

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Távközlési és Mesterséges Intelligencia Tanszék

Szladek Máté Nándor

**RAG és LLM technológiák alkalmazása tudományos publikációk hatékony feldolgozásához**

Integrating RAG and LLMs for Enhanced Understanding of Scientific PublicationsTartalomjegyzék

Konzulensek

Dr. Gyires-Tóth Bálint Pál

Ónozó Lívia Réka

BUDAPEST, 2025

[Összefoglaló 5](#_Toc196514396)

[Abstract 6](#_Toc196514397)

[1. Bevezetés 7](#_Toc196514398)

[1.1. Háttér és motiváció 7](#_Toc196514399)

[1.2. Célkitűzések 9](#_Toc196514400)

[2. Irodalmi áttekintés 12](#_Toc196514401)

[2.1. Természetes nyelvfeldolgozás és LLM-ek 12](#_Toc196514402)

[2.2. Retrieval-Augmented Generation (RAG) technológia 14](#_Toc196514403)

[2.3. Vektortárolási megoldások és dokumentum-visszakeresés 17](#_Toc196514404)

[2.4. Etikai és biztonsági kihívások az információ-visszakeresés és tudásalapú rendszerek területén 19](#_Toc196514405)

[2.5. Az internetes dokumentumalapú kérdezési funkciók és a RAG rendszerek kapcsolata 21](#_Toc196514406)

[3. Módszertan 25](#_Toc196514407)

[3.1. Tervezési szempontok és követelmények 25](#_Toc196514408)

[3.2. Technológiák és eszközök kiválasztása 27](#_Toc196514409)

[3.2.1. Modellek 27](#_Toc196514410)

[3.2.2. Vektortárolók 28](#_Toc196514411)

[3.2.3. A válasz nyelvi korlátjainak feloldása 29](#_Toc196514412)

[3.3. A LangChain keretrendszer 29](#_Toc196514413)

[3.4. Rendszerarchitektúra 31](#_Toc196514414)

[4. Implementáció részletei 36](#_Toc196514415)

[4.1. Kódstruktúra és komponensek 36](#_Toc196514416)

[4.1.1. Dokumentumfeldolgozó modul 36](#_Toc196514417)

[4.1.2. Vektortároló és beágyazó modul 37](#_Toc196514418)

[4.1.3. Kérdésosztályozó és nyelvfelismerő modul 38](#_Toc196514419)

[4.1.4. Válaszgeneráló és forrásmegjelölő modul 39](#_Toc196514420)

[4.1.5. Felhasználói interfész modul 41](#_Toc196514421)

[5. Kiértékelés 43](#_Toc196514422)

[5.1. A kiértékelési módszertan kialakítása 44](#_Toc196514423)

[5.1.1. A kérdésállomány összeállításának szempontjai 44](#_Toc196514424)

[5.2. Használt metrikák és számítási módjuk 45](#_Toc196514425)

[5.2.1. Precision és Recall definíciója és jelentősége 45](#_Toc196514426)

[5.2.2. Mean Reciprocal Rank (MRR) és a Score of First Relevant Document 46](#_Toc196514427)

[5.3. Eredmények bemutatása 48](#_Toc196514428)

[5.4. A módszer előnyei és korlátai 50](#_Toc196514429)

[5.5. A fordításon alapuló információkinyerés értékelése 51](#_Toc196514430)

[6. Összefoglalás 53](#_Toc196514431)

[7. Irodalomjegyzék 55](#_Toc196514432)

[8. Ábrajegyzék 61](#_Toc196514433)

[8.1. Ábrák 61](#_Toc196514434)

[8.2. Képletek 61](#_Toc196514435)

[8.3. Táblázatok 61](#_Toc196514436)

[8.4. Kódrészletek 61](#_Toc196514437)

[9. Függelék 63](#_Toc196514438)

[9.1. A 30 elemszámú méréshez használt kérdések 63](#_Toc196514439)

[9.2. A többnyelvű méréshez használt kérdések 65](#_Toc196514440)

[9.2.1. Magyar kérdés -Angol dokumentum 65](#_Toc196514441)

[9.2.2. Angol kérdés- Angol dokumentum 65](#_Toc196514442)

Hallgatói nyilatkozat

Alulírott **Szladek Máté Nándor**, szigorló hallgató kijelentem, hogy ezt a diplomatervet 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(k), 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, 2025. 04. 27.

