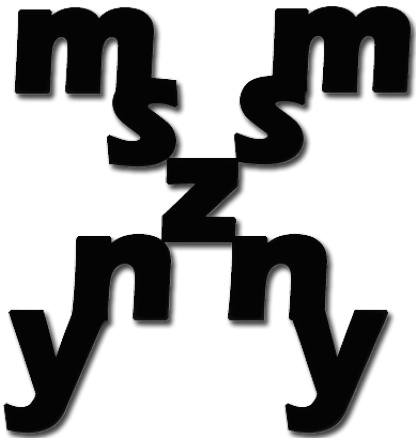
XXI. Magyar Számítógépes Nyelvészeti Konferencia



Szerkesztette:
Berend Gábor
Gosztolya Gábor
Vincze Veronika

Szeged, 2025. február 6-7.

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

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

ISBN: 978-963-688-034-7

Szeged, 2025. február

Az MSZNY 2025 konferencia szervezője:
HUN-REN–SZTE Mesterséges Intelligencia Kutatócsoport

Támogatók:
K&H Bank
Neumann János Számítógéptudományi Társaság
SZTE Informatikai Intézet

Előszó

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

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

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

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

A szervezőbizottság nevében,
Ács Judit,
Berend Gábor,
Gosztolya Gábor,
Ligeti-Nagy Noémi,
Nemeskey Dávid Márk,
Novák Attila,
Simon Eszter,
Sztahó Dávid,
Vincze Veronika

Tartalomjegyzék

Zárt és nyílt súlyú LLM-ek teljesítményértékelése RAG-alapú felsőoktatási chatbot alkalmazás példáján

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 nagy nyelvi modellek (LLM) terjedésével egyre népszerűbbek a célzott saját (custom) chatbotok. Ezek fejlesztése sokszor a Retrieval Augmented Generation (RAG) megoldásra épül. Azonban mind a fejlesztés és tesztelés, mind az alkalmazás szempontjából fontos döntés, melyik LLM-t használjuk a saját chatbot mögött: a választott alternatíva nem közömbös sem teljesítmény, sem költségek tekintetében. A kutatás keretében egy felsőoktatási célú chatbot esetén vizsgáltuk a legnépszerűbb zárt és nyílt súlyú LLM-ek API hívásokkal történő használatának minőségi és költség jellemzőit. A chatbotba az oktatók tölthetik fel tananyagaikat, a hallgatók kérdéseket tehetnek fel a modellnek, végül az oktatók megtekinthetik a tanulók kérdéseit és statisztikáit, így támogatva a személyre szabott tanulást. A vizsgálat rávilágított, hogy a költségek jelentős részét teszik ki a saját tudásbázis kiépítése és tesztelése. A lokális tudásbázis felépítése során az egyes modellek nagyon eltérő minőséget eredményeznek. A teszt és felhasználói kérdésekre adott válaszokban a költségek rövid távon nem jelentősek (a fejlesztéshez képest), ugyanakkor a nyílt súlyú modellek is jó eredményt tudnak biztosítani, megnyitva a költségek csökkentését hosszú távon.

Kulcsszavak: nyíltkérdés-megválaszolás, RAG, nagy nyelvi modell, LLM, generatív válaszolás, retriever-reader architektúra

1 Bevezetés: a felsőoktatás generatív MI kihívása

A nagy nyelvi modellek (LLM) és az azokra épülő különböző mesterséges intelligencia (MI) megoldások egyik legnépszerűbb alkalmazási területe a chatbotok. E chatbotok alkalmazása azonban számos kérdést vet fel, melyek egy része technikai problémákból (pl hallucináció), más része minőségi hiányosságokból (pl. hibás, nem adekvát vagy terjengős válasz) ered, emellett felvetődnek adatvédelmi kockázatok is (pl. a belső adatok feltöltése egy külső chatbotba). Az általános generatív modellekre épülő botok nem minden esetben adnak pontos választ egy-egy felhasználói kérdésre, különösen, ha azok egy specifikus szakterületre vagy tartalomra irányulnak, így ilyen esetekben gyakrabban adnak helytelen választ vagy akár hallucinálhatnak. Ugyanakkor konkrét alkalmazások esetén a felhasználó számára fontos lehet a pontos és akár ellenőrizhető válasz (amihez forrás megjelölés is kapcsolódhat) (Berkecz és mtsai, 2024). E problémák ellensúlyozására egyre többet figyelembe vett lehetséges megoldás a saját (ún. custom

vagy lokális) botok építése. A lehetőség előnye, hogy a tartalom közvetlenül az üzleti igényhez igazítható, és az üzemeltető nagyobb kontrolt gyakorolhat a tartalom és a lekérdezés felett (Gao és mtsai, 2023). Ugyanakkor kihívás, hogy a chatbotot ki kell fejleszteni – melyhez fejlesztési környezet és szaktudás kell –, illetve jól kell megválasztani a bot mögött működő nyelvi modelleket.

Egy saját bot kialakítása azonban nem csak fejlesztői kihívás, de tesztelni is kell a tartalmat, hisz ez az egyik legfőbb célja a bot önálló fejlesztésének, és figyelembe kell venni a fejlesztés és üzemeltetés költségeit is. Számos döntés kapcsolódik egy saját bot kialakításához, de ezek közül a legfontosabb, hogy a custom bot építése és használata során alkalmazott LLM milyen teljesítményre képes, és milyen várható költségekkel jár. A költségek tekintetében két lényegi megközelítés érhető el a piacon: az ún. nyílt súlyú modellek, melyek ingyenesen használhatók (akár netes eléréssel, akár lokális futtatással kapacitástól függően), míg a zárt modellek használatáért fizetni kell, ami lehet havi előfizetés (személyre szabottan) vagy használat (azaz token mennyiség) függő.

