Önálló laboratórium 2 beszámoló

Távközlési és Médiainformatikai Tanszék

Készítette:	**Szladek Máté Nándor**
Neptun-kód:	TGPZTT
Ágazat:	**Adattudomány és Mesterséges Intelligencia**
E-mail cím:	**szladekmate@gmail.com**
Konzulens(ek):	**Dr. Gyires-Tóth Bálint és Ónozó Lívia Réka**
E-mail címe(ik):	**tothb@tmit.bme.hu és onozo@tmit.bme.hu**

Téma címe: Dokumentumfeldolgozáson alapuló magyar nyelvű chat AI, RAG technológiával

Feladat

Egy olyan szoftverrendszer megtervezése és megvalósítása, amely lehetővé teszi a nagy méretű, strukturálatlan szöveges adatbázisok hatékony kezelését és a releváns információk gyors lekérdezését természetes nyelvi kérdések segítségével. A rendszernek alkalmaznia kell a természetes nyelv feldolgozás és gépi tanulás korszerű módszereit annak érdekében, hogy a közérthetően megfogalmazott lekérdezésekre pontosan a témához kapcsolódó válaszokat adja. A programnak magyar nyelven kell válaszolnia minden esetben, azonban a feldolgozott dokumentumok bármilyen nyelvűek lehetnek.

2023/2024. 2. félév

1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

1.1 Bevezető

A világ exponenciálisan növekvő adatmennyiségének jelentős hányada strukturálatlan formában, különböző szöveges forrásokból származik [3]. Ez az adattömeg kulcsfontosságú információkat rejt, amelyek kinyerése és hatékony felhasználása kimagaslóan fontos volna a tudományos kutatás, az üzleti döntéshozatal és számtalan más terület számára. Sajnos a jelenlegi módszerek gyakran nem elég rugalmasak és pontosak a nagy méretű, változatos adathalmazok kezelésére. A strukturálatlan szöveges adatok fő problémái a nagy mennyiség, a sokféleség és a változatosság. A természetes nyelvi feldolgozás (NLP) és a mély tanulás fejlődése új megoldásokat tett lehetővé ezen kihívások leküzdésére [4].

Természetesen a háttérben továbbra is valószínűségek és statisztikai számítások zajlanak, azonban ezek a rendszerek elérték azt a szintet, amikor egy valós intelligencia látszatát tudják kelteni. Ezek a megoldások lehetővé teszik a felhasználók számára, hogy természetes nyelven fogalmazzák meg a kérdéseiket, amelyekre a rendszer pontosan a lekérdezés tárgyához kapcsolódó válaszokat ad a rendelkezésre álló dokumentumokból, forrásokból [1]

Számos megközelítés létezik a strukturálatlan adatok lekérdezésére, mint például a kulcsszavas keresés, a szemantikus keresés és a kérdés-válasz alapú (Q&A) rendszerek, mint például az elsők közé sorolható IBM Watson nevű mesterséges intelligenciája, amely 2011-ben a TV-ben sugározva győzött le kvízjátékosokat [2]. Azonban ezek a módszerek gyakran nem elég rugalmasak vagy pontosak, különösen a nagy méretű és komplex adathalmazok esetében. Egy ígéretes lehetőséget jelentenek az úgynevezett "Retrieval Augmented Generation" (RAG) megoldások [1], amelyek ötvözik a retrieval, azaz az információ keresési, és a generatív módszereket a hatékony és pontos információkinyerés érdekében.

Jelen projekt célja egy olyan szoftverrendszer megtervezése és megvalósítása, amely a RAG megoldást alkalmazza a nagy méretű, strukturálatlan szöveges adatbázisok kezelésére és a releváns információk rugalmas lekérdezésére természetes nyelvi kérdések segítségével. Ez a megközelítés lehetővé teszi a felhasználók számára, hogy intuitív módon jussanak hozzá a számukra lényeges információkhoz anélkül, hogy manuálisan kellene áttekinteniük a teljes szakirodalmi listát.

1.2.1 Hagyományos megközelítések

A strukturálatlan adatok lekérdezésére hagyományosan számos módszert alkalmaznak, mint például a kulcsszavas keresés, a szemantikus keresés és a kérdés-válasz rendszerek.

A kulcsszavas keresés egyszerű szűrést végez a dokumentumokban fellelhető szavak előfordulása alapján. Bár hatékony és könnyen implementálható, gyakran pontatlan és irreleváns eredményeket ad, mivel a szövegek szemantikáját, kontextusát nem veszi figyelembe.

A szemantikus keresés ezzel szemben a szövegek jelentését is figyelembe veszi a keresésnél, nem csak a konkrét szavakat [5] . Ennek egyik módja a szavak vektoros reprezentációk előállítása, például Word2Vec [6] használata, amelyek képesek értelmezni a szavak közötti szemantikai kapcsolatokat. Módszertani korlátaik azonban, mint például az ún. out-of-vocabulary [1]probléma és a lexikai homonímia[2] kezelésének nehézsége, gyakran pontatlanságokhoz vezetnek.

[1] Out-of-vocabulary (OOV) probléma: Az OOV probléma akkor merül fel, amikor a modell olyan szavakkal találkozik a bemeneti szövegben, amelyek nem szerepelnek a betanítási korpuszban, így a modell nem rendelkezik előre betanított beágyazásokkal ezekre a szavakra. Ez a különösen ritka szavaknál fordul elő általában.

[2] Lexikai homonímia: A lexikai homonímia az, amikor egy szónak több, egymástól eltérő jelentése van.

