



Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Szombathy Tamás

NAGY NYELVI MODELL ALAPÚ CHATBOT FEJLESZTÉS

KONZULENS

Marik Gabriella

Dr. Horváth Gábor

BUDAPEST, 2024

Tartalomjegyzék

Összefoglaló	5
Abstract.....	6
1 Bevezetés	7
1.1 Chatbot fogalma, a technológia történelme	8
1.2 Chatbot fejlesztő platformok fogalma	9
1.3 Nagy Nyelvi Modell (LLM)	9
1.4 Beágyazások és beágyazó modell.....	10
1.5 Szemantikus keresés	10
1.6 Prompt és prompt tervezés.....	11
1.7 Retrieval Augmented Generation	11
1.7.1 Naiv RAG	12
1.7.2 Fejlett RAG.....	14
1.7.3 Előnyök és hátrányok.....	15
1.7.4 Alkalmazási területek	16
2 Tervezés	17
2.1 Felhasználási eset elemzése	17
2.2 Chatbotok architektúrája.....	18
2.3 Chatbot építő platformok bemutatása	18
2.4 Chatbotok teljesítmény kiértékelési módszere.....	19
3 Chatbotok implementálása	21
3.1 Chatbotok tudásbázisa	21
3.2 Dialogflow CX chatbot fejlesztése	21
3.3 Llama Index chatbot fejlesztése.....	24
4 Chatbotok kiértékelése	32
5 Chatbot építő eszközök összehasonlítása.....	39
5.1 Fejlesztési élmény, interakció az eszközzel.....	39
5.2 Költségek	40
5.3 Elérhető integrációk, kompatibilitás más eszközökkel.....	40
5.4 Konklúziók.....	41
6 Összefoglaló	42
7 Köszönetnyilvánítások.....	43

8 Irodalomjegyzék.....	44
-------------------------------	-----------

HALLGATÓI NYILATKOZAT

Alulírott **Szombathy Tamás** szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2024. 06. 02.

.....
Szombathy Tamás

Összefoglaló

A modern egyetemek gyakran szembesülnek azzal a kihívással, hogyan lehetne gyors és hatékony informálódási lehetőséget nyújtani hallgatóik és oktatóik számára. Számos más intézkedés mellett egy kérdések megválaszolására alkalmas chatbot használata kielégítő megoldást nyújthat a probléma enyhítésére. A mesterséges intelligencia gyors fejlődésének köszönhetően azonban új technológiák és architektúrák nyertek teret a chatbotok fejlesztéséhez, amelyek egyre természetesebb kommunikációt és komplexebb kérdések megválaszolását teszik lehetővé. Ilyen architektúrára és technológiára példa a Retrieval Augmented Generation és a Nagy Nyelvi Modellek. A szakdolgozat két, Retrieval Augmented Generation és Nagy Nyelvi Modelleket alkalmazó chatbot fejlesztő eszköz segítségével készült chatbot implementálását és több szempontot figyelembe vevő összehasonlítását mutatja be, hogy választ adjon a kérdésre: melyik új technológiákat alkalmazó chatbot építő eszközzel lenne érdemes egy egyetemi gyakran ismételt kérdéseket megválaszoló chatbot létrehozása. Az összehasonlítás több szempontot értékel, köztük a chatbotok válaszainak pontosságát, az eszközökkel való fejlesztés egyszerűségét és a chatbotok karbantarthatóságát, hogy átfogó elemzést nyújtson azok teljesítményéről és gyakorlati alkalmazhatóságáról egyetemi környezetben.

Abstract

Modern universities are confronted with the challenge of offering quick and efficient information to their students and their faculties. To overcome these difficulties, alongside other measures, deploying a chatbot that is capable of answering frequently asked questions can offer a satisfactory solution to mitigate this issue. With the rapid advancements in the field of Artificial Intelligence, new architectures and technologies arose in the field of chatbot technology, such as Retrieval Augmented Generation and Large Language Models, enabling more natural communication and the handling of more complex queries. This thesis presents the implementation and multi-faceted comparison of chatbots developed using tools that leverage Retrieval Augmented Generation and Large Language Models, and answers the question: which new technology-using chatbot building tool would be more suitable for creating a university chatbot that is capable of answering frequently asked questions. The study evaluates several key aspects, including the accuracy of the chatbots' responses, ease of development, and maintainability, providing a comprehensive analysis of their performance and practical applicability in an academic environment.

1 Bevezetés

A modern oktatási intézmények, különösen az egyetemek egyre több hallgatót oktatnak, akik adminisztratív információval való ellátása egyre nagyobb terhet jelent az oktatók és az egyetem szervezetei számára. Továbbá az online oktatásnak köszönhetően a hallgatók és az egyetem között létrejött egy még szabadabb, ám indirektebb kapcsolat is, amely tovább nehezíti a két fél közötti sikeres kommunikációt. Mindemellett a hallgatók elvárják és nagyra értékelik, ha az egyetem információs csatornája is legalább olyan kényelmes, gyors és pontos tájékoztatást tudnak adni, mint más kereskedelmi online platformok. Kijelenthető tehát, hogy a kommunikáció javítása az egyetem és a hallgatók között kulcsfontosságú, ugyanis kihat az egyetem hallgatók által való megítélésére, a hallgatók teljesítményére, az oktatók terheltségére és ezáltal az oktatás minőségére is.

A problémát hagyományos, emberi erőforrást igénybe vevő megoldásokkal, például tudakozó telefon, chat vagy email vonalakkal nehéz, néhány esetben lehetetlen és költséges lenne megoldani. A chatbotok azonban egy olcsóbb és skálázhatóbb megoldást tudnak nyújtani a probléma enyhítésére. A mesterséges intelligencia fejlődésének köszönhetően pedig a chatbotok korszakának egy újabb felvirágzását élhetjük, amely technológiai alapját a transzformátor modell architektúra [1] nagy mértékben befolyásolta. Az architektúra első sikeresebb implementációja a Google által fejlesztett BERT nagy nyelvi modell volt, de áttörő piaci sikert és fellendülést később az OpenAI vállalat által fejlesztett GPT nagy nyelvi modell piaci alkalmazása, ChatGPT chatbot szolgáltatás jelentett. A ChatGPT [2] sikere a generatív mesterséges intelligencia (GenAI) egy új korszakát hozta el, amely a technológiára épülő chatbotok elterjedését eredményezte.

Újabb mérföldkövet jelentett a Retrieval Augmented Generation (RAG) chatbot architektúra feltalálása és elterjedése, ugyanis az architektúra lehetővé tette a modellek számára, hogy az eredeti tanító adatokon kívül külső adatforrásokból származó adathoz férjenek hozzá, amely tovább növelte a GenAI alapú chatbotok használati területét és fejlesztésük egyszerűségét. Az architektúra egyik feltörekvő használati esete a vállalati adattal történő beszélgetés lehetősége, amely a dolgozók számára egy kérdés-válasz alapú chat beszélgetésen keresztül teszi lehetővé, hogy gyorsan és hatékonyan, cégen belüli

információhoz jussanak. A szakdolgozatomban implementálandó chatbotok felhasználási esete nagy mértékben hasonlít ehhez a feltörekvő használati esethez.

Az eddig felsorolt technológiákat több korábbi és újonnan létrejött chatbot építő platform is elérhetővé tette a GenAI trendtől fűtve és kínál ma lehetőséget arra, hogy saját, ChatGPT szerű, saját adatunkat felhasználó chatbotokat hozzunk létre a már megszokott és használt infrastruktúrákban, komolyabb fejlesztést nem igényelve. Azonban felmerül a kérdés, hogy ezek a chatbot építő platformok alkalmasak-e és ha igen, melyik a legjobb egy magyar nyelvű egyetemi chatbot létrehozására, amely gyakori kérdéseket hivatott megválaszolni egy egyetem esetében.

Szakdolgozatomban két gyakran ismételt kérdéseket megválaszoló chatbotot fogok létrehozni, amelyek két innovatív chatbot építő eszközt használva nyújtanak megoldást az ismertett problémákra. A két chatbot megvalósítása által és eredményeik kiértékelése során pedig képet kaphatunk a magyar nyelvű RAG architektúra és LLM alapú chatbotok platformokon való fejlesztésének helyzetéről, illetve választ kaphatunk a kérdésre, melyik chatbot építő platform szolgál jobb választásként az ismertett felhasználási esetben.

1.1 Chatbot fogalma, a technológia történelme

A chatbot egy olyan számítógépes program, amely mesterséges intelligencia (MI) és természetes nyelv feldolgozás (NLP) segítségével érti meg a felhasználó kérdését és válaszol arra, szimulálva egy emberi beszélgetést. Az első chatbot programot 1966-ban Joseph Weizenbaum fejlesztette ELIZA [1] néven, amely szimbolikus természetes nyelv feldolgozást alkalmazott az emberi nyelven írt szöveg megértésére. Egészen az 1980-as évekig ez a megközelítés volt a legfejlettebb, de a Gépi Tanuló (ML) algoritmusokat használó statisztikus NLP megjelenésével a régebbi kulcsszó és szabály alapú megoldások elavulttá váltak.

A statisztikus NLP lehetővé tette a kézzel írt szabályok elhagyását, helyette az algoritmusok maguk tanulták meg a természetes nyelvet elemezni. A szöveg elemzését kezdetben felügyelt, később részben felügyelt vagy felügyeletlen tanulás során sajátították el a gépi tanuló algoritmusok. Az új megoldásnak köszönhetően a chatbotok tudása könnyebben bővíthető volt és nem kellett a kulcsszó szó szerint szerepeljen a tanító adathalmazban, hogy az chatbot megfelelően válaszoljon a kérésre. Továbbá megnyílt a lehetőség arra, hogy a chatbotok a választ dinamikusan generálják. Azonban a statisztikus

NLP alkalmazásának is voltak hátrányai, például a tanító adathalmaz a modelleknek megfelelő struktúrában kellett legyen, jellemzőtervezésre (feature engineering) volt szükség, ami manuális és időigényes feladat, továbbá egy-egy megoldás mindig több algoritmusból álló rendszer volt, amelyeket külön kellett tanítani.

A 2010-es években a Mély Tanulás (Deep Learning) létrejöttével egy újabb korszak kezdődött az NLP történetében. A Mesterséges Neurális Hálózatokon (ANN) alapuló modellek jobb megoldásnak bizonyultak NLP feladatok során is, mint a statisztikus modellek, köszönhetően a végponttól-végpontig (end-to-end) tartó tanításnak és a reprezentáció tanulásnak (representation learning). További pozitívum, hogy nem egy megadott módon strukturált adattal lehet tanítani az ANN-eket. Az NLP ezen korszaka a mai napig is tart, a legutóbbi áttörés, amely ennek a technológiának a tökéletesítését célozta az a transzformátor modell, amely a korábban egyeduralkodó CNN és RNN modelleket utasította maga mögé teljesítményben.

1.2 Chatbot fejlesztő platformok fogalma

A chatbot építő platformok olyan fejlesztői környezetek, amelyek egy chatbot készítésénél szükséges modulokat és megoldásokat tartalmazzák. Ilyen modulok és szolgáltatások például a beszélgetés irányítása, a természetes nyelv megértési (NLU) modul más termékekkel való előre elkészített integrációk, tesztelés és karbantartási funkciók, illetve analitikai megoldások, amelyek lehetővé teszik a felhasználók chatbotokkal való interakcióinak elemzését. A chatbot készítő eszközöktől abban különböznek, hogy jellemzően biztosítanak grafikus fejlesztői felületet és egy teljes chatbot termék könnyű létrehozását teszik lehetővé, szemben a chatbot építő eszközökkel, amelyeknél a termék létrehozása általában programozással és könyvtárak által biztosított funkciók alkalmazásával elérhető. Ugyan a chatbot építő eszközök sokkal flexibilisebb felhasználást tesznek lehetővé, sokszor mélyebb technikai tudást igényelnek, szemben a platformok csökkentett szabadságával, de kényelmesebb fejlesztési élményével.

1.3 Nagy Nyelvi Modell (LLM)

A LLM-ek nagy, akár több száz milliárd paraméter számú előre tanított nyelvi modellek (PLM). A hagyományos nyelvi modellektől (LM) tanításuk és paraméter számukban térnek el. A nyelvi modellek felügyelt tanulással tanulnak, míg az előre tanított nyelvi modellek nagy adathalmazokon, felügyeletlen tanulással tanulnak, hogy

generikus reprezentációkat tanuljanak az adatból, amit több NLP feladatban is hasznosítani tudnak. LLM-ek közé tartoznak a ChatGPT mögött működő GPT [2] modellek, a Google által fejlesztett BERT [3] és PaLM 2 [4] a Meta által fejlesztett LLaMA [5], a nyílt forráskódú Falcon [6] és még sok más modell. Generatív chatbot rendszerekben válasz generálására használják.