Technikai megoldás szempontjából egyre népszerűbbek az ún. Retrieval Augmented Generation (RAG) megoldások (Lewis és mtsai, 2020). Egy RAG-alapú rendszer jellemzően öt fő részből áll, egy adatbeviteli funkcióból (mely a szakterület vagy célspecifikus információkat gyűjti és vektorizálja), egy lokális tudásbázisból (mely a szakterületi anyagokból készült embedded vektorokat eltárolja), egy hozzá kapcsolódó lekérdezőből és kontextus azonosítóból (mely kettő együtt megkeresi a felhasználói kérdések alapján a választ és a releváns kontextust, majd azt jellemzően prompttá kiegészítve továbbadja a nyelvi modellnek), egy (generatív) LLM-ből (mely a kontextushoz illő végső válaszokat nyelvtanilag helyesen legenerálja). Kell még (vég)felhasználói interfész is (mely minimálisan egy prompt-felületet, de más funkciókat is nyújthat).

Egy custom chatbot kialakítása jól meghatározható területek és lekérdezések esetén lehet sikeres, melyre jó példa a felsőoktatás, ahol egy-egy tantárgy tananyaga általában jól körülhatárolható. Lehetnek elérhető tankönyvFmtsaek és jegyzetek, vagy az oktató használhat cikkeket, előadás anyagokat és más forrásokat, melyek az adott tárgyban elvárt ismereteket, dedikált tudást tartalmaznak. Egy felsőoktatási kontextusban azonban a jelenlegi piaci körülmények között nem alternatíva az egyéni előfizetés (jellemzően a hallgatói létszám miatt, de a várható alacsony kihasználtság okán is), így tipikusan a tokenes (API) elérést vagy az ingyenes LLM-eket érdemes megvizsgálni. Természetesen a költségek mellett törekedni kell a minél pontosabb válaszadásra is.

Az e tanulmányban bemutatott kutatás fő célja annak vizsgálata, hogy egy felsőoktatásra tervezett, cél-orientált és tantárgyra szabható custom chatbot fejlesztése és üzemeltetése esetén milyen ár-érték arányt tudnak biztosítani a zárt modellek, és vajon a nyílt modellek fel tudják-e venni velük a versenyt.

2 RAG-alapú egyedi, testre szabott chatbot

2.1 A RAG megoldás és technikai komponensei

Egy cél-orientált, lokális chatbot lényege, hogy képes egy fókuszált területre vonatkozóan kérdéseket helyesen megválaszolni, és az eredményt kontextusban adott nyelven

megformált szövegként visszaadni. Az egyik legnépszerűbb és gyorsan felkapottá vált technológia ilyen jellegű kérdés-felelet (Q&A) bot építésére a Retrieval Augmented Generation (Gao és mtsai, 2023). A RAG megközelítéssel csökkenthetők a költségek, saját tartalmakat lehet kereshetővé tenni és úgy csökkenthető a hallucináció jelensége, hogy a saját tartalmakat nem kell feltölteni a nyelvi modell szolgáltatójának felhőjébe (Shuster és mtsai, 2021). Ilyen RAG megoldással kereshetővé 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 (promptok és megtalált válaszok leírása és statisztikái) is elérhetők. Ezek a jellemzők aztán felhasználhatók mind a RAG továbbfejlesztésére, mind a tartalom javítására, mind a kérdezők promptolási technikáinak képzésére.

A bot kiépítése során számos döntést kell hozni, melyek a fenti modulok implementálására vonatkoznak amennyiben a lehetséges technikai megoldások közül választani kell. A két legfontosabb döntés a bejövő anyagok feldolgozását végző modell kiválasztása, illetve a lekérdezésre adott végső választ legeneráló LLM megválasztása.

2.2 A felsőoktatási chatbot mint teszt környezet és a használt technológiák

Az LLM-ek tesztelését és eredményeik összehasonlítását egy olyan felsőoktatási chatbot fejlesztése során végeztük el mely lehetővé teszi PDF fájlok feltöltését és támogatja a hallgatókat az adott tananyagra vonatkozó kérdéseik megválaszolásában. A felsőoktatási chatbot implementációja során fontos szempont volt, hogy a hallgatók a rendszer által feldolgozott tananyagok alapján pontos, hallucinációktól mentes válaszokat kapjanak kérdéseikre. További célkitűzés volt, hogy részletes statisztikákat biztosítson az oktatók számára, hogy a tananyagokat és az oktatást teszteredmények alapján továbbfejleszthessék. A rendszer ezért a RAG megoldásra épül: az oktató feltöltheti a tantárgyi anyagokat, valamint megtekintheti a tanulók kérdéseit és a statisztikákat, így támogatva a személyre szabható tanulást. A hallgatói oldal egy prompt felületből és tudás-tesztelési funkcióból áll.

A bot létrehozása során két fő részre bontottuk projektet: (1) a PDF anyagok betöltésének, feldolgozásának, vektorizálásának kezdeti lépésére, amelyből a lokális tudásbázis keletkezett; (2) a kérdés-válasz RAG módszer szerinti implementációjára és a megoldás tesztelésére. Mindkét lépésnél figyelembe vettük a költségeket is és ennek megfelelően határoztuk meg az optimális megoldást.

A teljes RAG folyamat teszteléséhez egy angol nyelvű, 649 oldalas tankönyvet használtunk (Laudon, 2021) PDF formátumban, amelyben 393 oldal szkennelve, képként szerepel. A könyv bemutatja, hogyan használják a vállalatok az információs rendszereket üzleti céljaik elérésére és problémák megoldására.

2.2.1 Adatbevitel és tárolás

Minden egyes oldal, amin ábra található nem csak az ábrát, grafikont tartalmazza képként, hanem az oldalon található többi szöveget. Ennek következtében az első technológiai kihívás a képként tárolt oldalak transzformálásánál jelentkezett. További problémát okozott, hogy a könyvben gyakoriak a magyarázó ábrák, diagramok és minden fejezet 1-2 szövegdobozban kiemelt esettanulmányt is tartalmaz, amelyek félbeszakították a törzsszöveget. Ebből következően nem lehetett alkalmazni az alapértelmezett,

mindenhol használt szöveg összefűzési módszert, ahol az oldal alján véget érő szöveghez a következő oldal tetején található szöveget kötjük, mivel akár a következő oldal közepén vagy 2-3 oldallal később folytatódhatott a félbehagyott törzsszöveg. Ezután szemantikailag koherens kisebb részekre kellett tovább osztani a feldolgozott szöveget, mivel a teljes könyv folyamatos elküldése jelentősen megdrágította volna a válaszadást.

