

TDK-dolgozat

Szerző:

Guti Szilárd

DEBRECENI EGYETEM, INFORMATIKAI KAR

Python alkalmazás nagy nyelvi modellek mérésére és összehasonlítására

Belső témavezető:

Dr. Harangi Balázs (egyetemi docens)

Külső témavezető:

London András (Kodesage Zrt.)

Bírálatra való beküldéséhez hozzájárulok:

Szerző:

Guti Szilárd

Debrecen, 2024

Absztrakt

Az elmúlt évek mesterséges intelligencia fejlesztései átalakították az élet számos területét, köztük a munkahelyi folyamatokat, az oktatást és a szórakozást. A nagy nyelvi modellek kiemelkedő szerepet játszanak ebben a fejlődésben, több milliárd paraméterrel működve rendkívüli eredményeket produkálnak különféle területeken. Folyamatos fejlődésük szükségessé tette olyan módszerek kidolgozását, amelyek képesek hatékonyan mérni és értékelni ezeknek a modelleknek a teljesítményét.

Kutatásom során egy új, eddig nem létező eszközt hoztam létre, amely nagy nyelvi modellek mérésére szolgál. Ez az innováció már meglévő keretrendszerekre és technikákra épül, de a megvalósításában új perspektívát képvisel, mivel eddig nem alkalmazták a bemutatott módon. Az eszköz lehetővé teszi, hogy a nagy nyelvi modellek teljesítményét olyan környezetben mérjük, amely ipari szintű kihívásokat tükröz. Ilyen kihívás például az automatikus dokumentáció generálás, mely szorosan összekapcsolódik a kutatásommal, mivel ezáltal remekül kiértékelhetőek egy nagy nyelvi modell képességei.

A kutatás további célja az volt, hogy hozzájáruljak a Kodesage projekt fejlesztéséhez, amely egy ipari szinten alkalmazható mesterséges intelligencia megoldás. Ez a rendszer a legmodernebb technológiákat felhasználva szeretné megkönnyíteni a cégek munkáját, legyen szó programozásról, analitikai kérdésekről vagy a mindennapi munka felgyorsításáról.

Tartalomjegyzék

1. Bevezetés	4
2. Célkitűzés	5
3. A projekt felépítése	6
3.1 Előfeldolgozás.....	7
3.2 Szövegfeldolgozás	9
3.3 Hibrid visszakeresés.....	10
3.4 Generálás.....	12
3.5 Kiértékelés	13
4. Eredmények	16
4.1. A program működése	17
4.2. Egy szélsőséges eset bemutatása.....	24
5. Összefoglalás	25
6. Limitációk.....	25
7. Jövőbeli tervek.....	26
8. Irodalomjegyzék	27
9. Függelékek.....	29
1. függelék: A különböző konfigurációk által generált szövegek és LLM általi értékeléseik	29
2. függelék: A kiértékelés alatt használt mérőszámok	31

1. Bevezetés

A mesterséges intelligenciának (MI) az utóbbi években mutatott fejlődése befolyásolja az életünk minden részét, legyen az a munkahelyek dinamikájának alakulása, oktatás vagy szórakozás. Ez szorosan összeköthető a nagy nyelvi modellek (Large Language Model, LLM) forradalmasításával és széles körben elterjedésével. Ezek, az akár ingyenesen is elérhető, több milliárd paramétert tartalmazó modellek kiemelkedő eredményeket mutatnak több területen is, mint az ajánlási rendszerek, molekula felfedezés és jelentés generálás [2]. Mindehhez hatalmas mennyiségű tanító adat szükséges, valamint kifejlett modell architektúrák és mély tanulási technikák.

A nagy nyelvi modellek továbbfejlesztése és a generált eredmények javítása érdekében a RAG (Retrieval-Augmented Generation) külső adatforrásokból származó információkat vagy tudást használ fel, amelyek a bemeneti lekérdezés vagy az előállított kimenet kiegészítéseként szolgálnak [2]. Ilyen külső adatforrások lehetnek dokumentumtárak, tudás grafikonok (knowledge graphs) vagy vektor adatbázisok (vector databases), amelyek az adatokat vektorként reprezentálják, szemantikai jellemzőiket magasabb dimenziós térbe helyezve, ezáltal a szemantikai hasonlóságok és különbségek reprezentálódhatnak a vektorok közötti távolságokkal [3]. Ilyen módon a RAG befolyásolja a visszakeresési (retrieval) folyamatot, valamint az információkat a generálási szakaszban felhasználhatja, ezáltal pontosabb és relevánsabb választ adva. Így a RAG LLM-ek jóval kevesebb paraméterrel nagymértékben felülmúlják a visszakeresés nélküli nagy nyelvi modelleket [1]. Mindez az LLM-ek kontextuson belüli tanulása miatt lehetséges.

A kutatásom célja ezen RA-LLM-ek megértése, majd ezekhez egy kiértékelő szoftver megalkotása, utána pedig a Kodesage projektbe¹ integrálása. Ennek célja, hogy az ipari környezetben olyan mesterséges intelligenciát használó alkalmazásokat biztosítson más cégeknek, amik a szoftverfejlesztés minden folyamatát segíteni tudják, mint például a legacy kód megértése, vagy a mindennapi fejlesztési munka felgyorsítása. Mindehhez többféle tudásforrást, beleértve a teljes kódbázist, teszteket, dokumentációkat, a legelterjedtebb feladat és jegykezelő rendszereket is integrálja és felhasználja, hogy minél pontosabb válaszokat adjon. A csevegő alkalmazás lehetővé teszi, hogy természetes nyelvi kérdésekkel könnyebben megértse egy kezdő fejlesztő is az összetettebb kódrészleteket is, egy haladó kolléga bevonása nélkül. Egy másik alkalmazási területe a dokumentációgenerálás, ami jelentős terhet vehet le a

¹ <https://kodesage.ai/en>

fejlesztő csapat válláról. A kutatásom a nagyobb Kodesage projekt ezen részéhez kapcsolódik szorosan, ugyanis az általam elkészített kiértékelő rendszer, többek között a nagy nyelvi modellek hosszú szövegek generálásának tulajdonságait is méri. Rengeteg új LLM jelenik meg nap, mint nap, melyek kiértékeléséhez számos mérőszám és keretrendszer létezik, azonban olyan szoftveres megoldás, mely a felsoroltakat összefogja, és egységes módon biztosítja átfogó kiértékelésüket, valamint könnyen értelmezhető grafikus eredményeket biztosít, egyelőre hiányzik. Nagy nyelvi modellek, beágyazó modellek és egyéb paraméterek kombinációjának összehasonlításával bemutatjuk, hogy a projekt megfelelően működik.

A szoftver python programozási nyelven, a Google Colab platformon készült. Ezen ingyenesen felhasználható memória és tárhely, valamint integrált eszközök, mint a Gemini AI segítik a munkát. Az általunk felhasznált LLM keretrendszer a LlamaIndex, mely széleskörű eszközeivel és beépített moduljaival remekül felhasználható a fejlesztéshez. Az elkészült projekt a következő linken keresztül megtekinthető:

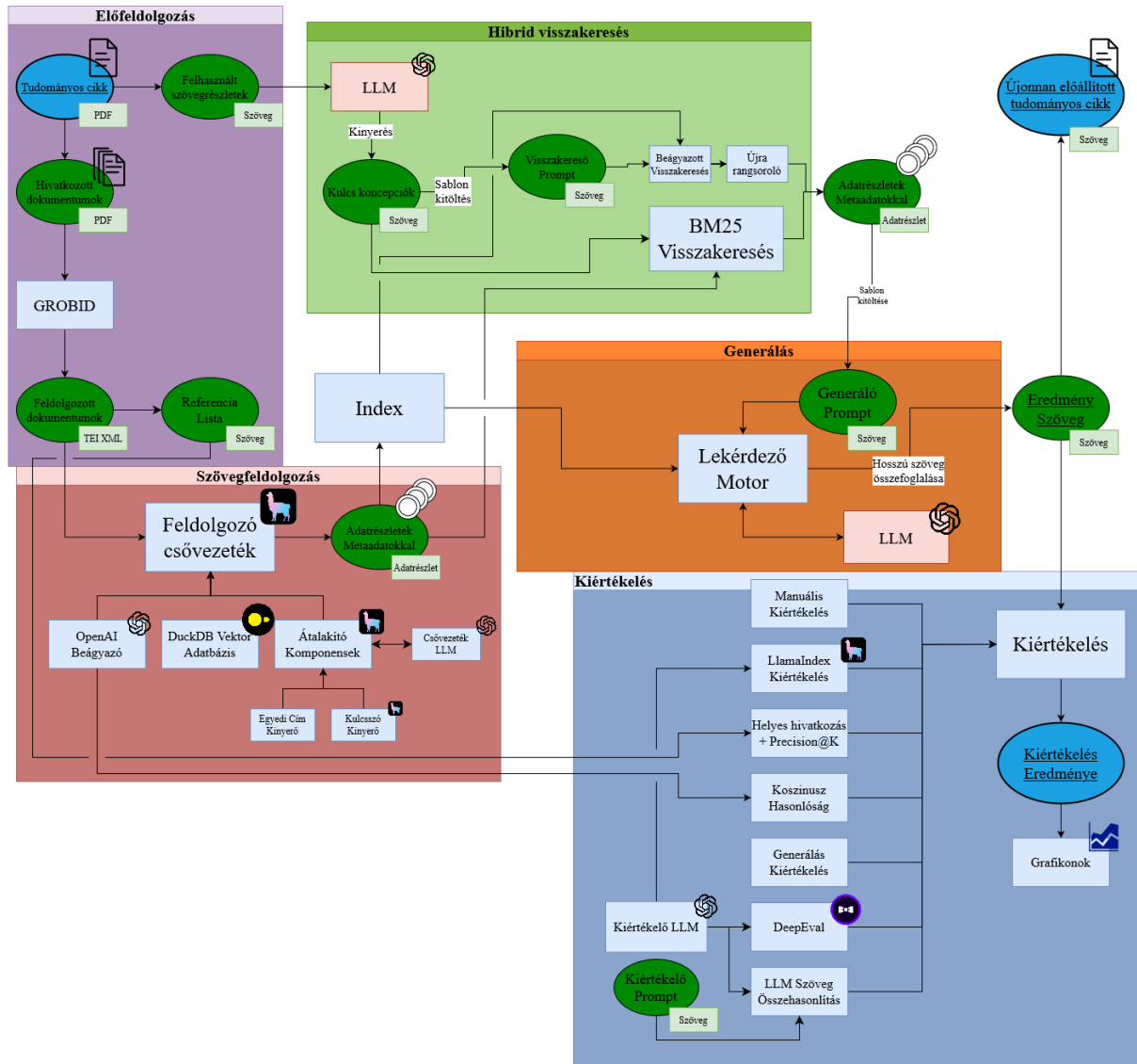
https://colab.research.google.com/drive/1O_df8Y6oQQYj0_WcdoiMozDgl-LIExVa?usp=sharing