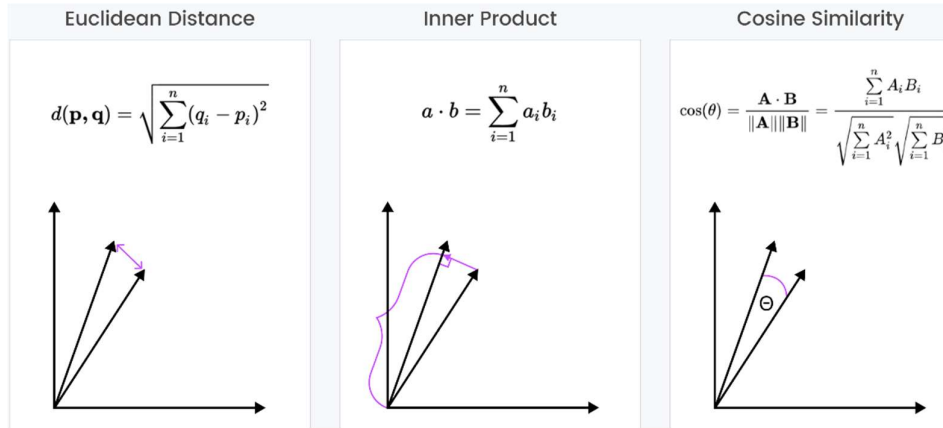
1.4 Beágyazások és beágyazó modell

A beágyazások a valós világ objektumait alakítják át komplex matematikai reprezentációkká, vektorokká [7]. A beágyazás vektorok fontos tulajdonságokat és kapcsolatokat zárnak magukba a valós világ objektumairól és a köztük lévő kapcsolatokról, ezáltal lehetővé téve a gépi tanuló rendszereknek a valós világból származó adattal való munkát. A beágyazás műveletének két formája a statisztikus és a neurális háló alapú. Azokat a neurális hálókat, amik beágyazások létrehozására vannak tanítva beágyazó modelleknek nevezzük [1]. Ilyen modellek például az OpenAI által létrehozott text-embedding modell sorozat, a Microsoft E5 modell és még sok más.

1.5 Szemantikus keresés

A szemantikus keresés egy kereső motor technológia, amely értelmezi a szavak és mondatok jelentését. Az így keletkezett keresési eredmények a keresés jelentésével fognak egyezni, nem pedig a keresésben szereplő szavakkal. A keresés a kereső szándékát és a keresés kontextusát veszi figyelembe, ahol a keresés kontextusa lehet bármilyen, a keresés körülményeihez tartozó adat, például keresési előzmény. A szemantikus keresés a vektor keresésre alapul, amely a vektorok hasonlóságának mérésére, pontosabban térben való egymástól mért távolságuk mérésére támaszkodik keresés során. Ezek a vektorok a beágyazások, amelyek hasonló jelentés esetén a térben közelebb lesznek egymáshoz. [8] A vektor keresést akkor válik szemantikus kereséssé, amikor szemantikus jelentést magukba záró vektor beágyazások között történik a keresés. A szemantikus keresést és a vektor beágyazások hatékony eltárolását a vektor adatbázisok teszik lehetővé, amik erre a feladatra specializált adatbázisok. [9]

A vektorok távolságának vizsgálatára három fő metrikát szoktak alkalmazni, a koszinusz hasonlóságot, a skaláris szorzatot, illetve az euklideszi távolságot.



1. ábra: Szemantikus kereséshez használt vektor távolsági metrikák [10]

1.6 Prompt és prompt tervezés

A prompt egy olyan összetett, strukturált, szöveges bemenet egy LLM számára, amely a válasz generálásához szükséges összes információt tartalmazza. Egy prompt több részből áll, állandó helyet kapnak benne a modellnek szánt instrukciók és a felhasználó üzenete. Ezen kívül tartalmazhat korábbi kérdés-válasz párokat a beszélgetésből és a válasz igazságtartalmát meghatározó kontextus dokumentumot. A prompt szerkezete és összetétele függ a használandó nyelvi modelltől, a prompt nyelvétől, illetve a modelltől elvárt elvégzendő feladattól. A prompt tervezéssel a prompt mérnökök foglalkoznak, akik célja olyan prompt előállítása, amely ráveszi a nagy nyelvi modelleket az adott feladat minél sikeresebb elvégzésére.

1.7 Retrieval Augmented Generation

Az előre feltanított nyelvi modellek, habár rengeteg információt képesek magukba zárni tanulásuk során, új információt csak lassú és költséges újratanulás, vagy gyorsabb és kevésbé költséges, ugyanakkor kevésbé mélyreható finomhangolás során tudnak magukévá tenni, továbbá nem képesek a kimenetük előállításának folyamatába betekintést nyújtani és fogékonyak lehetnek a hallucinációnak nevezett jelenségre. [11] Ennek oka, hogy a modellek a tanításuk során megtanult információt parametrikus memóriában tárolják, amely az információ modell paraméterekben való tárolása. Azonban a paraméterek mennyisége és pontossága korlátozott, az összes tanult információra érintetlenül tökéletesen rátanulnia a modellnek szinte lehetetlen és nem is ajánlott, ugyanis a túltanulás jelenségét idézi elő, amikor a modell a tanító adathoz igazodik és elveszti a képességét, hogy a tanító adaton kívüli adatra megfelelő működést

produkáljon. A legegyszerűbb analógia a jelenségre az emberi agy esetében a saját memóriánkra való hagyatkozás, az emberek sem képesek megjegyezni minden apró részletet pontosan úgy ahogy az van, esetenként nem is a valóságnak megfelelő információt jegyezzük meg, ebből kifolyólag helytelen információt közvetíthetünk. Mint az emberi fejlődés történelmében, úgy a generatív mesterséges intelligencia területén is az előbb ismertetett problémák megoldására a nem parametrikus memória vegyítése a parametrikussal tűnik jó megoldásnak. Nem parametrikus memória például egy könyv, egy kép vagy bármely adatot feldolgozatlanul megőrző információ halmaz.

A Retrieval Augmented Generation (RAG) egy olyan eljárás, amely ötvözi a parametrikus és nem parametrikus memóriát egy olyan rendszerben, ahol a parametrikus memóriát egy szövegből szöveg generáló modell, a parametrikus memóriát pedig egy beágyazásokat tartalmazó index halmaz tartalmazza. [11] Célja a hallucináció csökkentése, a rendszer válaszainak pontossága és relevanciájának növelése, illetve az új információhoz való adaptáció gyorsítása, amely gyorsabb és kevésbé költséges adaptációt eredményez olyan adathoz, amely a modell eredeti tanító adatában nem volt jelen. [12].

Az architektúra fejlődésének három szakaszát különböztetik meg jelenleg, amelyből szakdolgozatomban az első kettőt, a Naiv és a fejlett RAG szakaszokat fogom bemutatni. A Naiv RAG a legelső szakasz, ezért megismerése szükséges, a szakdolgozatomban létrehozott chatbotok architektúrája pedig a fejlett RAG szakaszhoz tartoznak.

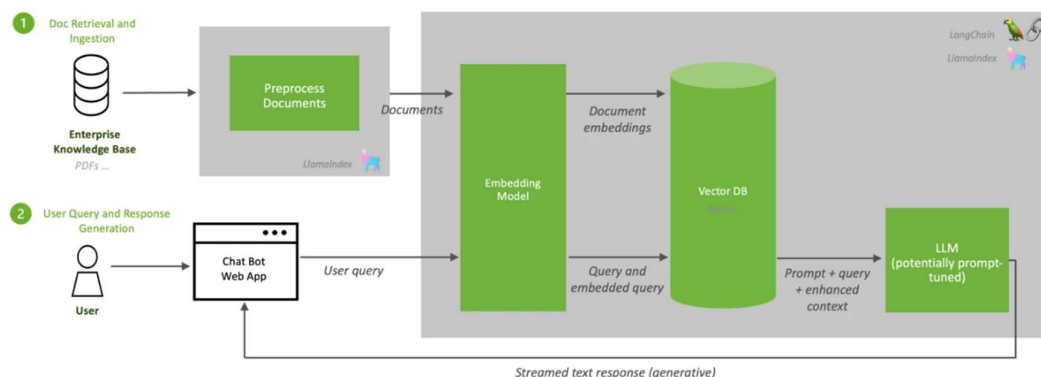
1.7.1 Naiv RAG

A Naiv RAG a tradicionális folyamatot követi, amely az indexelés, visszanyerés és generálás folyamatokat tartalmazza. Az indexelés folyamata adattisztítás, szeletelés, beágyazás és tárolás folyamatokból áll. [13] Adat tisztítás során minden fölösleges zajtól, szöveg alapú információ esetén weboldalaknál például megmaradt HTML címkéktől, megtisztítjuk az adatot. Szeletelés során az adat kisebb szeletekre bomlik mivel a szöveg generálást végző modellnek limitált a bemeneti token száma és később a szeleteket a generáló modell feldolgozza. Beágyazás során a létrejött szeletek egy beágyazó modellen keresztül vezetve vektor beágyazásokká transzformálódnak. A folyamat célja, hogy később a keresés során vektor hasonlóság alapú keresést lehessen alkalmazni. A beágyazásokat a rendszer ezután egy erre specializált adattárolóban menti el, amely a vektor adatbázis. A folyamat pontosságára nagy hangsúlyt kell fektetni, mert befolyásolja

a későbbi hasonlósági keresés pontosságát, ezáltal a rendszer által adott válaszok minőségét.

A visszanyerés folyamata során a rendszer ugyanazt a beágyazást generáló modellt alkalmazva a beérkező felhasználói szöveget vektor beágyazássá transzformálja. A keletkező vektor beágyazást a már tárolt beágyazásokkal vektor hasonlóság alapján összehasonlítja és a top-k mennyiségű, a felhasználói szövegének beágyazásához legközelebb álló szeleteket választja ki. A kiválasztott szeletek a válasz generáláskor a válasz generáló modellnek átadott prompt-ban foglalnak helyet, ahol elegendő információval látják el a modellt a helyes válasz generálásához.

Generálás során a felhasználó által megadott szöveg, a visszanyerés során megtalált kontextus és adott esetben a korábbi beszélgetés üzenetei egyesülnek egy utasítással, amely összességében a válasz generálásra szolgáló prompt. A prompt a modell bemenete, amely tartalmaz minden utasítást és adatot ahhoz, hogy a modell a kívánt módon helyes választ generáljon a felhasználó által megadott bemenetre. A folyamatot és az architektúrát a 2. ábra mutatja be.



2. ábra: Naiv RAG alkalmazás komponensei és folyamatai [14]

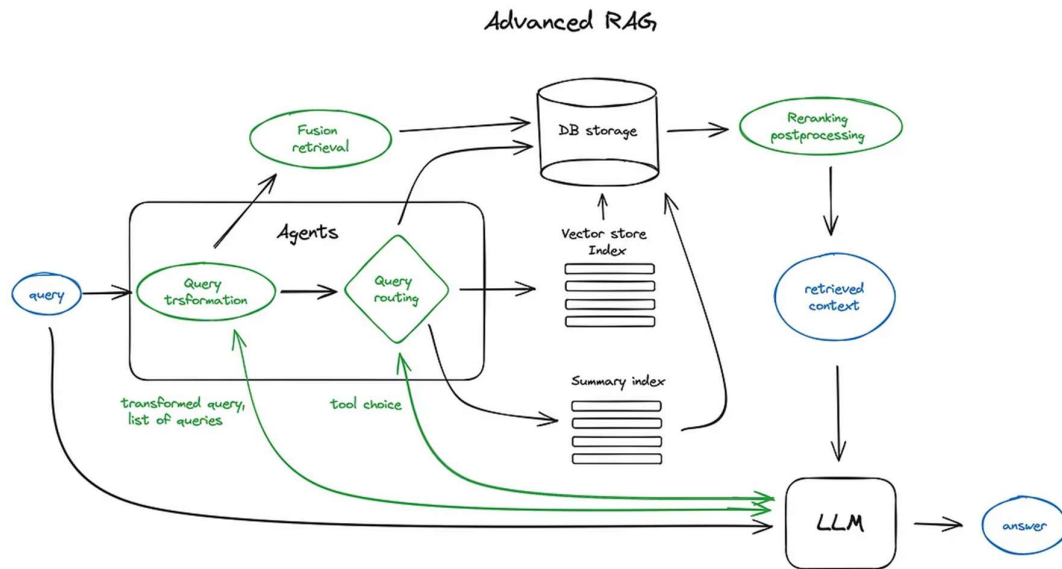
Habár a Naiv RAG is biztató eredményeket produkált, használata során több probléma is felmerült. Az információ visszanyerés folyamata sokszor irreleváns szeleteket ad vissza például mert a felhasználó eredeti kérése nem megfelelő a távolsági kereséshez, generálás során a hallucináció még mindig gyakran megtörtént, illetve a generált válasz a szeletek alapján néha értelmetlen volt, vagy akár túlságosan is vezette a nyelvi modellt, ami csak a szeletekben található információk megismétléséhez vezetett. [13] A megismétlés akkor baj például, ha a nyelvi modelltől jogosan elvárjuk hogy az

adatot átdolgozza a feladatnak megfelelően. A problémák orvosolására a RAG kutatás egy új szakaszba lépett, amelyet Fejlett RAG-nek nevezünk.