A kérdés-válasz (Q&A - Question Answering) rendszerek természetes nyelven feltett kérdésekre adnak választ az adatforrás(ok) elemzése alapján. Korai megközelítéseik elsősorban információkinyerési technikákon alapultak, mint szótag-felismerés, relációkinyerés stb. [7]. A mély tanulás megjelenésével a neurális Q&A rendszerek óriási fejlődésnek indultak, egyesítve a retrieval (információ keresési) és generatív modelleket [8]. Az IBM Watson[3] korai sikerei [2] után napjainkban egyre kifinomultabb architektúrák születnek ezen a téren.

1.2.2 Modern neurális megközelítések

A neurális hálózatok és mély tanulási módszerek megjelenése forradalmasította az információ-visszakeresés területét is. A hagyományos technikákat fokozatosan felváltják a kifejezetten erre a célra tervezett neurális architektúrák.

-A figyelem-alapú modellek a figyelmi mechanizmust [9] alkalmazzák a releváns információ kiemelésére a bemenetből. Az ún. token figyelmek súlyozást rendelnek az egyes tokenekezhez annak megfelelően, hogy mennyire fontosak a lekérdezés szempontjából. Ilyen pl. a ByteDance által javasolt BERT-Reranker [10].

-A kontextualizált beágyazási modellek a nyelvi kontextust is figyelembe vevő előtanított nyelvi modellek, mint a BERT [11], RoBERTa [12] vagy XLNet [13] jelentősen jobbak a hagyományos szóbeágyazásokhoz képest. Képesek megragadni a szavak többértelmű jelentését a kontextus alapján.

-A Retrieval Augmented Generation (RAG) modellek [1]ötvözik az információ-visszakeresést és a válaszgenerálást egyetlen architektúrában. Egy visszakeresési (retrieval) komponens keresi meg a releváns dokumentumokat, egy generatív komponens pedig választ generál ezek alapján.

1.2.3 Vektorizált információtárolás szerepe a RAG esetében

A RAG (Retrieval Augmented Generation) modellek hatékony működésének egyik legfontosabb feladata a vektorizált információtárolás, a másik pedig a hatékony vektorizált keresés megvalósítása. Ehhez nélkülözhetetlen a dokumentumok megfelelő vektorizálása, azaz beágyazása egy alkalmas vektortérbe, amelyben a hasonló dokumentumok egymáshoz közel helyezkednek el.

A vektorizálás folyamata jellemzően a következő lépésekből áll:

1. A szöveges dokumentumok tokenizálása megfelelő nyelvi modell segítségével, tokenekre bontva azokat.
2. A tokenekhez kontextusfüggő beágyazások rendelése előtanított nyelvi modellek felhasználásával, figyelembe véve a tokenek környezetét.
3. A token-beágyazások aggregálása dokumentum-beágyazássá, jellemzően egyszerű átlagolással (vagy speciálisabb összesítési műveletekkel).
4. A dokumentum-vektorok normalizálása, dimenziószám-csökkentése vagy más jellemző-előkészítési technikák alkalmazása a hatékonyabb indexelés és keresés érdekében.

Az így előállított vektorizált dokumentum-reprezentációk egy megfelelő indexelő és visszakereső rendszer segítségével tárolhatók és lekérdezhetők. Erre a célra az egyik legmodernebb és leghatékonyabb megoldás a Facebook AI kutatói által létrehozott Faiss[4] (Facebook AI Similarity Search) könyvtár (Johnson et al., 2021).

A Faiss számos különböző adatstruktúrát és algoritmust használ a nagy méretű, sűrű vektortér indexelésére és a hozzá legközelebbi szomszédok közelítő megkeresésére. Támogatja a GPU-k használatát is a folyamat felgyorsítására.

[3] A Watson egy, az IBM által kifejlesztett mesterséges intelligencia, mely arról lett híres, hogy 2011-ben a Jeopardy! című TV-s vetélkedőben (hasonló a magyarországi Mindent vagy semmit! játékhoz) két élő játékost legyőzött.

[4] A FAISS dokumentációja itt érhető el: faiss.ai

A RAG modellekben a Faiss indexelőrendszere tárolja el a vektorizált dokumentum-beágyazásokat, a lekérdezési fázisban pedig hatékonyan keresi vissza a lehető legjobban illeszkedő dokumentumokat vagy dokumentumrészleteket egy adott lekérdezési bemenet vektorizált reprezentációjához képest. Ez a visszakeresési komponens szolgáltatja a generatív nyelvi modell számára a bemenetként felhasznált releváns kontextust a válaszgenerálás feladatához.

1.2.4 A LangChain[5] keretrendszer

A tudásalapú rendszerek és információ-visszakeresés terén folyamatos kutatások és fejlesztések zajlanak a mesterséges intelligencia előretörésével. Ebben a környezetben a LangChain egy nagyon fontos keretrendszer, amely lehetővé teszi a fejlesztők számára, hogy hatékonyan integrálják a nagy nyelvi modelleket, adatbázisokat és egyéb összetevőket komplex, AI-vezérelt alkalmazásokba.

A LangChain egy Python alapú keretrendszer, amelyet a Harrison Chase fejlesztett ki és tett nyíltan elérhetővé 2022-ben. Célja, hogy egységesített absztrakciós réteget biztosítson a különböző nagy nyelvi modellek, dokumentumtárolók, láncolási stratégiák és más összetevők felett, leegyszerűsítve az AI alkalmazások fejlesztését [15]. A LangChain modularitása lehetővé teszi a fejlesztők számára, hogy könnyen kombináljanak és cserélgessék ezeket az összetevőket az igényeik szerint, elősegítve az alkalmazások rugalmasságát és skálázhatóságát.

