RAG módszerek implementálásának kihívásai egy célorientált chatbot rendszer kontextusában

Csáki Csaba [1] és Vándor Péter[1]

[1] Budapesti Corvinus Egyetem, Fővám tér 8,
H-1093 Budapest, Magyarország
peter.vandor@stud.uni-corvinus.hu
csaki.csaba@uni-corvinus.hu

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

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

1 Bevezetés

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

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

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

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

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

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

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

2 Módszertan

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

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

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

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

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

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

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

3 A RAG folyamat problémái és megoldási technikák

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

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

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

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

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


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


1. ábra: JSON struktúra (saját munka)

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

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

4 PDF feldolgozó módszerek és modell-specifikus kihívások

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

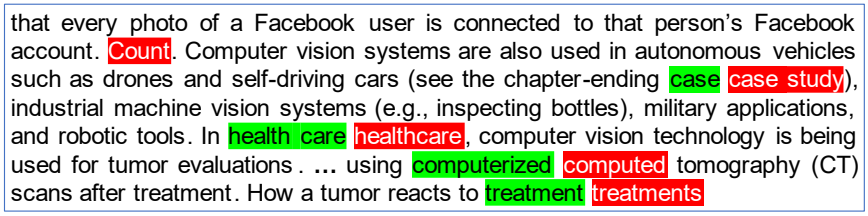
1. táblázat: Az LLM-ek és a Tesseract szoftver OCR feldolgozási eredményei

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

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

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

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



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

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

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

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

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

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

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

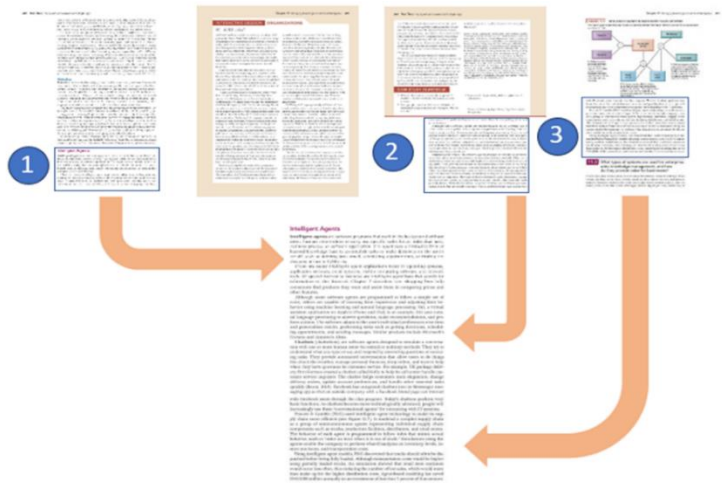
5 Összefűzés, a szöveg szemantikai koherenciájának biztosítása

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

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

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

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



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

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

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

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

6 Szöveg feldarabolása

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

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

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

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

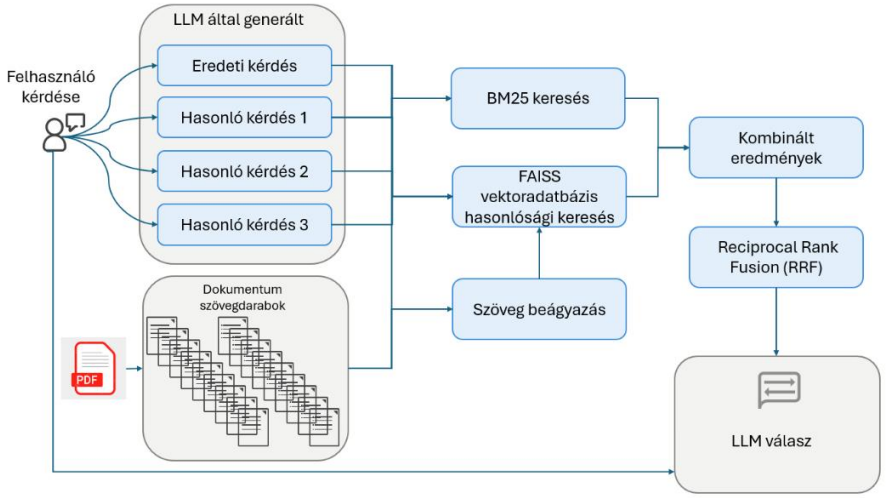
Az ágensalapú szövegdarabolás a kulcsfontosságú kritériumok (pl. szemantikai koherencia) körvonalazása után a promptban, teljes mértékben az LLM-re bízza a szöveg szegmentálását és a darabok létrehozását. A prompt a PDF feldolgozásnál használtnál sokkal rövidebb, tartalmazza pl. az 1000 karakteres limitet,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

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

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

7 Keresési módszerek és rangsorolás

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



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

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

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

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

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

8 Kérdés-válasz teszt eredmények

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

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

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

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

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

9 Tanulságok összefoglalása

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

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

Köszönetnyilvánítás

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

A chatbot fejlesztését és szerver szolgáltatását a Webra Kft. (webra.hu) támogatta.