1.7.2 Fejlett RAG

Hogy a Naiv RAG problémáira megoldást nyújtson, a Fejlett RAG az információ visszanyerési folyamatot elő- és utó- információ visszanyerési lépéssel bővíti. Az elő információ visszanyerési lépés célja az indexelés és a felhasználó által nyújtott bemenet optimalizálása. [13] Az indexelés folyamatát például precízebb szeletelési technikákkal, index struktúra optimalizálásokkal és a dokumentum metaadatainak felhasználásával lehet javítani. A felhasználói szöveget pedig kérdés újra fogalmazással, kérdés bővítéssel és egyéb technikákkal lehet javítani annak érdekében, hogy a keresési folyamat később a keresésnek jobban megfelelő adattal tudjon szolgálni a választ generáló modell számára, ezzel javítva az információ visszanyerési folyamat pontosságát. Szakdolgozatomban a kérdés újraírás technikát mindkét chatbot alkalmazni fogja.

Az utó információ visszanyerési lépés a talált szeletek kérdéshez való megfelelő integrációjára összpontosít. Az egyik ilyen megoldás típus a szeletek tömörítése például modellel való összefoglalással, vagy új sorrendbe helyezés attól függően, mennyire relevánsak az adott kérdéshez. Utóbbi folyamatot reranking-nek nevezik, amely egy modell segítségével képes relevanciát mérni a felhasználó kérése és a rendelkezésre álló szeletek között. A szakdolgozatban létrehozott chatbotok reranking és összegzés technikákat is alkalmaznak az adat visszanyerési folyamat javítására. Az összegzésre a modell számára átadott adat koncentrációja miatt van szükség, hogy félre ne vezesse a modellt valamilyen irreleváns, a szeletekben fellelhető mellék információ, amely a nem tökéletes szeletelés eredménye. A reranking pedig azon megfigyelésen alapul, hogy a vektor hasonlóság alapú keresés jellemzően nem elég precíz ahhoz, hogy alacsony top-k érték mellett a ténylegesen leghasznosabb szeleteket találja meg, azonban mint egy szűrő, kiválóan alkalmas a teljesen irreleváns szeletek kiszűrésére. A szűrés után fennmaradt szeleteket így a reranker hamar át tudja válogatni, hogy a leghasznosabbak szeletek legyenek az elsők, ekkor pedig az alacsony top-k érték már megengedhető lesz.



3. ábra: Fejlett RAG rendszer architektúra diagramja [15]

1.7.3 Előnyök és hátrányok

A RAG eljárásnak, habár sok kedvező tulajdonsággal rendelkezik, vannak hiányosságai is. A szemantikus hasonlóság nem minden esetben találja meg a megfelelő dokumentum darabokat az adott kérdéshez, ugyanis az beágyazást generáló modell is hibázik bizonyos százalékban. A dokumentum darabok ideálisan egy-egy jól elhatárolható témát kell, hogy felöleljenek. Azonban a legtöbb daraboló egy fix karakter hossz alapján szabdalja fel a dokumentumokat, amely problémát okoz az beágyazást generáló modellnek, ugyanis előfordulhat, hogy két, teljesen eltérő témában is a beágyazást generáló modell által fontosnak vélt információt hordoz az adott dokumentum darab. Ilyenkor a darabhoz tartozó vektor valódi információ tartalmát nehéz pontosan meghatározni, gyenge vagy helytelen lesz a leképzés, pontatlanságot visz a keresésbe. További probléma, hogy a távolságmérés sok dokumentum darab esetén egyre lassabbá válik, ha például K-közelebbi szomszéd (KNN) algoritmusokat használunk. Amennyiben áttérünk közelítő legközelebbi szomszéd (ANN) algoritmusokra, a pontosságból kell áldoznunk. Az egész rendszer pontosságát jelentősen befolyásolja továbbá, hogy hány dokumentumot használunk fel a válasz generálás során, amely paramétert top-k értéknek neveznek. A darabok méretét és a top-k értéket párban kell finom hangolni, azonban ez is egy költséges, időigényes, szakértelmet igénylő folyamat. Összeadva az LLM a beágyazást generáló modell és a keresés pontatlanságát könnyen előfordulhat, hogy rosszabb RAG rendszert építünk mintha egy teljes szöveges keresőt használnánk.

A RAG architektúrának továbbá anyagi és sebesség problémái is vannak. Nem tudnak versenyre kelni egy szándék felismerés alapú rendszerrel gyorsaságban és üzemeltetési költségekben. Ugyanakkor jóval nagyobb pontosságot és sokkal szélesebb körben alkalmazható szolgáltatásokat építhetünk velük, továbbá a fejlesztések idejét és a manuális munkát is csökkenti, amely egy más típusú architektúránál előfordul. Egy másik rivális megoldás a modell finomhangolás, amely egy kevésbé intenzív modell tanítási folyamat. Azonban a finomhangolás gyakran változó tanító adat esetén lassítja az adaptációt és költségessé teszi a működést, továbbá a csak modellt alkalmazó megoldások hátránya is megmarad, hogy a modell nem enged betekintést a belső működésébe, például, hogy milyen forrásból válaszolta meg az adott kérdést.

1.7.4 Alkalmazási területek

Az architektúra jellemző alkalmazási területe a chatbotok, különösen az oktatásban használt és a felhasználó támogató chatbotok, ugyanis ezekben az esetekben gyakran változó adat alapján kell tényszerű és pontos válaszokat adni. Hagyományos, csak generatív komponenst tartalmazó chatbot esetében ez a modell újra finomhangolásával lenne elérhető, ami az LLM-ek méretének növekedésével egyre bonyolultabb, költségesebb és időigényesebb folyamat. A RAG architektúra ígérete, hogy a finomhangolás helyett, amelyhez kell szakértelem, erőforrás és idő, egy MI-hez nem értő személy is gyorsan, komoly terhelést nem róva a rendszerre tud új információt juttatni a rendszerbe. Az architektúra másik előnye, hogy igazoltan csökkenti a modell hallucinálásnak nevezett jelenséget, amikor a modell új, nem létező és gyakran nem helyes információt generál. A jelenség a felsorolt felhasználási esetekben hatalmas problémát okozna.

2 Tervezés

A tervezés folyamatát két szakaszban valósítottam meg. Az első szakaszban a chatbotokat terveztem meg és platformokat azonosítottam a fejlesztésükhöz, majd a második szakaszban a chatbotok és a platformok kiértékelésének menetét terveztem meg.

A chatbotok tervezését a felhasználási eset elemzésével kezdtem, hogy egy már létező mérnöki kihívássá fordítsam le azt. A kihívás felismerése után létező architektúrákat azonosítottam, amelyek megoldást nyújthatnak a felismert problémára. Végül az architektúrát és annak alkotóelemeit támogató platformokat azonosítottam, amelyeket a chatbotok megvalósításához használtam. A tervezés ezen formája kikényszeríti a legújabb technológiák figyelembevételét és a mérnöki probléma megoldást, illetve alkalmaz már bizonyítottan működő szoftver fejlesztési folyamatokat, ezzel biztosítva a szakdolgozatban bemutatott eljárások frissességét és precíz kivitelezését.

2.1 Felhasználási eset elemzése

A felhasználási esetet úgy képzeltem el, mint egy valós igény, amely egy egyetem részéről érkezett. Az igénylő személye azért fontos, mert a platformok prioritását megváltoztatja. Az oktatási intézmények legtöbbje valamely nagy felhő alapú szolgáltatónak az infrastruktúráját alkalmazza, mivel ezek kényelmes és modern infrastruktúra kialakítást tesznek lehetővé. Ennek megfelelően platformok tekintetében prioritást élveznek a nagy felhő alapú szolgáltatást nyújtó cégek, mint a Microsoft, Amazon és a Google. További érv egy felhő alapú szolgáltatásokat nyújtó cég megoldása mellett pedig a skálázhatóság és a hardware nyújtása. A mesterséges intelligenciához szükséges nagy teljesítményű videokártyák nem gyakran állnak rendelkezésre egy-egy intézménynél, beszerzésük pedig pusztán egy chatbot kedvéért sok esetben nem előnyös. Karbantartásuk és üzemeltetésük pedig szakértelmet igényel, amely szintén nem minden egyetemen áll rendelkezésre.

Az igénylő személyén kívül fontos maga a megoldandó feladat is. Az igény egy chatbotról szól, amely az intézmény által biztosított fájlokból származó adat alapján képes válaszokkal szolgálni a felhasználók kérdéseire és egy weboldalon keresztül elérhető a felhasználók számára. A chatbotnak egyéni adat alapján kell egy meghatározott témában

segítséget nyújtania, ahol fontos, hogy ne adjon helytelen információt. További fontos követelmény, hogy a megoldás a téma kiírásnak megfelelően LLM-et kell alkalmazzon, illetve, hogy tudjon magyarul. Az igényt a felsoroltak alapján egy LLM alapú kérdés megválaszolási feladatnak azonosítottam, amelyre két lehetséges megoldást találtam.

2.2 Chatbotok architektúrája

Az azonosított feladathoz a RAG és az LLM finomhangolást találtam, mint lehetséges megoldások, amelyek kiaknázzák az LLM-ek által nyújtott lehetőségeket. Mindkét megoldásnak vannak előnyei és hátrányai, azonban a RAG esetünkben egy jobb megoldás. A RAG gyors és kevésbé erőforrás igényes tanító adathoz való adaptálódást tesz lehetővé, amely egy GYIK adathalmaznál vagy egy egyetemi működést leíró dokumentum esetében relatíve gyakori esemény. Egyetemi működést leíró és egyéb dokumentumok esetén nem áll rendelkezésre felcímkézett tanító adat a finomhangoláshoz, amely egy felügyelt tanítási folyamat. Továbbá a felhasználási eset nem igényel olyan szakzsargon vagy téma specifikus kifejezések tudását, amelyet egy előre tanított modell ne tudna. A felsorolt indokok alapján a RAG architektúrát választottam, amelyet részleteiben az 1.3 fejezetben fejtettem ki. Az architektúra azonosítása után pedig olyan platformokat kerestem, amelyek támogatják egy ilyen chatbot létrehozatalát.

2.3 Chatbot építő platformok bemutatása

A lehetséges platformok keresését a Budapesti Műszaki és Gazdaságtudományi egyetem által is használt Microsoft Azure környezetben kezdtem meg. A Microsoft Azure egy felhő alapú szolgáltatásokat nyújtó platform. Két RAG chatbot építésére alkalmas szolgáltatást találtam a platformon. A Copilot Studio egy kódmentes környezet, ahol pusztán a grafikus felület segítségével tudunk akár generatív asszisztenseket is létrehozni. Sajnos a platform azonban nem támogatta a magyar nyelvet, előzetes próbálkozásaim alapján pedig nem tudott válaszolni se magyar nyelven fogalmazott kérdésekre, se angolra lefordított, a tudásbázisban szereplő kérdésekre, ezért a platform használatát elvettem. Egy másik Azure alapú megoldás volt az Azure AI Studio, amely egy kevesebb kódolást igénylő platform, viszont tökéletesen alkalmas egy RAG chatbot létrehozására magyar nyelven is. Az eszköz azonban az Azure OpenAI szolgáltatást használja, amely csak várólistás jelentkezés és engedélyezés alapján érhető el vállalati

ügyfelek számára, ezért nem tudtam hozzáférést nyerni az eszközhöz a szakdolgozat végzésének ideje alatt.

Az Azure platformon való sikertelen kísérlet után egy másik jól ismert felhő alapú szolgáltatásokat tömörítő platform, a Google Cloud Platform ökoszisztémájában kerestem megoldást, ahol hamar rátaláltam a Dialogflow CX szolgáltatásra, amely RAG chatbotok létrehozására is alkalmas. A platform nagy múltú, eredetileg szándékfelismerés vezérelt felhasználó segítő alkalmazások készítésére lett létrehozva. Fontos megjegyezni, hogy a magyar nyelvet hivatalosan a RAG architektúrát megvalósító moduljai még csak teszt üzemben támogatják, mivel a termék állandó fejlesztés alatt áll és a szakdolgozat készítésének ideje alatt is frissítéseken esett át. Ennek megfelelően hangoltam az elvárásaimat a termékkel szemben, megadva az esélyt számára.

Második platformnak szerettem volna egy olyan eszközt választani, amely nem felhő alapú, menedzselt szolgáltatásokat nyújt. Ilyen platformok közül a Llama Index és a LangChain keretrendszereket találtam, melyek közül a Llama Index tűnt leírása és segítségével megvalósított referencia termékek alapján a jobb választásnak. Ennek oka, hogy a Llama Index hangsúlyosabban szól RAG alkalmazások építéséről, amely a Langchain keretrendszerről nem mondható el. A Langchain inkább általánosabb és széleskörűbb LLM alapú alkalmazások fejlesztésére lett létrehozva. [16] Gyakran azonban egymást kiegészítve is alkalmazzák a két keretrendszert MI-ra épülő alkalmazások fejlesztésére. A Llama Index egy Python és TypeScript programozási nyelven elérhető keretrendszer, amely alapvető alkalmazás komponenseket tartalmaz. Segítségével gyorsan és könnyen lehet egyedi felhasználói esethez szabott alkalmazásokat készíteni.

Fontos megemlíteni, hogy a két eszköz nem ugyanazt a termék kategóriát képviseli. A Dialogflow CX egy Google Cloud infrastruktúrában megtalálható Software as a Service (SaaS) termék, szemben a Llama Index eszközzel, amelyik egy alkalmazás fejlesztést elősegítő keretrendszer, program könyvtár.