...…………………………………………….

Szladek Máté Nándor

Összefoglaló

Az információs társadalom fejlődése és az internet elterjedése nyomán a strukturálatlan szöveges adatok mennyisége rohamosan növekszik, beleértve a tudományos publikációkat, hírcikkeket és közösségi média tartalmakat. Ezeknek az adatoknak a gyors és pontos feldolgozása komplex problémát jelent a magyar nyelv esetében, amely agglutináló szerkezete, gazdag ragozási rendszere és szabad szórendje miatt speciális megoldásokat igényel az természetes nyelvfeldolgozás területén.

A dolgozat célja egy magyar nyelvű, dokumentumalapú chat AI rendszer tervezése és megvalósítása, amely a Retrieval-Augmented Generation (RAG) technológiát használja a szövegek kezeléséhez. A rendszer lehetővé teszi a felhasználók számára, hogy szabadon feltett kérdéseken keresztül gyorsan és pontosan hozzáférjenek nagy méretű, strukturálatlan szöveges adatbázisok mély információihoz. A fejlesztés során külön figyelmet fordítottam a magyar nyelv jellegzetességeire és a felhasználói élményre.

A fejlesztési fázis alatt kezdetben a Mistral 7B és a Meta Llama 2 modelleket használtam, de az így elkészített rendszer korlátozott teljesítménye miatt inkább áttértem az OpenAI GPT-4o-mini modell használatára. Ez lényegesen javította a válaszok minőségét és a rendszer általános értelemben vett teljesítményét. A dokumentumok embedding folyamatára és tárolására a Chroma vektortárolót választottam a rugalmassága és metaadat-kezelési képességei miatt.

A rendszert komponens-alapú architektúrával készítettem el. Részei:

dokumentumfeldolgozó modul, a vektortároló és beágyazó modul, a kérdésosztályozó és nyelvfelismerő modul, a válaszgeneráló és forrásmegjelölő modul, valamint a felhasználói interfész.

A tesztelések során a rendszer sikeresen igazolta képességeit a felhasználói kérdések megértésében, információk visszakeresésében és koherens válaszok generálásában magyar és angol nyelven egyaránt. A dolgozat hozzájárul a magyar nyelvű természetes nyelvfeldolgozás fejlődéséhez, integrálva a RAG technológia gyakorlati alkalmazhatóságát és előnyeit a magyar nyelvű szövegek feldolgozásában és információkinyerésében.

A jövőbeni fejlesztési irányok között szerepel a vektortárolási megoldások teljesítményének alapos összehasonlítása és a rendszer automatizált bővítése új dokumentumokkal.

Abstract

The dramatic spread of the Internet and the creation of the information society are accompanied with an increasing volume of unstructured text data. These include, inter alia, science articles, journalists’ writings and social network comments. The processing of these data sets represents a particular challenge for the Hungarian language which, given its agglutinative features; rich inflectional morphology and free syntax calls for tailor-made approaches in the area of NLP.

The objective of this thesis is, therefore, to design and deploy a Hungarian, retrieval-augmented generation (RAG)-based, document-centered chat AI system for text processing. The system gives users the ability to formulate questions in natural language and obtain quick and accurate responses from large collections of unstructured textual data. In the course of development, much emphasis was made on the words of the Hungarian language which are of distinct nature, and creating a friendly interface for interaction with the system.

At first, the implementation relied on the Mistral 7B and Meta Llama 2 models. Since these models provided a very low performance of the system created with them, I decided to use the OpenAI GPT-4o-mini model which brought a visible improvement in terms of response accuracy and performance of the overall system. For document embedding and storing, I selected Chroma vector store, which provides many advantages in terms of flexibility and handling of metadata.

The system is structured with a modular architecture. These are document processing, vector embeddings storage and correspondence, question classifier and language detection, response generation and citation, and user interface components.

In the course of testing, the system was able to comprehend user’s inquiries, search for relevant information and provided responses in Hngarian and English languages. This thesis ‘s effort constitutes further steps towards enhancement of NLP for Hungarian which is combining RAG technology principles’ practical advantages in dealing with Hungarian text and information retrieval.

