sup>1</sup>HUN-REN NYTK Lexikológiai Intézet, <sup>2</sup>ELTE IK Mesterséges Intelligencia Tanszék, <sup>3</sup>HUN-REN NYTK Általános és Magyar Nyelvészeti Intézet, <sup>4</sup>HUN-REN TTK, <sup>5</sup>Debreceni Egyetem Egy Egészség Intézet és Regensburgi Egyetem Romanisztikai Intézet, <sup>6</sup>Magyar Nemzeti Levéltár

sass.balint@nytud.hu, indig.balazs@inf.elte.hu, kalivoda.agnes@nytud.hu, lagos.matyas@nytud.hu, lipp.veronika@nytud.hu, makrai.marton@ttk.hu, petho.gergely@etk.unideb.hu, simon.laszlo@nytud.hu, vadasz.noemi@mnl.gov.hu

**Kivonat** A konstruktikonok (szerkeztárak) egy nyelv konstrukcióit veszik számba. Az utóbbi években számos nyelvre (például: svéd Lyngfelt és mtsai (2018b), orosz Bast és mtsai (2021), észt Vainik és mtsai (2024)) készülnek ilyen lexikai erőforrások, ebbe a trendbe illeszkedik a magyar változat. A Magyar szerkeztár első verziójának a működését és a <https://szerkeztar.hu> címen szabadon elérhető felületét ismertetjük laptopos bemutató formájában.

**Kulcsszavak:** konstrukció, szerkezet, konstruktikon, szerkeztár

## 1. Háttér

### 1.1. A szavak is konstrukciók

A konstrukciós nyelvtan (Goldberg, 2006) szerint minden olyan nyelvi egység konstrukció (szerkezet), ami felfogható forma–jelentés (vagy forma–funkció) párként, amiben a jelentés vagy funkció nem kikövetkeztethető, hanem megtanulandó. Ide tartoznak mindenekelőtt a fix és szabad elemekkel is bíró összetett egységek, mint például az *áll valamiből*, a *kísérletet tesz valamire* vagy az *őrizetbe vesz valakit valami miatt*. De ide tartoznak a skála egyik oldalán az absztrakt nyelvtani szabályok (konstrukciós sémák (Diessel, 2023)), a skála másik oldalán pedig az egyes szavak, sőt morfémák is. Ma is vannak egymással szembenálló vélemények arról, hogy a szavakat konstrukcióknak tekintsük-e. Hilpert (2014, 2. oldal) híres megfogalmazása szerint „nyelvi tudásunk semmi másból nem áll, kizárólag konstrukciókból” („a person’s knowledge of language consists of nothing but constructions”), tehát a szavak is azok. Janda és mtsai (2020) vagy Haspelmath (2023) viszont amellet érvel, hogy csak a fenti összetett konstrukciókra érdemes koncentrálni.

A Magyar szerkeztár (röviden: Szerkeztár) koncepciójának kiindulópontja, hogy *minden* konstrukciót számba vegyen a morfémáktól a nyelvtani szabályokig. Éppen az a nagy előnye a konstrukciós nyelvtani keretnek, hogy ezt lehetővé teszi, hogy általa a nyelv egésze írható le. Nincs elkülönülés az általában

„szótár”-nak és „nyelvtan”-nak nevezett részek között, mivel egy skálára helyezve azonos típusú entitásként kezel minden nyelvi jelenséget a teljesen konkrétól a teljesen absztrakttig: konstrukcióként. Ennek köszönhetően egy lexikai erőforráson belül ábrázolni tudjuk a nyelvi egységek hálózatában lévő összes kapcsolatot, és a felhasználónak is kényelmes, hogy nem kell különböző lexikai erőforrások, szótárak között váltania, mert a szerkezetárban megtalál mindent.

Ahogy azt Janda és mtsai (2020) is megállapítják, a nyelv ilyenfajta teljes leírására való törekvés joggal tűnik utópisztikusnak. A Magyar szerkezetár projekt is annak tudatában tűzi ki ezt a célt, hogy csak valamilyen mértékű megközelítésére lesz képes.

## 1.2. Szótárból szerkezetár

A szerkezetárak sok esetben az adott nyelv FrameNet-jén (Lyngfelt és mtsai, 2018a) alapulnak. Magyar FrameNet hiányában, illetve mivel a mi megközelítésünk fókusza a konstruktikon felépítése, hálózatos struktúrája, azt az utat választottuk, hogy egy meglévő szótárból kiindulva alakítjuk ki a Magyar szerkezetár tartalmát.

Ez olyan módon történik, hogy a szótári szócikkek mélyén megtalálható kifejezésként vagy szókapcsolatként (azaz speciális jelentéssel bíró egységként) megjelölt szerkezeteket önálló egységekként kiemeljük, és a továbbiakban a szavakkal, illetve a többi konstrukcióval egyenrangú módon kezeljük (1. ábra). A *címszó* terminus mintájára nevezhetjük ezeket az önálló egységeket *címszerkezet*nek. Kiinduló szótárunk az Értelmező kéziszótár (Pusztai, 2003), a fentieknek megfelelően lesz az eredetileg a *fehér* címszó belsejében található *fehér asztal*, *fehér holló* stb. kifejezésekből saját jogú egység.

Azt a döntést tehát, hogy mit tekintünk szerkezetnek, a kiinduló szótárra hagyjuk, a szótárban manifesztálódó döntést elfogadjuk. Hangsúlyozzuk, hogy egyfelől a Szerkezetárban jelenleg szereplő egyszavas és többszavas egységek egyaránt kizárólag a kiindulásul szolgáló Értelmező kéziszótárból származnak; másfelől pedig minden ilyen egységet elfogadunk speciális jelentésűnek, azaz konstrukciónak, akkor is, ha merülnek fel ezzel kapcsolatban kétségek, mint az *asztalfiók* esetében. A fentieknek köszönhetően minden konstrukcióhoz eleve megvan a hozzá tartozó szótári definíció, és van egy bár nem tökéletes, de megalapozott érvünk arra, hogy mit tekintünk konstrukciónak.

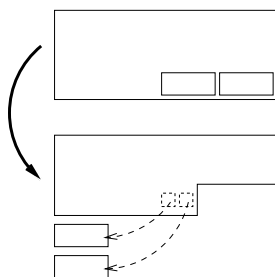
A fenti két fejezet részletesebb kifejtését az (Sass, 2023) cikkben olvashatjuk.

## 1.3. Általánosított online felület

A szerkezetárak általában bizonyos szintű nyelvészeti ismeret meglétét feltételezik a felhasználó részéről (vö. Lyngfelt és mtsai, 2018b, 92. oldal). A Magyar szerkezetár célközönsége ezzel szemben a laikusokat is magában foglalja: az a cél, hogy bárki tudja használni akár egyfajta egynyelvű szótárként, akár a nyelvben lévő hálózatok felfedezésére, mindenfajta konstrukciós nyelvtani tudás nélkül.

Ennek megfelelően alakítjuk ki a felületet: a felhasználó egy szövegdobozt lát, ebbe beírhat egy szót, néhány szót vagy egy rövid magyar szövegrészletet, és





1. ábra: Az eredeti szócikk (fent) mélyén található szerkezeteket önálló egységekként kiemeljük, belőlük az eredeti szócikkkel azonos módon kezelendő egységeket képezünk (lent). A szótárbeli egy darab szócikkból a szerkezetárban három egyenrangú konstrukció lett.

a rendszer feladata, hogy ebből az *összes* szerkezetet kihüvelyezze, és egyenként, kapcsolataikkal együtt bemutassa a felhasználó számára. Ez az újfajta felület több szempontból is a hagyományos online szótári felület általánosításának tekinthető az alábbiak miatt.

Egy online szótári felület alapvetően egy egyszavas inputot vár, azt szótári címszónak tekintve kikeresi az adatbázisból, és bemutatja. Az elvárást, hogy az input kanonikus szótári alak legyen, annyival szokták megkönnyíteni a szótárak, hogy a rendhagyó alakokat külön kezelik. A szabályos ragozott alakokat viszont nem, vagy nem teljes lefedettséggel, ezért azokra sokszor nem kapunk választ. Azt sem szokták megtenni az online szótárak, hogy több címszó beírása esetén mindet külön-külön visszaadják eredményként (vö. *black dog* az OALD-ben (Oxford University Press, 2023)).

A Szerkezetár felületén valamilyen módon nyilván meg kell engedni többszavas bemenetet, tekintve, hogy számos konstrukció többszavas. Mivel nem szeretnénk elvárni a felhasználoktól valamiféle kanonikus alak ismeretét – a konstrukciók esetleges kanonikus alakjának definiálása eleve nem triviális kérdés –, a Magyar szerkezetár tetszőleges szöveges inputot megenged. Egy ilyen, bemenetként megadott rövid szövegrészletben viszont több szerkezet is megjelenhet egymással összefonódva. Ezt az esetet is kezeljük: bemutatjuk az összes megjelenő konstrukciót, valamint kezeljük a szabályos alakokat is, felfedve a bennük rejlő konstrukciókat (ld. a (3) példát a 3. részben).

Az általánosítás tehát abban ragadható meg, hogy egyrészt nem várunk el kanonikus alakot, másrészt megengedünk többszavas bemenetet, harmadrészt az inputban megjelenő összes konstrukciót kezeljük.

## 2. Hogyan működik?

Jelenleg az egyszavas és a többszavas bemenetek esetében két külön eljárás működik az összes konstrukció azonosítása érdekében. Az előbbihez tokenizálást (Mittelholcz, 2017) és morfológiai elemzést (Orosz és Novák, 2013; Novák és mt-

sai, 2016) használunk, utóbbihoz szintaktikai elemzést is (Straka és mtsai, 2016), mindezt az *e-magyar* (Indig és mtsai, 2019) rendszerben integrálva.