2.4 Chatbotok teljesítmény kiértékelési módszere

A chatbotok kiértékelésénél természetes nyelv megértési (NLU) képességeit, azaz a válaszok minőségét értékelem ki, amely a legjobban befolyásolja a chatbot felhasználók által való megítélését. A chatbotok NLU képességeinek kiértékeléséhez egy olyan metodikát kellett találgatni, amely kérdés megválaszolás feladatban alkalmazott RAG

architektúrájú alkalmazások kiértékelésére alkalmas. A szakdolgozat készítésének idejében nem állt rendelkezésre egységesen elfogadott, kimerítő kutatással és alkalmazási példákkal alátámasztott metrika csomag RAG rendszerek kiértékelésére, azonban sok kutatás folyik a mai napig is a témában. A jelenlegi legfrissebb megoldások az LLM alapú kiértékelést alkalmazzák, különböző pontozási metrikák alapján. Habár a megoldás mellett sok érv szól, a manuális kiértékelést választottam inkább. Ennek oka, hogy a Dialogflow CX platformhoz nem elérhető egy ilyen automatikus kiértékelési rendszer és a termék maga sem tartalmaz erre még megoldást. Ennek megfelelően a chatbotok kiértékelését manuálisan végeztem. A manuális kiértékeléshez először négy különböző kérdés típust azonosítottam amire a chatbottól más és más reakciót vártam, a kérdések megválaszolásához a rendelkezésre álló adatok közül megjelöltem azokat a fejezeteket, amik alapján a kérdést megválaszoltam, majd az információ alapján létrehoztam egy választ, amely a lényegi információt tartalmazta.

A chatbotok teljesítményének kiértékelése fontos része a chatbot platformok kiértékelésének, de önmagában nem elégséges. Hogy egy átfogóbb képet adhassak és több szempontot is figyelembe vegyek, a platformok költségeit, elérhető integrációit, személyre szabhatóságukat és az eszközök segítségével készített chatbotok fejlesztéshez szükséges idejét is összehasonlítottam.

3 Chatbotok implementálása

A chatbotok implementálását a közös tudásbázisként szolgáló adat beszerzésével kezdtem, majd a Dialogflow CX asszisztens implementálásával folytattam, végül a Llama Index chatbot implementálásával fejeztem be. A következő néhány fejezet ezt a folyamatot mutatja be.

3.1 Chatbotok tudásbázisa

A chatbotok tudásbázisához használt adatként a Budapesti Műszaki és Gazdaságtudományi egyetem Központi Tanulmányi Hivatal gyakori kérdések oldalait és szintén az oldalon található archivált szabályzatokból néhány kiválasztott dokumentumot alkalmaztam, melyek publikusan elérhetőek. A fejlesztendő chatbotok felhasználási esetéhez ezek az adatok állnak a legközelebb, ezért esett a választásom ezekre a forrásokra. A dokumentumokat PDF és HTML fájlformátumban szereztem be manuális, böngésző alapú letöltéssel. Adattisztítási vagy adattranszformációs műveleteket a dokumentumokon nem végeztem, mindkét chatbot esetében ezt a feladatot az eszközre bízam. A gyakori kérdés oldalak szabványos HTML dokumentumok, amelyek az oldal némely navigációs és egyéb elemein kívül kérdés-válasz párokat tartalmaznak. A dokumentumok tematikája diákigazolvány, KTH ügyintézés és egyéb hallgatók számára fontos témakörökben tartalmaznak információkat. Ezzel szemben a szabályzatok hosszú jogi szövegek, fejezetekbe és paragrafusokra bontva, melyek az egyetem működését, a képzések szabályozását és egyéb működési szabályok leírását tartalmazzák. Az adatok megismerése és beszerzése után a tervezési fázisban bemutatott chatbotot először a Dialogflow CX eszközzel valósítottam meg.

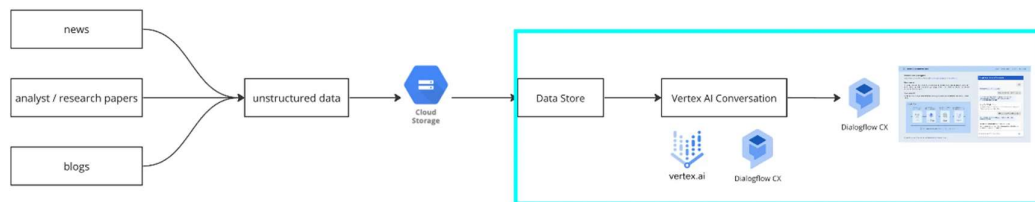
3.2 Dialogflow CX chatbot fejlesztése

A fejlesztés és a használt koncepciók könnyebb megértése érdekében először bemutatom magát a RAG implementációjához szükséges Google szolgáltatásokat, majd rátérek az implementáció fontosabb mozzanataira és a közben született fontosabb döntésekre, végül pedig a végtermék bemutatására.

A Google Dialogflow CX platform egy, a Google Cloud infrastruktúrában megtalálható Software as a Service (SaaS) termék. Segítségével szándék felismerés vezérelt, generatív mesterséges intelligenciát alkalmazó chatbotokat hozhatunk létre

kódolási ismeretek nélkül. Különlegessége a véges állapotú állapotgéppel való beszélgetés irányítása, amely lehetővé teszi a pontos irányítást és komplex, állapotokkal rendelkező beszélgetés menetek létrehozását. Ezen tulajdonságánál fogva kiválóan alkalmas felhasználók előre definiált folyamatokon való végig vezetésére és felhasználók támogatására. Ugyanakkor ezen képességét a szakdolgozatban implementált chatbot nem fogja kihasználni. A generatív mesterséges intelligencia megjelenését a platform is lekövette a Google Vertex AI Conversation szolgáltatás segítségével, lehetővé téve a RAG architektúra megvalósítását és akár ötvöztetését a régebbi szándék felismerés alapú megoldásokkal.

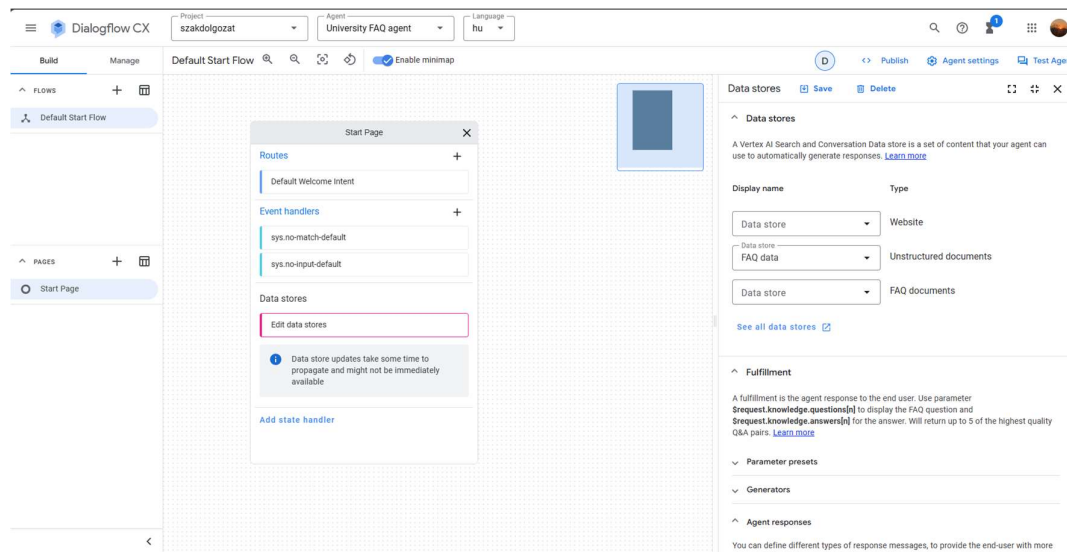
RAG alapú megoldásokat a Data Store Agent asszisztensekkel tudunk létrehozni az eszközzel, amely egy speciális típusú Dialogflow CX asszisztens. Különlegessége, hogy adattárolókat, úgynevezett Data Store-okat tud használni. Az adattárolók olyan eszközök, amelyek képesek szemantikus keresés alapján releváns információt találni a kérdésre, majd a legjobb 5 dokumentum közül kiválasztani azokat, amelyekből végül az eszköz válasza létrejön. A megoldás továbbá használni fogja a Google Cloud Storage Bucket szolgáltatást is, amelyben a dokumentumokat tárolja.



4. ábra: Data Store Agent architektúrája [17]

Az implementáláshoz létrehoztam egy Bucket-et a Google Cloud Storage szolgáltatásban, ahova feltöltöttem a tudásbázist alkotó dokumentumokat. Azért volt szükség pont a Google Cloud Storage használatára, mivel a szakdolgozat írásának idejében még csak ez a típusú adat tároló volt támogatva, ami alkalmas fájlok tárolására és DataStore létrehozására lehetett használni. Lehetőség lett volna kérdés-válasz párokat is megadni egy CSV vagy JSON fájl formájában is, ami közelebb állt volna a GYIK felhasználási esetekben használt adatokhoz, azonban a strukturált kérdés-válasz párokat a rendszer nem RAG válaszadási folyamatom keresztül alkalmazta volna, hanem átfogalmazás nélkül visszaadta volna a chatbot a felhasználó kérdéséhez legjobban illő kérdéshez tartozó választ.

A feltöltés után az Agent Builder segítségével létrehoztam először egy Data Store-t, amit összekötöttem a létrehozott Bucket-el. Egy fontos beállítást kiemelnék, a dokumentum feldolgozó beállítását, ugyanis itt a Google által ajánlott Layout Parser-t használtam. A Layout Parser a dokumentum szerkezetét is figyelembe veszi a dokumentumok feldarabolásánál, így garantálva, hogy a dokumentum eredeti szerkezete amennyire lehet megmaradjon. Ezután az Agent Builder felületén az alkalmazások menüpontban létrehoztam egy új Chat típusú alkalmazást. Fontos az alkalmazás típusa, ugyanis a Chat alkalmazás van csak arra tervezve, hogy megvalósítsa a RAG működést. Az alkalmazás létrehozásakor megadtam, hogy a létrehozott Data Store-t használja az alkalmazás és az alkalmazás kész is volt.



5. ábra: Data Store Agent a Dialogflow CX felületén

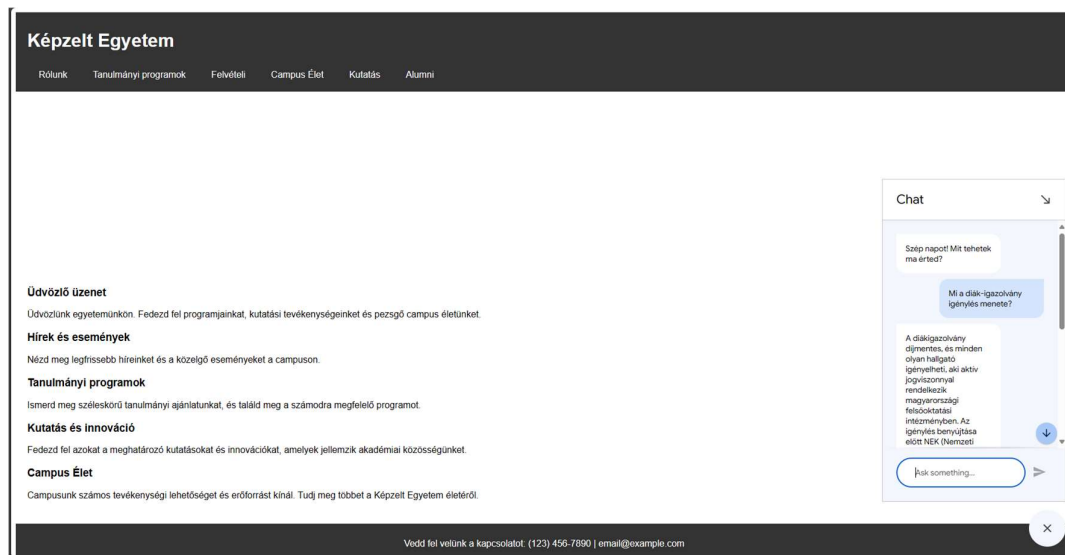
Az alkalmazás létrejötte után a Dialogflow CX fejlesztői környezetben további beállításokra és finomhangolásokra volt szükség. Először átállítottam az asszisztens nyelvét magyarra, majd a kezdő oldalon beállítottam az asszisztens választát. Ehhez meg kellett adnom, hogy a chatbot válasza mindig a generált válasz legyen.

További változtatást a Data Store prompt megadása jelentette, amely része a válasz generáláshoz használt promptnak. A prompt megírása valójában egy sémába való adat beillesztés volt, amit a platform alapvetően használ. Az asszisztens nevét, az identitását, a vállalat nevét, a vállalat rövid leírását és az asszisztens feladatát kellett megadnom, mint értékek a grafikus felületen. A teljes prompt a következő lett:

Your name is Mikrobi, and you are a helpful and polite chatbot at Budapest University of Technology and Economics, a hungarian university. Your task

is to assist humans by answering their questions based on relevant information.