A LangChain egyik fő erőssége az adat-/ismeretbázis-lekérdezés (knowledge-base querying) és a lekérdezési elemzés (retrieval-augmented generation, RAG) támogatása. Ez a funkció lehetővé teszi a rendszer számára, hogy a felhasználói lekérdezésekre válaszoljon úgy, hogy releváns információkat keres és von ki a betöltött dokumentumokból vagy adat-/ismeretbázisokból. Ez különösen hasznos olyan területeken, mint a tudáskezelés, ügyfélszolgálat, tartalomgenerálás és információ-visszakeresés.

A RAG folyamat a LangChain-ben általában a következő lépésekből áll:

1. Dokumentumbetöltés: A releváns dokumentumok betöltése különböző forrásokból, például fájlokból, weboldalakról vagy adatbázisokból.
2. Szövegfeldolgozás: A betöltött dokumentumok feldarabolása kezelhetőbb szövegrészletekre, amelyek könnyebben kezelhetők a későbbi lépésekben.
3. Vektorizálás: A szövegrészletek átalakítása numerikus vektorokká a megfelelő beágyazási modellel, lehetővé téve a hatékony hasonlósági keresést.
4. Vektortárolás: A vektorizált szövegrészletek tárolása egy kereshető indexben vagy vektortárolóban, esetemben a FAISS segítségével.
5. Lekérdezés: A felhasználói lekérdezések átadása a nagy nyelvi modellnek, amely a vektortárolóban keresi meg a releváns szövegrészleteket.
6. Láncolás: A nagy nyelvi modell a relevánsan visszakeresett információk alapján létrehozza a választ a felhasználói lekérdezésre.

Ez a folyamat lehetővé teszi a rendszer számára, hogy gazdag, kontextusfüggő válaszokat adjon a felhasználói kérdésekre azáltal, hogy összesíti a különböző forrásokból származó releváns információkat. A LangChain rugalmassága miatt ez a folyamat testre szabható és finomhangolható a különböző alkalmazási igényeknek megfelelően.

A LangChain által támogatott nagy nyelvi modellek közé tartozik a GPT-3, a BLOOM, a FLAN és számos más népszerű modell[15]. Emellett a keretrendszer integrálható különböző dokumentumtárolókkal, mint például a Chroma[6], a FAISS, a Weaviate[7] és a Pinecone[8], amik

[5] A LangChain dokumentációja itt érhető el: python.langchain.com
[6] A Chroma dokumentációja itt érhető el: docs.trychroma.com
[7] A Weaviate dokumentációja itt érhető el: weaviate.io

hatékony vektorizált adattárolást és keresést tesznek lehetővé.

A LangChain keretrendszer aktív fejlesztői közössége folyamatosan bővíti funkcionalitását és támogatott összetevőinek számát, lehetővé téve a fejlesztők számára, hogy lépést tartsanak a gyorsan fejlődő mesterséges intelligencia területével.

1.2.5 Etikai és biztonsági kihívások az információ-visszakeresés és tudásalapú rendszerek területén

A nagy nyelvi modellek robbanásszerű fejlődése és térnyerése az információ-visszakeresés és tudásalapú rendszerek terén számos etikai és biztonsági kérdést is felvet. Ezek a hatalmas modellek, melyek milliárdos nagyságrendű paramétertömegeket tartalmaznak, hajlamosak átvenni a betanító adatokból származó rejtett torzításokat, előítéleteket és nemkívánatos tartalmakat [16]. A kimenetükben így megjelenhetnek toxikus, sértő vagy akár veszélyes információk, ami súlyos károkat okozhat. Egy kutatás szerint a BERT típusú nagy modellek akár 60%-ban is átvehetik a betanító adatok toxicitását [17].

A nagy modellek által generált tartalom objektivitása, hitelessége és minősége is gyakran megkérdőjelezhető. A kimenetek hajlamosak túlzó állításokat tenni, hamis információkat közölni, pontatlanul vagy kiszínezve visszaadni a tényeket [18]. Ez súlyos problémákat okozhat olyan kritikus területeken, mint az egészségügy vagy a jogrendszer információellátása.

További etikai dilemmát jelent a nagy modellek betanítására használt óriási adathalmazok eredete, tisztasága és a velük kapcsolatos szerzői jogi kérdések [19]. Az adatvédelmi aggályok is felmerülnek, különösen személyes információk modellbe kerülésével kapcsolatban. Ilyen aggályok alapján például Olaszországban egy időre betiltásra került az OpenAI ChatGPT[9].

A nagy nyelvi modellek biztonsági kihívásai abból fakadnak, hogy ezek a rendszerek sérülékenyek lehetnek a bemeneti adatok manipulálására. Kutatók kimutatták, hogy a modellek bemenetének kis mértékű, célzott módosításával a rendszer eredeti működésétől eltérő, akár ártalmas viselkedésre lehet őket késztetni. Például egy eredeti "stop" utcanévtáblát a módosított bemenet alapján a modell "áteresztő" jelzésként értelmezhet, ami nyilvánvalóan súlyos kockázatokat rejt a valós alkalmazásokban [20]. Ez a probléma rámutat a nagy nyelvi modellek sebezhetőségére a rosszindulatú támadásokkal szemben, ami komoly biztonsági fenyegetést jelent a széleskörű alkalmazásukra nézve.

A megoldási lehetőségek egyrészt a modellek fejlesztése során alkalmazható módszereket ölelnek fel, mint a felügyelt tanulás az előítéletek csökkentésére [21] vagy a magánélet védelmét célzó kriptográfiai eljárások [22]. Másrészt az alkalmazás során is elengedhetetlen a szigorú minőség-ellenőrzés, a szennyezett adatok folyamatos ’tisztítása’, az irányelvek és szabályozás megalkotása az etikus felhasználásra, valamint az átláthatóság és elszámoltathatóság biztosítása a modellezési folyamatokban és eredményekben .