A PDF dokumentumok betöltésére és pontos feldolgozására már régóta születtek megoldások, amelyek alapvetően megpróbálták valamilyen determinisztikus algoritmus alapján felismerni az egyes szövegblokkok sorrendjét, a formázási kiemelések alapján kitalálni a címeket és szintjüket Markdown formátumban, valamint képek esetén OCR technikát alkalmazva felismerni a karaktereket. Előbbire példa a PyMuPDF (McKie és Liu, 2016), Camelot (Camelot Developers, 2021), Tabula (Ariga, 2016), PdfMiner (Shinyama, 2019), míg utóbbira a legjobban elterjedt Tesseract rendszer (Smith, 2007). A PDF elsősorban azonban nyomtatási, megjelenítési formátum és emiatt a szemantikai egység erősen csorbul. A szöveg értelmezése nélkül sok esetben nem lehet megállapítani a sorrendet, kapcsolódást, valamint a diagramokról sem tudnak összefoglaló leírást készíteni ezek a hagyományos szoftverek. A legújabb LLM-k közül több már rendelkeznek vision képességgel, amely kiválóan használható OCR kiváltására és annál magasabb szintű PDF feldolgozásra. A kutatás során 6 zárt (o1-preview, GPT-4-turbo, GPT-4o, GPT-4o-mini (OpenAI, 2024), Gemini PRO 1.5 (Gemini Team, 2024), Claude Sonnet 3.5 (Anthropic, 2024a)) és 7 nyílt modellt (Mixtral-8x22B-Instruct (Jiang és mtsai, 2024; Mistral AI team, 2024), Qwen2-72B VL (Yang és mtsai, 2024), Meta-Llama 3.1 405B (Llama Team, 2024), Llama 3.2 11B Vision, Llama 3.2 90B Vision (Meta, 2024), Mini CPM-Llama3 (Pondhouse Data, 2024), Google Gemma 2-27B (Gemma Team, 2024)) használtunk úgy, hogy az egyes feladatokhoz ezekből választottunk.

Tesztjeink alapján mind a hagyományos PDF betöltőknél (PyMuPDF, PyMuPDF4llm), mind a Tesseract OCR rendszernél jobb eredmény érhető el vision LLM alkalmazásával (amit a GPT-4o, Google Gemini PRO 1.5, Claude Sonnet 3.5, Qwen2 72B, Llama 3.2 11B Vision, Llama 3.2 90B Vision, Mini CPM-Llama3 tud). A nyelvi modellek pontosabbak a karakterfelismerésben főleg, ha diagramon, folyamatábrán szerepel a felirat, jobban követik a dokumentum struktúráját, képesek részletes leírást készíteni az ábrákról és alapvető kimeneti formátumuk a JSON struktúra, amely sokkal alkalmasabb további szoftveres feldolgozásra mint a Markdown. A tesztek során kipróbáltuk, hogy a PyMuPDF-el vagy Tesseracttal már felismertetett szöveget hozzáadjuk a prompthoz: a nagyobb modellek ezt teljesen figyelmen kívül hagyták, de a kis modelleket megzavarta és önismétlésbe kezdtek. A költségeket és feldolgozási időt figyelembe véve ezért a hagyományos rendszerek teljes elhagyása mellett döntöttünk a folyamatból. Az LLM-mel történő feldolgozás gyorsítását pedig aszinkron hívásokkal értük el ügyelve a szolgáltató (pl. OpenAI, Anthropic) által meghatározott lekérdezési korlátozásokra (rate limit).

A több oldalra szétszórt fejezetek problémáját felismerve a PDF oldalait egy általunk létrehozott JSON szerkezetbe töltöttük be, amely oldalanként tárolta a fejlécet, láblécet, külön az esetlegesen felismert oldalszámot, valamint kép, táblázat és törzsszöveg típusú blokkokban az oldalon megtalálható szövegrészeket, ábrák leírását. Minden szövegblokk tartalmazott címet is, ahová opcionálisan a kép, táblázat vagy a kezdődő fejezet címét írta be az LLM kérésünkre. Ha egy szövegblokk nem rendelkezett címmel, de valamelyik előző oldalakról származó szöveg folytatása volt, akkor egy speciális cím

beírására kértük a nyelvi modellt. Ezután az előző oldal végéről blokkonként visszafelé haladó algoritmussal összekötöttük az aktuális oldal elejétől indulva ezeket a speciális címmel ellátott JSON részeket. Mivel ezt szekvenciálisan oldalanként elvégeztük, ezért a több oldalra szétszórt fejezeteket láncolt listába tudtuk szervezni és sikeresen összekötni. Például a 468. oldal alján kezdődő fejezethez hozzákapcsoltuk a 470. oldal alján folytatódó bekezdéseket és végül a 471. oldal közepén található befejező részt.

Az LLM-k a PDF feldolgozás során sok esetben kényszeresen befejezték a félbehagyott mondatokat. Erről a szokásról a hőmérséklet 0-ra állításával és specifikusan megfogalmazott mondatokkal szoktattuk le őket. Például: „You must extract all text from the page image exactly as it appears in the image. Including the text at the top of the page, which in most cases is a continuation of the previous page, hence the lack of a title". A Google Gemini és a Claude Sonnet modellek esetén az oldalak tömeges feldolgozásakor újabb problémába ütköztünk, mivel ezek a modellek több oldal után hibával tértek vissza arra hivatkozva, hogy nem szöveges tartalmak szó szerinti visszaadására találták ki őket (Anthropic Privacy Center, 2024). Ennek valószínű oka az ellenük indított sajtóperektől való félelem lehetett. Ezeket az LLM-ket sajnos nem lehet OCR feladatokra használni. Az OpenAI GPT-4o pedig 1 oldal esetén tartalomszűrő (content filter) hibával tért vissza, ami azt jelenti, hogy veszélyesnek ítélte meg az arcfelismerő szoftverekről szóló tankönyvi esettanulmányt, amit a feldolgozandó PDF oldal tartalmazott. A prompt módosításával nem sikerült megkerülni a problémát ezért tartalék LLM (Llama 3.1 90B) bevetése mellett döntöttünk, amely képes volt feldolgozni az oldalt.