2. Célkitűzés

A kutatás célja egy eszköz megalkotása, mellyel egyszerűen és pontosan lehet mérni, valamint összehasonlítani nagy nyelvi modelleket. Célunk a keretrendszer kifejlesztésével további kutatások és tesztelések megkönnyítése, azaz hiperparaméterek egyszerű módosításával személyre szabható teszteseteket lehessen futtatni, melyekről könnyen értelmezhető és a valóságot jól tükröző grafikonok, mérőszámok, valamint szöveges értékelések keletkeznek. További célunk az elkészült szoftver integrálása egy nagyobb, ipari környezetben működő projektbe.

3. A projekt felépítése

Az alábbi folyamatábrán (1. ábra) láthatóak a projekt működésének lépései. Ezek magukban foglalják a manuális, a segédprogram, valamint az elkészített főprogram folyamatait. Ezen lépések, a színezéssel is jelölt, csoportokba sorolhatóak, melyek nagy vonalakban bemutatják egy RAG LLM működését.



1. ábra: A program működésének folyamatábrája

3.1 *Előfeldolgozás*

Az első csoportba tartozó folyamatok leginkább a manuális állomány összegyűjtésével és a segédprogram használatával foglalkoznak. Ezt gyakorlatilag tekinthetjük a fejlesztett főprogram elindítása előtt végrehajtott egyéb lépéseknek.

Egy tudományos dokumentum a bemeneti PDF (Portable Document Format) állományunk, mely az általunk meghatározott tudományos területekkel foglalkozik. A projekt alatt ezek a számítógépes orvosi képfeldolgozásról, gráfelméletéről és kereső motorokról végzett felmérések (survey) voltak, melyek rendkívül fontosak a kutatás szempontjából. Az ilyen felmérések általánosan a meglévő irodalom áttekintését, különböző megközelítések, technikák és módszerek elemzését jelentik. Fontos szerepet játszanak abban, hogy átfogó képet nyújtsanak a kutatási terület jelenlegi állapotáról, és feltárják a további kutatások irányait. Továbbá összefoglalják az egyes kutatási körökhöz tartozó rengeteg, sokszor több száz tudományos cikket, amit alap esetben jelentős munka lenne átnézni, ilyen módon tömörítik a már meglévő tudást.

A megfelelően sokrétű mérések érdekében több dokumentumot is felhasználtunk, melyek témája mind eltérő volt. Ezek mellett a szövegek felhasznált mennyisége is változó, míg a hibrid kereső motorokról szóló tudományos írás majdnem összes bekezdését, addig a matematika témájával foglalkozó cikk csak két szekcióját vettük igénybe. Különösen hasznos, hogy az egyes témák kifejtéséhez számos hivatkozást tartalmaznak, melyekkel könnyen ellenőrizhetjük a nagy nyelvi modellek visszakereső, majd generatív képességét is. Ilyen módon a hosszú szövegek generálásának problémájával is foglalkozik a kutatásom.

Azonban ezen írások struktúrája limitációkat is felvet, előfordul, hogy egy ilyen felmérés több száz hivatkozott dokumentumot tartalmaz, amelyeket olykor nehéz elérni annak érdekében, hogy a RAG folyamatban felhasználjuk. Ez a nehézség a hivatkozott tudományos cikkek eléréséhez szükséges anyagi feltételekre, valamint a szükséges manuális munkára egyaránt értendő.

A főprogram futása alatt a már kiválasztott kutatási terület felmérésének egy paragrafusát használtuk, melyet a folyamat alatt újra elő szeretnénk állítani. Mivel a program célja ezeket a szövegrészeket újra alkotni, ezért a továbbiakban előállítandó bekezdésekként hivatkozunk rá. A szoftver sima szöveggént várja el ezen részek feltöltését, külön állományokban. A projekt elkészítése közben ezeket a szövegeket azért, hogy jobban átlássuk és megértsük a generálási folyamatot, manuálisan válogattuk ki.

A program futtatásához továbbá szükséges volt a kutatási terület felmérésének dokumentumában hivatkozott további tudományos cikkekre és megjelenésekre. Jelenleg ezen dokumentumok letöltésének folyamata manuális, mely sok időt igényel, így a jövőben kifizetődő lenne automatizálni.

Az így megkapott szövegek karakteres azonosításához és feldolgozásához a GROBID (GeneRation Of Bibliographic Data) nyílt forráskódú eszközt használtuk, amely automatikusan képes tudományos cikkek és egyéb dokumentumok szövegét, valamint metaadatait, mint szerzők, címek, referenciák kinyerni. Mindezt összetett technológiák, például mélytanulási modellek és keretrendszerek felhasználásával. A projekt GitHub-on² érhető el, ahol folyamatosan fejlesztik. A kutatás szempontjából azért fontos, mert hatékonyan képes strukturált adatokat előállítani különböző formátumokból, mint például a hivatkozott dokumentumok, ami jelentősen megkönnyíti a nagy adathalmazok automatizált feldolgozását.

Lokális futtatásához Docker konténerben konfiguráltuk a környezetet, majd egy szövegfeldolgozó segédprogramot fejlesztettünk hozzá. Ez a python program képes kezelni az állományok beolvasását és mentését, valamint a kommunikációt a konténerizált GROBID végpontjaival. A minél gyorsabb és hatékonyabb végrehajtás érdekében a programban figyelembe vettük a párhuzamos dokumentum feldolgozó képességét is az eszköznek.

Az előfeldolgozó segéd szoftverünk tehát képes a megadott cikkeket átalakítani egy univerzálisan használt formátumra (TEI/XML). Az előre definiált struktúrája alapján könnyen feldolgozható, az egyes részeket, bekezdéseket, címet, szerzőket és hivatkozott dokumentumokat egyszerűen elérhetjük. A segédprogram ezen formátum bejárásához a BeautifulSoup³ nevezetű modult használja.

Limitációt jelentett, hogy egyes helyeken, mint néhány dokumentum címében, a szövegfeldolgozó a különleges karaktereket vagy a komplexebb elrendezéseket rosszul ismerte fel. Ehhez a jelenséghez hozzájárult, hogy a tudományos cikkek különféle megjelenési helyeik miatt sokszor teljesen más felépítésűek. Ez szükségessé tette, hogy manuálisan átnézzük az egyes kimeneti állományokat, mely ezen lépés jövőbeli automatizálását hátráltatja.

Szükséges volt továbbá a bemeneti állományban található hivatkozások listájának kinyerése, a megfelelő hivatkozások visszakeresésének ellenőrzéséhez. A listát egy előre definiált formátumban adjuk meg, amely könnyen beolvasható szótár adatstruktúrába.

² <https://github.com/kermitt2/grobid>

³ <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

A szövegfeldolgozó segédprogram által létrehozott állományokban tehát összességében jól működött a karakterfelismerés és szöveg tagolás is a kutatás alatt. Néhány esetben, leginkább tesztelés közben, azonban észrevehetőek voltak kisebb hibák, melyek manuális korrigálást igényeltek.

3.2 Szövegfeldolgozás

A szövegfeldolgozás a már az előfeldolgozó lépések által megfelelő formátumra hozott szövegrészeket vektor adatbázisba, valamint index adatstruktúrába rendezését jelenti. Ezen lépések biztosítják, hogy dokumentumrészek és metaadatok a RAG LLM számára elérhetőek és felhasználhatóak legyenek.

Az általunk használt keretrendszer részeként a feldolgozó csővezeték felelős az adatok előfeldolgozásáért és kisebb darabokra (chunk) bontásáért. Ez a folyamat azért fontos, mivel a beolvasott hosszabb dokumentumok kezelése így sokkal hatékonyabbá válik, és megkönnyíti a későbbi visszakeresést. Ezen kívül további lépéseket is biztosít, mint például a beágyazások előállítás vagy különböző metaadatok hozzárendelése az adatdarabokhoz. Ez lehet egy minden más lépéstől független LLM által generált kulcskonceptiók megadása vagy a megfelelő cikk címének hozzátársítása.

Az utóbbi példa a visszakeresés kiértékeléséhez különösen fontos lépés, melynek nem létezett ebben a keretrendszerben a csővezetékbe építhető megoldása, ezért egyedi implementálás volt szükséges. A feldolgozó csővezeték hasonló átalakító komponensekkel szabható személyre, általában sokféle metaadat hozzárendelését biztosítják az adatdarabokhoz. Ezek nagy részének működéséhez egy független LLM is szükséges.