1.3 A munka állapota, készültségi foka a félév elején

Az előző félévben egy egyszerű modellt, a magyar PULI modellt finetuningoltam gazdasági adatokon. Ezeket főként nyilvános cikkekből nyertem. Ebben a félévben több téma iránt is érdeklődtem, és több területen is aktívan olvastam, kipróbáltam az új lehetőségeket. Kísérleteztem a prompt engineeringgel, amelyet kategorizációhoz használtam fel magyar nyelven, továbbá a magyar nyelvű NER modelleket is kipróbáltam. Ezekkel a technikákkal létrehoztam programokat, melyeket a dokumentum végén található GitHub repoban helyeztem el. A fő téma azonban a RAG maradt, ami teljesen új volt számomra, és bár az előző féléves munkám során szerzett ismeretek nem voltak közvetlenül alkalmazhatóak, az LLM-ek terén

[8] A Pinecone dokumentációja itt érhető el: docs.pinecone.io
[9] Részletesebb hír: bbc.com

szerzett tudásomat hasznosnak találtam.

2. Az elvégzett munka és eredmények ismertetése

2.1.1 Nvidia kurzus

A félév során elvégeztem az Nvidia által kínált ingyenes online kurzust, amelynek címe "Building RAG Agents with LLMs" volt. Konzulensem javasolta, hogy tekintsem meg, mivel a nagy nyelvi modellek (LLM-ek) alkalmazása és a velük való munka egyre nagyobb szerepet kap a mesterséges intelligencia területén, és ez a kurzus sok újdonságot mutathat számomra. (A kurzus internetes elérhetősége megtalálható a dolgozat végén.)

A kurzus átfogó betekintést nyújtott a retrieval-augmented generation (RAG) felépítésébe és működésébe. Ezek a rendszerek hatékonyan képesek lekérdezni és felhasználni különböző dokumentumokat, eszközöket és külső információforrásokat a felhasználói kérdések megválaszolásához.

A kurzus tananyaga részletesen foglalkozott az LLM-ek működésével, a párbeszédkezelés technikáival, a dokumentumfeldolgozással, a szemantikus hasonlóságon alapuló szűrési módszerekkel, valamint a vektortárolók használatával a hatékony információ-visszakeresésnél. Mindez a LangChain, Gradio és LangServe keretrendszerek segítségével került bemutatásra.

Bár a kurzus online formában zajlott és mindössze 8 órás volt, a bemutatott koncepciók és kódminták alapos megértése és kipróbálása jóval több időt vett igénybe. Egyes témakörökben, mint például a párbeszédkezelésben vagy a vektortárolók implementálása részben, mélyen el lehetett merülni.

A kurzus hasznosnak bizonyult az LLM-alapú RAG alkalmazások fejlesztésének megismeréséhez. A gyakorlati példákon keresztül megértettem, hogyan lehet hatékonyan használni ezeknek a fejlett rendszereknek a képességeit különböző területeken, legyen szó akár dokumentum-elemzésről, kérdés-megválaszolásról vagy bármely más, szövegfeldolgozást igénylő feladatról.

2.1.2 Adatok és azok feldolgozása

A projekt egyik legfontosabb lépése a megfelelő adatforrások beszerzése és előkészítése volt. Erre a célra négy angol nyelvű tudományos publikációt töltöttem le az ArXiv oldaláról, amelyek mind Magyarországgal, a digitális gazdasággal, az elektronikus kereskedelemmel és a fogyasztói elfogadással kapcsolatos témákat érintettek. A letöltést nem manuálisan, hanem a programból végeztem az ArXivLoader segítségével. A letöltött nyers dokumentumokkal kezdtem meg a munkát, melyek előfeldolgozására a RecursiveCharacterTextSplitter Python osztályt alkalmaztam. Ennek segítségével a dokumentumokat körülbelül 1000 karakter méretű, de átfedő szövegszeletekre bontottam, biztosítva azt, hogy a tartalmi kontextus ne vesszen el a darabok határain. Az optimális értéket nagyban befolyásolják a bemeneti adatok, illetve a modell kontextusmegértő képessége is, azonban az Nvidia kurzus ajánlása szerint az 1000 karakteres méret egy megfelelő kiindulási érték általában. Ezért ezt választottam. A darabolási folyamat során kiszűrtem a 200 karakternél rövidebb töredékeket, mivel ezek túlságosan rövid terjedelműek lettek volna a normális értelmezéshez.

A következőkben bemutatásra fog kerülni néhány modell, amiket mind az Nvidia kreditjeim felhasználásával tudtam használni. Hallgatók és kutatók számára az Nvidia biztosít 4000 kreditet, azonban alapvetően ezért a szolgáltatásért fizetni kell. A modellek a build.nvidia.com oldalon érhetőek el kategorizálva és részletes leírásokkal. A használat előtt ki is lehet próbálni ezeket egy webes szimulált környezetben, hogy könnyebb legyen a választás. A kreditek a kérdések mennyiségének függvényében fogynak. Az oldal API kulcsokat generál, amiket a

programkódba kell írni, vagy a megfelelő könyvtár alkalmazásával a futtatáskor a bejelentkező szövegdobozba kell illeszteni az első használatkor. A legújabb és legmodernebb modellek is megtalálhatóak itt, így innen választottam mindet.