2.2.2 Releváns szövegrészek azonosítása felhasználói kérdéshez

A releváns szövegrészek keresése szempontjából a fejezetek kiváló kiindulási pontot biztosítanak, mert a megfelelően szerkesztett könyvek, cikkek egy fejezetben ugyanahhoz a témakörhöz kapcsolódó gondolatokat fogalmazzák meg, viszont továbbra is túl hosszúak általában kivéve a képeket és táblázatokat. Az előbbiekhez az LLM által készített leírások hossza átlagosan 500-600 karakter, így alatta marad a tapasztalati alapon meghatározott 1000 karakteres határnak. A táblázatokat pedig nehéz lenne szétvágni szemantikusan, de a tapasztalatok alapján nem haladják meg ezek sem az előbb említett határt. Az 1000 karakternél hosszabb szöveges fejezeteket először rekurzív módon a bekezdések végét jelző dupla sortöréseknél, mondatok végén igyekeztünk szétvágni a jól ismert RecursiveCharacterTextSplitter metódussal, de több esetben is egybetartozó szövegrészeket vágott ketté. Ezután egymás után következő 1 mondatban különböző 3 mondatos blokkok távolságát (1 – koszinusz hasonlóság) vizsgáltuk, és a távolságok 95%-nál nagyobb értékek esetén vágtunk, de ebben az esetben teljesen kiegyensúlyozatlan lett az egyes eredményként kapott szövegrészek hossza. Végül ágens, azaz LLM-k segítségével végzett darabolás mellett döntöttünk, mert az LLM értelmezte a szöveget, és az egyes gondolati összefüggések végén határozta meg a vágási pontokat (itt a GPT-4o-t és a 4o-mini-t alkalmaztuk). Ezt erősítette, hogy sok esetben bekezdések végén darabolta a szöveget pedig a promptban ilyen irányú instrukció nem szerepelt, hanem csak a szöveg szemantikai egybetartozására vonatkozó utasításokkal láttuk el.

A legtöbb helyes válasz elérése szempontjából célszerű a legjobban illeszkedő, legnagyobb relevanciával rendelkező dokumentumrészeket átadni az LLM-nek, hogy válaszát ezekre a szövegrészekre alapozza és mást ne vegyen figyelembe, ne hallucináljon. Így jelentősen lehet csökkenteni a kitalált tények mennyiségét, mivel ha nem

határozható meg az átadott szövegrészek alapján a válasz, akkor a promptban arra utasítottuk az LLM-t, hogy térjen vissza a kontextus alapján nem adható válasz szöveggel. Kétfelé egymástól teljesen eltérő keresési módszert alkalmaztunk, hogy megtaláljuk az összes kapcsolódó dokumentumrészt. A FAISS vektoradatbázisba betöltöttük a vektorokká alakított dokumentum darabokat és a felhasználó eredeti kérdését, valamint LLM-mel generált 3 db hasonló kérdést is vektorizáltunk (Ma és mtsai, 2023), majd koszinusz távolság alapján megkerestük a jelentéstartalom alapján közel álló szövegrészeket és a relevancia pontszámát is meghatároztuk. Ugyanezt végeztük el a BM25 módszerrel is, ami nem vektorizál, hanem szavakat, kifejezéseket keres és a kulcsszavak, valamint a dokumentumok hosszának figyelembevételével állapítja meg azok relevancia pontszámát (Robertson és Zaragoza, 2009). Az algoritmus különösen alkalmas olyan alkalmazásokhoz, amelyeknél a kulcsszavak megjelenése fontos szerepet játszik a releváns találatok kiválasztásában. A BM25 súlyozza a dokumentumban előforduló szavak gyakoriságát, ugyanakkor normalizálja a hosszabb dokumentumok értékelését, így elkerüli a hosszabb szövegek túlértékelését.

Ezután normalizáltuk a kapott relevancia pontszámokat és a FAISS, BM25 találati listákat a normalizált pontszámok alapján egyesítettük külön az eredeti felhasználói kérdésre és külön a 3db generált hasonló kérdésre. Így 4 relevancia listát kaptunk, amelyeket a Reciprocal Rank Fusion módszerrel újra rendeztünk (re-ranking) és összefűztünk. Ennek a végső listának a tartalmát kapta meg kontextusként a válaszadásra kiszemelt LLM a felhasználó eredeti kérdése mellé.

Kipróbáltuk még a hasonló kérdések generálása helyett, hogy a dokumentum fejezeteiről egy bekezdés hosszúságú összefoglalókat generálunk LLM segítséggel és a hasonlósági kereséssel vektoradatbázisban megtalált releváns szövegrészekhez ezeket csatoljuk (Anthropic, 2024b). A célja ennek elsősorban az volt, hogy az adott szövegrész kapcsolódását lehessen látni a fejezet egészéhez, miközben nem növeljük jelentősen a bemeneti tokenek számát.