A feldolgozás egyik fontos lépése az embedding (beágyazó, magasabb térbe átalakító) algoritmusok végrehajtása, amelyek vektormátrixokat hoznak létre, így a dokumentumokat egy numerikus térben, a gyakorlatban lebegőpontos számok listájában, reprezentálják. A két vektor közötti távolság a hasonlóságokat méri, azaz a kis távolságok nagy, míg a nagy távolságok pedig alacsony szemantikai egyezést jeleznek [9], valamint gyorsan kiszámíthatóak. A kutatás közben használt embedding megoldásokhoz, a nagy nyelvi modellekhez hasonlóan, az OpenAI felhő alapú szolgáltatásait használtuk. A tesztelés jelentős részében két ilyen, név szerint a *text-embedding-ada-002* és *text-embedding-3-small* beágyazó modellt alkalmaztunk. Ezek a használat alapján, a tokenek száma szerint határozzák meg a fizetendő összeget.

Az így keletkezett vektorok egy ezek kezelésére alkalmas adatbázisba kerülnek, melyek száma a nagy nyelvi modellek elterjedésével jelentősen megnőtt az utóbbi években. Mivel a

különböző tudományterületeket átfogó összehasonlítások a vektoros keresési algoritmusok és rendszerek között ritkák [10], ezért a kutatásban egy könnyen kezelhető és hatékony vektor adatbázis kezelő rendszer használatára törekedtünk, ahelyett, hogy további méréseket vagy összehasonlításokat alkalmaztunk volna. Ezen céljainknak a *DuckDB* megfelelt, egyszerűen kezelhető, ugyanakkor nagyon gyors feldolgozási időt tesz lehetővé a beágyazott vektorokon.

Az adatdarabok reprezentálásához LlamaIndex keretrendszerbe épített, visszakereső algoritmusok és feldolgozó csővezeték eredményeinek típusát használtuk⁴. Ez magában foglalja az eredeti szöveg feldarabolt részeit, ezekhez tartozó egyedi azonosítót, hozzájuk tartozó metaadatokat, valamint a beágyazott értékeit. Lehetőség van a tárolt adatok hosszának változtatására, amely különösen hasznos hosszabb dokumentumoknál. Általánosabban az adatstruktúra tartalmazhatja képek vagy más formátumok részleteit is. A keretrendszerben könnyen, magas szinten használható a típus, azonban lehetőséget nyújt a testre szabásra is.

A csővezeték futása után az adatok mentésre kerülnek JSON formátumban, ami ezután a későbbi futtatásokkor könnyen újratölthetők és újrahasználhatók anélkül, hogy újból erőforrásigényes és költséges folyamatokat kelljen futtatnunk. Továbbá ez jelentős időmegtakarítást eredményez, különösen nagy adatállományok feldolgozásakor.

3.3 Hibrid visszakeresés

A visszakeresés, valamint generálás lépéseihez felhasználtunk egy nagy nyelvi modellt, pontosabban azt, melyet ki fog értékelni az elkészített szoftver. Ehhez használható külső, valamilyen végponton keresztül elérhető, vagy lokálisan futtatott is. A kutatás alatt OpenAI által biztosított GPT családba tartozó modelleket használtunk, melyek használat alapúak, tehát minden bemeneti és kimeneti token után kötelező fizetni egy, az LLM komplexitásától és hatékonyságától függő összeget. A szoftver fejlesztése alatt elsőnek GPT-3.5 Turbo, majd GPT-4o mini típusú nagy nyelvi modelleket alkalmaztunk és teszteltünk. A jövőben érdemes lehet más LLM családból, mint például LLaMA, választani vagy egy eltérő architektúrával rendelkező modellt, mint BERT, is kipróbálni [4].

Továbbá szükséges lesz az index adatszerkezet, amely lehetővé teszi, hogy gyorsan elérjük a releváns kontextust egy lekérdezéshez. A keretrendszer esetében ez a RAG alapja, amely a dokumentumokból épül fel, valamint ebből képezhetőek lekérdező vagy csevegő motorok. A már előzőleg említett adatrészlet objektumok tárolják az adatokat. Az egyik

⁴ https://docs.llamaindex.ai/en/stable/module_guides/loading/documents_and_nodes/usage_nodes/

leggyakrabban használt típusa a vektorokra épülő változat [6], melyet a kutatás alatt felhasználtunk. Az index továbbá a generáló lépésekben is jelentős szerepet játszott.

Első lépésként az előállítandó bekezdésből nyertük ki a témához legjobban kapcsolódó, úgynevezett kulcskonceptiókat. Ezen lista létrehozásához az általunk kiértékelni akart nagy nyelvi modellt használtunk fel, egy megfelelő prompt (szöveges utasítás) segítségével, mely tartalmazza az eredeti szöveg témáját is. A téma elhelyezésével általában elmondható, hogy jobb eredményeket kaptunk, ezt azonban befolyásolhatja az LLM saját tudása is. A BM25 rangsoroló funkció alapján működő visszakereső egység működéséhez elegendő a prompt nélküli lista megadása.

Ezen előre megadott sablonba kerülnek bele a kulcskonceptiók, ilyen módon megalkotva a beágyazott visszakereséshez felhasználható prompt-ot. Ez az utasítás segít a visszakereső algoritmusnak abban, hogy releváns adatokat adjon vissza a külső forrásokból, majd ezeket felhasználjuk a válasz generálása közben. Azaz a folyamat során biztosítjuk a megfelelő kontextus figyelembevételét, amely javítja a visszakeresés hatékonyságát és a generált válasz minőségét.

Az általunk használt hibrid visszakeresés, amely ötvözi a szórványos BM25 és a sűrű beágyazáson alapuló módszereket, számos előnnyel járhat. A BM25 visszakeresés jól teljesít olyan lekérdezések esetében, amelyek szorosan kapcsolódnak az adott szöveg lexikai elemeihez, vagyis kulcsszó-alapú egyezéseknél, ezért alkalmazhattuk magában, sablon nélkül a kulcskonceptiókat. Ezzel szemben a sűrű beágyazás a lekérdezések mélyebb, szemantikai hasonlóságaira fókuszál, így képes olyan tartalmakat is visszaadni, amelyek nem foglalják magukba a kérdés konkrét szavait, azonban fogalmilag relevánsak. Ehhez a visszakeresési technikához belátható, hogy segítséget nyújt egy jól megfogalmazott prompt, mely a kérdés szemantikáját magyarázza.

A két technika kombinálásával olyan rendszert hozhatunk létre, amely mindkét megközelítés hátrányait kiküszöböli, valamint az előnyeit egyesíti. [8] Az ilyen rendszerek különösen hatékonyak lehetnek nagy adathalmazok esetében, ahol mind a sebesség, mind a találatok mélysége kritikus jelentőségű. Más kutatási területeken kimutatták, hogy a BM25 és a sűrű beágyazás együttes használata növelte a visszakeresett dokumentumok relevanciáját és javította a válaszok minőségét [12].

A sűrű visszakereső algoritmushoz a LlamaIndex beépített modulját használtuk, mely a keretrendszerben használt vektor index adatstruktúrában teszi lehetővé nagy nyelvi

modelleknek a beágyazott dokumentumok közötti keresést. Az adatok ilyen vektoros ábrázolása lehetővé teszi a hatékony szemantikai alapú visszakeresését, a vektorok közötti térbeli távolságot felhasználásával. Ez különösen hasznos nagyméretű szövegtárak esetén, ahol a kulcskonceptiók szerinti egyezés helyett mélyebb fogalmi kapcsolatokat kell keresnünk.

Ebben a lépésben paraméterként megadható, hogy mennyi legpontosabb találatot szeretnénk visszakapni. A kutatás alatt a húsz legjobb dokumentumrészlettel dolgoztunk, melyből az újra rangsoroló algoritmus a négy legpontosabbat választja ki, ugyanis ennyivel már megfelelően jó értékeket kapunk. Továbbá az LLM bemeneti korlátozására is figyelniünk kellett. Ezen lépés után tehát egy újra rangsoroló algoritmus fut, amely egy nagy nyelvi modell segítségével finomítja a találatok sorrendjét, javítva ezzel a végső eredmények minőségét.

A szórványos visszakeresési algoritmushoz szintén a keretrendszerben már implementált BM25 megoldást alkalmaztunk. Ez egy valószínűségi keresési modell, amely a kifejezés gyakoriság és az inverz dokumentumgyakoriság alapján pontozza a dokumentumokat. Jelentős előrelépést jelent a mesterséges intelligencia RAG-ban, mivel pontosabb és relevánsabb keresési eredményeket biztosít [13]. A lépés folyamán tetszőlegesen állítható a legpontosabban illeszkedő dokumentum részletek száma, melyet a kutatás alatt a sűrű visszakeresésnél beállított paraméterrel megegyezően négynek adtam meg.

3.4 Generálás

Ezen lépések többek között a nagy nyelvi modell generatív folyamatait tartalmazzák. A végrehajtáshoz szükség lesz a visszakereső technikák által adott adatrészletekre, valamint a már tárgyalt index adatszerkezetre.

A korábban említett módon az indexből létrehozható egy lekérdező motor⁵, mely lehetőséget nyújt természetes nyelvű lekérdezéseket futtatni az adatokon. Lehetséges több és komplexebb indexből is összeállítani, amely további testreszabhatóságot biztosít. A projektben ez a lekérdező motor került felhasználásra a kulcskonceptiók előállítására, valamint a visszakeresett adatrészletek összefoglalására is. A csevegő motorral ellentétben az egyes lekérdezések egymástól elkülönülnek, nem osztoznak a kontextuson, ezáltal átláthatóbban és egyértelműbben tudjuk értékelni az egyes lépéseket.