## 2.1. Egyszavas egységek

Egy önálló, speciális jelentéssel bíró toldalékolt szó egy konstrukció (például: *valójában*), egy kompozicionális jelentésű toldalékolt szó viszont több konstrukcióból áll: minden morféma saját funkcióval bíró külön konstrukció (például: *asztalában*). Az egyszavas egységek kezelését egy elemekre bontásból és összevonásból álló kétlépéses algoritmus valósítja meg. Először morfológiai elemzéssel morfémákra bontjuk a szóalakot, majd az egymás melletti morfémák potenciális összevonásával alakítjuk ki a végleges konstrukciókat. Két vagy több egymás melletti morfémát természetesen akkor tudunk összevonni, ha a szerkezetárban megtalálható konstrukcióként az összevont forma. Az összevonás balról jobbra, a lehető leghosszabb összevont egység kialakításával történik. Az ún. „*szavankénti*” algoritmus lépései tehát a következők:

1. ha a lekérdezés megtalálható a szerkezetárban, akkor konstrukció, különben
2. morfológiai elemzést végzünk;
3. a morfémákat lehetőség szerint konstrukciókká vonjuk össze;
4. a megmaradó morfémák is önálló konstrukciók lesznek.

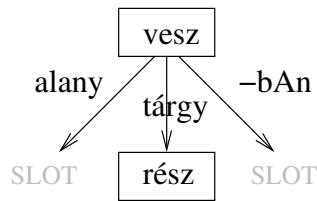
Az *asztalfiókba* (ld. az (5) példát a 3. részben) bemenetet a morfológiai elemzés három elemre bontja, de mivel az eredeti szótárban – és ennek nyomán a Szerkezetárban – önállóan, speciális jelentéssel bíró egységként szerepel az *asztalfiók*, ezért az *asztal* és *fiók* elemeket összevonva ezt és a *-ba/-be* ragot kapjuk meg eredményül.

## 2.2. Többszavas egységek

A többszavas – sok esetben nem fix, nem rögzített szórendű vagy nem folytonos – konstrukciókat függőségi fákhhoz hasonló struktúrák formájában tároljuk az adatbázisban, és ugyanilyen függőségi elemzésnek vetjük alá az input szöveget. Kulcsfontosságú, hogy a Szerkezetárban nyilvántartott szerkezetekben a bennük lévő szabad slotokat beazonosítsuk. Ez jelenleg annyit jelent, hogy a konstrukciók eredeti szótári bejegyzéseiben lévő *valaki/valami* elemet a feldolgozás során szabad slotná alakítjuk. A függőségi elemzés *nmod:obl* (oblikvusz) címke néven egy kalap alá veszi az összes esetraggal megjelenő bővítményt. Számunkra elegendhetetlen ennek felosztása az egyes esetragoknak megfelelően. A céljainknak megfelelő, fentiek szerint módosított függőségi fákat slot-filler fákna nevezünk. Így lesz az eredeti *részt vesz vmiben* alakból a 2. ábrán látható reprezentáció.

Miután közös reprezentációban rendelkezésre állnak a címszerkezetek és az input szöveg, az a feladat, hogy e kettőt egymásra illesszük. Ez a következő ún. „*fatöredék*” rekurzív algoritmussal történik:

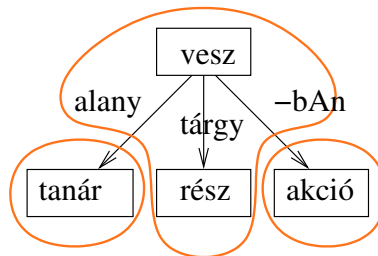
1. végighaladva az adatbázison megnézzük, hogy



2. ábra: A *részt vesz valamiben* szerkezet slot-filler fa reprezentációja a Szerkezetárban.

2. mely szerkezetek illeszkednek az input szöveg fájának gyökerére;
3. ezek közül kiválasztjuk a leghosszabbat (a legtöbb csomópontból állót);
4. a kiválasztottat kivonjuk (eltávolítjuk) a fából;
5. a fa így kisebb megmaradó részekre esik szét;
6. ezeken a részekén rekurzívan újból lefuttatjuk az algoritmust.

A második pontbeli illeszkedés azt jelenti, hogy a fix elemeknek egyezni kell, a címszerkezetek szabad slotjainak megfelelő helyen pedig bármi szerepelhet az inputban. Az algoritmus működésére egy példa a 3. ábrán látható. Látjuk, hogy az algoritmus – helyesen – nem fogja eredményül adni sem a tárgyas *vesz* igét, sem a *rész* szót. Előbbit azért nem, mert van hosszabb illeszkedő szerkezet, utóbbi pedig azért nem, mert ez a bizonyos hosszabb illeszkedő szerkezet már lefedte. Az eljárás elnevezése onnan ered, hogy az egyes szerkezetek végeredményben az eredeti bemeneti fa-reprezentáció részeinek, töredékeinek felelnek meg.



3. ábra: Szerkezetek az *a tanár részt vesz az akcióban* bemenetben. A gyökérnél illeszkedő leghosszabb címszerkezet éppen a 2. ábrán látható szerkezet. A névelőket az egyszerűség kedvéért nem tüntettük fel.

A működés részletesebb bemutatása a (Sass, 2023, 2024) cikkekben olvasható. Azt gondoljuk, hogy a konstrukciók túlnyomó többsége kezelhető a fenti módon. A gyökérben slotot tartalmazó, valamint a függőségi fák formájában esetleg egyáltalán nem megragadható konstrukciók kezelése a jövő feladata.

Megjegyezzük, hogy a fenti két algoritmus struktúrája absztrakt szinten meglehetősen hasonló: mindkettőben van egy elemekre bontó és egy összevonó lépés. A fő különbség, hogy ez az első esetben egy lineáris, a második esetben pedig egy fa-struktúrára történik. A jövőben ez lehetőséget adhat egy egyesített algoritmus kialakítására, ami közös reprezentációban kezeli a morfológiát és a szintaxist.

### 3. Demó

Az alábbiakban példákon keresztül mutatjuk be a Magyar szerkezetár működését.

- (1) *asztal*
- (2) *asztalos*
- (3) *asztalokra*
- (4) *faasztal*

Az (1-2) példák önálló szócikként szerepelnek az eredeti szótárban, egyszerűen az eredeti szótárból származó szócikküket kapjuk eredményül. Utóbbi esetben morfológiai elemzés és összevonás után (vö. a „szavankénti” algoritmus a 2.1. részben). A (3) példa természetesen nem szerepel az eredeti szótárban, a morfológiai elemzésnek köszönhetően a benne lévő három morféma három külön konstrukcióként – három önálló funkcióval bíró egységként – fog megjelenni az eredményben: *asztal* + *-k* + *-rA*. A (4) esetében hasonlóan: *fa* + *asztal*.

- (5) *asztalfiókba*

Az (5) példát a morfológiai elemzés három elemre bontja, de mivel az *asztalfiók* az eredeti szótárban önálló szócikk, ezért az első két elemet összevonjuk, így az eredmény két konstrukció: *asztalfiók* + *-bA*.

- (6) *fehér asztal*
- (7) *sárga asztal*

Említettük, hogy ami a kiinduló szótárban kifejezésként vagy szókapcsolatként szerepel, azt elfogadjuk konstrukciónak. A (6) az eredeti szótárban egy önálló jelentéssel bíró konstrukció, amit kiemeltünk (vö. 1.2), jelentése: ‘étkezésre terített asztal’; (7) ezzel szemben két konstrukció szabad kombinációja: *sárga* + *asztal*. Ez tehát két, formailag azonos, de konstrukciók szempontjából eltérő példa.

Az alábbi példákon érdemes összevetni az eredeti szótár és a Szerkezetár működését (vö. 1.3). Elérhetőségek a 5. részben.

- (8) *avokádó*
- (9) *fűbe harap*
- (10) *fűbe*

(11) *fű*

(12) *aki fűbe harap*

(13) *nem harapott fűbe*

(14) *fűvet harap*

Az eredeti szótárban csak a (8)-ra és a (11)-re kapunk eredményt, a Szerkezetárban mindre. Az előbbi esetében teljesen azonos az kimenet, utóbbi esetében viszont látjuk, hogy a szerkezeteket – köztük (9)-et – önálló egységekként kiemeltük, és csak kereszthivatkozást hagytunk a helyükön (vö. 1. ábra). A (9) természetesen egy konstrukció, szemben a (10)-zel, ami kettő, és a (14)-gyel, ami három. A (12) mutatja, hogy szöveg részeként is felismerjük a szerkezeteket, a (13) pedig azt, hogy ragozott alakban, más szórenddel és nem folytonos formában is. Utóbbihoz – és az alább következő összes példához – már szükséges 2.2. részben ismertetett „fatöredék” algoritmus.

(15) *a tanár részt vesz az akcióban*

(16) *a tanár kenyeret vesz az üzletben*

(17) *az alapító tag részt vesz az akcióban*

(18) *az okos gyerek részt vesz az akcióban*

A fentebb (vö. 3. ábra) ismertetett példát látjuk (15)-ben. A „fatöredék” algoritmusnak köszönhetően felismerjük a *részt vesz valamiben* szerkezetet. A (16) formailag azonos, de itt minden elem külön konstrukció lesz. Hasonlóan a (6-7) példákhoz, az *alapító tag* önálló egység az eredeti szótárban, tehát számunkra egy konstrukció, szemben az *okos gyerek* jelzős főnévvel, ami kettő. Ennek megfelelően ad eredményt a rendszer a (17-18) példákra. A (17) példa azt is demonstrálja, hogy a „fatöredék” algoritmus a rekurzivitásának köszönhetően a slot-filler fa alsóbb részein is képes a többszavas konstrukciók feltárására, egyben példa a konstrukcióknak az 1.2. részben említett összefonódására.

(19) *tanár részt vett az akcióban*

(20) *tanár vesz részt az akcióban*

(21) *tanár vesz az akcióban részt*

(22) *tanár részt vesz a munkában*

(23) *tanár kollégáival részt vesz az akcióban*

A (19-21) példákban látszik, hogy a különböző ragozott és más szórendű változatokat is felismeri az algoritmus, a (22-23) példák pedig azt mutatják, hogy természetesen az eltérő szabad elemek és az esetleges további bővítmények sem zavarják a felismerést.