2.2.3 Felhasználói kérdések (promptok) tesztelése nyelvi modelleken

A kérdés-válasz rész tesztelése a felhasználóktól várható jellemző kérdések segítségével történt mely során az adott tantárgy tesztkérdéseit használtuk: 69 kérdéssel teszteltünk és értékeltünk 8 kiválasztott nyílt és zárt LLM-t (GPT-4-turbo, GPT-4o, GPT-4o-mini, Claude Sonnet 3.5, Mixtral-8x22B-Instruct, Qwen2-72B-Instruct, Meta-Llama-3.1-405B-Instruct, Google Gemma-2-27b-it), hogy a pontosság és költségek szempontjából optimális megoldást találjunk. Az értékelés során pontos egyezést kerestünk a feleletválasztós és igaz/hamis kérdésekre, míg a fennmaradó kérdéseket a GPT-4o-val értékeltük, összehasonlítva a várt és a kapott válaszokat 0-1 skálán. A végső eredményeknél a manuális ellenőrzés tapasztalatai szerint csak a 0,8 feletti értékeket fogadtuk el, mert ezek mindig helyes válasznak bizonyultak a tesztesetekben.

3 A tesztek leírása és a vizsgált nagy nyelvi modellek költségei

A RAG implementálása után a kutatás célja annak kiderítése volt, hogyan lehet a költségeket csökkenteni egyrészt a tananyag feltöltési folyamat optimalizálásával (beleértve a keletkező tudásbázis építését és tesztelését LLM-mel), másrészt a hallgatói

kérdések megválaszolása során alkalmazott nagy nyelvi modell költséghatékony megválasztásával (mivel ma már elérhetők zárt-súlyú modellek különböző árú fizetős előfizetésekkel, de vannak nyílt-súlyú, azaz ingyenes megoldások is, melyek futtahatók lokálisan vagy a felhőben).

Az 1. táblázat mutatja az általunk tesztelt összes (12) modell jelenlegi árazását bemeneti és kimeneti tokenek számának függvényében 1 millió tokenre vetítve. A könyv 1 oldala feldolgozási árának meghatározásához figyelembe vettük az általunk készített, részletes instrukciókat tartalmazó OCR feldolgozási lépést leíró prompt hosszát (1096 token) a bemeneti tokenek árával, valamint kiszámoltuk a könyv oldalairól 200 dpi felbontással készített teljes oldalas képek (1654px x 2339px) bemeneti költségeit is. A kimeneti költségeket több oldal hosszának átlagolásával (891 token) határoztuk meg. A bemeneten megadott képek árazása felbontásfüggő és eltérő az egyes szolgáltatóknál, többen tokenekre konvertálják, míg az OpenAI, Google költségszámítást biztosít. Például GPT-4o 0,0028 USD-t kalkulál 1 oldal képére, míg a Claude $\frac{(szélesség \times magasság)}{750}$ képlettel számolva 5158 tokenre konvertálja át a képet, amelyből a bemeneti tokenek árával felszorozva megkapjuk a képfeldolgozás költségét. Az ingyenes modelleket a Together.ai szolgáltatónál vettük igénybe, amely optimális ár-érték arányt nyújt a nyílt-súlyú modellek futtatásához. Az összes modellt a Mixtral 8x22B-től kezdve ennél a szolgáltatónál teszteltük. Ezekben az esetekben a kép költségét a $\min\left(2, \max\left(\frac{magasság}{560}, 1\right)\right) \times \min\left(2, \max\left(\frac{szélesség}{560}, 1\right)\right) \times 1601$ kell számolni, ami 6404 plusz bemeneti tokent eredményez. Mivel 1 oldal költsége viszonylag kicsi, ezért kiszámoltuk a teljes 649 oldalas könyv feldolgozási összköltségét, ahol az OCR költségekhez még hozzáadtuk az ágens alapú feldaraboló prompt hosszát és annak kimenetét.

Az OCR lépést csak olyan modellekkel lehet megvalósítani, amelyek rendelkeznek vision képességekkel (lásd 1. táblázat 4. oszlop). A többi modellt csak a kérdés-válasz tesztelésnél lehetett felhasználni (de a teljes könyv költségnél az OCR helyett a Together.ai oldal árát vettük figyelembe, ezért dőlt betűvel jeleztük, hogy ez az érték csak elméleti). Látható, hogy a legdrágább megoldásnak a vision modellek közül a GPT 4 Turbo (52,51 USD) bizonyult, amelynek árát az OpenAI szándékosan magasan tartja, hogy inkább a GPT-4o felé terelje a felhasználókat a saját erőforrásainak védelme érdekében. A Claude Sonnet 3.5 (41,7 USD) lett a 2. helyezett, míg az OpenAI GPT-4o (18,71 USD) követi a 3. helyen. Mindkét modell 100%-os eredményt ért el az OCR tesztjeink során, ezért az ár alapján érdemesebb a GPT-4o-t választani.

A tokenek árazása tapasztalataink alapján erős korrelációt mutat a modellek méretével. Ebből következően a 8-13 milliárd paraméteres tartományban mozgó LLM-ek a legolcsóbbak, sőt a táblázatban nem szereplő MiniCPM Llama3 8B modellt lokális GPU hardveren futtattuk, így ingyenesnek tekinthető, ha a GPU beruházási költségeit nem vesszük figyelembe. Azonban hiába kerülnek jóval kevesebbe ezek a modellek, sajnos az eredményeik is sokkal pontatlanabbak, mint a nagyobb modellek esetén. Például a MiniCPM 59,28%-ban volt pontos, a Llama 3.2 11B szavak egyezése szempontjából ugyan 98%-ot ért el, de a JSON kimenetben megduplázta az egyik bekezdést egy további szövegblokkot hozzáadva az oldal tartalmához.

A várható költségekről előzetes becslést végeztünk a teljes választott könyv és 1000 kérdés-válasz párra (1. táblázat utolsó 2 oszlop). A kérdés-válasz párok esetén több kérdés bementét és kimenetét átlagoltuk a számításhoz. A kérdéshez hozzávettük a RAG módszer által megtalált releváns szövegdarabok hosszát is. Az 1000 kérdésre

vetített költség alapján a legdrágább az o1-preview lett, amelynek erőssége a bonyolultabb matematikai-logikai következtetések, így nem érdemes kontextus alapú kérdés megválaszolásra használni, hanem a GPT-4o sokkal költséghatékonyabb és gyorsabb opció helyette. A 2. helyen a GPT 4 turbo végzett, amit a Claude Sonnet 3.5, majd a Llama 3.1 405B követ.