A szövegszeletek vektorizálása volt az előkészítés következő fontos lépése, amihez a Nvidia által kínált nvolveqa_40k modellt használtam, ami a francia mixtral-8x7b-instruct modellhez lett fejlesztve. Ez a nagyteljesítményű modell képes volt a dokumentumokat vektorokká alakítani egy sokdimenziós térben, lehetővé téve, hogy a rendszer megtalálja a hasonló információkat tartalmazó szövegrészleteket. Az elkészült vektorizált dokumentumrészleteket ezután a FAISS (Facebook AI Similarity Search) vektortárolóba mentettem, amely hatékony kereső- és rangsoroló funkciókat biztosít a későbbi lekérdezések kiszolgálásához. Ezek a lépések alapvetően nagyon intuitívak és nem igényelnek nagyobb, komplexebb gondolkozást.

A FAISS indexelése viszont összetett folyamat volt, amelynek során különböző dokumentumtárolókat hoztam létre az extra információknak, metaadatoknak, valamint a tényleges szövegszeleteknek. Fontosnak tartottam ugyanis azt, hogy amikor a chat AI válaszol, akkor ne csak a válasz, hanem a dokumentum hivatkozása is látható legyen a válaszban. Tehát nem volt elég a szövegeket eltárolnom, hanem a különböző metainformációk is fontosak voltak. Ezeket a különálló dokumentumtárolókat egy iteratív eljárással egyesítettem és optimalizáltam egyetlen, konszolidált vektortárolóba. Az elkészült, aggregált adatbázis végül 166 dokumentumrészletet tartalmazott a tudományos publikációkból származó, részben átfedő információkkal, megfelelően előkészítve a Retrieved Augmented Generation (RAG) architektúra megvalósításához. Más dokumentumbemenetekkel természetesen más mennyiségű dokumentumrészletet kapunk. A program nem kifejezetten ezekre a dokumentumokra lett fejlesztve, így szabadon cserélhetőek. Különböző paraméterezésű, különböző számú dokumentumokhoz a program tökéletesen alkalmazkodik. Egyedüli gondot a karakterkészletek változékonysága jelentheti, azonban mivel a legtöbb dokumentum angol nyelvű, valamint jelenleg minden dokumentum az Arxiv adatbázisából származik, ami megfelelő kódolást használ, így ezzel a problémával nem foglalkoztam, azonban egy esetleges ázsiai kiterjesztés során ez már gondot okozhat.

2.1.3 Felhasznált modellek és módszerek

A dokumentumok előkészítését és vektorizálását követően rátértem a rendszer központi komponenseinek kidolgozására, ahol az Nvidia kurzusból megismert különböző nyelvi megoldásokat alkalmaztam. A legfontosabb a már fentebb is említett mixtral-8x7b-instruct nagy nyelvi modell, melyet a lekérdezések megválaszolására használtam. Ez a fejlett generatív modell képes természetes nyelvi bemeneteket értelmezni, és azokra részletes, kontextusfüggő válaszokat generálni a dokumentumokból kinyert információk felhasználásával. Próbálkoztam más modellekkel is, azonban a kontextusablak mérete megkötéseket jelentett. Jó kompromisszumos megoldást jelent, ha a párbeszédet megfelelő kontextusablakon belül tartjuk, azonban ez információveszteséggel is jár. Minden egyes kérésnél majdnem 5000 karakteres adatcsomag kerül elküldésre a modellnek, ami tartalmazza a feltett kérdést és a dokumentumból kinyert releváns információkat is. A nyelvi modell ezekből alkotja meg a választ. Egy kisebb kontextusablakkal rendelkező modell esetén a dokumentumokból kinyert információk méretét is korlátozni kell, ami kevesebb, vagy esetlegesen téves információkhoz is vezethet a válaszban. Ennek a kompromisszumnak a mérlegelése azonban minden modell esetében egyedi. Annak érdekében, hogy a kontextusablakban maradjak, a chunk_size és a chunk_overlap paraméterek megfelelő beállításával végeztem. A felhasznált modell által nyújtott lehetőségeken belül kellett maradnom, de az Nvidia ajánlásnak is eleget szerettem volna tenni, így a chunk_size 1000 karakter, a chunk_overlap pedig 100 karakter lett. Ez utóbbi azt állítja be, hogy a szomszédeos szeletek 100 karakternyi átfedésben vannak egymással. Nagyobb kontextusablak esetén sem

érdemes egyébként túlzottan nagy chunk_size értékeket választani, mivel ekkor feleslegesen sok információ kerülhet a kontextusba, ami veszélyezteti a válasz pontosságát. Az értékek minden esetben kompromisszumosak, de az általam használtak megfelelő választásnak bizonyultak.

A modell hatékony működéséhez egy ún. "prompt engineering" folyamatra volt szükség, hogy megfelelelően elkészítsem az inputokat a modell számára. Ezen lépéshez a LangChain keretrendszert használtam, ahol a ChatPromptTemplate osztállyal definiáltam a kérdés-válasz mintákat.

```
chat_prompt = ChatPromptTemplate.from_messages([("system",
    "You are a document chatbot. Help the user as they ask questions about
documents."
    " User messaged just asked: {input}\n\n"
    " From this, we have retrieved the following potentially-useful info: "
    " Conversation History Retrieval:\n{history}\n\n"
    " Document Retrieval:\n{context}\n\n"
" (Answer only from retrieval. Only cite sources that are used. Make
your response conversational.)"
), ('user', '{input}')])
```

1. ábra: prompt engineering technika

Az 1. ábrán látható módon a sablonba belefoglaltam utasításokat, amiket a modell számára írtam, hogy a dokumentum-kivonatokból és a korábbi beszélgetés-történetből származó információkat is felhasználja a válaszok összeállításához.