A folyamatábrán (1. ábra) látható LLM megegyezik a kulcskonceptiók kinyerésénél felhasználttal. A generálás folyamata gyakorlatilag a visszakeresett adatrészletek, ezáltal egy

⁵ https://docs.llamaindex.ai/en/stable/module_guides/deploying/query_engine/

hosszú szöveg, összefoglalása. Ennek kiértékelésére számos megoldás létezik⁶, melyeket a kutatás alatt felhasználtunk.

Továbbá szükséges volt egy generáló prompt, ami a generáló lépésben használ fel a lekérdező motor, és ezáltal az LLM is. Megfogalmazásában a lehető legpontosabban adtuk meg az elvárt lépéseket, amelyek magukban foglalták a többek között az összegzés feladatát, a feldolgozandó szövegrészleteket, valamint kisebb segítségeket a lehető legjobb létrehozás érdekében. Ilyenfajta segítség volt az elvárt szöveg hossza, valamint témája. Azért, hogy a nagy nyelvi modell a lehető legkevesebb saját tudását használja, külön kiemeltük a fontosságát, hogy csak az általunk megadott szöveget használja fel.

A generálási folyamat eredménye tehát egy új szövegrészlet, melyet a kiértékelésnél használunk fel. Az ilyen módon létrehozott sima szövegek összessége fogja kiadni az újonnan generált dokumentumot. A kutatás alatt kontextusban megfelelő és elég nagy pontosságú fejezeteket sikerült generáltatni, melyeket a továbbiakban tárgyalt kiértékelési részek is bizonyítanak.

3.5 *Kiértékelés*

Az itt tárgyalt lépések az újonnan létrehozott szövegrészleteket kiértékelő folyamatok egyvelege, amely magában foglal egyszerű és manuális, valamint komplex és automatikus kiértékelő lépéseket. Ezen lépések adják a projekt egyik végeredményét, azaz a nagy nyelvi modellt minősítő mérőszámokat. Az ilyen módon kapott értékeket a későbbi feldolgozáshoz egy általunk implementált osztályba szervezzük, ezen objektumokat tároljuk, majd lementjük.

A kiértékelő lépések közül a legegyszerűbb, azonban a legtöbb munkát igénylő, a szövegek manuális összehasonlítása. Azaz az elvárt és generált szövegrészletet szükséges összehasonlítanunk, főleg a pontosságot és kontextust figyelve. Belátható, hogy a szemantika összeméréséhez a kiértékelőnek saját háttértudásra van szüksége.

Az általunk használt keretrendszer tartalmaz LLM kiértékeléshez felhasználható modulokat. Ezek közül kettőt, a faithfulness-t (forráshoz való hűség) és a relevancy-t (relevancia) alkalmaztuk. Ezekhez egy numerikus mérték tartozik, amelynek legnagyobb és ezáltal legjobb értéke 1 lehet, valamint egy logikai érték, ami azt jelzi, hogy átlépi-e az általunk választott küszöbértéket, ezáltal elfogadjuk-e helyesnek. A faithfulness azt biztosítja számunkra, hogy a modell nem hallucinált [5], azaz nem generált olyan szöveget, ami

⁶ https://cookbook.openai.com/examples/evaluation/how_to_eval_abstractive_summarization

szemantikailag helyesnek tűnik, azonban nem alátámasztható vagy szimplán helytelen. A relevancy mérőszám abban hasznos, hogy megmérjük és eldöntsük, az adott válasz ténylegesen megválaszolja-e a lekérdezést [7].

A hibrid visszakeresési módszer, ezáltal az LLM, kiértékeléséhez egy egyedinek mondható megoldást alkalmaztunk. A tudományos cikkekre jellemző szövegek közötti hivatkozásokat emeltük ki az egyes részletekből, melyeknek címét egy előre összeállított listából határoztuk meg. Az így kapott értékeket képesek vagyunk összehasonlítani a visszakeresés alatt kapott adatrészletek metaadatokban tárolt címével. Ehhez az összehasonlításhoz nem teljes szöveges alapú módszert, hanem egy szofisztikáltabb, RapidFuzz nevezetű külső modult használtunk. Ez a könyvtár hatékonyan képes karaktersorozatok nem egyértelmű hasonlóságát észrevenni, mely az előfeldolgozó kisebb hibáit hivatott kiküszöbölni.

Az eredmények könnyebb értelmezhetősége érdekében a címeket egy táblázatban jelenítjük meg a felhasználó felé. Ezt egy általunk készített HTML sablon kitöltésével érjük el, melyben elsőnek csak az eredeti szövegben megtalálható, de a visszakereső által nem megtalált címek állnak. Ezt követik az eredeti szövegben és a visszaadott adatrészek között szintén megjelenő elemek, a táblázatot pedig csak a visszakereső által visszaadott címek zárják.

Ezen táblázat logikája alapján implementáltunk egy, a visszakeresést értékelő mérőszámot, melynek neve `precision@k`. Ez a metrika segít kiértékelni a rendszer hatékonyságát abban, hogy mennyi releváns találatot ad vissza a legmagasabban rangsorolt visszakeresett adatrészletek között. Kiszámítása viszonylag egyszerű, a szövegrészletben hivatkozott és a visszakeresés folyamat végén visszaadott dokumentumok számának, valamint az összesen visszaadott adatrészletek számának hányadát tekintjük.

További kiértékelési lépésben a generált és az eredeti szövegrészletet az eddig is használt beágyazó algoritmus segítségével egy magasabb dimenziós térbe helyezzük, majd ezen numerikus vektorok között koszinusz értéket számolunk. Ezen vektorok között számolt koszinusz-hasonlóság azonban nem képes tökéletesen kezelni a szövegek szemantikai jelentését [11], ennek ellenére ez egy elterjedt módja a szövegek hasonlóságának mérésére. Az egyes beágyazó modellek másféle tartományokban határozzák meg ezt az értéket, a kutatás alatt használt hosszabb szövegek hasonlósági értéke sosem csökkent 0.60 érték alá (0-tól 1-ig tartó skálán), valamint sosem sikerült tökéletes egyezést, ezáltal 1-es értéket elérnünk.

Felhasználtunk egy nyílt forráskódú LLM kiértékelő keretrendszert⁷, amely magas szintű és nagyon könnyen használható eszközöket biztosít erre a feladatra. Többek között a legtöbbet használt mérőszámokat is támogatja, melyeknek megoldását tudományos cikkekre alapozzák. Komplexebb feladatokhoz testreszabhatóak az egyes mérési lépések. Az eszközök hozzájárulnak mind a kvalitatív, mind a kvantitatív kiértékeléshez, amelyek segítik a modellek erősségeinek és gyengeségeinek azonosítását. Egyes mérőszámok működéséhez elengedhetetlen egy a program többi részétől független, általában alacsony hőmérsékletű LLM, mely a kiértékelést végzi. Ezzel ellentétben léteznek beépített eszközök, melyek nem igényelnek ilyen modellt.

Egy másik lépésben egy független nagy nyelvi modell hasonlítja össze az elvárt, valamint a generálás folyamat végén kapott szövegrészletet. Ehhez egy kiértékelő prompt-ot használ fel, melynek szerepe létfontosságú a folyamatban. A projektben ez a kiértékelő lépés szemantikai és szintaktikai összehasonlítást is végez, erősen az előbbire koncentrálva. A visszakapott értékelés tehát tartalmaz egy hasonlósági százalékot 0-100% skálán, valamint egy rövid összefoglalót és magyarázatot, hogy miért ezt a választ adta. Ezzel a prompt alkotási technikával önreflexióra készítjük a nagy nyelvi modellt, mely által jobb eredményeket várunk.

Az említett kiértékelő prompt egy újabb hiperparaméternek tekinthető, amely szabadon megváltoztatható. A projekt fejlesztése alatt azt tapasztaltuk, hogy amennyiben bizonyos százalékonként példákat és magyarázatot adunk meg, úgy sokkal jobb eredményt kaphatunk. Ugyanakkor ez azt is jelenti, hogy sokkal többször jelenik meg egy példával ellátott érték, mint egy nem említett.

A kiértékelés több lépésében is szükséges egy már említett független LLM, melynek a lehető legkisebb a hőmérséklet paramétere, azaz a lehető legdeterminisztikusabb. Ez szükséges a megbízható és pontos értékeléshez. Ajánlott kiértékelő nagy nyelvi modellnek az éppen vizsgált modellnél több paraméteres, már helytállt modellt választani. Az elkészült programban egyszerűen lehet ezt a paramétert módosítani.

A generált válaszok kiértékeléséhez további két gyakran használt eszközt alkalmaztunk. Ezek a ROUGE (Recall-Oriented Understudy for Gisting Evaluation) és a BERTScore, melyek az elvárt és a létrehozott szövegeket hasonlítják össze, és a szövegrészlet különböző aspektusait elemzik. Az előbbi kizárólag a szövegek szintaktikai átfedéseit vizsgálja, azonban ezeket több

⁷ <https://docs.confident-ai.com/>

kombinációban is. Ilyen módon beszélhetünk ROUGE-1, azaz önmagában álló szavak, ROUGE-2, tehát szó párok vagy ROUGE-L mérőszámokról, melyek közül az utolsó a közös leghosszabb szekvenciát vizsgálja.

Ezzel ellentétben a BERTScore a szövegek közötti szemantikai hasonlóságot vizsgálja. A BERT (Bidirectional Encoder Representations from Transformers) nyelvi modell segítségével numerikus vektorokká alakítja a szavakat, majd ezen vektorok hasonlóságát méri. Ezzel a jelentésben található, mélyebb összefüggéseket is figyelembe véve.

Az eredmények tárolása érdekében létrehoztunk egy egyedi osztályt, mely az értékeket kezelni képes. Továbbá ezáltal valósítjuk meg az eredmények kimeneten való megjelenítését vagy az állományba való mentéshez szükséges szerializálást is. A struktúráját tekintve egy ilyen objektum tartalmazza a prompt-ot, a generált választ, az elvárt szövegrészletet, valamint az összes általunk használt mérőszámot.