1. táblázat: PDF feldolgozás és Q&A költségei zárt és nyílt modellek esetén

(Minden költség USD)	Bemeneti token USD/1M	Kimeneti token USD/1M	1 oldal OCR költség	Kérdés-válasz költség	Teljes könyv 649 oldal	1000 kérdés-válasz
o1-preview	15	60	*nincs*	0,0311	*94,32*	31,08
GPT-4-turbo	10	30	0,0405	0,0189	52,51	18,93
GPT-4o	2,5	10	0,0144	0,0052	18,71	5,18
GPT-4o-mini	0,15	0,6	0,0062	0,0003	8,08	0,3108
Gemini PRO 1.5	1,25	5	0,0062	0,0026	7,99	2,59
Claude Sonnet 3.5	3	15	0,0321	0,0068	41,70	6,753
Mixtral-8x22B-Instr.	1,2	1,2	*nincs*	0,0018	*13,07*	1,842
Qwen2-72B VL	1,2	1,2	0,0101	0,0018	13,07	1,842
Meta-Llama 3.1 405B	3,5	3,5	*nincs*	0,0054	*38,12*	5,3725
Llama 3.2 11B Vision	0,18	0,18	0,0015	0,0003	1,96	0,2763
Llama 3.2 90B Vision	1,2	1,2	0,0101	0,0018	13,07	1,842
Google Gemma 2-27B	0,8	0,8	*nincs*	0,0012	*8,71*	1,228

A feldolgozás sebességét aszinkron hívások segítségével gyorsítottuk fel. Ez gyakorlatilag azt jelenti, hogy időben egyszerre küldjük el a feldolgozandó oldalakat az LLM API-nak, amelyek saját erőforrásaik védelme érdekében korlátozzák az egyidőben elküldhető API hívások számát. Általában a korlátok ötféleképpen mérhetők: percenkénti lekérdezések (RPM), napi lekérdezések (RPD), percenkénti tokenek (TPM), napi tokenek (TPD) és percenkénti képek (IPM). A felhasználás függvényében az elsőként elért limit alapján kerül alkalmazásra a korlátozás. Ezen kívül az API hívások használatára előre befizetett összegtől és a használt nyelvi modelltől is függ a korlátozások mértéke. A 2. szint (Tier 2) legtöbb esetben 50 USD befizetésével érhető el és néhány nap várakozási időt is megkövetelnek a szolgáltatók a szint eléréséhez. Az Anthropic Claude Sonnet 1000 RPM-t, 80 000 TPM-t és 2,5 millió TPD-t, míg az OpenAI GPT-4o 5000 RPM-t és 450000 TPM-t, Together.ai modellfüggetlenül pedig 1800 RPM-t és 250000 TPM-t engedélyez. Látható, hogy a Claude esetén a legszigorúbbak a korlátozások. A feldolgozás gyakorlati tapasztalatai alapján mind a Claude, mind a Together.ai esetén mesterséges várakoztatást (sleep) kellett alkalmaznunk ahhoz, hogy a korlátozást elkerüljük. A kódba automatikus háromszori újra próbálkozást építettünk be arra az esetre, ha az API hívás rate limit hibával tért vissza.

Összesen 69 hivatalos tesztadatbankból származó vizsgakérdést teszteltünk automatizáltan a választott tankönyv 11. fejezetéhez kapcsolódóan. A tesztadatbank tartalmazta a kérdésekhez tartozó hivatalos válaszokat is. A tesztadatbázis feleletválasztós (43 db), igaz/hamis (16 db) és esszé (10 db) kérdést tartalmazott, amelyeket pontos

minta alapú (pattern matching) egyezéssel, illetve a GPT-4o modell használatával értékeltünk ki. A GPT-4o az esszé kérdések helyességét 0-tól 1-ig terjedő skálán állapította meg kérésünkre (prompt), ahol a határt manuális ellenőrzés után 0,8-nál állapítottuk meg.

4 Eredmények

4.1 A tudásbázis-építés tapasztalatai

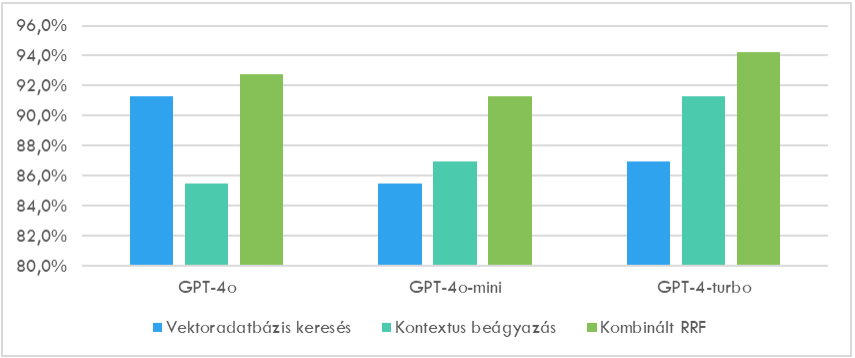
A tesztelt hét modell és egy OCR szoftver közül 100%-os pontossággal a GPT-4o és a Claude Sonnet 3.5 végzett az élen. Az utóbbi viszont kiesett egy, a tömeges feldolgozás során előkerült említett probléma miatt: a Claude Sonnet több oldal beküldése után a saját használati feltételeire hivatkozva megtagadja az oldalak OCR feldolgozását, mivel készítői szerint nem arra tervezték, hogy tartalmakat szó szerint visszaadjon („regurgitating"). A következő a nyílt Qwen2 72B lett 99,87%-kal, majd ezt követte az ugyancsak nyílt Llama 3.2 90B 98,96%-kal. Ez világosan mutatja, hogy az OCR feladatok területén a nyílt modellek felzárkóztak a zárt LLM-ekhez. A Llama 3.2 11B is ilyen eredményt ért el, de megduplázta az egyik bekezdést a JSON kimenetben, ami jelentősen rontja a későbbi használhatóságot. A Tesseract látszólag jó eredményt ért el, de a többi hasonló eredményhez képest sokkal több szót rosszul ismert fel vagy feleslegesen tett bele a dokumentumba, ami leértékeli az elért eredményt. A kis modellek érték el a legrosszabb eredményt. A lokálisan futtatott MiniCPM csak 59%-ban adott helyes eredményt.