Future development directions embrace the clear analysis of differences of vector storage options and automatic update of the system to include additional documents.

# Bevezetés

## Háttér és motiváció

Az információs társadalom folyamatos fejlődése és az internet elterjedése eredményeként a világban elérhető adatmennyiség rohamosan növekszik. Ennek jelentős része strukturálatlan szöveges adat, például tudományos publikációk, hírcikkek, feljegyzések, blogbejegyzések és közösségi média tartalmak formájában jelenik meg. Ezek az adatok hatalmas mennyiségű értékes információt hordoznak, amelyek hatékony feldolgozása és kiaknázása nagyon fontos a tudományos kutatás, az üzleti élet és számos egyéb terület számára. A tudományos publikációk írásakor a szakirodalom elemzése, olvasása a legtöbb kutató számára bár tanulságos, de a hatékonyság és eredményesség szempontjából nem mindig kielégítő. Ennek egyik oka, hogy hosszú, fáradalmas munka a pontosan kapcsolódó szakirodalmak megtalálása. Mindemellett pedig a megfelelő források megtalálása koránt sem triviális, hiszen a felgyorsult világban nehéz megtalálni a legfrissebb, témához legjobban kapcsolódó cikkeket. A világ teljes adatmennyiségét 2018-ban 2025-re vetítve 175 Zetabytera becsülték (Reinsel, Gantz and Rydning, 2018). Az aktuális statisztikák is becslések nyomán készülnek, azonban világosan megmutatják, hogy a 2018-ban tett becsléseket már túlszárnyalta az emberiség. 2023-ban 123, 2024-ben 149 ZB adat becsülhető. Ezzel a léptékkel 2028-ra elérheti a világ a 394 ZB adatmennyiséget is (Petroc, 2024).

A mesterséges intelligencia és a gépi tanulás területén elért fejlődésnek, előrelépéseknek köszönhetően új módszerek váltak elérhetővé a strukturálatlan szöveges adatok elemzésére, amik eddig nem álltak az emberiség rendelkezésére széles körben. Ennek nem csupán az akadémiai, matematikai módszerek ismeretét érintő fejlettség a kulcsa, hanem erős technológiai, kapacitáskorlátokat érintő vetülete is van a kérdésnek. Általánosságban tapasztalható, hogy a számítási kapacitások növekedése, a kellő mennyiségű és minőségű erőforrás ugyanis a legtöbb módszer használatánál szűk keresztmetszetként, kapacitáskorlátként jelenik meg. A nagy nyelvi modellek (Large Language Models, LLM-ek), mint például a GPT-3 és a GPT-4, képesek komplex nyelvi feladatok megoldására, beleértve a szövegértést, hosszú szövegek összefoglalását, a különböző nyelvek közötti fordítást és a szöveg-, válaszgenerálást (Brown et al., 2020). Ugyanakkor ezek a modellek elsősorban angol nyelven működnek optimálisan, és a kisebb nyelvek, mint a magyar, kevésbé vannak reprezentálva a tanító adathalmazokban. Láthatunk persze kivételeket és vannak nyelvi modellek, amik rendkívül jól használják a kisebb nyelveket, mint például a magyart, azonban a legritkább nyelvekkel továbbra sem boldogulnak kielégítően ezek sem. A folyamatos fejlődés és a tanítás következményeként ezek a jövőben várhatóan változni fognak az erőforrásokhoz való hozzáférhetőség megléte miatt, azonban a mintázat, ami szerint a nagyobb nyelvekkel jobban boldogulnak ezek a nyelvi modellek várhatóan megmarad, még ha nem is szembetűnő módon. Ez a különböző szakzsargonok, regionálisan használt nyelvváltozatok esetében lehet szembetűnő a mindennapi felhasználó számára.

A magyar nyelv sajátosságai, például az agglutináló szerkezet, ami a ragok „ragasztását” jelenti, a gazdag ragozási rendszer és a szabad szórend, különleges feladatot jelentenek a számítógépes feldolgozásban (Rehm and Uszkoreit, 2012). Emiatt kiemelten fontos olyan megoldások kidolgozása, amelyek a magyar nyelv kezelésére kifejezetten alkalmasnak tekinthetőek.