Bibliográfia

Anthropic. (2024a). Claude 3.5 Sonnet Model Card Addendum. *https://paperswithcode.com/paper/claude-3-5-sonnet-model-card-addendum*

Berkecz, P., Zombori, T., Banga, G., Szabó, G., Szántó, Zs., Novák, A. & Farkas, R. SHunQA: egy nyíltkérdés-megválaszoló rendszer. XX. Magyar Számítógépes Nyelvészeti Konferencia online kiadás, Magyarország, Szegedi Tudományegyetem (2024) pp. 73-84., 12 p.

Camelot Developers. 2021. Camelot. https://github.com/camelot-dev/camelot. Letöltve: 2024.12.23.

Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009*. https://doi.org/10.1145/1571941.1572114

Fenniak, M., Kulkarni, A. & Witham, S. 2014. PyPDF. https://github.com/py-pdf/pypdf. Letöltve: 2024.12.23.

Gemini Team Google. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv.org*. https://doi.org/10.48550/arxiv.2403.05530

Gemma Team, Google DeepMind. (2024). Gemma 2: Improving Open Language Models at a Practical Size. *arXiv.org*. https://arxiv.org/abs/2408.00118v3

Introducing Contextual Retrieval. (2024b). Anthropic.com. https://www.anthropic.com/news/contextual-retrieval

Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., Casas, D.D., Hanna, E.B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L.R., Saulnier, L., Lachaux, M., Stock, P., Subramanian, S., Yang, S., … Sayed, W.E. (2024). Mixtral of Experts. arXiv, abs/2401.04088. *arXiv.org*. https://arxiv.org/abs/2401.04088

Laudon, K. és Laudon, J. (2021). Management Information Systems: Managing the Digital Firm. 17th (Global) edition. Pearson Education. Harlow UK.

Lewis, P. S. H., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Neural Information Processing Systems*, *33*, 9459–9474.

Llama Team. (2024). The Llama 3 Herd of Models. *arXiv.org*. https://arxiv.org/abs/2407.21783v2

LLM Document Extraction: How to use AI to get structured data from legacy documents. (2021). Pondhouse-Data.com. https://www.pondhouse-data.com/blog/document-extraction-with-llms

Ma, X., Gong, Y., He, P., Zhao, H., & Duan, N. (2023). Query Rewriting in Retrieval-Augmented Large Language Models. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/2023.emnlp-main.322

McKie, J. X.; és Liu, R. 2016. PyMuPDF. https://github.com/pymupdf/PyMuPDF. Letöltve: 2024.12.23.

Meta. (2024). Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta.com. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

Mistral AI team. (2024). Cheaper, Better, Faster, Stronger. Mistral.ai. https://mistral.ai/news/mixtral-8x22b/

OpenAI. (2024). GPT-4o System Card. *arXiv.org*. https://arxiv.org/abs/2410.21276

Pethő, G., Sass, B., Simon, L., Lipp, V. OCR-hibák kvantitatív elemzése több szövegváltozat összehasonlításával. XX. Magyar Számítógépes Nyelvészeti Konferencia online kiadás, Magyarország, Szegedi Tudományegyetem (2024) pp. 17-29., 13 p.

Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, *3*(4), 333–389. https://doi.org/10.1561/1500000019

Shuster, K., Poff, S., Chen, M., Kiela, D., & Weston, J. (2021). Retrieval Augmentation Reduces Hallucination in Conversation. *Conference on Empirical Methods in Natural Language Processing*.

Smith, R. (2007). An overview of the Tesseract OCR engine. *Proceedings of the International Conference on Document Analysis and Recognition*, 629–633. https://doi.org/10.1109/icdar.2007.4376991

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., … Fan, Z. (2024). Qwen2 Technical Report. *arXiv.org*. https://arxiv.org/abs/2407.10671v4

Yao, Y., Yu, T., Zhang, A., Wang, C., Cui, J., Zhu, H., Cai, T., Li, H., Zhao, W., He, Z., Chen, Q., Zhou, H., Zou, Z., Zhang, H., Hu, S., Zheng, Z., Zhou, J., Cai, J., Han, X., & Zeng, G. (2024). MiniCPM-V: A GPT-4V Level MLLM on Your Phone. *arXiv.org*. https://arxiv.org/abs/2408.01800

Why am I receiving an “Output blocked by content filtering policy” error? Anthropic Privacy Center. (2024). Anthropic.com. https://privacy.anthropic.com/en/articles/10023638-why-am-i-receiving-an-output-blocked-by-content-filtering-policy-error