A szoftver az eredmények egyszerű értelmezéséhez különféle grafikonokat generál. Ezek megalkotásához a Plotly⁸ által biztosított nyílt forráskódú grafikus könyvtárat használtuk fel. A modulban számos grafikonfajta megtalálható, melyek közül egyelőre a hisztogram, pókhálódiaagram, hő térkép és buborékdiagram került be a projektbe.

Az összes elvárt szövegrészleten lefuttatott lépések végeredményül egy újonnan létrehozott tudományos cikket alkotnak. Ezeket az új dokumentumokat manuálisan összevethetjük az eredeti megfelelőjükkel. Érdekes kutatási témát jelenthet emberi tesztelés formájában összemérni ezeket, azaz, hogy melyik az érthetőbb, pontosabb vagy csak összességében jobban megfogalmazott.

4. Eredmények

A kutatás alatt jelentős fejlesztői munkával sikeresen megalkottunk egy összetett és jól működő eszközt a nagy nyelvi modellek értékelésére, valamint összehasonlítására. A projekt számos részén lehetséges a hiperparamétereket, mint a darabolás mértéke, nagy nyelvi modellek vagy beágyazó modellek cseréje, ami által könnyen testreszabható anélkül, hogy a működést meg kellene változtatni. Ezen eredmények bemutatásához egy konkrét példán keresztül szemléltetjük a program működését kitérve egy szélsőséges esetre is.

⁸ <https://plotly.com/python/>

4.1. A program működése

A program felépítésénél tárgyalt lépéseknél több *konfigurációt* is szemléltetünk. A továbbiakban egy ilyen konfiguráció az adott futtatáskor felhasznált LLM neve, a beágyazott modell neve, a kiértékelő LLM neve, az adatrészlet nagysága, átfedése, valamint a beágyazott és BM25 visszakereséshez használt top k és csomópont darab hiperparaméter együttesét jelenti.

A tesztelés alatt felhasználtuk a dokumentumok közül az orvosi képfeldolgozás témáját tárgyaló szövegrészletet (1. szövegdoboz). A további lépések bemutatásához ezt a szöveget alkalmazzuk. Fontos észrevennünk, hogy a részlet tartalma csak egy konkrét téma köré rendeződik, pontosabban a Swin-Unet hálózati architektúráról ír, valamint található benne hivatkozás, mely a kiértékelés szempontjából lesz érdekes.

2.4 Swin-Unet

Swin-Unet network architecture was proposed by Cao et al. [9] in 2023. The model structure is depicted in Fig. 4. Unlike Trans-Unet, which replaces the convolutional blocks in the U-Net encoder with Transformer blocks, Swin-Unet utilizes Swin Transformer blocks [11] to extract hierarchical features from the input image. Swin-Unet is the first purely Transformer-based U-shaped architecture. Swin Transformer extends the traditional Transformer's one-dimensional sequence to two-dimensional image blocks and adopts a hierarchical attention mechanism to capture features over a larger receptive field. This structure is similar to the hierarchical structure in convolutional neural networks and serves feature extraction. Additionally, Swin Transformer introduces the mechanism of shifting windows on top of the self-attention mechanism. By limiting the attention calculation to windows in the vicinity of the current region, Swin-Unet better preserves positional information and further improves the model's performance.

In Swin-Unet, Swin Transformer is applied in the encoding, bottleneck, and decoding modules. Importantly, the compression of each layer's features in Swin-Unet is smaller than in TransUNet. Instead of adding additional Transformer modules, Swin-Unet replaces the convolutional modules with Transformer modules, effectively reducing the number of model parameters. Overall, Swin-Unet leverages the advantages of Swin Transformer and U-Net to provide a promising approach for medical image segmentation. It has demonstrated competitive performance in various segmentation challenges and benchmarks.

1. szövegdoboz: A további lépésekben felhasznált szövegrészlet

A szövegrészletből kiemelt kulcs fogalmak és koncepciók gyakorlatilag tömörítik a szöveg tartalmát. Ez az LLM azon tulajdonságát teszteli, hogy mennyire képes megérteni a részlet szemantikai lényegét. Manuális kiértékeléshez ezeket a koncepciókat a kimeneten az 1. táblázatnak megfelelően jelenítjük meg. A konfiguráció 3-nál tapasztalható hosszabb kulcsmondatok valószínűleg annak tudhatók be, hogy az LLM képes többféleképpen értelmezni a koncepció jelentését. Azt, hogy a nagy nyelvi modell mennyi kulcsértékben tömöríti egy fejezet lényegét egy újabb hiperparaméter, melynek beállítása további kutatást igényel.

	KONFIGURÁCIÓ 1	KONFIGURÁCIÓ 2	KONFIGURÁCIÓ 3
LLM	gpt-4o-mini	gpt-3.5-turbo-0125	gpt-4o-mini
Beágyazott modell	text-embedding-3-small	text-embedding-3-small	BAAI/bge-small-en-v1.5
Kiértékelő LLM	gpt-4o-mini	gpt-4o-mini	gpt-4o-mini
Adatrészlet méret	1024	1024	1024
Adatrészlet átfedés	20	20	20
Top_k	4	4	4
Csomópont darab	4	4	4
Kulcskonceptiók	1. Swin-Unet 2. Transformer blocks 3. Hierarchical features 4. U-shaped architecture 5. Self-attention mechanism 6. Two-dimensional image blocks 7. Shifting windows 8. Feature extraction 9. Medical image segmentation 10. Model performance	Swin-Unet, Cao et al., 2023, Transformer blocks, hierarchical features, first purely Transformer-based U-shaped architecture, Swin Transformer, hierarchical attention mechanism, shifting windows, feature extraction, compression of features, model parameters, competitive performance.	1. Swin-Unet network architecture 2. Purely Transformer-based U-shaped architecture 3. Swin Transformer blocks for hierarchical feature extraction 4. Hierarchical attention mechanism for larger receptive fields 5. Shifting windows mechanism for improved positional information 6. Encoding, bottleneck, and decoding modules utilizing Swin Transformer 7. Reduced feature compression compared to Trans-Unet 8. Replacement of convolutional modules with Transformer modules 9. Reduction in model parameters 10. Competitive performance in medical image segmentation challenges

1. táblázat: Példák az egyes konfigurációk használatával visszkapott kulcskonceptiókra

A hibrid visszakeresési lépésben a már előállított kulcskonceptciókat felhasználva kapjuk vissza a konfigurációban feltüntetett darabszámú adat csomópontokat. Azaz a RAG visszakeresés javításához alkalmaztunk beágyazott, valamint BM25 rangsoroló keresést is.

Ahhoz, hogy ezen lépést egyszerűen értékelni tudjuk, egy saját mérőszámot hoztunk létre, amelynek működése a szövegrészletekben található, egységes formátumú hivatkozásokra alapul, valamint a hivatkozás egyezés nevet viseli. A működtető algoritmus kiemeli, majd felsorolja, hogy milyen hivatkozott tudományos cikkek vannak jelen a szövegben, ezután összeegyezteti a visszakereső lépés által kapott dokumentum részletekkel. Ezt a kimeneten egy táblázatban (2. táblázat) jelenítjük meg a könnyű értelmezhetőség érdekében, ahol az egyező címek háttere zöld, míg a nem egyező dokumentumok háttere fekete színű. Ezen tulajdonság az eddig elérhető szoftver csomagokra nem jellemző, általában nem biztosítanak ilyen vizuális kimenetet a kiértékelés lépéseiről.

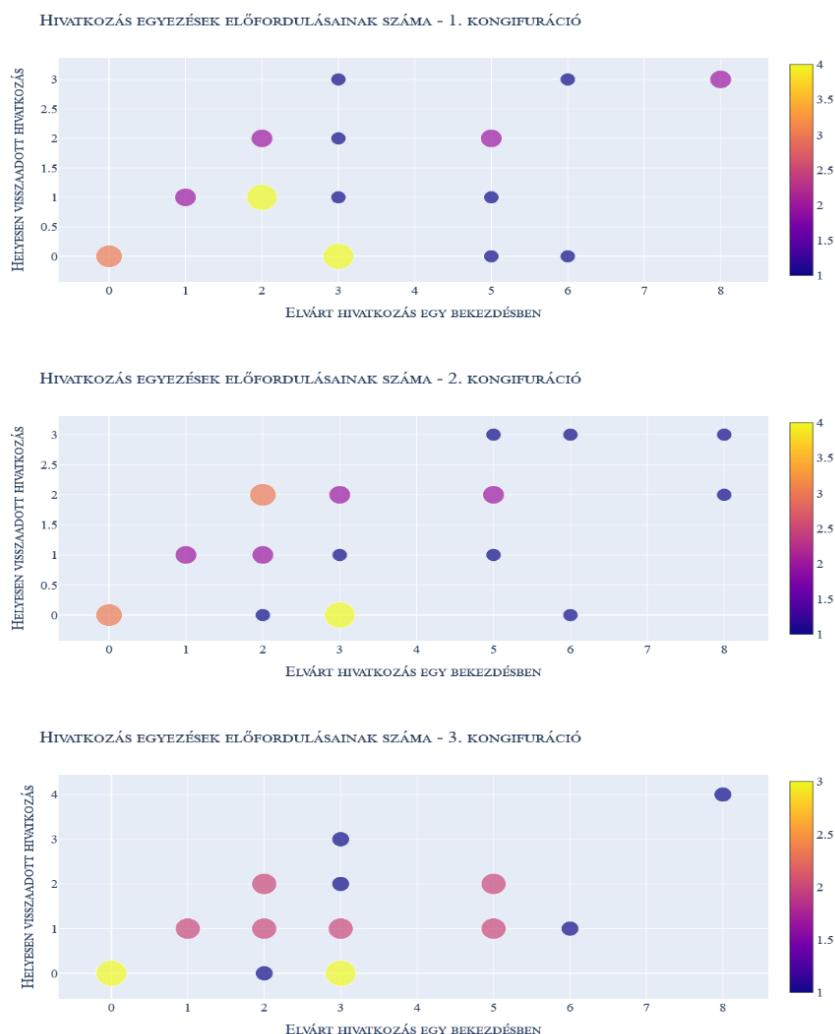
Az alábbi képen visszaadott dokumentumok címét összenézve könnyedén kiszámíthatjuk a precision@k mérőszámot, ugyanis ebben az esetben csak hasznosnak tekinthető adatrészleteket adott a visszakereső lépés. Így tehát a mérőszám értéke 1, ami egy kiválóan teljesítő visszakeresést mutat.