A releváns dokumentumtartalmak kinyerése a FAISS vektortárolóból történt egy ún. retriever mechanizmus segítségével. Az IndexFlatL2 index módszert használtam a kereséshez, ami egy gyors, bár memóriaigényes megoldás. A legközelebbi szomszédok megtalálásán alapul az eljárás. A felhasználói lekérdezéshez hasonló vektorizált inputokat kerestem a FAISS indexében, majd a legközelebbi illeszkedéseket adtam át kontextusforrásként a modellnek. Az egész folyamatot egy stream_chain pipeline-ba szerveztem, mely összekapcsolta a lekérdezés-átalakítást, dokumentumkinyerést és a modell általi válaszgenerálást.

A rendszer kimenetéhez azonban további funkciókat alkalmaztam, mint például a hosszú kontextusok átrendezése a LongContextReorder transzformer segítségével, hogy a legfontosabb részek a válasz közepére kerüljenek átrendezésre. Ez azért jó, mivel a nagy modellek több figyelmet fordítanak a középre eső részekre a válaszadáskor. Emellett egy beépített fordítási funkciót is implementáltam az OpenAI könyvtár felhasználásával, hogy a rendszer angol nyelvű válaszait magyar nyelvre tudja konvertálni egy külön lépésben. Erre azért volt külön lépésként szükség, mivel nem találtam olyan kontextusablakkal rendelkező modellt, ami ismerte volna a magyar nyelvet kellő mértékben. A generáláshoz használt modell bár megérti a magyar nyelvet, de generáláskor nem tűnik természetesnek a magyar nyelven adott válasza. A magyar nyelvet ismerő modelleknek azonban valamivel kisebb a kontextusablakuk, így ezért döntöttem úgy, hogy egy külön modellt használok a fordításra. Ez a modell a Llama3 lett, ami a Meta fejlesztése és 2024. április 18-án került bemutatásra. Ez tehát egy kimondottan új modell a dolgozat megírásakor. A nehézséget ekkor az jelentette, hogy a felhasznált Nvidia fiókomhoz tartozó API kulcsok egy másodpercen belül több hívást is indítottak az Nvidia felé, aminek hatására vélhetően biztonsági okokból letiltásra kerültek a kulcsok. Az egyik hívás a kérdés és a dokumentumválaszok darabjai voltak (egyben), a másik pedig a fordítási kérés volt. Megoldásként egy rövid, 2 másodperces késleltetést implementáltam a fordítás előtt, továbbá különböző API kulcson keresztül kértem a kéréseket. A probléma ezzel többnyire elhárult és csak rendkívül ritkán észleli támadásnak az Nvidia a kéréseket. Szükség esetén az időablak növelhető.

2.1.4 tágabb architektúra

Az egyes komponensek és modellek integrálását követően rátértem az összetett rendszer végső összeállítására. Az implementáció Python programozási nyelven történt, kihasználva a LangChain keretrendszer nyújtotta lehetőségeket a nyelvi modellek és erőforrások hatékony kezeléséhez. A program GPU optimalizáció nélkül készült, ugyanis nem tartalmaz olyan részeket, amik túlzottan GPU igényesek lennének.

A rendszer magjában a korábban ismertetett stream_chain pipeline áll, mely a felhasználói lekérdezések fogadásáért, a FAISS vektortárolóból való dokumentumkinyerésért, valamint a mixtral modell általi válaszgenerálásért felel egy egységes folyamatban. Emellett több kiegészítő réteget is integráltam a jobb felhasználói élmény biztosítása érdekében.

2. ábra: A pipeline stációi (a példa nem szerepelteti a fordítást)

Ahogyan a 2. ábrán látható, a program a kérdés feldolgozása után összegyűjti a releváns információkat a vektortárolóból, majd átadja egy következő rétegnek, ami a nyelvi megfogalmazásért felel. A kezelőfelület kialakítására a Gradio[10] nevű, Python-alapú könyvtárat alkalmaztam, mely lehetővé tette egy interaktív, chat-stílusú felhasználói felület kialakítását webes környezetben. A Gradio beépített Chatbot funkciójával teremtettem meg a kapcsolatot a felhasználó és a RAG rendszer között.

A frontend (Gradio) és backend (RAG rendszer) összekapcsolása érdekében a chat_gen függvényben definiáltam a lekérdezések fogadásának és a válaszok visszaküldésének folyamatát. Itt hívtam meg a stream_chain pipeline-t, valamint itt zajlott a válaszok opcionális lefordítása is magyar nyelvre a Meta Llama3 modell felhasználásával.

Az eredmények megjelenítésére a Gradio felületén egy iteratív megoldást alkalmaztam, amely karakterenként továbbította és frissítette a modell általi válaszgenerálási folyamatot, ezáltal természetesebb kommunikációs élményt nyújtva a felhasználónak.

Ez azonban csak az angol nyelvnél működik így, ugyanis a fordításhoz nem szabad karakterenként továbbítani az üzenetet, mivel az annyi hívást kezdeményezne a fordítónak, ahány karakterből áll a szöveg. Ráadásul a karakterek lefordításának nincsen nyelvi relevanciája, tehát felesleges is. A generált mixtral modellből érkező szöveget a fordításos változatban egyben adtam át a fordító modellnek, és az ebből érkező válasz került kiírásra a chat ablakon. A fordítás esetében a válaszokat egyben adtam vissza a felhasználónak. Ezt csak azért