Következő lépésként a példa weboldal integrációt valósítottam meg, amelyet a Dialogflow Messenger segítségével értem el. A Dialogflow Messenger egy weboldalakra beágyazható JavaScript felhasználásával készített felhasználói felület, amit a Dialogflow CX mint kommunikációs integráció tartalmaz. Az integrációk között kiválasztva kapunk egy HTML objektumot, amelyet bemásolva egy weboldal törzsébe elérhetővé válik az asszisztensünkkel való kommunikáció egy chat felületen. A szemléltetés kedvéért és webfejlesztői jártasságom hiánya miatt ChatGPT segítségével készítettem egy példa egyetemi weboldalt, amelybe belehelyeztem a Dialogflow Messenger chat felületet, amelynek eredményét az 6. ábra mutatja:



6. ábra: Példa weboldal Dialogflow Messenger chat integrációval

Ahogy a képen is látható, az eszköz ezek után működött, a chatbot válaszokat adott. Ennek megfelelően figyelmemet a második chatbot platformon megvalósítandó chatbotra helyeztem át.

3.3 Llama Index chatbot fejlesztése

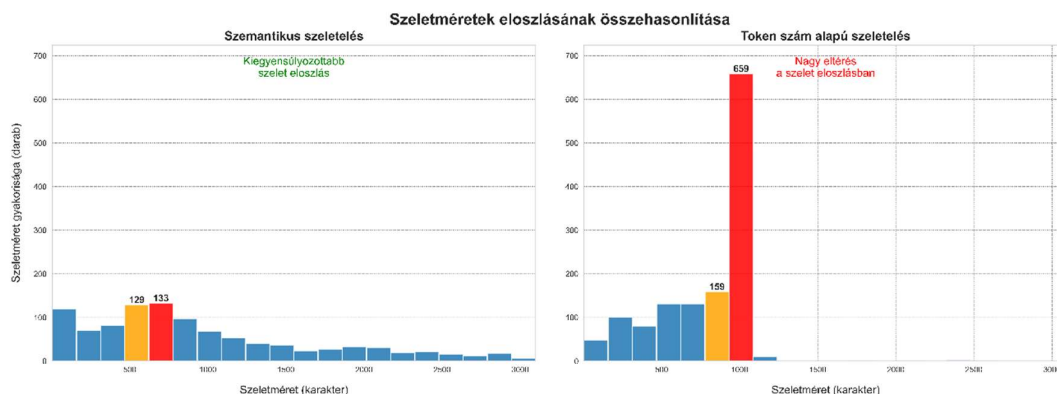
A chatbot implementációját az adatok feldolgozásával és vektor adatbázisba való betöltésével kezdtem, ugyanis a Dialogflow CX termékkel szemben a keretrendszer nem tartalmaz teljes egészében előre megvalósított funkciókat, viszont az építő kockákat az adatfeldolgozáshoz igen. A két típusú adatforrásnak két külön feldolgozási eljárást definiáltam, amelyek az egyes dokumentum típusok egyediségére való tekintettel

készültek el. Az eszköz erre egy közös interfészt definiál, amely segítségével adatfeldolgozási eljárásokat definiálhatunk, amelyek egyszeri definiálás után újrahasznosíthatók. A feldolgozási eljárásokban megadhatjuk milyen adattranszformációs műveletek szekvenciális végrehajtásából áll a feldolgozás. Ezek az adatfeldolgozási eljárások például szeletelési eljárások, illetve a beágyazások generálása. Az egyes fájlformátumoknak külön meg tudtam adni azt az előre elkészített beolvasó eljárást, amely beolvasta a fájlokat dokumentum objektumokká. PDF fájlok esetén a PyMuPDF könyvtárt alkalmazó beolvasót alkalmaztam, HTML dokumentumok esetén pedig nem alkalmaztam semmilyen különleges betöltési eljárást, később a szeletelésnél kezeltem csak különlegesen a fájlok tartalmát.

A dokumentumok beolvasása után több dokumentum felszeletelési eljárás közül tudtam választani. A legjobb szeletelési eljárásnak, működési elve alapján, a szemantikus szeletelés tűnt, ugyanis a többi szeletelő megoldással ellentétben figyelembe veszi a dokumentum tematikus struktúráját és ennek megfelelően igazítja az egyes szeletek méretét. Az eljárást Greg Kamradt mutatta be először [18], a Llama Index eszközben pedig SemanticSplitterNodeParser osztály implementálja. Az eljárás mondatokra bontja szét a teljes szöveget regex kifejezések segítségével, majd az éppen kiválasztott mondat előtti és utáni ablak méret által meghatározott mennyiségű mondatot összeilleszti a kiválasztott mondattal, létrehozva egy több mondatból álló szöveget. Az így kapott szövegekhez kiszámolja a vektor beágyazásokat és koszinusz távolságot mér a kialakult beágyazások között, de mindig csak az előző szöveg és a következő szöveg beágyazásai között mér hasonlóságot. Az így kialakult listányi koszinusz távolságok között pedig a kiugró értékek határozzák meg az egyes szelethatárokat. A kiugró értékek lehetnek például az adat 95. percentilisei.

Hogy megbizonyosodjak a szeletelés minőségéről, a szabályozást leíró dokumentumokat felszeleteltem az öt különböző szeletelési eljárással. A szakdolgozatba egy fix szeletmérettel és egy dinamikus szeletmérettel dolgozó eljárás eredményeit választottam ki és hasonlítottam össze. Az összehasonlítás a szeletek karakterben mért hosszának eloszlása és beágyazásainak koszinusz hasonlósága alapján vetette össze az egyes szeletelési eljárások eredményeit. Utóbbi összehasonlítás a beágyazás generáló modell minőségét is bemutatja. A szelet méretek eloszlása képet ad az eljárás természetes dokumentum struktúrájához való alkalmazkodási képességéről. A dokumentum struktúrához való pontosabb alkalmazkodás kihatással van a szeletek

tartalmának kohéziójára, ezáltal a későbbi vektor hasonlósági keresés pontosságára. Ha például az egyes szelet határok úgy lennének meghatározva, hogy az az előző fejezet végét és a következő fejezet elejét tartalmazza és a két fejezet eltérő témákat tárgyal, a szelet beágyazása gyenge információ tartalommal bírna, ugyanis nem hasonlít eléggé se az egyik, se másik fejezet témájához. A következő ábra a fix szeletméretet alkalmazó token és a dinamikus szeletméretet alkalmazó szemantikus eljárások által generált szeletek hosszának eloszlását tartalmazza.



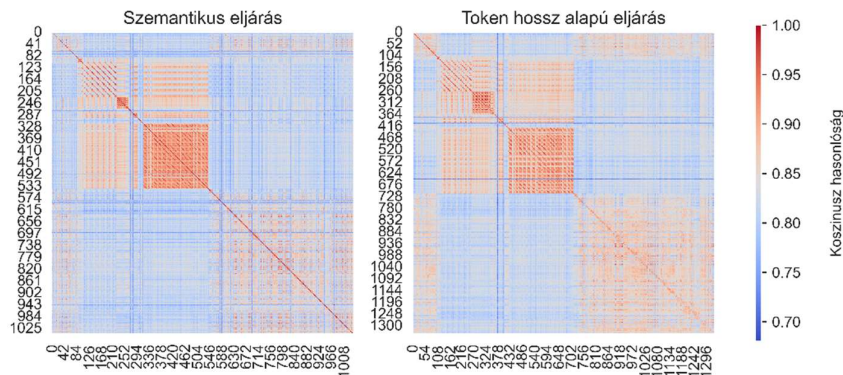
7. ábra: Token alapú és szemantikus szeletelő szelet méret eloszlások összehasonlítása

Az ábrán jól látható, hogy a token alapú szeletelés kiugróan magas arányban 1000 körüli karakterszámú szeleteket készített, 500 darabbal többet, mint a következő leggyakrabban előforduló szelet méret csoportból. A szemantikus szeletelés azonban ennél jóval kiegyensúlyozottabb munkát végzett, a két leggyakoribb szeletcsoport között az előfordulások különbsége csak 4, mindemellett az is megfigyelhető, hogy jóval többféle szelet méretet produkált.

Azonban mindez nem elég bizonyíték arra, hogy a stratégia jól működik. Meg kell bizonyosodnunk arról, hogy a szeletelés közben olyan szeletek keletkeztek, amelyek nem tartalmaznak több mint egy témát. Továbbá egy fontos elemet, a beágyazásokat generáló modellt sem figyeltük meg, mennyire pontos. Beágyazó modellnek először az OpenAI text-embedding-ada-002-v2 modellt próbáltam ki, azonban többször is előfordult, hogy API kapcsolati problémák merültek fel, használati limitekbe botlottam, illetve lassú volt a feldolgozás. Így végül a HuggingFace modell könyvtárból szereztem be az intfloat/multilingual-e5-large modellt [19], amelyik a Microsoft által fejlesztett E5 modellen alapuló többnyelvű beágyazásokat generáló modell. A chatbotban ezt a modellt alkalmaztam végül. A következő ábrán a két stratégia által készített szeletek

beágyazásainak egymáshoz mért koszinusz hasonlóságát figyelhetjük meg egy hő térképen ábrázolva.

Szelet beágyazások koszinusz hasonlóságának hő térképei

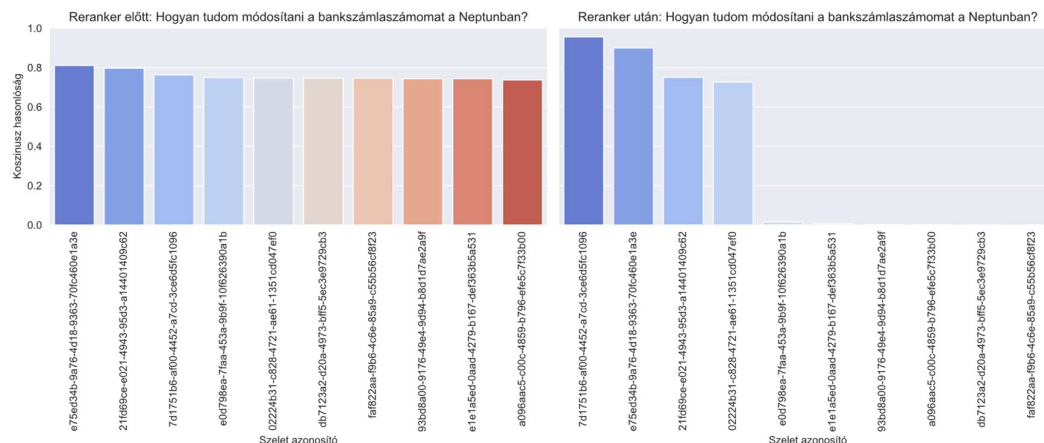


8. ábra: Szelet beágyazások hő képei

Az ábrán a vöröses árnyalat nagyobb hasonlóságot jelent. A vízszintes és függőleges tengelyeken pedig az egyes szeletek sorszámát láthatjuk. A vízszintes tengelyen az értékek 0.7-től 1-ig terjednek, amelyek alából 0 és 1 között kéne mozogjanak, de ez nem hiba, a modell dokumentációja alapján a modell így működik. A négyszög alakú zónák olyan pontok, ahol sok, egymást követő szelet beágyazása hasonlít egymásra. A token alapú eljárás meglepően jó eredményt ért el, azonban általánosságban látható, hogy több hasonló jelentéssel bíró szeletet hozott létre, mint a szemantikus eljárás. Az adatot visszaellenőrizve kiderült, hogy ez a viselkedés nem helyénvaló, ugyanis az egyes szeletek kevert témájúak voltak. A témák keveredése növeli annak az esélyét, hogy az egyes szeletek hasonlítani fognak egymásra. A szemantikus eljárás azonban nem mosott össze különböző témákról szóló szeleteket, ennek köszönhetően kevesebb egymásra enyhén hasonlító szelet keletkezett. Az egyes forró négyzetek és a körülöttük lévő hűvös elválasztó vonalak pedig jól kirajzolják a dokumentumokat és a dokumentumokon belüli fejezet határokat. A szemantikus eljáráshoz tartozó hő térképen ez jobban látszik. Összességében az ábrák alapján tehát a szemantikus eljárás jobb szeletelési eljárásnak ígérkezett, ennek megfelelően ezt az eljárást alkalmaztam a chatbotban. A létrejött beágyazásokat és szeleteket egy lokálisan, fájl alapon működő ChromaDB vektor adatbázisba helyeztem el ezután, amely lépés része volt a feldolgozási eljárásnak. A HTML és PDF dokumentumokat külön kollekcióban helyeztem el az adatbázisban, hogy ne keveredjenek a szeletek.

Az adatok betöltése és tárolása után a chatbot válasz generálási folyamatát implementáltam. A megvalósításhoz a Llama Index OpenAI asszisztens elemét alkalmaztam. Az asszisztens képes a korábbi üzenetekre visszaemlékezni és különlegessége, hogy eszközöket tudunk definiálni számára, amelyek kimeneteit tetszőlegesen felhasználhatja. Ennek megfelelően két QueryEngineTool eszközt definiáltam a GYIK és szabályzat dokumentumokhoz. A QueryEngineTool eszköz vektor hasonlósági keresést alkalmaz az adaton, majd a legjobb szeletekből tetszőleges módon egy kimenetet generál. A megoldásomban az eszközök szimpla összefoglalással állították elő kimeneteiket, amelyeket az asszisztens tudott hasznosítani. Az összegzések után az eszközök kimenete az LLM-hez kerül, amely a végső választ generálja.

