A kontextusablakon kihajolni nem veszélyes: jogi szövegek hatékony szemantikus keresése

Csányi Gergely Márk[1], Lakatos Dorina[1], Vadász János Pál[2], Nagy Dániel[1], Üveges István[1]

[1]MONTANA Knowledge Management Ltd.
[2]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 alappillérét 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 válóperes 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 hasonlítunk 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 ismertetett 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 rövidségéből 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 bíró, 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 teszthalmazként (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, Ítélőtáblák, 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 parafrazálja 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 készítettek 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* oszlopoban a tényállásra vonatkozó adatok láthatók.

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

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.

	T	**T (vázlat)**	**T-vázlat**	**kar./tok.**
hubert **sbert_hubert** **danieleff**	969,17	697,36	82,02	4,694
e5_large **e5_base** **cohere** **bge_m3** **jina_v3**	1140,09	819,21	108,41	3,862
mcontriever	1492,35	1091,38	139,82	2,908
openai_ada **openai_3_large**	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 tokennel.

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 [%]
openai	97,01	60,59
bge_m3, jina_v3	98,55	63,02

A magas lefedettségi 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 e problémát kezelni annak érdekében, hogy elkerüljük az alacsony hatékonyságú szövegreprezentációk 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.

[1] https://txt.cohere.com/introducing-embed-v3/ (2024.09.25)
[2] 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 [1]	cohere	512	1024
facebook/mcontriever-msmarco (Izacard és mtsai, 2021)	mcontriever	512	768
OpenAI text-embedding-ada-002 [2]	openai_ada	8191	1536
OpenAI text-embedding-3-large [2]	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 mondatszintű 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` modell 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) modell. 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[1], 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[1] 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 szöveghasonlósági feladatokban az MTEB ranglistán[3] (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 teljesítményveszteség nélkül.

Az OpenAI v3 beágyazási modelljeit szintén Matryoshka Representation Learning technikával tanították, amely lehetővé teszi a beágyazások lerövidítését anélkül, hogy azok fogalomreprezentáló tulajdonságai sérülnének (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

[3] 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 faktorral 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:

$$MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{r_i} \tag{1}$$

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 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. a táblázat ismerteti. Az MRR pontszámokat 100-zal megszoroztuk, így az elméleti maximális érték 100. A továbbiakban erre a megszorzott 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 átfedéses stride-ot jelenti, amely egyenértékű a chunked opcióval,
– **16**: a 16 tokenes átfedéses 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 LCS	trunc False	0 False	0 True	25% False	25% True	16 False	16 True
bge_m3	**93,67**	**93,67**	**93,67**	**93,67**	**93,67**	**93,67**	**93,67**
cohere	92,99	93,23	94,87	90,69	92,21	94,27	**95,03**
danieleff	77,09	77,09	**79,88**	76,46	78,35	78,23	**79,88**
fasttext	**53,72**	**53,72**	**53,72**	**53,72**	**53,72**	**53,72**	**53,72**
hubert	7,69	**12,12**	8,79	8,01	7,69	10,65	8,79
e5_base	85,13	86,81	88,01	86,67	87,45	87,42	**88,77**
e5_large	87,82	89,00	**92,45**	87,83	89,37	88,51	91,81
jina_v3	**93,08**	-	-	-	-	-	-
mcontriever	81.46	89.53	90.62	90.34	91.14	89.23	**91.42**
openai_3_large	**93,05**	-	-	-	-	-	-
openai_ada	**83,65**	-	-	-	-	-	-
sbert_hubert	34,81	**60,43**	57,31	57,17	57,91	58,56	58,11

A legrosszabb teljesítményt a `hubert` nyújtotta, amely rámutat a feladatspecifikus 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.

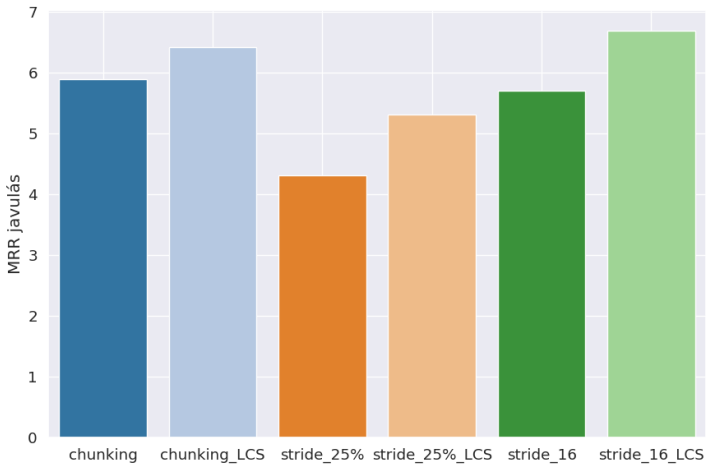
A következő szintet a `danieleff` modell képviselte, amelyet kérdés-válasz (Q&A) feladatra finomhangoltak, kevesebb, mint 200 példán. A modell kontextusablaka négyszer nagyobb (512), mint az `sbert_hubert` modellé (128). Az eredmények arra utalnak, hogy a kontextusablak méretkülönbsége és a feladatspecifikus 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 újrafelhasználhatósága 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 amp; 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)