Költségeket tekintve a kis modellek bizonyultak a legolcsóbbnak, de pontosságuk igen gyenge. A GPT-4o esetén 18,71 USD a teljes 649 oldalas könyv feldolgozása, míg a Qwen2 72B és a Llama 3.2 90B felhasználásával csak 13,07 USD. Ez az árkülönbség azonban nem jelentős, ha a pontosságot is figyelembe vesszük, mivel a GPT-4o 100%-ot ért el, míg az utóbbi 2 modell kevesebbet. A Google Gemini PRO 1.5 pontossága is hasonló a két nyílt modelléhez, és az ára jóval alacsonyabb, 7,99 USD, azonban a Claude Sonnethez hasonló hiba merült fel a tömeges OCR végrehajtása során, amelyet a Google tiltott recitationnak nevez.

A fentiek alapján a chatbot megvalósításánál választásunk a GPT-4o-ra esett, tartalomszűrő hiba esetén a Llama 3.2 90B tartalék LLM bevetésével.

4.2 Keresési módszerek és rangsorolás

Három módszert hasonlítottunk össze a 69 kérdésen a választott GPT-4 három változatán (lásd 1 ábra): vektoradatbázis keresés, kontextus beágyazás és kombinált megközelítés. Ezek közül a kombinált RRF (Cormack és mtsai, 2009) módszer teljesített a legjobban, különösen a GPT-4-turbo modell, amely 94,2%-os pontosságot ért el teljes egyezés és 98,55%-ot a 80%-os egyezés esetén.



1. ábra: A 3 keresési módszer tesztelése: GPT-4o, GPT-4o mini és GPT-4 turbo (saját munka)

4.3 Kérdés-válasz teszt eredmények

Az előző teszt eredményeképpen bebizonyosodott, hogy a legjobban teljesítő módszer a vektor adatbázis és BM25 kereséssel kombinált RRF algoritmus. Ezért a korábban kiválasztott 8 LLM-et egységesen teszteltük mind a 69 tesztkérdésre a kombinált RRF módszer felhasználásával. Csak egy kisebb modellt választottunk (Google Gemma 2 27B), mivel előzetes tapasztalataink alapján a kis paraméterszámmal rendelkező modellek nem teljesítettek jól. Mint a 2. táblázatban látható, ez itt is beigazolódott, mivel a Google Gemma eredménye (86,96%) az utolsó helyre volt elegendő, de azért nem sokkal maradt el a csúcsmodellek eredményétől. A legjobb eredményeket a paraméterek számának függvényében megvizsgálva észrevehető, hogy az bár alapvetően befolyásolja a teljesítményt, a jó eredmény nem csak ezen múlik. A Qwen2 72B nyílt modell (mely a közepes méretű LLM-ek táborához tartozik) végzett a 2. helyen (95,65%) a GPT-4-turbo (98,55%) mögött (mely egyes források szerint 8x222 milliárd paraméterrel dolgozik). Ehhez képest a 405 milliárd paraméteres Llama 3.1 már picivel rosszabbul teljesített.

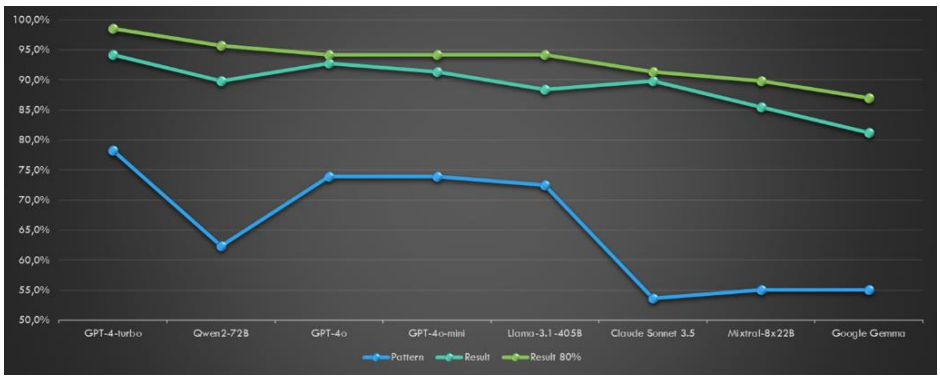
A mintázat alapú egyezés a feleletválasztós és az igen/nem kérdések esetén volt alkalmazható. Ekkor a megfelelő betűjelet vagy az igen/nem értékeket kerestük a válaszban, és ezt hasonlítottuk össze a hivatalos eredménnyel. Az esszé kérdések így eleve kiestek ebből a körből, és meglepő módon a kérdésben egyértelműen nagybetűvel megjelölt válaszlehetőségek betűjelét sem mindig írták bele az LLM-ek válaszukba. A legtöbb esetben így is helyes volt a válasz, azonban ez már csak a GPT-4o-val elvégzett kiértékelés során derült ki. A modellt arra kértük megfelelő prompt, az eredeti kérdés, a kapott válasz és a hivatalos válasz megadásával, hogy hasonlítsa össze az eredményt a hivatalossal, és 0-1-ig terjedő skálán pontozza. A GPT-4o minden esetben 1 tizedesre kerekített pontszámokat (pl. 0,9, 0,8) adott, és az egyes különböző pontszámokhoz tartozó válasz darabszámokat külön is kigyűjtöttük.