## 4. Tervek

A számos továbbfejlesztési lehetőség közül az alábbiakat tervezzük megvalósítani először.

A kiinduló szótárnak köszönhetően szavakat és morfémákat, a kiemelésnek köszönhetően pedig fix és szabad elemekkel is bíró összetett szerkezeteket tartalmaz a Szerkezzettár. Viszont egyelőre hiányoznak az absztrakt nyelvtani konstrukciók, mint például a jelző+főnév, a birtokos+birtok vagy az ige+tárgy. Tervben van az ezekkel – vagy az ezek közül a leggyakoribbakkal – való kiegészítés. Ehhez a feladathoz hasznos forrás lehet a Croft (2022)-es kötetének végén található nyelvfüggetlennek és teljesnek szánt gyűjtemény.

Látjuk, hogy egyszavas input esetén, amennyiben ragozott a szó, külön szerkezetekként megkapjuk az egyes morfémákat (ld. a (3) példát), viszont ha összetett szerkezetet azonosítunk be, akkor nem (ld. a (13) és (19) példát). Tervben van ennek a megoldása. A kulcs minden bizonnyal az összetett szerkezet gyökerében lévő szónak a morfológiai elemzése lesz, mivel általánosnak tűnik, hogy az összetett szerkezet morfológiai jellegzetességeit a gyökér hordozza. Ez lényegében a 2. fejezet végén említett algoritmus megvalósítását jelentené.

A kiinduló szótárból első körben az XML struktúrában kifejezésként vagy szókapcsolatként megjelölt elemeket emeltük ki konstrukcióként. Az a tapasztalat, hogy a szótárban példaként megjelenő egységek is legtöbb esetben konstrukciók. Példában találjuk meg többek között a *kísérletet tesz vmire* és a *szívesen vesz* konstrukciót. Ezt meg kell vizsgálni, és ha bebizonyosodik, hogy valóban így van, akkor érdemes a példákat is hozzávenni teljes jogú szerkezetként a Szerkezzettár adatbázisához, jelentősen növelve ezzel lefedettségét.

## 5. Hozzáférhetőség

A felület szabadon elérhető a <https://szerkezzettar.hu> címen minden érdeklődő számára. A működés tanulmányozása során érdemes összevetni a Szerkezzettár által adott kimenetet azzal, amit az ÉKSz. (Pusztai, 2003) <https://eksz.nytud.hu> címen szintén szabadon elérhető online felületén kapunk.

Reményeink szerint ez az újfajta szótárszerű, szótárhelyettesítő lexikai erőforrás hasznos lesz a nyelvészeti vizsgálatokon túl a magyart mint idegen nyelvet tanulók számára, illetve az iskolai magyarórákon is, határon innen és túl.

## 6. Köszönetnyilvánítás

Az ebben a publikációban ismertetett kutatást a Nemzeti Kutatási, Fejlesztési és Innovációs Hitaval (NKFIH) OTKA (K 147452) projektje támogatta. A K 147452 számú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával, a K\_23 pályázati program finanszírozásában valósult meg.

## Hivatkozások

- Bast, R., Endresen, A., Janda, L.A., Lund, M., Lyashevskaya, O., Mordashova, D., Nessel, T., Rakhilina, E., Tyers, F.M., Zhukova, V.: The Russian Constructicon. An electronic database of the Russian grammatical constructions (2021), <https://constructicon.github.io/russian>
- Croft, W.: Morphosyntax. Constructions of the World's Languages. Cambridge University Press (2022)
- Diessel, H.: The Constructicon. Cambridge University Press (2023)
- Goldberg, A.E.: Constructions at work: The nature of generalization in language. Oxford University Press (2006)
- Haspelmath, M.: On what a construction is. *Constructions* 15(1) (2023)
- Hilpert, M.: Construction Grammar and Its Application to English. Edinburgh University Press (2014)
- Indig, B., Sass, B., Simon, E., Mittelholcz, I., Vadász, N., Makrai, M.: One format to rule them all – the **emtsv** pipeline for Hungarian. In: Proceedings of the 13th Linguistic Annotation Workshop. pp. 155–165. Association for Computational Linguistics, Florence, Italy (aug 2019), <https://www.aclweb.org/anthology/W19-4018>
- Janda, L., Endresen, A., Zhukova, V., Mordashova, D., Rakhilina, E.: How to build a constructicon in five years: The Russian example. *Belgian Journal of Linguistics* 34, 162–175 (12 2020)
- Lyngfelt, B., Borin, L., Ohara, K., Torrent, T.T. (szerk.): Constructicography: Constructicon development across languages. John Benjamins, Amsterdam (2018a)
- Lyngfelt, B., Bäckström, L., Borin, L., Ehrlemark, A., Rydstedt, R.: Constructicography at work: Theory meets practice in the Swedish Constructicon. In: Lyngfelt és mtsai (2018a), pp. 41–106
- Mittelholcz, I.: emToken: Unicode-képes tokenizáló magyar nyelvre [emToken: Unicode-aware tokenizer for hungarian]. In: Berend, G., Gosztolya, G., Vincze, V. (szerk.) MSZNY 2017. pp. 61–69. Szegedi Tudományegyetem, Informatikai Intézet (2017)
- Novák, A., Siklósi, B., Oravecz, Cs.: A new integrated open-source morphological analyzer for Hungarian. In: Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odiijk, J., Piperidis, S. (szerk.) Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). European Language Resources Association (ELRA), Paris, France (may 2016)
- Orosz, G., Novák, A.: PurePos 2.0: a hybrid tool for morphological disambiguation. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013). pp. 539–545 (2013)
- Oxford University Press: Oxford advanced learner's dictionary (2023), <https://www.oxfordlearnersdictionaries.com>
- Pusztai, F. (szerk.): Magyar Értelmező Kéziszótár. Akadémiai Kiadó (2003)
- Sass, B.: From a dictionary towards the Hungarian Constructicon. In: Medved', M., Měchura, M., Tiberius, C., Kosem, I., Kallas, J., Jakubiček, M., Krek,

- S. (szerk.) Electronic lexicography in the 21st century (eLex 2023): Invisible Lexicography. Proceedings of the eLex 2023 conference. pp. 534–544. Lexical Computing CZ s.r.o. (2023)
- Sass, B.: The “dependency tree fragments” model for querying a constructicon. In: Š. Despot, K., Ostroški Anić, A., Brač, I. (szerk.) Lexicography and Semantics. Proceedings of the XXI EURALEX International Congress. pp. 275–282. Institute for the Croatian Language (2024)
- Straka, M., Hajič, J., Straková, J.: UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In: Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. (szerk.) Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). pp. 4290–4297. European Language Resources Association (ELRA), Portorož, Slovenia (May 2016), <https://aclanthology.org/L16-1680>
- Vainik, E., Paulsen, G., Sakhai, H., Kallas, J., Tavast, A., Koppel, K.: From a dictionary to a constructicon: Putting the basics on the map. In: Š. Despot, K., Ostroški Anić, A., Brač, I. (szerk.) Lexicography and Semantics. Proceedings of the XXI EURALEX International Congress. pp. 209–216. Institute for the Croatian Language (2024)



# Evaluation Library for the Hungarian Language Understanding Benchmark (HuLU)

Péter Hatvani<sup>1,2</sup>, Kristóf Varga<sup>1</sup>, Zijian Győző Yang<sup>1</sup>

<sup>1</sup>HUN-REN Research Center for Linguistics

family name.(middle name).first name@nytud.hun-ren.hu

<sup>2</sup>Pázmány Péter Catholic University, Doctoral School of Linguistics

**Abstract.** This paper introduces Hungary’s first fine-tuning and evaluation Python library for the Hungarian Language Understanding Benchmark Kit (HuLU), designed to streamline prototyping and support effective language modeling in Hungarian. The library lets researchers quickly validate their results on the official HuLU leaderboard, handling the full pipeline from training to submission. It supports a range of architectures, including encoders, decoders, and custom PyTorch modules, and it’s ready for both smaller models with standard fine-tuning and larger models using low-rank adaptation and reduced-precision fine-tuning. The library is highly configurable and supports huggingface integration, giving researchers the flexibility of modern training frameworks. Two interfaces are included: a simple command-line interface and a programmatic API for more in-depth customization.

**Keywords:** Language Modeling, Evaluation, Hungarian NLP, Benchmarking, Training Automation, Fine-Tuning, Low-Rank Adaptation, HuLU

## 1 Introduction

Recent years have resulted in a big push to evaluate language models easily. State-of-the-art language modeling methods require fast and easy-to-use tools for evaluation, as industry leaders have proven for years now. For Hungary, the most widely used and state-of-the-art evaluation benchmark is HuLU (Ligeti-Nagy et al., 2024). This is why we created a Python client library that is available free and open source at<sup>1</sup> to facilitate fine-tuning and validation of this benchmark. The tool is capable of training models without regard to size and architecture thanks to the underlying technology from huggingface tools. The tool is capable of training decoder and encoder architectures or custom PyTorch modules. This library is designed with rapid prototyping in mind. Researchers can achieve quick results from models because, prior to training, custom classification layers are automatically attached to the model.

---

<sup>1</sup> <https://github.com/nytud/HuLU-evaluate>

## 2 Related works

Leaderboards such as the mteb (Muennighoff et al., 2023) and GLUE (Wang et al., 2019) are creating an intense competitive environment. For such spaces to exist, we need a reliable evaluation methodology. This methodology is usually implemented in ways of testing harnesses. In this context, the mteb leaderboard was made possible by the Eleuther AI evaluation harness (Gao et al., 2024). This harness offers quantization, adapter evaluation, and the entire tool set of the huggingface transformers library (Wolf et al., 2020). The harnesses standardize how a benchmark should be used and expose parameters that are transparent to other competitors. Another famous space is the Open LLM Leaderboard by huggingface (Leaderboard, 2024). This leaderboard includes many state-of-the-art benchmarks such as the MMLU-PRO (Wang et al., 2024) benchmark. This leaderboard uses the same evaluation harness and processes models at no cost to the model creators.