[10] A Gradio dokumentációja itt érhető el: https://www.gradio.app/docs

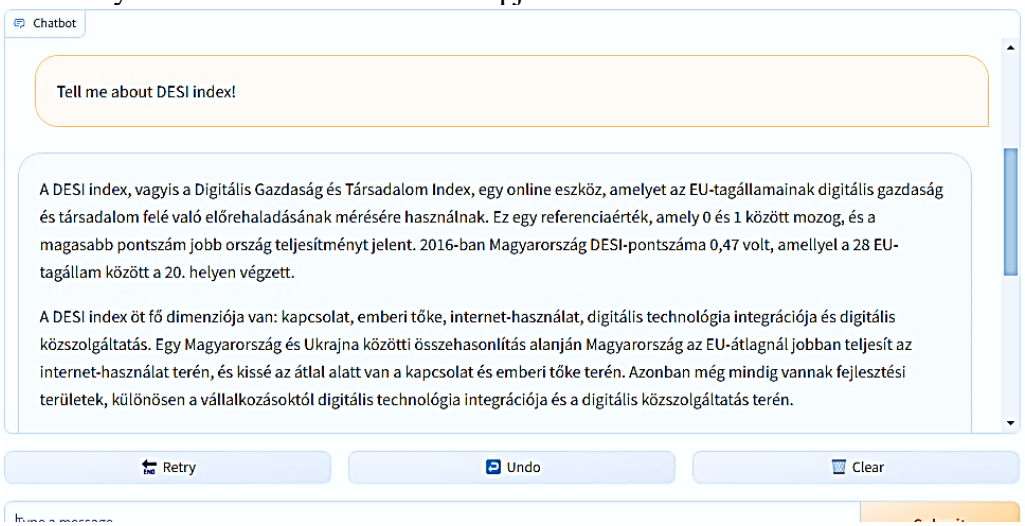
csináltam így, hogy ne kelljen várni a szavakra. Megfelelő sebességgel implementálható a szavankénti kiírás effektus a fordításnál is, ami természetessé teszi a párbeszédet. Azonban mivel komplex és hosszabb válaszokról beszélünk, így úgy gondoltam, hogy a gyorsabb tartalom fontosabb a párbeszédes felhasználói élménynél. Természetesen felhasználói preferenciáktól függően nagyon egyszerűen megoldható a karakterenkénti – szavankénti kiírás a fordítás esetében is.

A teljes alkalmazás futtatása Jupyter notebook környezetben történt a google colab szerverein.

2.1.5 Eredmények és kiértékelés

A rendszer sikeres megvalósítását és integrációját követően lehetőségem nyílt annak kipróbálására és kiértékelésére különböző kreatív kérdésekkel. Az interaktív Gradio felületen keresztül tetszőleges kérdéseket tehettem fel a rendszernek az előzetesen feldolgozott tudományos publikációk témáiban.

A kipróbálás során világossá vált, hogy a RAG architektúra rendkívül hatékony módszert kínál a strukturálatlan dokumentumokban rejlő információk kinyerésére. A program nagyon pontos válaszokat adott és ezeket ráadásul hivatkozásokkal is ellátta. Ezek alapján megállapítottam, hogy a mixtral modell képes volt releváns válaszokat generálni a FAISS vektortárolóból kinyert dokumentumrészletek alapján.



3. ábra: Gradio chatbot ablak (frontend)

Ahogyan a 3. ábrán látható, a rendszer jól teljesített olyan általánosabb témájú kérdések esetén, mint „Mondj valamit Magyarországról!” vagy „Mi az a DESI index?”, visszaidézve és összegezve a publikációk legfontosabb megállapításait.

Emellett sikeresen boldogult specifikusabb információk kinyerésével is, mint például statisztikai adatok, definíciók, folyamatok ismertetése a dokumentumok szövegeiből. A prompt engineering és az egyéb fentebb említett technikák, mint a hosszú kontextusok átrendezése, tovább javították a válaszok pontosságát és érthetőségét.

Egy rendkívül hasznos funkcionalitás volt a válaszok automatikus magyarra fordítása is. Bár a publikációk eredeti nyelve angol volt, a felhasználó magyar kérdéseire is koherens, anyanyelvi válaszokat kapott a rendszertől.

Természetesen akadtak gyengeségek és korlátok is. Túlzottan összetett, többrészes kérdések esetén olykor releváns, de kissé szétszórt válaszokat adott a rendszer. Az sem minden esetben működött zökkenőmentesen, amikor egy lekérdezésre a különböző publikációkból származó információkat kellett volna szintetizálni egységes válasszá. Jól lektorált környezetben az

ellentmondó információk esélye nem nagy, azonban előfordulhat az, hogy egy állítás kétszeres tagadással van leírva egy szövegben, amit nehezen értelmez a program.

Azonban összességében a magyar RAG rendszer megvalósítása sikeresnek tekinthető kísérleti projekt volt. Demonstrálta, hogy a nagy nyelvi modellek és modern vektorizációs technikák integrálásával hatékony, természetes nyelvfeldolgozáson alapuló keresési és információ-visszanyerési megoldások hozhatók létre tudományos szakirodalmak számára. A továbbiakban, és a közeljövőre gondolva arra számítok, hogy akár a Llama3, akár egy másik modell ki lesz bővítve olyan kontextusablakkal, amekkora esetében már nem kell külön fordításos megoldást alkalmaznom. Futási időket és skálázhatóságot jelen esetben nem vizsgáltam, mivel csak proof-of-concept megközelítéssel végeztem a kísérletezést. A továbbiakban ebben az irányban érdemes lehet folytatni a munkát.

2.2 Összefoglalás

A félév során az Önálló laboratórium 2 tárgyban egy olyan természetes nyelvi feldolgozáson alapuló rendszer megvalósításán dolgoztam, amely képes strukturálatlan tudományos publikációkban rejlő információkat kinyerni. A probléma rendkívül releváns, hiszen a tudományos szakirodalmak növekedése mellett nagyon fontos az adatokban rejlő tudás hatékony feltárása és hozzáférhetővé tétele.