Elvárt dokumentumok	Visszkapott dokumentumok
Swinunet: Unet-like pure transformer for medical image segmentation	Swinunet: Unet-like pure transformer for medical image segmentation
Swin transformer: Hierarchical vision transformer using shifted windows	Swin transformer: Hierarchical vision transformer using shifted windows
-	Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation
-	Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation
-	Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation
-	Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation
-	Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation
-	Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

2. táblázat: A kimeneten megjelenített táblázat az elvárt, valamint a ténylegesen visszakeresett dokumentum címekről

A hivatkozás egyezés mérőszám megjelenítéséhez a program futása végén automatikusan buborékdiagramot generál (1. grafikon), amelyek tovább segítik az egyes konfigurációk, ezáltal

a nagy nyelvi modellek, összehasonlítását. Ezen grafikonok a buborékok nagyságával szemléltetik az esetek előfordulását. A tengelyeken az elvárt, valamint az adott futás alatt sikeresen egyeztetett hivatkozások értendők. Az alábbi grafikont kiértékelve láthatjuk, hogy a mostani példában a harmadik konfigurációnál fordult elő legkevesebbszer az, hogy egy helyesen egyeztetett hivatkozás se legyen.

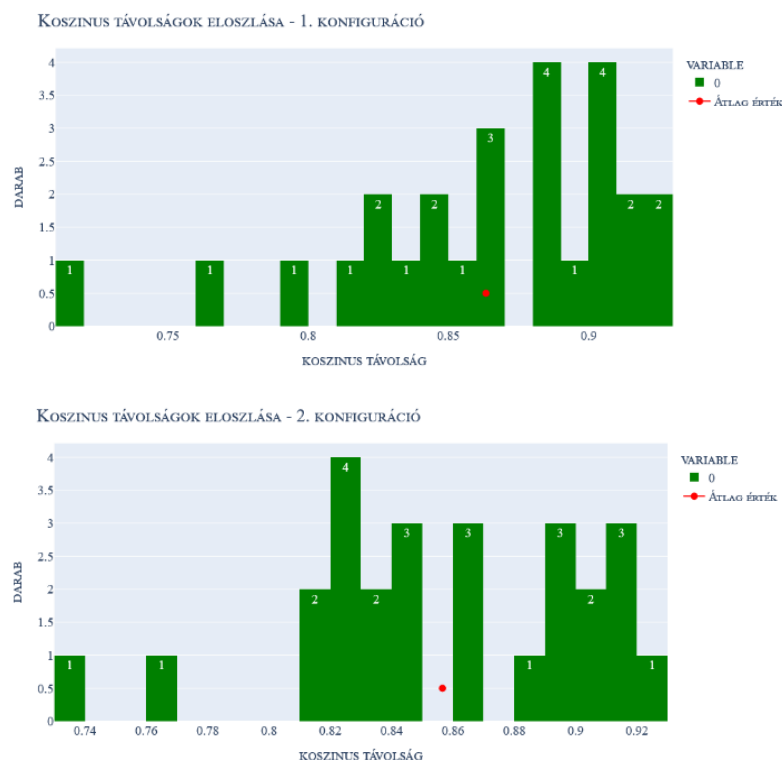


1. grafikon: *Automatikusan generált buborékdiagramok a hivatkozás egyezés mérőszámhoz*

A generáló lépés kiértékelésének támogatásához a kimeneten elérhetővé tesszük az előállított szövegrészletet, ezáltal manuálisan tesztelhetővé válik. Belátható, hogy ezen szövegek értelmezéséhez és összehasonlításához specifikus háttértudás szükséges. A probléma megoldására egy független nagy nyelvi modellel hasonlítjuk össze az elvárt, valamint az újonnan létrehozott szöveget. Ez a lépés magyarázatot biztosít a két szöveg szemantikai és

szintaktikai eltéréseit illetően. Az 1. függelékben található példán észrevehetjük az egyes konfigurációk által generált szövegrészletek különbségeit, mind tartalomban, mind hosszban és pontosságban. Az első két szöveg (70%-os egyezés) pontosabbnak és lényegretörőbbnek hat, míg a harmadik (60%-os egyezés) több nem szorosan kapcsolódó témát is felkarol. Ezt az eredményt hűen tükrözik a nagy nyelvi modellek által visszaadott százalékos értékek, valamint azok magyarázatai is.

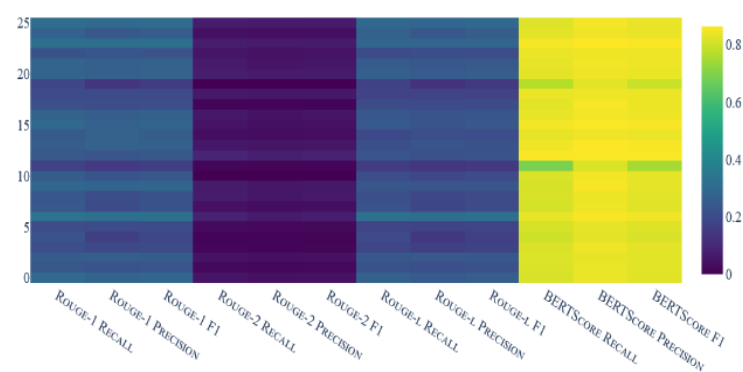
A szövegek közti hasonlóság fontos mérőszáma a koszinusz távolság, amely a mélyebb szemantikai jelentést képes felismerni [11]. A program automatikusan megalkot egy hisztogramot (2. grafikon), mely ezen számok eloszlását jeleníti meg egy futás eredményén. A grafikonon továbbá látható a hasonlóságok átlagos értéke is egy pontként. Az alábbi ábrán észrevehetjük, hogy az oszlopok viszonylag eltérőnek tűnnek, azonban az átlagok értéke szinte ugyanott helyezkedik el. Továbbá elmondható, hogy a második konfigurációnál kevesebb egy elemű oszlopot láthatunk, erről arra következtethetünk, hogy ez az LLM konzisztensebb az eredményeit illetően.



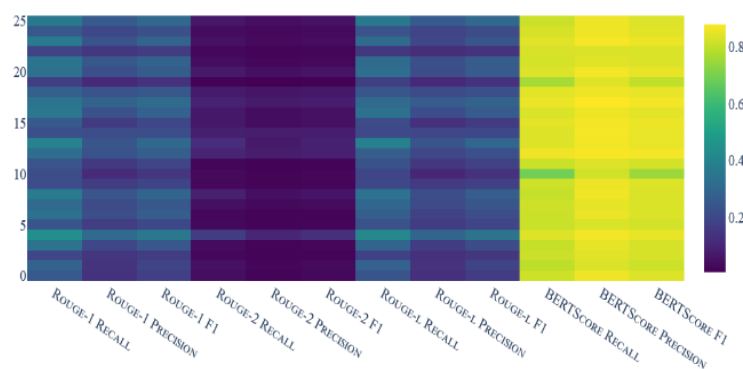
2. grafikon: A koszinusz hasonlóságok eloszlása első és második konfigurációban

A hosszú szövegek összegzése egy ismert feladat, melynek kiértékelésére a ROUGE és BERTScore mérőszámokat használtuk. Ez a két eszköz együtt használva remekül kiegészítik egymást, ugyanis amíg a ROUGE szimplán szavak és szóösszetételek jelenlétét elemzi, addig, a BERT modellre épített BERTScore szemantikus hasonlóságokat képes észrevenni. Ezekhez az értékekhez a szoftver egy hő térkép diagramot alkot meg (3. grafikon), melyről nem a pontos értékek leolvasása a lényeg, minél inkább a gyors és egyszerű átláthatóság. Az alábbi grafikonról megállapíthatjuk, hogy a második konfiguráció színei észrevehetően világosabbak, ezáltal jobb eredményeket jelenthetnek.