## 3 Corpora

The library obtains the training and development sets for all HuLU evaluation tasks from the official repository<sup>2</sup>. These are: Hungarian Commitment Bank (HuCB), Hungarian Corpus of Linguistic Acceptability (HuCOLA), Hungarian Choice of Plausible Alternatives Corpus (HuCOPA), Hungarian Recognizing Textual Entailment (HuRTE), Hungarian Stanford Sentiment Treebank (HuSST) Hungarian Winograd Schema Challenge (HuWNLI).

### 3.1 Corpus preprocessing

Each task requires a different tokenization process. The keys for each dataset are different, and the tasks require diverse training objectives. We have to distinguish between keys like "hypothesis", "premise", or "Sent". In terms of training objectives, we have used sequence classification for: HuCOLA, HuCOPA, HuRTE, HuWNLI, and HuSSt. For HuCB, we have created a custom head for multiple-choice classification. Labels are normalized for training.

*HuCOPA* We have followed the work of (Sileo, 2022) and prepared the labels from zero. Furthermore, we have constructed sentences for the model to make judgments about. The two categories are asked about.

```
item["question"] = (  
f'{item["premise"]} Mi volt ennek az eseménynek az előzménye?'  
    if item["question"] == "cause"  
    else f'{item["premise"]} Ennek következtében mi történt?'  
)
```

---

<sup>2</sup> <https://github.com/nytud/HuLU>

*HuRTE* For implementing the multiple choice task, we adopted a pre-processing strategy inspired by TaskNet’s methodology (Sileo, 2022). This method combines the premise with conjunctive phrases based on the type of question (cause or effect). Specifically, the premise is concatenated with *ezért* (therefore) for the effect questions and *mivel* (because) for the cause questions, and then paired with the answer choices. The implementation of this approach is shown in Code Snippet 1.1.

**Listing 1.1.** Preprocessing function for multiple-choice input in HuRTE

```
def preprocess_function_for_multiple_choice(examples):

    first_sentences = []
    second_sentences = []
    for premise, question, choice1, choice2 in
    zip(examples["premise"], examples["question"],
    examples["choice1"], examples["choice2"]):
        if question == "effect":
            first_sentences.extend([f"{premise}, ezért "] * 2)
        else: # "cause"
            first_sentences.extend([f"{premise}, mivel "] * 2)

    second_sentences.extend([choice1, choice2])
    ...
    return {k: [v[i : i + 2] for i in range(0, len(v), 2)]
            for k, v in tokenized_examples.items() }
```

As shown in Code Snippet 1.1, this function processes examples by iterating through the premise, question, and choices. The output consists of paired sentences prepared for tokenization and grouped for input into the model.

### 3.2 Custom Heads

For most tasks, we attach a custom classification head to the models, tailored to handle specific output requirements. The primary distinction arises with the Commitment Bank task, which necessitates a multiple-choice head rather than a standard classification head. This difference lies in the number of output dimensions: while the classification head outputs the number of classes (in our case, consistently three), the multiple-choice head outputs a single dimension, representing the likelihood of each choice. For the Commitment Bank, the labels mean "entailment", "contradiction", and "neutral".

In our setup, we implemented a linear layer followed by a dropout to enhance generalization and reduce overfitting. The classification head processes each sequence independently, using the dropout layer to stabilize training and mitigate variance due to model complexity. For the multiple choice head, we adapt the architecture to compare the likelihoods between choices within a single instance, allowing the model to better assess the relevance of each option within the context. The custom head is implemented because for most of the generative models,

such a head does not exist, and for BERT-like models, encoder models, the heads must be retrained as well.

## 4 Training

The training process is designed for ease of use, taking advantage of the Hugging Face transformers library through the *Trainer* class. This library provides state-of-the-art machine learning tools, with extensive arguments available in *Trainer* to configure model training for various needs. To enhance the adaptability of the tool, we included support for Low-Rank Adaptation (LoRA) (Hu et al., 2022), which allows efficient parameter tuning in large language models under resource constraints. Additionally, integration of the bitsandbytes library (Dettmers and Zettlemoyer, 2023) enables mixed precision training, which optimizes memory usage and computational efficiency, making the tool suitable for large- and small-scale models.

*Low-Rank Adaptation (LoRA)* LoRA enables more efficient training of large-scale language models by introducing low-rank matrices into the model’s architecture, allowing for fewer parameters to be fine-tuned without sacrificing performance. This approach is particularly advantageous for the HuLU benchmark, as it minimizes the required computational resources, making fine-tuning feasible on readily available standard hardware. In our setup, LoRA configurations allow researchers to selectively update only a subset of the model parameters, significantly reducing memory and processing needs. This method proved especially useful when working with models that have large embedding sizes and numerous layers, as it retains core parameters while enabling targeted tuning through low-rank representations.

*Mixed Precision Training* Mixed precision training, facilitated by bitsandbytes integration, is implemented to reduce computational costs by allowing selective precision scaling. Rather than using full 32-bit floating point precision across all operations, mixed precision training allows for 16-bit or even 8-bit precision in certain model operations, particularly for gradients and noncritical model parameters. This approach drastically reduces memory usage, enabling larger batch sizes and increased model depth within GPU memory constraints. This method reduced memory usage by 25-33%. Our testing has shown that a standard 8 billion parameter large language model can be fine-tuned in 48GB of VRAM with this method.

*bitsandbytes Integration* The bitsandbytes library (Dettmers and Zettlemoyer, 2023) supports 8-bit and 16-bit precision, making it an essential component of our training set for models with significant memory requirements, such as LLMs with more than 3 billion parameters. However, the definition of large model is in flux, but it is impossible to load without quantization a 7B model for training. A common method is normalizing quantization and low rank adaptation and

use as one term QLoRA. With QLoRA it is possible to fine-tune a 13 billion parameter model with just 9GB of memory. By quantizing model parameters and optimizing memory allocation, bitsandbytes allows researchers to fit larger models into limited GPU memory, further extending the capabilities of mixed precision. This integration is especially beneficial when working with large models. Additionally, bitsandbytes’s support for low-precision computations aligns well with LoRA configurations, enabling the tool to maximize efficiency across model sizes and training scenarios.

*Submitting Results* Our tool provides an efficient way to prepare the results for submission to the HuLU leaderboard. The user only needs to specify the folder to which the results should be saved, and for each task a json file will be created in the required format of the HuLU website. Based on insights from (Yang et al., 2023), batch size and context length are defining factors in fine-tuning. These adjustable settings allow users to experiment with the optimal configurations for their models. Additionally, our observations revealed significant performance variance depending on the number of micro-batches processed in a single GPU cycle, leading us to include batch handling configurations for enhanced consistency and reproducibility across runs.

*Command Line Interface (CLI)* To increase accessibility, we developed a Command Line Interface (CLI) that allows users to configure and execute training and evaluation tasks efficiently. The CLI includes options to set key parameters such as batch size, learning rate, and context length, allowing for rapid and flexible experiment setup. Additionally, users can easily toggle the mixed precision and LoRA options, tailoring memory and computational demands to suit available resources. An automated validation feature confirms that all submission criteria are met, standardizing the evaluation process between experiments. The CLI thus minimizes setup time and lowers the learning curve for those new to the Hugging Face transformers library, while providing advanced configuration options for experienced users engaged in Hungarian NLP tasks.

## 5 Training Arguments

Training arguments are crucial to controlling and defining the behavior of the model throughout the training process. Table 1 lists the main training arguments used in our setup, including model selection, batch size, learning rate, and other hyperparameters essential to optimize model performance. For example, we set the number of training epochs to 10 by default, based on the methodology of Yang et al. (2023), while the batch size and the learning rate are adjusted based on the size of the model. Larger models require lower learning rates to achieve stable convergence, reducing the likelihood of overfitting and improving training consistency.

## 5.1 Optimizer

In this setup, we use the AdamW optimizer, which has proven to be effective for transformer-based models due to its adaptive learning rate mechanism. It is the standard in many researches, and we can treat it as a default. We pair this with cosine annealing as a learning rate scheduler, allowing the training to start with a higher learning rate that gradually decreases, thus facilitating refined learning in later epochs. This scheduling is particularly beneficial given our limited dataset size, as it allows for effective convergence over multiple epochs.

model_name	bert-base-uncased
train_epochs	5
train_batch	16
train_lr	0.0001
train_warmup	500
train_maxlen	256
train_seed	123
train_loss	focal_loss
precision	fp16
lora_r	4
lora_alpha	32
lora_dropout	0.05
tasks	cola

**Table 1.** Default Training Arguments

Table 1 presents the main configuration values and hyperparameters, ensuring reproducibility across different training runs. The combination of these arguments allows for a flexible setup while optimizing model performance and resource usage.

## 6 Models

To assess the effectiveness of our evaluation methodology, we analyzed several models not previously evaluated in this context. In particular, two of these models are expected to perform well in Hungarian language tasks based on prior evaluations.

### 6.1 EuroLLM

EuroLLM (Martins et al., 2024) is a multilingual language model developed by the UTTER consortium (Unified Transcription and Translation for Extended Reality), a Horizon Europe-funded initiative focused on applications for translation and multilingual language processing. EuroLLM consists of two variants: a general-purpose raw model and an instruct model fine-tuned for prompt-based

tasks. Both versions are designed with multilingual capabilities, providing potential utility for languages of lower resources, including Hungarian. In this paper, we use the instruct version. The model has 7 billion parameters and was trained on 1% of the 2x100 billion token training corpus. The training was aimed at the highest quality available datasets.

## 6.2 mBERT

The original multilingual BERT model (Devlin et al., 2019) has remained effective for multilingual language processing since its release in 2019. With 179 million parameters, BERT supports more than 100 languages and is widely used for classification and general NLP tasks. Its pre-training approach, based on masked language modeling and next-sentence prediction, established a robust baseline for multilingual understanding and continues to be relevant for comparison across language models.