2. táblázat: A tesztkérdések kiértékelése 4 zárt és 4 nyílt LLM-el tesztelve

	Zárt				Nyílt			
FAISS és BM25 keresés RRF-el rangsorolva	GPT-4-turbo	GPT-4o	GPT-4o-mini	Claude Sonnet 3.5	Qwen2-72B-Instruct	Meta-Llama-3.1-405B-Instruct	Mixtral-8x22B-Instruct	Google Gemma-2-27b-it
Minden kérdés	69	69	69	69	69	69	69	69
Mintázat alapú egyezés	54	51	51	37	43	50	38	38
GPT kiértékelt eredmények	15	18	18	32	26	19	31	31
Összes helyes eredmény (1.0 + mintázat)	65	64	63	62	62	61	59	56
Eredmény (%)	94,20	92,75	91,30	89,86	89,86	88,41	85,51	81,16
Eredmény (0.9)	2	0	0	0	2	1	2	1
Eredmény (0.8)	1	1	2	1	2	3	1	3
Eredmény (0.7)	0	1	1	0	0	0	1	2
Eredmény (0.5)	0	0	0	0	1	1	1	0
Eredmény (0.0)	1	3	2	6	2	3	5	7
Futásidő (sec)	127	127	127	815	375	821	489	635
Eredmény (%) 0.9, 0.8 tartalmazva	**98,55**	**94,20**	**94,20**	**91,30**	**95,65**	**94,20**	**89,86**	**86,96**
Minta összes (kiv. esszé %)	78,26 (91,53)	73,91 (86,44)	73,91 (86,44)	53,62 (62,71)	62,32 (72,88)	72,46 (84,75)	55,07 (64,41)	55,07 (64,41)

Bár csak a mintázat alapú egyezéseket figyelembe véve (táblázat utolsó sora) több modell rosszul teljesített, a GPT-4o-val végzett 1 pontot kapó kiértékeléseket is figyelembe véve már minden modell 81% felett teljesített (táblázat Eredmény % sora). Megvizsgálva a nyelvi modell által végzett automatikus kiértékelés pontszámait, megállapítottuk, hogy a 0,8 és 0,9 pontszámmal rendelkező válaszok is elfogadhatók jó megoldásnak. A legtöbb esetben azért vont le a GPT-4o 1 tizedet, mert a válasz terjengősebb volt, és olyan állítások is bekerültek, amelyek nem feltétlenül voltak szükségesek. Ellenben a 0,7 és 0,5 pontszámok esetében a válasz nem volt jó és több esetben rossz következtetéseket is tartalmazott. A 0 esetén leginkább a promptban megfogalmazott kérésünknek tett eleget a vizsgált LLM, mert azt válaszolta, hogy a kapott kontextus alapján nem tudja a megoldást.

A 2. ábra a 2. táblázat 3 legfontosabb eredmény sorát ábrázolja. A kék vonal a mintázat alapú egyezés százalékos eredményeit mutatja, a világoszöld a mintázat mellett 1 pontot kapó eredményeket, míg a zöld a végső eredményt, amely a 0,8 és 0,9 pontszámokat kapó válaszokat is tartalmazza. Ez utóbbi alapján rendeztük csökkenő sorrendbe az egyes modelleket. A legjobb eredményt a GPT-4-turbo érte el 98,55%-kal, ami azért érdekes, mert a GPT-4o fejlettebb modellnek számít. A 2. helyen egy nyílt modellt találunk, a Qwen2 72B-t 95,65%-kal, amelynek az előzőekben ismertetett token költsége is nagyon alacsony.



2. ábra: A legjobb, RRF alapú módszer tesztelése a 8 választott LLM rendszeren (saját munka)

5 Tanulságok összefoglalása

A felsőoktatási, RAG módszerekkel testre szabott chatbot esetében részletes költség- és teljesítményelemzést végeztünk többféle méretű nyílt és zárt súlyú LLM összehasonlításával. A 649 oldalas, képeket is tartalmazó PDF feldolgozási folyamatban 100%-os karakterfelismerési pontosság legkedvezőbben 18,71$-ért érhető el (GPT-4o). A GPT-4o-mini vagy open-source modellek használatával ez az összeg tovább csökkenthető. Az OCR jellegű felhasználásban a nyílt modellekre mindig támaszkodhatunk, mivel a szabály alapú, standard feldolgozóknál pontosabb, strukturáltabb eredményt adnak és zárt társaik a tartalmi szűrések miatt néhány esetben nem végzik el a feladatot. A tartalomszűrű intézkedés miatt a 13,07$-os nyílt súlyú Llama 3.2 Vision 90B a tartalék modell a TogetherAI szolgáltatást használva, a teszt alapján 98,96%-os pontosságot ér el.

A kérdés-válasz feladatban a legjobb RAG módszert 8 LLM-re teszteltük 69 hivatalos tesztkérdéssel. Ezek közül a GPT-4-turbo 98,55%-ban helyes válaszokat adott, a legjobb pontosságot elérve. Az eredmények az mutatják, hogy a nyílt súlyú rendszerek elérhetik a SOTA modellek szintjét a RAG alapú kérdések megválaszolásában (megfelelően megválasztott tudástárolási megoldások és lekérdezések alkalmazásával), mivel a Qwen2-72B 95,65%-os pontossága felülmúlta a GPT-4o (94,2%) és Claude Sonnet 3.5 (91,3%) megoldását. A Llama 3.1 405B egy szinten teljesített a legújabb OpenAI modellekkel. Az 1000 kérdésnél fellépő költségeket figyelembe véve a GPT-4o-mini a legjobb költséghatékony választás, míg a Qwen2-72B a nagyobb zárt modellek költségei alatt biztosít jobb teljesítményt. Mivel zárt modellek esetén a kimeneti token drágább, így érdemes arra optimalizálni.

Köszönetnyilvánítás

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

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

Bibliográfia

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

Ariga M. 2016. Tabula. https://github.com/chezou/tabula-py. Letöltve: 2024.12.23

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

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

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

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997. *arXiv.org.* https://arxiv.org/abs/2312.10997

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

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

Shinyama Y. 2019. PDFMiner. https://github.com/euske/pdfminer. Letöltve: 2024.12.23

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

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

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

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