ROUGE és BERTSCORE ÉRTÉKEK - 1. KONFIGURÁCIÓ



ROUGE és BERTSCORE ÉRTÉKEK - 2. KONFIGURÁCIÓ

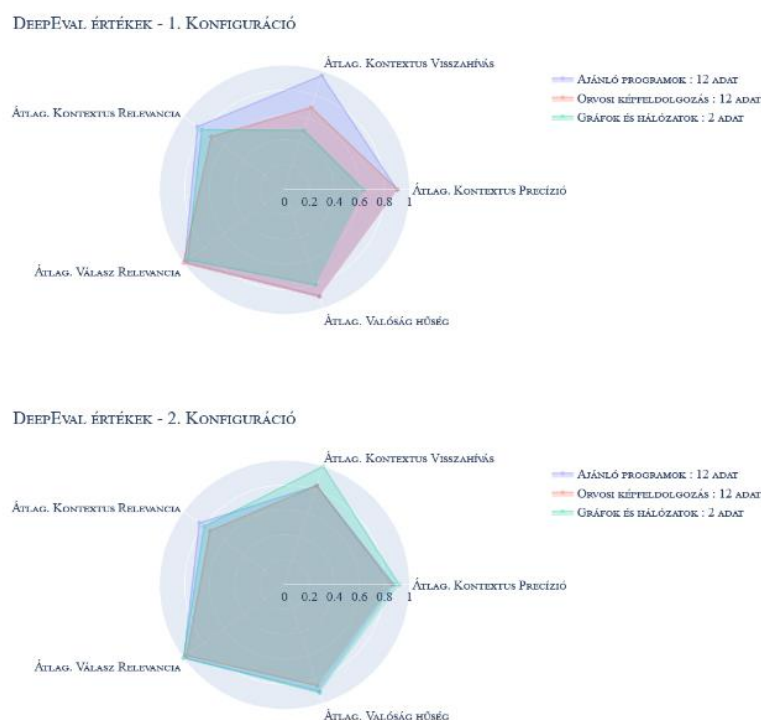


3. grafikon: *Hő térkép a ROUGE és BERTScore értékekről*

Az általunk használt DeepEval kiértékelő keretrendszer számos hasznos mérőszámot biztosít, melyek közül mi a generált szöveg kontextusával foglalkozókat részesítettük előnyben. Ezek alapján értékelni tudjuk többek között a válasz precízióját, relevanciáját és valóság alapját. Hogy ezen tulajdonságokat minél könnyebben értelmezhetővé tegyük a

szoftver pókháló diagramokat (4. grafikon) hoz létre az egyes témákra szétbontva. Ezek a generált válasz által egyfajta profilt alkotnak az adott konfigurációról.

Az alábbi grafikonon egyértelműen láthatjuk, hogy a második konfiguráció ezen futásban kontextus visszahívásban, tehát a bemenet kontextusa alapján releváns információk pontos rögzítésében és visszakeresésében két témában is jobban teljesített. A többi mérőszám jelentősen nem tér el, ami tovább erősíti az eredeti állításunk, miszerint mindkét konfiguráció hasonlóan jó eredményt kapott, azonban a második válasz eredmény pontosabbnak tűnik.



4. grafikon: A DeepEval kiértékelő keretrendszer által kapott értékek pókháló diagramon

Az eddig tárgyalt értékek összesítve is megtekinthetők a program által előállított táblázatban (2. függelék). Ez ugyan kevésbé áttekinthető, azonban analitikus szemmel rengeteg plusz információt tartalmazhat. Egy ilyen táblázat az adott szövegrészletre értendő, azaz az alábbi példa a már korábban tárgyalt fejezetre vonatkozik. Azt tapasztalhatjuk, hogy az eddig állítottakkal szemben a harmadik konfiguráció érte el a legtöbb mérőszámban a legnagyobb értéket, mely valószínűleg ahhoz köthető, hogy ez a modell generálta a leghosszabb szöveget, így a ROUGE értékeknél előnyben lehetett.

4.2. Egy szélsőséges eset bemutatása

A továbbiakban egy szélsőséges, vagyis nem gyakran előforduló esetet mutatunk be. A tesztelés folyamán észrevettünk olyan érték párokat, melyek nem egyeznek az elképzeléseinkkel (3. táblázat). Meglepően magas értékeket kapott egy olyan fejezet előállítása, mely egyetlen hivatkozást se tartalmaz. Ezáltal a visszakeresés mérőszámai alacsony, míg a generáláshoz tartozó értékek magas jószágot mutatnak.

A probléma megmagyarázásához érdemes lenne megvizsgálnunk azt, hogy visszakeresési lépés nélkül milyen szöveget generál az LLM. Amennyiben az alap kontextus, azaz a kulcskonceptiók megadásával is elég jó értékeket ad a folyamat, azt feltételezhetjük, hogy a saját betanított tudásából képes ezt elérni. A lefuttatott tesztek alapján pontosan ezt tapasztaltuk, meglepő módon koszinusz hasonlósága nagyobb érték lett a visszakeresés nélküli generálásnak. A modellek alaptudása és ennek a kiértékelésre mért torzító hatása egy nagyon fontos szempont és kérdés, amit megéri a jövőben kutatni.

Mérőszámok-	Visszakereséssel	Visszakeresés nélkül
Koszinusz hasonlóság	0.92129	0.94808
Kontextus relevancia	0.70315	-
LLM értékelés	Hasonlóság: 70%	Hasonlóság: 70%
Hivatkozás egyezés	0/0	0/0
Precision@K	0	0

3. táblázat: Visszakereséssel és visszakeresés nélkül kapott mérőszámok

5. Összefoglalás

Az utóbbi időkben hatalmas teret hódítottak a nagy nyelvi modellek, mind az emberek hétköznapijában, mint a kutatásban. Teljesen megváltoztatták az emberek munka dinamikáját, ugyanis eddig nem látott eredményekre képesek egyes modellek. A RAG technikák tovább képesek javítani a visszakeresés és generálás folyamatán, valamint számos új felhasználási lehetőséget nyitnak meg.

A projekt keretein belül sikeresen megalkottam egy eszközrendszert, amely összegyűjt már létező kiértékelési módszereket és mérőszámokat és magas személyre szabhatósággal, valamint könnyen értelmezhető, grafikus kimenettel teszi ezeket elérhetővé. Az így kapott szoftver egy hiánypótló ötlet, amely elősegítheti a nagy nyelvi modellek tesztelését. A projektet felhasználhatjuk ipari környezetben összehasonlítási céllal is.

A kutatás gördülékenyebbé tétele érdekében lefejlesztettem egy segédprogramot, mely nagyban megkönnyíti az állományok előfeldolgozását.

A dolgozatomban bemutatom az eszközrendszer felépítését, valamint működésének lépéseit. Egy konkrét példán keresztül belátható, hogy az eredeti manuális kiértékelés által kapott megérzésünket a mérőszámok is alátámasztják, így a szoftver jól működőnek mondható. Kitérek egy szélsőséges esetre is, mely azt bizonyította számunkra, hogy a modellek alaptudása nem belátható módon befolyásolja a kiértékelést, ezért érdekes kutatási témát jelenthet a jövőben.

6. Limitációk

A projekt során két nagyobb limitációba ütköztünk. Ebből az egyik a nagy nyelvi modellek használatának költsége, legyen szó lokálisan vagy távolban futtatott verzióról. Az anyagi korlátok ilyen módon nagyban befolyásolták a tesztek futtatásának hosszát és gyakoriságát.

A másik limitáció ahhoz köthető, hogy a tudásbázis kiépítése jelenleg még manuális módszereket igényel. Egyes tudományos cikkek, főleg a területeket áttekintő és elemző dokumentumok sokszor több száz hivatkozást tartalmaznak. Ezek összegyűjtése mind időben, mind szintén anyagilag korlátokat vetnek fel, ugyanis rengeteg dokumentum csak a szükséges hozzáféréssel vagy egyenleggel érhető el.

7. Jövőbeli tervek

A jövőben további fejlesztésekkel bővítenénk és egyszerűsíténénk a szoftver használatát. Egy ilyen jelentős javítás lenne az előfeldolgozás teljes mértékben automatizálása. Azaz a hivatkozott dokumentumok beszerzését, majd ezeken a szövegfelismerő programot alkalmazni. Ehhez szükséges lehet a GROBID és hozzá tartozó segédprogram integrálása.

A dolgozat több érdekes kutatási témát is felvet. Ilyen lehetne annak a megmérése, hogy a nagy nyelvi modellek saját tudásuk mennyiben torzítják a kimérések adatait. Valamint érdemes lehetne a mostani rendszer egy már fejlettebb változatával több LLM család, architektúra, vektor adatbázis, beágyazó modell és egyéb hiperparaméterek kipróbálása.

8. Irodalomjegyzék

- [1] Asai, A., Min, S., Zhong, Z., Chen, D. (2023). *Retrieval-based language Models and Applications*. ACL 2023 Tutorial.
URL: <https://acl2023-retrieval-lm.github.io/>
- [2] Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T., Li, Q. *A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models*. Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 24, 2024, pp 6491 - 6501.
DOI: <https://doi.org/10.1145/3637528.3671470>
URL: <https://arxiv.org/html/2405.06211v2>
- [3] Jing, Z., Su, Y., Han, Y., Yuan, B., Xu, H., Liu, C., Chen, K., Zhang, M. (2024). *When Large Language Models Meet Vector Databases: A Survey*. Cornell University.
DOI: <https://doi.org/10.48550/arXiv.2402.01763>
URL: <https://arxiv.org/pdf/2402.01763>
- [4] Kniazieva, Y. (2024). *Types of LLMs: Classification Guide*. LabelYourData.
URL: <https://labeledyourdata.com/articles/llm-fine-tuning/types-of-llms>
- [5] LlamaIndex. Faithfulness Evaluator.
URL: https://docs.llamaindex.ai/en/stable/examples/evaluation/faithfulness_eval/
[Letöltve: 2024.október]
- [6] LlamaIndex. *Indexing*.
URL: https://docs.llamaindex.ai/en/stable/module_guides/indexing/
[Letöltve: 2024.október]
- [7] LlamaIndex. Relevancy Evaluator.
URL: https://docs.llamaindex.ai/en/stable/examples/evaluation/relevancy_eval/
[Letöltve: 2024.október]
- [8] Nguyen, I. (2023). *Hybrid Document Retrieval*. Haystack.
URL: <https://haystack.deepset.ai/blog/hybrid-retrieval#a-hybrid-retrieval-pipeline-in-haystack>
- [9] OpenAI Platform. *Embeddings*.
URL: <https://platform.openai.com/docs/guides/embeddings/what-are-embeddings>
[Letöltve: 2024.október]