## 6.3 Twitter/twhin-bert-base

TWHIN-BERT (Zhang et al., 2023) is a 279 million parameter BERT-based model developed by Twitter, pre-trained on a large corpus of tweet interactions. The model leverages a joint contrastive social loss combined with masked language modeling, enabling it to capture both semantic and social context. TWHIN-BERT has shown notable results in informal text processing, outperforming multilingual BERT by 3% in Hungarian hashtag prediction tasks, illustrating its utility for tasks involving short-format web text.

## 6.4 SZTAKI-HLT huBERT

HuBERT (Nemeskey, 2020) (Nemeskey, 2021), developed by Dávid Nemeskey and affiliated with HUN-REN SZTAKI, is a Hungarian adaptation of BERT, pre-trained on Webcorpus 2.0, a Hungarian dataset compiled from Common Crawl (Common Crawl, 2024). This model is specifically optimized for Hungarian, capturing linguistic nuances unique to the language. HuBERT serves as a valuable resource for Hungarian NLP, addressing a gap in language-specific pretrained models. This is the smallest model with 110 million parameters, but it is the state-of-the-art encoder model for Hungarian.

# 7 Results and Evaluation

Table 2 summarizes the results of the evaluation of several language models on key Hungarian NLP benchmarks. These benchmarks span a variety of tasks, including textual entailment, linguistic acceptability, commonsense reasoning, sentiment analysis, and natural language inference. The evaluation process leverages

a streamlined and standardized setup, enabling consistent and reliable comparisons among models. This unified approach is particularly beneficial for benchmarking models across lower-resource languages like Hungarian, where such standardization is crucial for resource development and research progress.

The table presents official metrics such as the F1 score for HuCB (Hungarian CommitmentBank), the Matthews correlation coefficient (MCC) for linguistic acceptability (HuCOLA) and common sense reasoning (HuCOPA), and accuracy for sentiment classification (HuSST) and natural language inference (HuWNLI). These diverse tasks highlight the strengths and limitations of each model in a spectrum of linguistic challenges.

## 7.1 Performance Insights

- **HuCB (F1 Score)**: The HuCB task, which evaluates textual entailment and inference capability, shows that the SZTAKI huBERT model achieves the best F1 score (14.27), slightly outperforming the euroLLM instruct model (13.56). Both models demonstrate their capacity to understand and process nuanced Hungarian textual inferences, but there is significant room for improvement.
- **HuCOLA (MCC)**: Linguistic acceptability, measured by HuCOLA, reveals the strong language understanding capabilities of huBERT (65.49 MCC). This model significantly outperforms euroLLM instruct (45.05 MCC), mBERT (39.54 MCC), and twhin-BERT (32.31 MCC). These results underscore huBERT’s robust handling of grammatical and syntactic nuances in Hungarian.
- **HuCOPA (MCC)**: For commonsense reasoning, huBERT again leads the performance (28.40 MCC), with euroLLM instruct trailing at 8.38 MCC. This task demonstrates the challenge of adapting pretrained models to reasoning-based tasks in Hungarian, as all models exhibit relatively low scores.
- **HuRTE (MCC)**: On the HuRTE natural language inference task, euroLLM instruct achieves the best performance (51.57 MCC), indicating its effectiveness in handling entailment-based tasks. huBERT follows closely (48.50 MCC), while mBERT and twhin-BERT perform less competitively (43.37 MCC and 21.67 MCC, respectively).
- **HuSST (Accuracy)**: The results of the sentimental classification show that huBERT excels with an accuracy of 79.66%, followed by the euroLLM instructive at 74.59%. These high scores highlight the effectiveness of these models in understanding sentiment in the Hungarian text. Both mBERT (63.61%) and twhin-BERT (65.41%) lag behind, indicating their relatively weaker adaptation to sentiment analysis tasks in Hungarian.
- **HuWNLI (Accuracy)**: The HuWNLI task evaluates natural language inference in a binary format. mBERT and twhin-BERT are tied with an accuracy of 63.43%, closely followed by the euroLLM instruct (62.69%). huBERT, surprisingly, scores lower in this task (52.98%), suggesting potential challenges in adapting its architecture to binary inference tasks.



*Caveats* These results are only indicative for this inquiry it is out of the scope of this paper to challenge the state-of-the-art performance of models. The results are shown as a testimonial of the library’s working. For comparisons with the state-of-the-art models, see (Yang et al., 2023),

## 7.2 Discussion of Model Strengths and Limitations

The results demonstrate that **SZTAKI huBERT** consistently performs well on a range of Hungarian language tasks, particularly excelling in linguistic acceptability (HuCOLA) and sentiment classification (HuSST). Its robust performance reflects its specialization in Hungarian NLP and its fine-tuned adaptation to local linguistic structures. **EuroLLM**, while slightly behind huBERT in overall performance, showcases competitive results in natural language inference (HuRTE) and sentiment analysis (HuSST), making it a valuable tool for broader applications in Hungarian.

**mBERT** and **twhin-BERT**, despite their strong general-purpose architectures, struggle to match the performance of more localized models. This gap underscores the importance of fine-tuning models in specific languages to handle complex linguistic features and achieve competitive performance.

## 7.3 Standardized Evaluation Framework

The ease of submission and the standardized evaluation setup ensure that all models are tested under uniform conditions. This consistency minimizes variations due to differing implementations and experimental setups, enabling direct and fair comparisons across diverse architectures and pre-training strategies. By streamlining the benchmarking process, the framework reduces the barrier to entry for researchers and promotes the adoption of these benchmarks for Hungarian NLP research. This approach fosters the development of models tailored to the unique challenges of Hungarian, advancing progress in this lower-resource language.

Metric	euroLLM	m-bert-base	twhin-bert	hubert-base
HuCB (f1)	13.56	0.00	0.00	14.27
HuCOLA (MCC)	45.05	39.54	32.31	65.49
HuCOPA (MCC)	8.38	2.38	1.54	28.40
HuRTE (MCC)	51.57	43.37	21.67	48.50
HuSST (ACC)	74.59	63.61	65.41	79.66
HuWNLI (ACC)	62.69	63.43	63.43	52.98

**Table 2.** Model Evaluation Results by Metric

In conclusion, the results in Table 2 highlight the varying strengths of the evaluated models, with huBERT standing out as a strong performer in most

tasks. The standardized framework not only accelerates the evaluation process, but also provides a foundation for future work in benchmarking and improving Hungarian NLP models.

## 8 Conclusion

This paper presented the first dedicated library for evaluating Hungarian language models on the HuLU benchmark, streamlining the process of model training, fine-tuning, and submission to the official leaderboard. The tool supports a wide range of model architectures, including both small models with fine-tuning and larger models with low-rank adaptation, addressing the varying computational needs of researchers.

Through its integration with Hugging Face transformers and support for mixed-precision training, the library enables efficient experimentation, facilitating accurate benchmarking for Hungarian language understanding. Our results demonstrated competitive performance across key metrics, with models like SZ-TAKI huBERT and EuroLLM instruct performing well in multiple tasks, underscoring the potential of standardized evaluations in advancing NLP resources for lower-resource languages.

Overall, this library provides a practical, accessible platform for researchers working with Hungarian NLP, contributing to the consistency and transparency of model evaluation efforts within the community. Future work will focus on expanding support for additional tasks and improving scalability to accommodate more diverse architectures and larger datasets.

## 9 Further Work

Future work will focus on expanding the benchmarks available within this library to provide a more comprehensive evaluation suite for Hungarian NLP. While the current implementation covers key HuLU tasks, additional benchmarks will enable a deeper analysis of diverse linguistic aspects, including syntax, entity recognition, and semantic similarity, which are critical yet understudied for Hungarian.

We also aim to incorporate broader evaluation metrics to capture model performance more holistically, especially for generative tasks. Planned improvements to the tool include extended logging, better error analysis, and options for fine-tuning configurations to support a wider range of model architectures.

Through these expansions, we hope to enhance the utility of the library for researchers and contribute to the development of Hungarian NLP resources in a practical and accessible way.

## References

Common Crawl: Common crawl (2024), <https://commoncrawl.org>, accessed: 2024-10-27

- Dettmers, T., Zettlemoyer, L.: The case for 4-bit precision: k-bit inference scaling laws. In: International Conference on Machine Learning. pp. 7750–7774. PMLR (2023)
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), <https://aclanthology.org/N19-1423>
- Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac’h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., Zou, A.: A framework for few-shot language model evaluation (07 2024), <https://zenodo.org/records/12608602>
- Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (2022), <https://openreview.net/forum?id=nZeVKeeFYf9>
- Leaderboard, O.L.: Open llm leaderboard blog (2024), <https://huggingface.co/spaces/open-llm-leaderboard/blog>, accessed: 2024-10-27
- Ligeti-Nagy, N., Ferenczi, G., Héja, E., Laki, L.J., Vadász, N., Yang, Z.G., Váradi, T.: HuLU: Hungarian language understanding benchmark kit. In: Calzolari, N., Kan, M.Y., Hoste, V., Lenci, A., Sakti, S., Xue, N. (eds.) Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024). pp. 8360–8371. ELRA and ICCL, Torino, Italia (May 2024), <https://aclanthology.org/2024.lrec-main.733>
- Martins, P.H., Fernandes, P., Alves, J., Guerreiro, N.M., Rei, R., Alves, D.M., Pombal, J., Farajian, A., Faysse, M., Klimaszewski, M., Colombo, P., Had-dow, B., de Souza, J.G.C., Birch, A., Martins, A.F.T.: Eurollm: Multilingual language models for europe (2024), <https://arxiv.org/abs/2409.16235>
- Muennighoff, N., Tazi, N., Magne, L., Reimers, N.: Mteb: Massive text embedding benchmark (2023), <https://arxiv.org/abs/2210.07316>
- Nemeskey, D.M.: Natural Language Processing Methods for Language Modeling. Phd thesis, Eötvös Loránd University (2020)
- Nemeskey, D.M.: Introducing huBERT. In: XVII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2021). p. TBA. Szeged (2021)
- Sileo, D.: tasknet, multitask interface between Trainer and datasets (11 2022), <https://github.com/sileod/tasknet>
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: A multi-task benchmark and analysis platform for natural language understanding (2019), in the Proceedings of ICLR.

- Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al.: Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. arXiv preprint arXiv:2406.01574 (2024)
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020), <https://www.aclweb.org/anthology/2020.emnlp-demos.6>
- Yang, Z.G., Dodé, R., Ferenczi, G., Héja, E., Jelencsik-Mátyus, K., Kőrös, A., Laki, L.J., Ligeti-Nagy, N., Vadász, N., Váradi, T.: Jönnek a nagyok! bert-large, gpt-2 és gpt-3 nyelvmodellek magyar nyelvre. In: XIX. Conference on Hungarian Computational Linguistics (MSZNY 2023). pp. 247–262. University of Szeged (SZTE) Institute of Informatics, Szeged, Hungary (2023)
- Zhang, X., Malkov, Y., Florez, O., Park, S., McWilliams, B., Han, J., El-Kishky, A.: Twhin-bert: A socially-enriched pre-trained language model for multilingual tweet representations at twitter (2023), <https://arxiv.org/abs/2209.07562>

# Egy magyar nyelvű táblázatos kérdésmegválaszolás kiértékelő adatbázis

Tóth Gábor<sup>1</sup>, Farkas Richárd<sup>1</sup>, Szántó Zsolt<sup>1</sup>

<sup>1</sup>Szegedi Tudományegyetem, Informatikai Intézet

Szeged, Árpád tér 2.

`gabor.toth.103@gmail.com`

`{rfarkas,szantozs}@inf.u-szeged.hu`

**Kivonat** Cikkünkben nagy nyelvi modellek táblázat értelmezési képességeit vizsgáljuk. Ehhez készítettünk egy magyar nyelvű kiértékelőadatbázist, amiben Wikipédiáról származó táblázatok és azok alapján megválaszolható kérdés-válasz párok találhatóak. A kérdés-válasz párokat a táblázat környezetében található szövegek alapján automatikusan generáltuk, majd ezeket kézzel ellenőriztük. Az így kapott korpuszon összehasonlítottuk 14 nagy nyelvi modell pontosságát. Azt konkludálhatjuk, hogy 2024 végén a 8 milliárd paraméteres modellek körülbelül 10 százalékponttal, míg a 30 milliárd körüli modellek 5 százalékponttal teljesítenek rosszabbul, mint a GPT4o.

**Kulcsszavak:** kérdésmegválaszolás, táblázat, nagy nyelvi modellek

## 1. Bevezetés

A táblázatok széles körben elterjedtek és gazdag információforrást jelentenek az interneten és különféle dokumentumokban. Statisztikai adatok szerint az internetes weboldalakon található táblázatok száma elérte a több százmilliót (Lehmsberg és mtsai, 2016); a vállalati környezetben pedig az Excel-szerű fájlokban lévő táblázatok száma meghaladta a 115 milliót (Wang és mtsai, 2020). A táblázatokból származó releváns információk pontos keresése kulcsfontosságú számos valós alkalmazásban, például pénzügyi elemzésekben vagy a tudományos kutatásokban.

Az elmúlt években a nagy nyelvi modellek (Large Language Models, LLM-ek) figyelemreméltó fejlődése (Brown és mtsai, 2020; Chowdhery és mtsai, 2023; Touvron és mtsai, 2023) átalakította az táblázatokból történő információkinyerést is. Az LLM-ek legfőbb előnye, hogy képesek általánosítani és összetett nyelvi struktúrákat értelmezni, ami kiemelkedően fontosá teszi őket a nyelvi adatok feldolgozásában. Bár ezek a modellek elsősorban angol nyelvre optimalizáltak, egyre nagyobb igény mutatkozik arra, hogy más nyelveken, például magyarul is sikeresen alkalmazhatók legyenek.

Az egyik érdekes alkalmazási terület a táblázatos kérdésmegválaszolási feladat (TQA) (Jin és mtsai, 2022), amely során a modellnek egy táblázat alapján kell a feltett kérdésekre választ adnia. Egészen pontosan a TQA feladat során a

bemenet egy vagy több táblázat és egy természetes nyelven megfogalmazott kérdést, amelyet a táblázat(ok)ban található információk alapján kell megválaszolni. A feladat sikeres végrehajtásához a modellnek képesnek kell lennie arra, hogy a kérdés alapján valamilyen módon a táblázat megfelelő celláiban található adatokat azonosítsa és felhasználja. Ez magában foglalhat egyszerű adatlekérdezéseket (pl. „Mennyi az X termék ára?”), összetett műveleteket (pl. „Melyik évben volt a legnagyobb növekedés az Y termék értékesítésében?”), valamint több lépésben elérhető információk aggregálását is.

Ebben a cikkben bemutatunk egy magyar nyelvű TQA adathalmaz-előállító eljárást, ami javarészt automatikusan képes azt létrehozni magyar Wikipédia szócikkek táblázatos adatait és a hozzájuk kapcsolható szöveges tartalmakat hasznosítva. Az ezekből az adatokból készített TQA adathalmaz lehetőséget nyújt az LLM-ek tesztelésére magyar nyelvű adaton, valamint a TQA algoritmusok hatékonyságának vizsgálatára.

Cikkünkben továbbá több nagy nyelvi modell pontosságát és sebességét hasonlítjuk össze az elkészült magyar nyelvű TQA adathalmazon.

## 2. TQA megközelítések

Az angol nyelvű TQA feladatra számos megoldás született az elmúlt néhány évben. Az encoder-decoder modellek, mint a TAPEX (Liu és mtsai, 2021), és az MLM (Masked Language Modelling) módszerek (Herzig és mtsai, 2020), feldolgozzák a strukturált táblázatos adatokat majd a szemantikus térben ezek alapján történik meg a kérdésmegválaszolás. Ebben a megközelítésben a modellek közvetlenül állítják elő a válaszokat anélkül, hogy a természetes nyelvi kérdéseket átalakítanák. Ugyanakkor ezek a módszerek memóriakorlátaik miatt nehezen alkalmazhatók nagyméretű, valós adathalmazokon és az MLM bemene- tének hosszlimitációja is problémát jelent.

Egy másik megközelítés a kérdések formális lekérdezésekké való átalakítása. Ezek a módszerek a természetes nyelvű kérdéseket valamilyen formális lekérdező nyelvre, például SQL-re alakítják át, amelyet aztán a táblázatos adatok lekérdezésére használnak (Shi és mtsai, 2020). Bár ezek a módszerek hatékonyak a strukturált adatbázisokban történő lekérdezésekben, korlátozottak nem szabványos táblázatok esetében, és a kérdések SQL-re történő pontos fordítása is nehézségekbe ütközhet. Megjegyezzük, hogy léteznek hibrid megközelítések is, amelyek ötvözik a formális leképezések és az encoder-decoder módszerét, például a TAPAS (Herzig és mtsai, 2020) vagy a TAGOP (Zhu és mtsai, 2021) modellek.

A mai modern LLM-ek már táblázatos struktúrákat is tudnak kezelni, így az elmúlt 2 évben a zero-shot és few-shot megközelítés lett egyeduralgó. Cikkünkben mi is ezt a megközelítést követjük, és 14 LLM-hez szerkesztünk promptot, amivel magyar TQA teljesítményüket hasonlítjuk össze.

### 3. Egy Magyar Nyelvű TQA adatbázis

#### 3.1. Adatgyűjtés

A manuális annotáció nagyon drága. Ahhoz, hogy ezt a költséget csökkentsük, egy nagyrészt automatikus módszert dolgoztunk ki magyar nyelvű TQA kiértékelőadatbázis előállítására. A magyar nyelvű TQA adatbázis létrehozása a magyar Wikipédián található táblázatok és kapcsolódó szövegek automatizált gyűjtésével és feldolgozásával történt. Az alábbiakban részletesen bemutatjuk az adatbázis összeállításának folyamatát, a Wikipédiából való adatgyűjtést, az adatok tisztítását és az adathalmaz struktúráját.

A magyar Wikipédiában található cikkek jelentős része tartalmaz olyan táblázatokat, amelyek különféle adatokat és kategóriákat rendeznek strukturált formába. Az adatgyűjtési folyamat során célzottan a következő típusú információkat gyűjtöttük:

- **Táblázatos adatok:** A cikkekben található táblázatok, amelyek számos adattípust, például statisztikai adatokat, időpontokat, országokat, földrajzi helyeket vagy történelmi eseményekre vonatkozó információkat tartalmaznak.
- **Kapcsolódó szöveges kontextus:** A táblázatok körül található szöveges tartalmakat elemeztük, mivel ezek gyakran magyarázzák vagy részletezik a táblázatok tartalmát. Ezeket a táblázathoz köthető mondatokat használjuk majd fel kérdések generálására.

Az összegyűjtött táblázatok HTML-struktúrájának feldolgozása során az adatokat egy strukturált formába alakítottuk át, amelyben minden táblázatsor és -oszlop különálló entitásként kezelhető. Így könnyebben társíthatóak a táblázat celláihoz kapcsolódó kérdések.

Mivel a Wikipédián előfordulhat, hogy egyes táblázatok ismétlődnek vagy hasonló adatokat tartalmaznak, ezért eltávolítottuk a redundáns bejegyzéseket, hogy az adathalmaz minél változatosabb és informatívabb legyen.

A táblázatok egyes cellái esetenként hiányosak vagy nem tartalmaznak releváns információt. Ezeket a cellákat vagy kitöltöttük, ha a kontextus alapján lehetett következtetni a helyes adatértékre, vagy jelöltük, hogy ezek az adatok nem használhatóak a kérdés-válasz feladatban.

A táblázatok celláiban szereplő adatokat típus szerint csoportosítottuk (pl. dátum, szám, szöveg), hogy a későbbi feldolgozás során könnyebben használhatók legyenek. Például az időpontokat és számadatokat formailag és tartalmilag is egységesítettük.