A cél egy olyan megoldás kidolgozása volt, amely a nagy nyelvi modellek legújabb fejlesztései, köztük a nvolveqa_40k, mixtral-8x7b-instruct és a Meta Llama3 modelljeinek integrálásával teszi elérhetővé a publikációkban rejlő tartalmat egy egyszerűen használható, természetes nyelvi interfészen keresztül.

A megvalósítás során a Retrieved Augmented Generation (RAG) architektúrát alkalmaztam, amely a dokumentumok vektorizált ábrázolását kombinálja nagy nyelvi modellek generatív képességeivel a kérdések megválaszolására. Az adatelőkészítéshez és indexeléshez a FAISS vektortárolót, a LangChain keretrendszert pedig a rendszerkomponensek összeállításához használtam fel.

Az elkészült rendszer képes volt magyar nyelvű lekérdezésekre, releváns és részletes válaszokat adni a korábban feldolgozott tudományos cikkek tartalmának felhasználásával, valamint a beszélgetést folytatni az előző kérdések és válaszok relációjában. A válaszok minőségét a prompt engineering és egyéb technikák, mint például a kontextusátrendezés (a fontos részletek középre rendezése) tovább javították. Az Meta Llama3 modelljének köszönhetően az eredetileg angol publikációkból is magyar válaszok születtek.

Bár a programnak vannak korlátai az összetett lekérdezések kezelésében, a projekt egyértelműen demonstrálta a nyelvtechnológiák hasznosságát a tudományos szakirodalmak feltárásában.

3. A tanulmányozott irodalom jegyzéke:

[1] P. Lewis *és mtsai.*, „Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”, 2020, Elérés: 2024. május 5. [Online]. Elérhető: https://github.com/huggingface/transformers/blob/master/

[2] IBM, „IBM Watson to watsonx”.

[3] D. Reinsel, J. Gantz, és J. Rydning, „The Digitization of the World From Edge to Core”, *IDC*, nov. 2018.

[4] T. Young, D. Hazarika, S. Poria, és E. Cambria, „Recent Trends in Deep Learning Based Natural Language Processing [Review Article]”, *IEEE Comput Intell Mag*, köt. 13, sz. 3, o. 55–75, 2018, doi: 10.1109/MCI.2018.2840738.

[5] V. Uren, Y. Lei, V. Lopez, H. Liu, E. Motta, és M. Giordanino, „The usability of semantic

search tools: A review”, *Knowledge Eng. Review*, köt. 22, o. 361–377, máj. 2007, doi: 10.1017/S0269888907001233.
[6] T. Mikolov, K. Chen, G. Corrado, és J. Dean, „Efficient Estimation of Word Representations in Vector Space”, jan. 2013.
[7] D. Jurafsky és J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, köt. 2. 2008.
[8] P. Baudi, „YodaQA: A Modular Question Answering System Pipeline”, 2015. [Online]. Elérhető: https://api.semanticscholar.org/CorpusID:196088771
[9] D. Bahdanau, K. Cho, és Y. Bengio, „Neural Machine Translation by Jointly Learning to Align and Translate”, szept. 2014.
[10] Z. Dai és J. Callan, „Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval”, okt. 2019.
[11] J. Devlin, M.-W. Chang, K. Lee, és K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, okt. 2018.
[12] Y. Liu *és mtsai.*, „RoBERTa: A Robustly Optimized BERT Pretraining Approach”, júl. 2019.
[13] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, és Q. V. Le, „XLNet: Generalized Autoregressive Pretraining for Language Understanding”, jún. 2019.
[14] J. Johnson, M. Douze, és H. Jégou, „Billion-Scale Similarity Search with GPUs”, *IEEE Trans Big Data*, köt. 7, sz. 3, o. 535–547, 2021, doi: 10.1109/TBDATA.2019.2921572.
[15] LangChain Inc., „LangChain”. Elérés: 2024. május 6. [Online]. Elérhető: https://python.langchain.com/docs/get_started/introduction
[16] E. M. Bender, T. Gebru, A. McMillan-Major, és S. Shmitchell, „On the Dangers of Stochastic Parrots”, in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, New York, NY, USA: ACM, márc. 2021, o. 610–623. doi: 10.1145/3442188.3445922.
[17] S. Gehman, S. Gururangan, M. Sap, Y. Choi, és N. A. Smith, „RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models”, szept. 2020.
[18] X. Liu, D. Pathak, és K. M. Kitani, „HERD: Continuous Human-to-Robot Evolution for Learning from Human Demonstration”, dec. 2022.
[19] T. B. Brown *és mtsai.*, „Language Models are Few-Shot Learners”, máj. 2020.
[20] H. Jégou, M. Douze, és C. Schmid, „Product Quantization for Nearest Neighbor Search”, *IEEE Trans Pattern Anal Mach Intell*, köt. 33, sz. 1, o. 117–128, 2011, doi: 10.1109/TPAMI.2010.57.
[21] A. Radford *és mtsai.*, „Learning Transferable Visual Models From Natural Language Supervision”, febr. 2021.
[22] C. Song és A. Raghunathan, „Information Leakage in Embedding Models”, márc. 2020.

Csatlakozó egyéb elkészült dokumentációk / fájlok / stb. jegyzéke:

Az alábbi GitHub linken elérhető, illetve akár futtatható is a program:
https://github.com/TGPZTT/Onlab2

Figyelem! Bár a kód mellett szerepel az API kulcs, amelyet használtam, de a kreditek esetleges kimerülése, vagy a kulcsok letiltása esetén ezek nem működnek. Ebben az esetben szükséges regisztrálni a https://build.nvidia.com/explore/discover oldalon és ingyenes vagy megvásárolt kreditek kérdésenkénti beváltására generálható szabadon API kulcs.

A dolgozatban említett Nvidia kurzus ezen a linken érhető el:
https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-FX-15+V1