- [10] Pan, J. J., Wang, J., Li, G. (2024). Survey of Vector Database Management Systems. *The VLDB Journal*, 33(5), pp 1-25.
URL: <https://arxiv.org/pdf/2310.14021>
- [11] Rahutomo, F., Kitasuka, T., and Aritsugi, M. *Semantic Cosine Similarity*. The 7th International Student Conference on Advanced Science and Technology (ICAST 2012), Seoul, South Korea, October, 2012.
URL: https://www.researchgate.net/profile/Faisal-Rahutomo/publication/262525676_Semantic_Cosine_Similarity/links/0a85e537e3b675c1e000000/Semantic-Cosine-Similarity.pdf
- [12] Seo, J., Lee, T., Moon, H., Park, C., Eo, S., Aiyanyo, I. D., Park, K., So, A., Ahn, S. and Park, J. (2022). Dense-to-Question and Sparse-to-Answer: Hybrid Retriever System for Industrial Frequently Asked Questions. *Mathematics*, 10(8), 1335.
DOI: <https://doi.org/10.3390/math10081335>
- [13] Wasti, N. (2024). Google Search Algorithm Insights and the Advancement of Retrievers in AI RAG. Medium.
URL: <https://medium.com/@nabilw/google-search-algorithm-insights-and-the-advancement-of-retrievers-in-ai-rag-fdeef7b19223>

9. Függelékek

1. függelék: A különböző konfigurációk által generált szövegek és LLM általi értékeléseik

	Konfiguráció 1	Konfiguráció 2	Konfiguráció 3
Generált szöveg	The Swin-Unet is proposed as a novel approach for medical image segmentation, leveraging a pure Transformer-based U-shaped Encoder-Decoder architecture. The architecture consists of an encoder, bottleneck, and decoder, all built using the Swin Transformer block. Medical images are divided into non-overlapping patches, which are treated as tokens for the encoder to learn deep feature representations. The decoder employs a patch expanding layer to up-sample extracted context features, which are fused with multi-scale features from the encoder through skip connections, effectively restoring spatial resolution for segmentation predictions. Extensive experiments on multi-organ and cardiac segmentation datasets demonstrate that Swin-Unet achieves superior segmentation accuracy and robust generalization compared to traditional CNN-based methods. Key contributions include the development of a symmetric architecture with self-attention capabilities, a novel patch expanding layer that enhances up-sampling without convolutions, and the effective use of skip connections within a pure Transformer framework. This work highlights the potential of Transformers in medical image segmentation, addressing limitations of CNNs in capturing global and long-range semantic information while maintaining localization accuracy.	The Swin-Unet model for medical image segmentation utilizes a Transformer-based U-shaped Encoder-Decoder architecture with skip connections. The encoder employs hierarchical Swin Transformer with shifted windows for context feature extraction, while the decoder includes a patch expanding layer for up-sampling. The model outperforms methods using full-convolution or a combination of transformer and convolution. Additionally, a CNN-Transformer hybrid model is introduced in TransUNet, where CNN acts as a feature extractor. A cascaded upsampler (CUP) with multiple upsampling steps is utilized for decoding the hidden features. Experiments on multi-organ and cardiac segmentation datasets demonstrate the effectiveness and robustness of the proposed methods. The Swin-Unet architecture showcases the potential of leveraging Transformers for accurate medical image segmentation tasks.	Recent advancements in convolutional neural networks (CNNs) have significantly impacted medical image analysis, particularly through U-shaped architectures that utilize skip connections. Despite their success, CNNs struggle with learning global and long-range semantic interactions due to the inherent locality of convolution operations. This paper introduces Swin-Unet, a novel Unet-like pure Transformer model designed for medical image segmentation. The model employs a Transformer-based Encoder-Decoder architecture, where input medical images are tokenized into patches that facilitate local-global semantic feature learning. The encoder utilizes a hierarchical Swin Transformer with shifted windows to extract contextual features, while the decoder incorporates a patch expanding layer for up-sampling, restoring spatial resolution for segmentation predictions. Experimental results demonstrate that Swin-Unet outperforms traditional CNN-based methods and hybrid approaches combining CNNs with Transformers on multi-organ and cardiac segmentation tasks. The architecture comprises an encoder, bottleneck, and decoder, all constructed from Swin Transformer blocks. The integration of skip connections enhances the model's ability to retain spatial information. Overall, Swin-Unet represents a significant step forward in leveraging pure Transformer architectures for enhanced accuracy and robustness in medical image segmentation tasks.

LLM értékelés	Similarity Score: 70% Semantics (Concepts and Ideas): Both texts discuss the Swin-Unet architecture, emphasizing its role in medical image segmentation and its use of the Swin Transformer blocks. They both highlight the architecture's encoder-decoder structure and its advantages over traditional convolutional neural networks (CNNs). Text 1 focuses more on the technical details of the Swin-Unet's structure, such as the hierarchical attention mechanism and the compression of features, while Text 2 emphasizes the practical applications and performance of Swin-Unet in segmentation tasks. Despite these differences in focus, the core ideas about the architecture's purpose, structure, and advantages are largely aligned, leading to a high semantic similarity score. Syntax (Grammar, Word Order, Structure): The syntax in both texts differs in terms of structure and phrasing. Text 1 is more technical and detailed, using longer sentences with specific references to figures and studies, while Text 2 is more concise and structured, focusing on the contributions and results of the Swin-Unet. The differences in sentence length and complexity do not significantly hinder understanding, but they do reflect a notable variation in style. This syntactical difference contributes to a slightly lower score than would be expected if the texts were more closely aligned in structure.	Similarity Score: 70% Semantics (Concepts and Ideas): Both texts discuss the Swin-Unet model, emphasizing its architecture and application in medical image segmentation. They highlight the use of Swin Transformer blocks and the model's advantages over traditional convolutional networks and other hybrid models like TransUNet. Text 1 provides a detailed explanation of the Swin-Unet's structure, including its hierarchical feature extraction and the mechanism of shifting windows, while Text 2 summarizes these points and mentions the model's performance in experiments. Although both texts convey similar core ideas about the Swin-Unet's architecture and its effectiveness, Text 2 introduces additional elements like the patch expanding layer and the CNN-Transformer hybrid model, which are not mentioned in Text 1. This results in a moderate difference in depth and focus, leading to a score of 70%. Syntax (Grammar, Word Order, Structure): The syntax in both texts is generally clear and coherent, but they differ in structure. Text 1 is more detailed and technical, with longer sentences and a more formal tone, while Text 2 is more concise and structured with shorter sentences. The differences in sentence length and complexity do not significantly impact the clarity of the ideas presented, but they do reflect a notable variation in style. This minor syntactical difference does not heavily influence the overall score, as the primary focus remains on the semantic alignment.	Similarity Score: 60% Semantics (Concepts and Ideas): Both texts discuss the Swin-Unet architecture and its application in medical image segmentation, highlighting its advantages over traditional convolutional neural networks (CNNs) and other hybrid models. They both emphasize the use of Swin Transformer blocks and the hierarchical attention mechanism, which allows for better feature extraction and retention of spatial information. However, Text 1 focuses more on the technical details of the Swin-Unet architecture, including its structure and specific mechanisms like shifting windows, while Text 2 provides a broader context about the limitations of CNNs and the significance of the Swin-Unet model in overcoming these challenges. This results in a moderate semantic alignment, as the core ideas are similar but expressed with different emphases and details. Syntax (Grammar, Word Order, Structure): The syntax in both texts is generally clear and well-structured, but there are notable differences in sentence construction and phrasing. Text 1 is more technical and detailed, with a focus on specific architectural components, while Text 2 adopts a more general overview style, discussing the implications and performance of the model. These differences in structure and word choice contribute to the overall score, but they do not significantly hinder understanding.

2. függelék: A kiértékelés alatt használt mérőszámok

Konfiguráció 3	Konfiguráció 2	Konfiguráció 1	
2.00000	2.00000	2.00000	Citation Matching Expected
2.00000	2.00000	2.00000	Citation Matching Both Num
0.92674	0.90332	0.91704	Cosine Similarity Result
1.00000	1.00000	1.00000	LlamaIndex Faithfulness Score
1.00000	1.00000	1.00000	LlamaIndex Relevancy Score
1.00000	0.91508	0.82540	Deepeval Contextual Precision
1.00000	1.00000	1.00000	Deepeval Contextual Recall
0.52239	0.69355	0.71622	Deepeval Contextual Relevancy
1.00000	0.66667	1.00000	Deepeval Answer Relevancy
0.76923	1.00000	0.75000	Deepeval Faithfulness Score
0.30000	0.35366	0.27273	Rouge-1 Recall
0.30233	0.22481	0.25581	Rouge-1 Precision
0.30116	0.27488	0.26400	Rouge-1 F1
0.08235	0.06364	0.05625	Rouge-2 Recall
0.06829	0.03415	0.04390	Rouge-2 Precision
0.07467	0.04444	0.04932	Rouge-2 F1
0.26923	0.32927	0.23967	Rouge-l Recall
0.27132	0.20930	0.22481	Rouge-l Precision
0.27027	0.25592	0.23200	Rouge-l F1
0.84559	0.83078	0.83611	BERTScore Recall
0.85877	0.85697	0.85106	BERTScore Precision
0.85213	0.84367	0.84352	BERTScore F1
0.875	1	1	Precision@K

Köszönetnyilvánítás

Szeretném megköszönni a Kodesage projekt, valamint a Sonrisa Kft. munkatársainak, hogy biztosították számomra a kutatásban való részvétel lehetőségét. Rengeteg új ismeretet szereztem és egy új területen is kipróbálhattam magamat.

Külön köszönet illeti a témavezetőimet, London Andrást és Dr. Harangi Balázst. Észrevételeik és útmutatásuk nagy segítségemre volt.

Köszönöm Tímeának, hogy segítséget nyújtott a dolgozat formázásában és megírásának folyamatában.