#### 3.2. Kérdések generálása

A táblázat közvetlen környezetében található szövegek egy része a táblázat tartalmát magyarázza, kiemelve annak fontos, érdekes részeit. Ezeket a magyarázatokat vettük a kérdésgenerálás alapjául, mégpedig úgy, hogy az azokban megfogalmazott állításokat fogalmaztattuk át kérdésekké. Vegyük például az alábbi szövegrészletet, ami a Szeged népességéről szóló táblázat mellett található:

a legmagasabb befejezett iskolai végzettség szerint az érettségi végzettséggel rendelkezők élnek a legtöbben a városban 52 347 fő, utánuk a következő nagy csoport a diplomával rendelkezők 44 502 fővel

Ebből az alábbi kérdést generáltuk:

Milyen végzettséggel rendelkezők élnek a legtöbben a városban?

A kontextusban található szövegek alapján lehetőség van többféle kérdéstípus generálására is:

- **Egyszerű információt kérjenek:** Például egy adott cellában szereplő konkrét adat megkérdezése, mint „Mi Magyarország fővárosa?” egy országokra vonatkozó táblázatban.
- **Összetett lekérdezéseket igényeljenek:** Olyan kérdések, amelyek több cella vagy sor összehasonlítását, összegzését vagy kiszámítását követelik meg, például „Melyik ország népessége a legnagyobb?”.

Kiválasztjuk azokat a mondatokat, amelyek tartalmazzák a közvetlen közelben szereplő táblázat valamely celláinak tartalmát. Ezeknek az állító mondatoknak az átfogalmazásával generálunk kérdéseket, azaz egy mondat és abban előforduló érték (ami egyben egy táblázat valamely cellájának értéke) alapján olyan kérdést generálunk, ami az értékre kérdez rá. Jelen verzióban ezt a kérdés-széalakítást a GPT4o-mini modell promptolásával valósítottuk meg.

Az így összeállított adathalmaz strukturájában a WikiTableQA-t (Pasupat és Liang, 2015) követtük. Megjegyezzük, hogy a WTQA-ban szerepelnek listával megválaszolható kérdések, azaz olyanok, amik több helyes válasszal rendelkeznek. Az egyszerűség kedvéért, mi kiszűrtük azokat a kérdéseket, amire több helyes válasz is adható egy táblázatból. Az adathalmaz a következő elemeket tartalmazza:

- **Táblázat:** A forrástáblázat maga, amely a Wikipédiáról származik, és strukturált formában van tárolva.
- **Kérdések:** Minden táblázathoz kapcsolódóan több kérdés-válasz pár található.
- **Válaszok:** Minden kérdéshez tartozik egy referencia-válasz, amely az elvárt kimenetként szolgál a kiértékelés során.

Például: adott egy táblázat, amely Szeged népességének alakulását mutatja be. A fejezetben található szövegrészlet átalakításával a kapott kérdés: "Melyik évben éltek a legtöbben a városban?" A válasz pedig az, hogy 1990-ben.

Az adatgyűjtés során az volt a célunk, hogy csak a megbízható táblázatokat és állításokat elemjük ki, hogy minél kevesebb manuális ellenőrzésre legyen szükség. A vizsgált 2000 szócikkből összesen 1086 olyan szövegrészletet vagy mondatot azonosítottunk, amelyben szerepeltek a táblázat közelében (Wikipedia szócikk fejezete) a táblázatok celláiban megtalálható kifejezések. Azonban ezek többsége nem állt releváns kapcsolatban a megfelelő táblázattal. Ennek következtében



szigorítottunk a keresési feltételeken: az elemzést kizárólag a táblázatok közvetlen környezetére (a szereplő legalacsonyabb szintű alfejezete) szűkítettük, és több szűrési technikát is alkalmaztunk.

Ezek a szűrési eljárások magukban foglalták például annak vizsgálatát, hogy a táblázat cellájában található célérték többször előfordul-e a szövegben, valamint a túl hosszú mondatok kizárását, amelyek kevésbé informatívak vagy nem egyértelműek.

Ezek az eljárások a vizsgált szócikkekben a helyes adatpontokat megtartották. Az így szűkített adathalmaz végül néhány száz potenciális adatpontot tartalmazott, amelyeket manuálisan validáltunk. Ez az eljárás további szűrők bevezetésével még pontosabbá tehető.

Problémát jelent még, hogy a táblázatok nem jól strukturáltak, ezért ezek a HTML-ből nyert adatok hiányos vagy elcsúszott cellákat, néha üres oszlopneveket tartalmaznak, amivel nehezebbé teszik a modellek számára a táblázat értelmezését. Fontos megjegyezni, hogy míg a WTQ adatbázis legalább 8 sorral és 4 oszloppal rendelkező táblázatokat tartalmaz, addig itt ilyen szűrőt nem alkalmaztunk. Ez azt jelenti, hogy megfelelően felismert táblázat esetén a feladat lényegesen könnyebb lehet más adatbázisokhoz képest.

Az automatikus szűrőkön átment példákat manuálisan ellenőriztük. Egyrészt értékeltük, hogy a GPT4o-mini modell helyes magyarságú és tényleg a cellaadatra vonatkozó kérdést generált-e. Másrészt vizsgáltuk, hogy a kérdés megválaszolható-e kizárólag a táblázat tartalma alapján. Összesen 54 érvényes adatpontot találtunk tökéletesen helyesnek. Megközelítésünkben így 2000 magyar Wikipédia-szócikk feldolgozásából mindössze 54 érvényes adatpontot sikerült előállítani, ami 32 egyedi táblázatot használ fel. Ez azt jelenti, hogy táblázatonként átlagosan 1-2 releváns mondatot találtunk, amely kifejezetten kapcsolódott az adott táblázat tartalmához, és a cellákban szereplő értékekre vagy azok magyarázatára utalt. A táblázatok közvetlen környezetében található szöveges információk túlnyomó többsége nem felelt meg az előzetesen meghatározott relevancia-kritériumoknak, így ezek további szűrést vagy pontosítást igényeltek. Habár átlagosan csupán egyetlen adatpont jutott 37 szócikkre, figyelembe kell venni, hogy a magyar Wikipédia több mint 1 millió lapot tartalmaz, így feltételezhető, hogy a módszerrel akár tízezres nagyságrendű adathalmaz is kinyerhető.

Az elkészült magyar TQA adatbázis publikusan elérhető a <https://github.com/szegedai/TableQA> címen és felhasználható a magyar nyelvű nagy nyelvi modellek (LLM-ek) táblázatértelmezési képességének kiértékelésére.

## 4. LLMek kiértékelése a TQA adatbázison

Az elkészült adatbázis felhasználásával, kezdeti kísérletként 14 nagy nyelvi modellt hasonlítottunk össze. A nyelvi modelleket próbáltuk minél nagyobb változatossággal kiválasztani, a szolgáltatásként elérhető két OpenAI GPT4 variánson kívül szerepeltek kisebb és nagyobb modellek 1 milliárd paramétertől 72 milliárdig, valamint kipróbáltunk kifejezetten magyar nyelvre fókuszálva továbbtanított modelleket a PULI Llumix 32K (Yang és mtsai, 2024) és a SambaLingo

Modellnév	Paraméterek száma (milliárd)	Pontosság	Sebesség (it/s)
Llama 3.2 1B	1	0.207	0.77
Llama 3.2 3B	3	0.530	0.53
Llama 3.1 8B	8	0.681	0.34
Llama 3.1 70B	70	0.755	0.18
Mistral NeMo	12	0.667	1.89
Phi-3.5-mini	3.8	0.463	0.71
Qwen2.5-72B	72	0.752	0.08
Gemma.2 27B	27	0.722	0.17
Cohere Aya Expanse 32B	32	0.722	0.09
C4AI Command-R	35	0.667	0.08
GPT4o-mini (2024-07-18)	?	0.741	0.34
GPT4o (2024-08-06)	?	0.778	0.25

1. táblázat. Nyelvi modellek hatékonyságának a vizsgálata az elkészült táblázat-adatbázis fölött.

Hungarian (Csaki és mtsai, 2024). A 12 lokálisan futatott modell esetén a futás-idők méréséhez 4db A100-as videokártyát használtunk. Ezen felül a két OpenAI modell API szolgáltatás alapú volt.

Minden LLM-nek ugyanazt a feladatszöveget adtuk meg, viszont ezt az adott modell ajánlásainak megfelelően helyeztük el a promptban. A pontosság kiszámításához csak teljesen egyező választ fogadtunk el, azaz ha az LLM válasza karakterre megegyezett a helyes cella szövegével. Ha az LLM több megoldását is adott, akkor ha a listában szerepelt a helyes válasz, akkor azt elfogadtuk. Minden LLM-et 5 alkalommal futtattunk minden kérdésre, az 1. táblázat az 5 futtatás eredményének az átlagát tartalmazza.

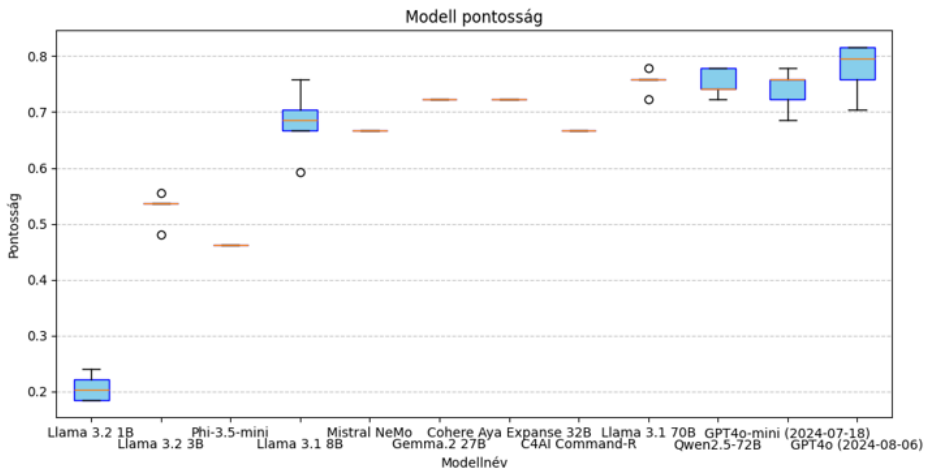
Egyes modellek, mint a MistralNemo és a CohereAya/32, kiegyensúlyozott teljesítményt nyújtott a kérdés-válasz feladatok során annak ellenére, hogy a dokumentációjuk alapján nem használtak magyar adathalmazt az előtanítás alatt.