A chatbot működésének definiálása után kísérletezni kezdtem az eredmények javítása érdekében. Elsőként az adat kinyerésén szerettem volna javítani, másodjára pedig a válasz generáláson. Az adat kinyerési folyamat javításához a reranker komponens bevezetésével próbálkoztam, amelyet a Coherence reranking szolgáltatás rerank-multilingual-v3.0 [20] modelljét használva próbáltam ki. Az eredmények alapján a komponens sokszor felülbírált a koszinusz távolság alapon kiértékelt szelet sorrendet, habár drasztikus sorrend változtatásokat nem végzett. A következő ábra ezt a jelenséget mutatja be.



9. ábra: Reranking hatása a dokumentumok sorrendjére

A 9. ábra bal oldalán a koszinusz hasonlóság, míg a jobb oldalán a reranker által meghatározott pontszám alapján vannak a szeletek rendezve. Az ábra arról tanúskodik, hogy bár a reranker abban egyetértett nagyjából a koszinusz hasonlóság alapú

kiválasztással, mely három szelet a legjobb, a sorrendjükben azonban nem értett egyet. Például a koszinusz hasonlóság alapján harmadiknak ítélt szeletet első helyre tette. Ez a döntés azonban hibás, mivel a legjobb szelet, amely alapján egy ember megválaszolta volna a kérdést, csak a harmadik helyre került. A koszinusz hasonlóság alapú keresés eredményében a legjobb szelet a második helyen áll. Azt azonban el lehet mondani, hogy a reranker képes volt leszűkíteni a felhasználandó szeletek számát, mert habár a sorrend nem tökéletes, de képes volt a valóban releváns szeleteket előtérbe helyezni és a korábban a koszinusz hasonlóságon alapuló kereséssel ellentétben irrelevánsnak minősíteni valóban irreleváns szeleteket. Ennek a pozitív hozadéka, hogy ha egy küszöb értéket szabunk meg, amely szűri a szeleteket pontszám alapján, akkor az eredeti 10 helyett csak 4 szelet jutna tovább az LLM-hez, amely mind a költségeket, mind a hallucinálás vagy értelmetlen válaszok lehetőségét csökkenti. Az eljárás sikerességének kiértékeléséhez a 4. fejezetben szereplő táblázatban definiált kérdéseket szegeztem a chatbothoz és megfigyeltem a válaszokat reranker használatával, illetve anélkül is. Az eredményekből az derült ki, hogy sok esetben nem segített a reranker az eredmények javításában. Előfordult, hogy rontott a helyzeten, hallucinálásra kényszerítve a chatbotot volt, ahol pedig csak a válasz megfogalmazásában javított, a válaszok információ tartalma nem változott. A jelenséget annak tudtam be, hogy a koszinusz hasonlóság alapú keresés eleve eléggé jól működött, illetve a reranker modell nem tudott elég jól magyarul. A következtetést abból vontam le, hogy néha a reranker nélküli szelet sorrend volt közelebb a helyes megoldáshoz, a reranker nélkül is tartalmazta a legjobb 10 dokumentum a legjobb dokumentumot, illetve a 9. ábra is mutatja, hogy a koszinusz hasonlóság szerinti sorrendben drasztikusan a reranker nem változtatott, például nem láthatunk olyan esetet, ahol az utolsó 3 szeletből lett valamelyik az első.

A reranker használata után a válasz generálási folyamat javításával foglalkoztam. Elsőként a promptok és eszköz leírások javításán dolgoztam, amelyek azonban csak a nem triviális kérdések esetén tudták rávenni a chatbotot, hogy ne válaszoljon a kérdésre. Sajnos sok esetben a válasz minőségén a prompt nem javított, azonban rá tudtam venni a modellt, hogy a nem-megválaszolando kérdéseket udvariasan utasítsa el. Magyar és angol promptokkal is kipróbáltam a rendszert, angol promptok mellett a rendszer sokkal pontosabban tudta követni az utasításokat. A végső promptokat és eszköz leírásokat a 10. ábra mutatja be.

```

qna_tool_metadata = ToolMetadata(
    name="BME_FAQ_Website_Data_Store",
    description=(
        "The tool contains frequently asked questions and answers about Central Study Office administration, the use of Neptun, academic matters, finances, student ID cards, and student loans at BME, a Hungarian University."
    ),
),
legal_documents_tool_metadata = ToolMetadata(
    name="BME_Rule_Book_Data_Store",
    description=(
        "The tool contains legal regulations and rules about how BME, a Hungarian University works."
    ),
),

system_prompt = """Personality:
You are an FAQ agent for a Hungarian university called BME. Your task is to provide informative answers to users using only the tools at your disposal.
Instructions:
- You have two tools. One contains answers to frequently asked questions, and the other contains more detailed information regarding the university's operations and regulations.
- Always use the information from the available tools, never respond based on information from other sources.
- If you receive a question to which you do not know the answer, politely inform the user.
- If you receive a question not related to the university, politely inform the user that you cannot answer it, as you are a university assistant.
- Always respond to questions in Hungarian"""

query_engine_prompt = """
Based on the provided context, answer the posed question in Hungarian.
If the answer to the question is not clear based on the context, respond with: I did not find information for the question.
Context: {context_str}
Question: {query_str}
Your answer:
"""

```

10. ábra: Promptok és eszköz leírások a Llama Index chatbothoz

Az ábrán két eszköz leírás és prompt látható angol nyelven. Az eszköz leírás az LLM-nek fontos, ugyanis név és a leírás alapján tud dönteni, hogy az adott eszközt használja-e vagy sem. Az első eszköz a GYIK adatokat tartalmazza, ennek megfelelően az eszköz leírásba belefoglaltam az adat típusát, továbbá az eszköz adatában tárgyalt témákat. Hasonló módon jártam el a következő eszközzel is, amely a szabályzatokat tartalmazta. Az első prompt a rendszer prompt, amelyik általános utasításokat tartalmaz a chatbot számára. A prompt kialakításánál Zero-shot prompt technikát alkalmaztam. A Zero-shot prompt írási technika nem definiál semmilyen példát az LLM-nek a kérdés megválaszolásához. A prompt két részből tevődik össze, a személyiség leírásából, ahol a chatbotot egy szerepbe helyeztem, illetve az utasításokból, amelyek egyértelművé teszik a modell számára azokat a szabályokat, amik mentén a választ generálnia kell. Az utolsó prompt az eszközökből származó adatok összefoglalására szolgál. Típusát tekintve szintén Zero-shot típusú. A benne látható kapcsos zárójelekben lévő rész egy változó, amelyet a Llama Index kezel és futási időben helyezi be az egyes változókat a promptba.

A promptok kipróbálása után a válasz különböző válasz generáláshoz és összegzéshez használt modell és top-p érték párokat próbáltam ki. A Top-p, vagy Nucleus Sampling [21] értékkel meghatározhatjuk, milyen valószínűséggel rendelkező szavakat vegyen a modell egyáltalán számításba a válasz generálás során, ugyanis minden szónak van egy valószínűsége, hogy az előző szó után jön. A hőmérsékleti érték a válaszok diverzitását növeli, kisebb hőmérsékleti értéknél a modell kisebb valószínűséggel generál ugyanarra a bemenetre eltérő választ. Egy RAG rendszerben, különösen tesztelés során a

hőmérsékleti értéket tehát érdemes alacsonyan tartani. Ennek megfelelően a hőmérsékleti értéket 0 értéken fixáltam. Az eredmények alapján mindegyik modell 0.25-ös top-p érték mellett érte el a legjobb eredményét, a reranker azonban sokszor rontott a válaszok minőségén, néha félrevezetve a modelleket. A legjobban teljesítő modell, számomra meglepő módon a GPT3.5-Turbo volt, ugyanis ez a modell hallucinált a legkevesebbszer. Az eredmények bemutatásához kiválasztottam egy pár GPT4-Turbo és egy GPT3.5-Turbo által generált választ:

Kérdés: Milyen bizottságok működnek állandóan az egyetemen belül?

GPT4-Turbo válasz: Az egyetemen belül az alábbi állandó bizottságok működnek: 1. Oktatási Bizottság 2. Oktatási és Vizsga Bizottság 3. Tudományos Bizottság 4. Hallgatói és Kollégiumi Bizottság 5. Könyvtári és Információ Technológiai Bizottság 6. Fegyelmi Bizottság

GPT3.5-Turbo válasz: Az egyetemen állandóan működő bizottságok közé tartozik az Egyetemi Kreditátviteli Bizottság (EKB) a kreditátviteli koordinációért felelős, a Felvételi Bizottság, amely az felvételi folyamatokat felügyeli, az Egyetemi Tanulmányi Bizottság (ETB) az akadémiai és vizsgaügyi koordinációt kezeli, az Egyetemi Szociális Bizottság (ESzB) a diákok szociális ügyeit kezeli, valamint a Kari Tanulmányi Bizottság (KTB) foglalkozik az akadémiai és vizsgaügyekkel a kar szintjén.

A válaszokból látszik, hogy a GPT4-Turbo válasza nem a megtalált kontextus alapján jött létre. A modell válaszában sem a kontextusban, sem a valós életben nem szereplő bizottságok lettek megemlítve. A GPT3.5 modell ezzel szemben a kontextusban szereplő, valóban létező bizottságok neveit és helyes rövidítéseiket említette meg. A kontextus a teszt során minden egyes modell esetén azonos volt, így minden modell egyelő esélyekkel indult. Az eredményeket tekintve tehát úgy döntöttem, hogy top-p értéknek 0.25-öt adok meg, LLM-ként a GPT3.5-Turbo modellt alkalmazom, illetve reranker modult nem használok a végső teszteknel.

4 Chatbotok kiértékelése

A chatbotok kiértékeléséhez kérdés alapú manuális kiértékelést végeztem, mivel a szakirodalomban használatos automatizált, pontozás alapú módszerek működése módszertanilag kérdéses, ugyanis az automatikus kiértékelést és a válasz generálást is ugyanúgy egy LLM végzi. Habár a legtöbb kutatás magas korrelációt mutatott az emberi vélemény és az LLM által alkotott kiértékelés között [22], egyetlen kutatást sem találtam, amely magyar nyelv esetén vizsgálta volna a jelenséget, márpedig az LLM-ek teljesítménye nyelvenként eltérő, ezért helytelen azzal a feltételezéssel élni, hogy az angol, vagy más elterjedt nyelvekkel történt megfigyelések a kevésbé elterjedt nyelvekre is pont ugyanúgy érvényesek. További problémát jelentett, hogy a Dialogflow CX eszközhöz nem rendelkezik egyik automatikus kiértékelési eszköz sem integrációval és az eszköznek sincs ilyen jellegű beépített modulja.

A kiértékeléshez először azonosítottam a különböző kérdés típusokat, amelyek segítenek a rendszer működésének felmérésében, majd egy adathalmazt állítottam elő, amely típusonként két kérdést tartalmazott, illetve a kérdések mellé a válasz generáláshoz használt szeleteket és magát az elvárt választ. Az adathalmazt abban az esetben amikor a téma nem érthető közemberek számára, például nehéz jogi szövegekből áll a tudásbázis, hozzá értő szakemberek szokták összeállítani emennyiben lehetséges, ezzel javítva a kiértékelés minőségét. Az általam választott tudásbázis adat átlagember számára is értelmezhető, ezért kivételesen én állítottam össze a teszteléshez szükséges adathalmazt.

Kérdés típusokból triviális, nem-triviális és nem-megválaszolando kérdéseket azonosítottam. Triviális kérdés esetén a kérdés és a válasz is szó szerint szerepel a tanító adatban. A kérdés célja a visszanyerő és a generáló komponens működésének kiértékelése ideális körülmények között. A teszt eredménye a kapott válasz és a tanító adatban szereplő válasz vektor beágyazásának koszinusz távolsága, ahol a kisebb érték jobb eredményt jelent, mivel azt feltételezzük, hogy a rendszer képes visszaadni a megfelelő választ egy kérdésre, és érzékeli, hogy a kérdésre a talált szelet módosítás nélkül is tökéletes válasz. Ezáltal megbizonyosodhatunk arról is, hogy a modell érti a felhasználó üzenetét és a kapott szelet tartalmát is. A kérdések előállításához a tanító adatból választottam ki véletlenszerűen két kérdést, amelyekhez a hozzátartozó válasz hossza átlagosnak tekinthető.