1. ábra jól szemlélteti, hogy az LLM-ek legújabb generációjánál a modell méretének függvényében szinte monoton módon nő azok pontossága.

Modellnév	Paraméterek száma (milliárd)	Pontosság	Sebesség (it/s)
PULI LlumiX 32K	7	0.444	0.18
SambaLingo-Hungarian	7	0.407	0.23

2. táblázat. Nyelvi modellek eredményei az elkészült táblázat adatbázison megengedő kiértékeléssel.

A PULI LlumiX 32K modell instrukciókra hangolt változata nem érhető el szabadon és a SambaLingo-Hungarian-nak is csak chatelésre finomhangolt változata érhető el. Mivel ezek a modellek nincsenek kérdésmegválaszolásra hangolva,



1. ábra: Modellek pontossága méret szerint rendezve.

ezeknél a modelleknél a generált kimenetet kézzel ellenőriztük és megengedően értékeltük, hogy a generált szöveg megválaszolja-e a kérdést. Ezek eredménye a 2. táblázatban található.

## 5. Hiba analízis

A kérdéseket két kategóriába osztottuk az alapján, hogy egyszerű információt kérnek (38 db), amihez elég a megfelelő oszlop és sor kiválasztása vagy valamilyen összetettebb feladatot kell az LLM-nek megoldani (16 db), pl. elemek megszámlálása, vagy a legnagyobb/legkisebb érték kiválasztása valamilyen feltétel alapján. Ezek alapján megvizsgáltuk a legjobban és legrosszabbul teljesítő 2-2 modell hatékonysága mennyire tért el feladattípusonként.

Modellnév	Egyszerű	Összetett
Llama 3.1 70B	0.784	0.688
Qwen2.5-72B	0.758	0.717
Phi-3.5-mini	0.518	0.313
Llama 3.2 1B	0.242	0.125

3. táblázat. Modellek teljesítményének összehasonlítása egyszerű és összetett kérdések felett.

A 3. táblázatban látható, hogy bár minden modell jobban teljesített az egyszerű kérdések megoldásában a Qwen2.5-72B esetén ez mindössze 4 százalékpont különbséget jelent, míg a Llama 3.1 70B esetén körülbelül 10 százalékpontot.

A kisebb modellek esetén látható, hogy a bonyolultabb feladatokon lényegesen rosszabbul teljesítettek.

Összehasonlítottuk, hogy a modellek ugyanarra a feladatra mennyire konzisztensen válaszolnak az 5 futtatás alapján. A két hasonló architektúrájú Llama modellt nézve az 1 milliárd paraméteres modell átlagosan 3.1 különböző választ generált 5 kísérletből, míg a 70 milliárd paraméteres modell nem csak jobban válaszolt, hanem sokkal konzisztensebb is volt, átlagosan 1.72 különböző választ adva.

Az egyik legösszetettebb kérdésforma volt, ahol nem csak valamilyen matematikai műveletet kellett elvégezni (mint például a legnagyobb elem megtalálása) hanem a művelet által megkapott érték alapján kellett egy másik sor vagy oszlop tartalmát visszaadni, ilyen kérdés például a **‘Milyen végzettséggel rendelkezők élnek a legtöbben a városban?’**. Az ilyen összetettségű következtetést elváró kérdéseken a két legkisebb modell rendre rossz választ adott, még akkor is, ha ezt az adatot egy 2-3 soros táblázatból kellett kinyerni, míg a 70 milliárdos Llama sok esetben még ezekre az bonyolult összetett kérdésekre is helyesen tudott válaszolni. Annak ellenére, hogy sokszor helyesen meg tudta válaszolni ezeket a kérdéseket is a Llama 3.1 70B, például a hasonló logikájú **‘Melyik évben éltek a legtöbben a városban?’** kérdésre a ötből négyszer rossz évszámot adott válaszul.

## 6. Összegzés

Cikkünkben beszámoltunk egy automatikus eljárással létrehozott magyar nyelvű táblázatos kérdésmegválaszoló adatbázissal kapcsolatos első kísérleteinkről. Magát az eljárást, a szűrőket, a kezelt kérdéstípusokat, az állító mondatból kérdés generálást még számos ponton fejleszteni tervezzük a jövőben. Például lehetne a listás kérdéseken felül időbeli és logikai kapcsolatokra kérdező állító mondatokat is generálni: Például **‘Hány év telt el Magyarországon uniós csatlakozása óta?’** vagy **‘Melyik esemény történt előbb?’**, amivel az ilyen kérdésekkel az LLMek logikai következtetési képességét vizsgálhatnánk. Ha ezek a fejlesztések elkészülnek egy nagyobb méretű kiértékelő adatbázist tervezünk építeni és közzétenni.

Cikkünkben 14 darab nagy nyelvi modellt értékeltünk ki egy 54 elemű, magyar nyelvű táblázatos kérdésmegválaszoló feladaton. Az eredményekből az látszik, hogy a modellek méretétől függ azok pontossága, illetve, hogy a két publikusan elérhető magyarra tanított LLM egyértelműen rosszabbul teljesít, mint a méretben hasonló, sok nyelvre tanított, de újabb architektúrát alkalmazó versenytársaik.

## Köszönetnyilvánítás

Köszönjük a lehetőséget, melyet a Kormányzati Informatikai Fejlesztési Ügynökség által üzemeltetett Komondor biztosított.

A kutatás az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében.

## Hivatkozások

- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., és mtsai: Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. pp. 1877–1901 (2020)
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., és mtsai: Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24(240), 1–113 (2023)
- Csaki, Z., Li, B., Li, J.L., Xu, Q., Pawakapan, P., Zhang, L., Du, Y., Zhao, H., Hu, C., Thakker, U.: SambaLingo: Teaching large language models new languages. In: Sälevä, J., Owodunni, A. (szerk.) Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024). pp. 1–21. Association for Computational Linguistics, Miami, Florida, USA (Nov 2024), <https://aclanthology.org/2024.mrl-1.1>
- Herzig, J., Nowak, P.K., Mueller, T., Piccinno, F., Eisenschlos, J.: Tapas: Weakly supervised table parsing via pre-training. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 4320–4333 (2020)
- Jin, N., Siebert, J., Li, D., Chen, Q.: A survey on table question answering: recent advances. In: China Conference on Knowledge Graph and Semantic Computing. pp. 174–186. Springer (2022)
- Lehmborg, O., Ritze, D., Meusel, R., Bizer, C.: A large public corpus of web tables containing time and context metadata. In: Proceedings of the 25th international conference companion on world wide web. pp. 75–76 (2016)
- Liu, Q., Chen, B., Guo, J., Ziyadi, M., Lin, Z., Chen, W., Lou, J.G.: Tapex: Table pre-training via learning a neural sql executor. arXiv preprint arXiv:2107.07653 (2021)
- Pasupat, P., Liang, P.: Compositional semantic parsing on semi-structured tables. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 1470–1480 (2015)
- Shi, T., Zhao, C., Boyd-Graber, J., Daumé III, H., Lee, L.: On the potential of lexico-logical alignments for semantic parsing to sql queries. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 1849–1864 (2020)
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., és mtsai: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)

- Wang, B., Shin, R., Liu, X., Polozov, O., Richardson, M.: Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7567–7578 (2020)
- Yang, Z.G., Dodé, R., Ferenczi, G., Hatvani, P., Héja, E., Madarász, G., Ligeti-Nagy, N., Sárosy, B., SZaniszló, Z., Váradi, T., és mtsai: The first instruct-following large language models for hungarian. In: IEEE 3rd Conference on Information Technology and Data Science (CITDS) Proceedings. University of Debrecen (2024)
- Zhu, F., Lei, W., Huang, Y., Wang, C., Zhang, S., Lv, J., Feng, F., Chua, T.S.: Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 3277–3287 (2021)

## Szerzői index, névmutató

- Ács, Judit, 185
- Barta, Botond, 185
- Berend, Gábor, 113
- Bóna, Judit, 59
- Csáki, Csaba, 97, 233
- Csányi, Gergely Márk, 3
- Dalotti, Ágoston, 73
- Dobsinszki, Gergely, 87
- Dodé, Réka, 153
- Dömötör, Andrea, 219
- Farkas, Richárd, 269
- Fegyő, Tibor, 87
- Ferenczi, Gergő, 153
- Gosztolya, Gábor, 59
- Hamerlik, Endre, 185
- Hatvani, Péter, 153, 257
- Héja, Enikő, 29, 153
- Hoffmann, Ildikó, 59
- Holl, András, 123
- Indig, Balázs, 219, 247
- Kádár, Máté Soma, 87
- Kahla, Mram, 41
- Kalivoda, Ágnes, 247
- Kiss, Gábor, 73
- Krishna, Swaroop, 137
- Lagos, Cortes Mátyás, 247
- Lakatos, Dorina, 3
- Lengyel, Mariann, 153
- Ligeti-Nagy, Noémi, 123, 153, 171
- Lipp, Veronika, 247
- Madarász, Gábor, 123, 153
- Mády, Katalin, 87
- Makrai, Márton, 247
- Mihajlik, Péter, 87
- Nagy, Dániel, 3
- Nemeskey, Dávid Márk, 197, 219
- Nyist, Milán Konor, 185
- Osváth, Mátyás, 29
- Pethő, Gergely, 137, 247
- Prótár, Noémi, 197
- Sárossy, Bence, 153
- Sass, Bálint, 247
- Simon, László, 247
- Skrabák, Boglárka, 171
- Svindt, Veronika, 59
- Szántó, Zsolt, 269
- Szécsényi, Tibor, 17
- Tóth, Gábor, 269
- Tóth, László, 59
- Üveges, István, 3
- Vadász, János Pál, 3
- Vadász, Noémi, 247
- Vándor, Péter, 97, 233
- Váradi, Tamás, 123, 153
- Varga, Kristóf, 153, 257
- Varga, Tamás, 153
- Virág, Nándor, 17
- Yang, Zijian Győző, 41, 123, 153, 257