Nem triviális kérdés esetén se a kérdés, se az elvárt válasz nem szerepel a tanító adatban pontos egyezéssel, azonban a tanító adatban szerepel elégséges információ a kérdés pontos megválaszolásához. A nem triviális kérdéseknek is két fajtáját különböztettem meg a válasz generáláshoz szükséges dokumentum részletek mennyisége alapján. Egy dokumentum részlet alapján megválaszolható kérdések esetén a válasz generáló modell kulcs információ kinyerési és az kinyert információból való válasz generálási képességét értékelem ki. Több dokumentum részlet esetén arra is választ kapok, hogy a chatbot válasz generáló modellje mennyire képes több dokumentum részletből is kiemelni a lényegi információt és pontos választ generálni a kérdésre, illetve, hogy a keresés hogyan kezeli, ha az információ több dokumentum részletben van szétszórva. A kérdések előállításához a tanító adat tüzetesebb átnézésére volt szükség. Itt fel kell hívjam a figyelmet, hogy normális esetben ezeket a kérdés-válasz párokat az adott foglalkozáshoz értő emberek készítik el. A rendelkezésre álló tanító adat minősége és a benne foglalt, hozzáértő kollégák által létrehozott kérdés-válasz párok azonban elég támogatást nyújtottak ahhoz, hogy a chatbotok limitált tudásához kellően pontos kérdés-válasz párokat generáljak önmagam is. Mindemellett jelen kiértékeléshez elégséges, ha a chatbot a rendelkezésre álló kulcs információkat kinyeri majd pontosan és hiánytalanul integrálja megfogalmazott válaszában.

Nem-megválaszolandó kérdés esetén se a kérdés, se a válasz, se a válaszhoz szükséges információ nem áll rendelkezésre a tanító adatban és a chatbotnak nem szabad érdemben válaszolnia, hanem tudatnia kell a felhasználóval, hogy a kérdésre nem válaszolhat. A kérdés célja felmérni a keresés pontosságát fals pozitív esetre, illetve a válasz generáló modell hallucinálásra való hajlamát.

A kérdés típusok definiálása és ismertetése után a teszt adathalmazt állítottam össze. Az adathalmaz a kérdés típusából, a kérdésből, a kérdés megválaszolásához általam használt dokumentum szeletből és az elvárt válasz tartalmának kivonatából. A teszt adathalmazt az alábbi táblázat tartalmazza:

Kérdés típus	Kérdés	Szelet	Válasz lényegi tartalma
Triviális kérdés	Hogyan tudok pénzt visszautalni a számlámra?	BME GYIK gyűjtőszámla, pénzügyek: Hogyan tudok pénzt visszautalni a számlámra?	A Neptun rendszerben a Pénzügyek menü, Tranzakciós lista alatt, a gyűjtőszámláról bármikor

			<p>visszaülalható az ott lévő összeg. A folyamat a Szűrések opcióval pontosítható, ahol minden félésre és a gyűjtőszámlára utalás típusát kiválasztva, a megjelenő Visszaülalás gombra kattintva indítható el a tranzakció. A minimális visszaülalható összeg 100 Ft. A visszaülalások több időt vehetnek igénybe, akár a kérelem benyújtását követő második hélig is eltarthat, mire az összeg megérkezik a célszámlára.</p>
	<p>Hogyan tudom módosítani a bankszámlaszámomat a Neptunban?</p>	<p>BME GYIK gyűjtőszámla, pénzügyek: Hogyan tudom módosítani a bankszámlaszámomat a Neptunban?</p>	<p>A Neptunban az új bankszámlaszám felvételéhez menj a Pénzügyek, majd Beállítások és Számlaszámok menüpontok alá, ahol az új számlaszámot rögzítheted és mentheted. Ezután állítsd be alapértelmezettnek az új számlaszámot. A régi számlaszámot nem lehet törölni, de ha rajta sikertelen kifizetés történt, a helyes számlaszám rögzítése után a következő</p>

			utalásnál újra próbálkozunk.
Nem-triviális kérdés	Milyen bizottságok működnek állandóan az egyetemen belül?	BME TVSZ: 3/A. § 15, 4. § 17, 5. § 27, 74/B. § 162, 173. §	Egyetemi Szociális bizottság, Kari tanulmányi bizottságok, Egyetemi, Hallgatói esélyegyelőségi bizottság, kreditátviteli bizottság, Doktori tanulmányi bizottság
	Mi a képzés vezetésének definíciója?	BME képzési kódex: 6. §	A képzés vezetésének definíciója az egyetemi polgárok viselkedésének tudatos befolyásolása a BME Intézményfejlesztési Tervében meghatározott képzési célok elérésére irányuló tevékenységek összessége.
Nem-megválaszolható kérdés	Mit tudsz a fagyikról?	-	Sajnos erre a kérdésre nem tudok válaszolni.
	Hogyan lehet egy fűnyírót beüzemelni?	-	Sajnos erre a kérdésre nem tudok válaszolni.

Elsőként a Google Dialogflow CX asszisztent teljesítményét vizsgáltam meg a teszt kérdések segítségével. Megfigyeléseim alapján a triviális kérdések nem okoztak problémát a chatbot számára, pontos és a kérdésre kielégítő választ adott mindkét esetben. A kiválasztott dokumentum szeletek relevánsak voltak és a generátor helyesen válaszolt a feltett kérdésre. Példaként szolgál erre a következő kérdés-válasz pár:

Kérdés: Hogyan tudok pénzt visszautalni a számlámra?

Válasz: A gyűjtőszámlán lévő pénzt bármikor vissza lehet utalni bármely a Neptunban megadott bankszámlára. (Pénzügyek - Beállítások - Számlaszámok) Ezt a Neptunban a Pénzügyek-Tranzakciók lista menüpontot kiválasztva lehet megtenni. A Szűrések alatt >> Félév: minden félév, Fizetési típusok >> utalás a gyűjtőszámlára, Gyűjtőszámla: #### Ekkor megjelenik a Visszaautalás gomb, amire kattintva lehet a gyűjtőszámláról pénzt visszaautalni. A legkisebb visszaautalható összeg 100 Ft. A visszaautalások hosszú időt vesznek igénybe, mert be kell várni a megfelelő nagyságú összeget, ezért az is előfordulhat, hogy a visszaautalástól számított második héten érkezik csak meg a pénz a célszámlára.

Azonban az első nem-triviális kérdésre egy hallucinált választ adott. A válasz ugyanis nem tartalmazott információt a megtalált dokumentum szeletekből és látszólag a modell saját, tanításából eredő tudását felhasználva keletkezett. Ebből fakadóan a válasz nem volt helyes. A hallucinálás oka ebben az esetben a talált dokumentum szeletek hosszára és hasznos információ tartalmára vezethető vissza. A kérdéshez megtalált három dokumentum szeletből kettő tartalmazott a kérdés megválaszolásához szükséges információt, amely önmagában nem probléma. Azonban a dokumentum szeletek hossza egyenként átlagosan 3000-4000 karakter volt, melyből a hasznos információ körülbelül 29 karakter. Ilyen információ sűrűség mellett a modell nem tudta kinyerni a lényeges információt a szövegből. A helyzetre potenciális megoldás lehet a szeletek méretének csökkentése, vagy egy olyan modell választása, amely pontosabban tudja azonosítani a kérdés megválaszolásához szükséges információt a szövegben. Azt is megfigyelhetjük, hogy a modell a szöveg generálását félbehagyta, ez azonban a maximális válaszként generálandó szöveg méret miatt van, ami egy állítható paraméter, így ez nem tekinthető hibának. A kérdés és a hallucinált válasz a következő volt:

Kérdés: Milyen bizottságok működnek állandóan az egyetemen belül?

Válasz: A következő bizottságok működnek állandóan az egyetemen belül:

- * Az oktatási bizottság felelős az oktatási programok és szolgáltatások minőségének biztosításáért.
- * A kutatási bizottság felelős a kutatási programok és szolgáltatások minőségének biztosításáért.
- * A szolgáltatási bizottság felelős a szolgáltatások minőségének biztosításáért.
- * A finanszírozási bizottság felelős a pénzügyi források biztosításáért.
- * A személyzeti bizottság felelős a személyzet minőségének biztosításáért.
- * A létesítményi bizottság felelős a létesítmények minőségének biztosításáért.
- * A biztonsági bizottság felelős a biztonsági szolgáltatások minőségének biztosításáért.
- * A kommunikációs bi

A második nem-triviális kérdésre a chatbot, habár nem az elvárt definíciót idézte fel, helyes választ adott, ugyanis a képzés definíciója két külön dokumentumban kétféleképpen van definiálva. Az egyik esetben egy absztraktabb definícióval van leírva a

képzés vezetése, a másik definíció viszont a folyamatok leírásával definiálja a tevékenységet. Ezért a választ helyesként értékeltem.

A nem-megválaszolando kérdésekre nem válaszolt a chatbot, azonban a két esetben különböző okok miatt. A „Mit tudsz a fagyikról?” kérdés esetén a chatbot a kérdést gyűlölet beszédnek azonosította, ezért a válaszadást megtagadta. Az eszköz hivatalosan a magyar nyelvet csak előlnézeti módban támogatja, azaz egyes funkciói még nem, vagy nem megbízhatóan tudnak magyar nyelven működni. A viselkedés ennek tudható be. A „Hogyan lehet egy fűnyíró beüzemelni?” kérdésre a rendszer a generatív alapértelmezett válasz lehetőséget választotta, amely tudatja a felhasználóval, hogy nem tud segíteni a kérdésben, ami az elvárt működéssel megegyezik.

A Llama Index eszköz esetén a válaszok közel kiválóak voltak. Nem volt a válaszokban hallucináció és mindegyik kérdésre sikerült a chatbotnak helyes választ adnia. Egyetlen kifogás a válaszokkal kapcsolatban a válaszok rövidegsége, illetve a válaszokban nem teljesen szerepelt mindig az az adat mennyiség, amely a kontextus alapján a modell rendelkezésére állt. Szerencsére minden esetben az adat csak a válasz teljességéhez adott volna hozzá, nem kulcsfontosságú információ veszett el. A Dialogflow CX chatbottal való összehasonlítás érdekében a következő néhány sor a korábban feltett kérdésekre a Llama Index chatbot által adott válaszokat fogja tartalmazni.

A pénzt a gyűjtőszámláról bármely a Neptunban megadott bankszámlára lehet visszautalni. Fontos tudni, hogy a visszautalások hosszú időt vehetnek igénybe, és a legkisebb visszautalható összeg 100 Ft. Ha az utalás összege 3 munkanapon belül nem jelenik meg a gyűjtőszámlán, kérhető az eset kivizsgálása a megfelelő dokumentumok csatolásával.

Az egyetemen állandóan működő bizottságok közé tartozik az Egyetemi Tanulmányi Bizottság (ETB), Egyetemi Szociális Bizottság (ESzB), Egyetemi Kreditátviteli Bizottság (EKB) és Kari Tanulmányi Bizottság (KTB). Minden bizottságnak specifikus felelősségei vannak az egyetemen belüli oktatási és adminisztratív ügyekkel kapcsolatban.

Összefoglalva az eredményeket, a Dialogflow CX chatbot egy nem-triviális és egy nem-megválaszolando kérdésnél hibázott, míg a Llama Index chatbot egyetlen teszt kérdés során sem hibázott. A jó válaszokat megfigyelve a két chatbot válasza között nincs hatalmas különbség. A Dialogflow CX chatbot hosszabb válaszokat adott, amelyek több kontextusból származó információt tartalmaztak, a Llama Index chatbot pedig rövidebb és tömörebb válaszokat adott. A Llama Index chatbot mindemellett a nem megválaszolando kérdések esetén udvariasan tudatta válaszként a felhasználóval, hogy csak az egyetemmel kapcsolatos kérdésekben tud segítséget nyújtani. A Dialogflow CX

chatbot ezzel szemben csak egyszer válaszolt hasonlóan, a másik ugyanebbe a kategóriába tartozó kérdésnél tévesen ítélte meg a felhasználó kérdését.

5 Chatbot építő eszközök összehasonlítása

A chatbot építő eszközök összehasonlítása során igyekeztem mérhető és nem mérhető, a fejlesztés során tapasztalt élmények alapján is összehasonlítani a két eszközt, hogy átfogóbb képet tudjak formálni a két eszközről.

5.1 Fejlesztési élmény, interakció az eszközzel

A Dialogflow CX egy kódolást nem igénylő webes szerkesztő felületet biztosít, ahol minden funkció és beállítás kényelmesen, néhány gombnyomással elérhető. Ezen a felületen egy bizonyos mélységig lehet a működésre hatással lenni, amely számomra elégségesnek bizonyult az eszköz lefejlesztéséhez. Amennyiben nem lettem volna elégedett a személyre szabhatósággal, úgy a termék egy API-t is a fejlesztők rendelkezésére bocsát. Fontos kiemelni, hogy ha nagyon részleteiben szerettem volna az eszközt személyre szabni erre az eszköz nem lett volna alkalmas. Részletes személyre szabás alatt például az egyes eszközök végső válaszának előállítási módját értem. Ha nem összegzést szerettem volna, hogy végezzen az eszköz a talált szeleteken, hanem hogy kiválassza az egyetlen legjobbat és annak szövegét adja vissza válaszként, a fejlesztés nem sikerült volna. Mindemellett az eszköz körül az évek során egy aktív fejlesztői közösség alakult ki, a dokumentációk részletesek és egyértelműek. Egyetlen hátrány, amit a fejlesztés során érzékeltem, amely lehet előny is, hogy a termék folyamatosan változik, gyors ütemben. A szakdolgozat készítésének idejében az eszköz éppen Generatív MI funkciókkal bővült, amelyet én is használtam, azonban ezekhez a modulokhoz már kevesebb a dokumentáció. Ennek ellenére a fejlesztést nagyjából két hét alatt sikerült befejeznem.

Mindez a Llama Index esetében nem mondható el. A keretrendszerhez nem tartozik semmilyen grafikus fejlesztői felület, a fejlesztendő alkalmazást a választott programozási környezetben kell implementálni, amit én Jupyter Notebook-ban oldottam meg. Referenciaként a Microsoft hasonló megoldása a Semantic Kernel a Prompt Flow Visual Studio Code bővítményével grafikus felületet is kínál. A dokumentáció struktúráját számomra nehéz volt megérteni először, mivel nagyon sok apró részletet alkalmazási példákon keresztül mutatott be, amelyek minden kombinációra nem terjedtek ki. Ebből fakadóan sokszor volt elakadásom, amikor nem pont úgy akartam alkalmazni egy-egy komponenst, ahogy az a példákon keresztül be volt mutatva. Pozitív élmény volt

azonban, hogy megfelelő idő ráfordításával az eszköz által készített terméket részletesen személyre lehetett szabni, amely végül jobb eredményt ért el, mint vetélytársa.

5.2 Költségek

A Llama Index esetén a keretrendszerért nem kell fizetnünk, azonban a felhasznált erőforrásokért igen. Ennek megfelelően nehéz pontos becslést adni a költségekre, ugyanis a felhasználható erőforrások és azok költsége felhasználási esetenként változik. A költségek a válasz generálásért felelős nyelvi modell, a beágyazásokat generáló modell, az esetlegesen alkalmazott vektor adatbázis futtatása, a Llama Index applikáció futtatása, illetve a felhasználói felület futtatása között oszlik szét. A szakdolgozatban bemutatott chatbot fejlesztésére összesen 22\$ értékű kreditet költöttem el az Open AI API-n keresztül, amely tartalmaz sok sikertelen próbálkozást, a tesztek futtatását és a három különböző modellel való válasz generálást.

A Dialogflow CX esetén a költségek becslése egyszerűbb, bár több komponensből épül fel az ár. A dokumentáció alapján a Chat Agent típus, amit a dolgozatban alkalmaztam 12\$-ba kerül 1000 üzenetenként, amihez az adatfeldolgozáshoz szükséges összeg is hozzáadódik, amely 10GiB-nyi adat alatt ingyenes, onnantól pedig 5\$/1GiB/hónap költséggel lehet számolni. A két érték bizonytalanságánál fogva mélyre menő következtetést nem tudtam levonni, azonban a Llama Index lehetővé teszi, hogy saját hardveren futtassuk a megoldásunkat, amely adott esetben egy olcsóbb megoldás is lehet. Összeadva azzal a lehetőséggel, hogy teljesen személyre szabható a végtermék, a Llama Index költségek terén sok esetben lehet olcsóbb megoldás.

5.3 Elérhető integrációk, kompatibilitás más eszközökkel

A Dialogflow CX 4 kontakt center és 17 chat integrációt támogat. Chat integrációk közül támogat szociális média felületeket, mint például a Discord és Telegram, mindemellett Slack integrációval is rendelkezik. A kontakt center integrációk lehetővé teszik, hogy a chatbot telefonon keresztül is elérhető legyen, a chat integrációk pedig könnyen elérhetővé teszik különböző közösségi média és más chat platformokon. A termék azonban jelenleg csak a Google által fejlesztett LLM-eket támogatja, ezek közül is csak a text-bison1, text-bison2, illetve a gemini1.0-pro modelleket. A Llama Index eszköz azonban más típusú integrációkat biztosít. Kommunikációs csatornának két, javarészt csak dedikált weboldalon elérhető, amely a Streamlit és a Chainlit. A

Dialogflow CX chatbottal szemben semmilyen kontakt center integrációt nem támogat, illetve nem tartalmaz semmilyen hang alapú integrációt. A keretrendszer azonban tartalmaz vektor adatbázis, LLM és egyéb más komponens integrációkat, amely nagyobb flexibilitást nyújt a fejlesztők számára.

5.4 Konklúziók

A Dialogflow CX eszköz gyors és könnyű fejlesztést tett lehetővé a kényelmes fejlesztési felületével és részletesen kidolgozott dokumentációjával. A termékkel elért eredmények nem teljesen tökéletesek, amely a magyar nyelv béta állapotban lévő támogatása miatt történhetett. A chatbot a Google Cloud infrastruktúrában működik, skálázódik és a fenttartásával, illetve a működtetéséhez szükséges hardverekkel nem kellett foglalkozni. A sok integrációnak köszönhetően pedig a termék szinte bármely közösségi média platformra, illetve weboldalra integrálható. További pozitívum a kontakt center integráció, amely lehetővé teszi a chatbottal való kommunikációt hang alapon is.

A Llama Index ezzel szemben időigényesebb, de jobban személyre szabható termék fejlesztését tette lehetővé, amely végül pontosabb eredményeket tudott produkálni vetélytársánál. Integrációk terén is elmarad az eszköz, amely azonban a szakdolgozatban szereplő chatbot esetében nem volt probléma, mivel weboldal alapú integrációkat támogat az eszköz. A szakdolgozat indító kérdésére, miszerint melyik eszköz lenne jó választás egy egyetem számára gyakran ismételt kérdésekre válaszoló chatbot létrehozására az elvégzett tesztek alapján egyiket sem ajánlanám. A Dialogflow CX pontosságát tekintve nem áll készen a magyar nyelven való generatív MI alapú működésre. A Llama Index alkalmazás pedig túl sok szakértelmet, időt és infrastruktúrát igényel egy átlagos egyetem számára. Az eset más lenne akkor, ha egy cég egy a keretrendszeren alapuló kész termékkel állna elő. Természetesen az egyetem is megvalósíthatja a chatbotot, amennyiben rendelkezik mesterséges intelligenciával foglalkozó szakemberekkel, illetve megfelelő infrastruktúrával.

Fontos kiemelni azonban, hogy a két eszköz nem teljesen ugyanazt a termék kategóriát képviseli. A Dialogflow CX egy teljes platform és a Google terméke, a Llama Index pedig egy nyílt forráskódú, ingyenesen felhasználható keretrendszer, amely nem egy kész termék.

6 Összefoglaló

A szakdolgozat részeként a Llama Index eszköz segítségével megtapasztalhattam, milyen nehéz is egy jól működő RAG architektúrát alkalmazó chatbotot fejleszteni „csak” egy keretrendszert használva. Azt is megtapasztalhattam, amely hátráltató tényező volt, mennyire kevés RAG architektúrát megvalósító, magyarul helyesen tudó, könnyen hozzáférhető chatbot építő platform létezett a szakdolgozat készítésének idejében.

Visszanézve sok helyen lehetett volna továbbfejleszteni a kivitelezést, például megnéztem volna az automatikus kiértékelési eszközök mennyire bizonyulnak hasznosnak a manuális kiértékeléssel szemben ebben az esetben, illetve, ha lett volna hozzáférésem időben más eszközökhöz, Llama Index helyett biztosan valamilyen menedzselt platformot választok a Dialogflow CX mellé az összehasonlításhoz. Egyrészt mert vélhetően sok időt spórolt volna meg amelyet például a kiértékelés bővítésére fordíthattam volna, másodrészt pedig hasonló súlycsoportú eszközöket hasonlítottam volna össze. Sajnos ez egy kényszer megoldás volt, ugyanis nem jutottam időben hozzá más eszközökhöz. Szerencsére a Llama Index egy olyan eszköz, amely egy egyetemi chatbot készítésénél biztosan a lehetséges megoldások listáján szerepelne, mint pontos és személyre szabható chatbot fejlesztését lehetővé tevő chatbot fejlesztő eszköz, amit a szakdolgozat is alátámaszt.

A chatbot programozással kapcsolatban a projekt rávilágított arra, mennyi mindent kell még tanuljak majd ahhoz, hogy valóban minőségi chatbot rendszereket tudjak készíteni a közeljövőben. A szakdolgozat készítés során is rengeteget tanultam, viszont a mondás igaznak bizonyult, miszerint minél többet tudsz annál jobban érzed, hogy semmit sem tudsz valójában. A realizáció azonban nem elrettentésként hatott rám. Amikor az első jobb eredményeket megláttam a chatbottól miután órákat töltöttem feszített figyelemmel a kód felett, rendkívül nagy örömmel töltött el.

7 Köszönetnyilvánítások

Ezúton is kiemelten szeretném megköszönni egyetemi konzulensem, Dr. Horváth Gábor hatalmas segítségét és támogatását, ami nélkül a szakdolgozat nem készült volna el ebben a formájában.

8 Irodalomjegyzék

- [1] J. Weizenbaum, „ELIZA - a computer program for the study of natural language communication between man and machine,” *Commun. ACM*, 1. kötet9, 1. szám1, pp. 36-45, 1966.
- [2] A. Radford, K. Narasimhan, T. Salimans és I. Sutskever, *Improving language understanding by generative pre-training*, OpenAI, 2018.
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser és I. Polosukhin, „Attention is all you need,” *Advances in Neural Information Processing Systems*, 1. kötet30, p. 15, 2017.
- [4] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, E. Chu, J. H. Clark, L. E. Shafey, Y. Huang, K. Meier-Hellstern, G. Mishra és Eri, „PaLM 2 Technical Report,” *arXiv*, 1. kötet2305, p. 10403, 2023.
- [5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave és G. Lample, „LLaMA: Open and Efficient Foundation Language Models,” *arXiv*, 1. kötet2302, 1. szám1., p. 13971, 2023.
- [6] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, É. Goffinet, D. Hesslow, J. Launay, Q. Malartic, D. Mazzotta, B. Noune, B. Pannier és G. Penedo, „The falcon series of open language models,” *arXiv preprint arXiv:2311.16867*, 1. kötet2311, p. 16867, 2023.
- [7] A. W. S. Inc., „What Are Embeddings In Machine Learning?,” Amazon Web Services Inc., 2024. [Online]. Available: <https://aws.amazon.com/what-is/embeddings-in-machine-learning/>. [Hozzáférés dátuma: 17. 04. 2024.].
- [8] Elasticsearch, „What is semantic search?,” Elasticsearch B.V., 2024. [Online]. Available: <https://www.elastic.co/what-is/semantic-search>. [Hozzáférés dátuma: 18. 04. 2024.].

- [9] Microsoft, „What is a vector database?,” Microsoft, 18. 03. 2024. [Online]. Available: <https://learn.microsoft.com/en-us/semantic-kernel/memories/vector-db>. [Hozzáférés dátuma: 18. 04. 2024.].
- [10] Y. Tang, „Similarity Metrics for Vector Search,” Zilliz, 11. 12. 2023. [Online]. Available: https://assets.zilliz.com/Similarity_Metrics_for_Vector_Search_Zilliz_43396d4adb.png. [Hozzáférés dátuma: 18. 05. 2024.].
- [11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel és D. Kiela, „Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” *ArXiv*, 1. kötetabs/2005.11401., 2020.
- [12] Amazon, „Amazon Bedrock - Knowledge bases - How it works,” Amazon, 2024. [Online]. Available: <https://docs.aws.amazon.com/bedrock/latest/userguide/kb-how-it-works.html>. [Hozzáférés dátuma: 17. 04. 2024.].
- [13] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang és H. Wang, „Retrieval-augmented generation for large language models: A survey,” *arXiv*, 1. kötetarXiv:2312.10997., 2023.
- [14] NVIDIA, „Technical Brief,” NVIDIA, 09. 02. 2024. [Online]. Available: <https://docs.nvidia.com/ai-enterprise/workflows-generative-ai/0.1.0/technical-brief.html>. [Hozzáférés dátuma: 18. 04. 2024.].
- [15] I. Ilin, „Advanced RAG Techniques: an Illustrated Overview,” Medium, 17. 12. 2023. [Online]. Available: <https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6>. [Hozzáférés dátuma: 02. 05. 2024.].
- [16] Ming, „Comparing LangChain and LlamaIndex with 4 tasks,” Medium, 11. 01. 2024. [Online]. Available: <https://lmy.medium.com/comparing-langchain-and-llamaindex-with-4-tasks-2970140edf33>. [Hozzáférés dátuma: 18. 05. 2024.].
- [17] S. Li, „How to Build Your Own Google AI Chatbot Within 5 Minutes,” *Medium*, p. 1., 11. 02. 2024.
- [18] G. Kamradt, *The 5 Levels Of Text Splitting For Retrieval*, YouTube, 2024.

- [19] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder és F. Wei, *Multilingual E5 Text Embeddings: A Technical Report*, arXiv preprint arXiv:2402.05672, 2024.
- [20] S. Shi és N. Reimers, „Introducing Rerank 3: A New Foundation Model for Efficient Enterprise Search & Retrieval,” Cohere, 11. április 2024. [Online]. Available: <https://cohere.com/blog/rerank-3>. [Hozzáférés dátuma: 28. 05. 2024.].
- [21] A. Holtzman, J. Buys, L. Du, M. Forbes és Y. Choi, „The curious case of neural text degeneration,” *arXiv preprint arXiv:1904.09751*, 2019.
- [22] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez és I. Stoica, „Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena,” *Advances in Neural Information Processing Systems*, 1. kötet36, 2024.