A nyelvi modellek tanításával azonban nem kizárólag a nyelvet tanulja meg a nyelvi modell, hanem a különböző szavak kontextusát is, ami azt jelenti, hogy a betanított szövegkörnyezetből képesek ezek a modellek akár valós tények visszagenerálására is. Ez a kognitív képességek fejlődésének látszatát kelti. A látszat azonban ebben a környezetben valóban látszatként jelenik meg, ugyanis a betanított szövegekből az információ kinyerése pusztán statisztikai lehetőségként jelenik meg, és nem funkcióként ezekben a modellekben. A betanított szövegek mennyiségének növelésével és egyéb módszerekkel javítható az igaz, hiteles információk aránya, azonban ennek fejlődésével a hiteles szövegalkotás aspektusa is javul, így az emberek egyre nehezebben veszik észre a helytelen információkat, amiket a modell válaszul adhat. Ezt a hamis tényközlést nevezi a köznyelv hallucinációnak. Tapasztalataim alapján ráadásul ez a kognitív képességek látszatának és a hiteles információk visszaadásának valószínűségi aránya tematikusan igen eltérő. Az emberek autizmusnak nevezett idegrendszeri rendellenességhez hasonlítanám a jelenséget, ugyanis vannak témák, amelyek esetében ezek a nyelvi modellek kiemelkedőnek mondhatók hiteles információk közlése szempontjából, míg más esetekben ezek a képességek jócskán elmaradnak. A nyelvi modell nem tudja eldönteni természetesen, hogy milyen okos a különféle témákból, a valószínűségek alapján történő optimalizációs válaszadás miatt pedig hitelesként állíthat be valótlan információkat.

A Retrieval-Augmented Generation (RAG) technológia az előzőek tükrében azonban új megközelítést adhat az információkinyerés és szöveggenerálás területén, ahogyan a neve is jelenti. A RAG rendszerek a visszakeresett információkat generatív nyelvi modellekkel ötvözik, hogy pontos válaszokat adjanak. A visszakeresett információk nem a nyelvi modell részei, hanem valós, ellenőrizhető adatbázisból állnak elő. Egyszerűen úgy is mondhatnánk, hogy a generatív nyelvi modell képességeiből ekkor nem a modell tanításakor keletkezett háttértudást használjuk fel, hanem lehetőség szerint csak a nyelvtanilag pontos fogalmazás képességét.

Korábbi munkám során egy egyszerűbb modellt finomhangoltam nagymennyiségű gazdasági adatokon, főként nyilvános, a magyar gazdaságot érintő cikkek felhasználásával. Bár ezek az ismeretek közvetlenül nem alkalmazhatók az új rendszer fejlesztésében, az LLM-ekkel szerzett tapasztalataim hozzájárultak a jelenlegi projekt alapjainak megteremtéséhez. Az ilyen módon tanított modellből ki lehetett nyerni valós információkat, illetve a szövegek gazdasági nyelvezete is jelentősen javult, azonban a válaszok hitelessége, valóságtartalma nem lett kielégítő. Úgy is mondhatnánk, hogy bár valós információkat is mondott, de ezzel együtt hitelesebben tudott valótlan információkat generálni, könnyebben becsapta a felhasználót, amit nagyon veszélyesnek tartok. Ezen a területen végzett további tanítás valószínűleg csökkentette volna a várható hamis információk arányát, azonban csak ebben a témakörben. Az ipari környezetben található erőforrások mennyisége idővel nő, ami a modellek fejlődését is gyorsítja, azonban a tanítási corpus ismerete nélkül nem tudja a felhasználó, hogy milyen témákban, mekkora valószínűséggel számíthat hiteles információkra. Ez a probléma sajnos olyan szinten is tapasztalható mindenki számára, hogy amikor az LLM modelleket működtető szolgáltatások használatával próbálunk forrást is kérni az információk felhasználásáról, akkor a rendszerek olykor valótlan, de pontos akadémiai hivatkozásrendszer formájában adott forrásokkal próbálják a hitelesség látszatát kelteni, ami nagyon veszélyes és félrevezető.

A dolgozat célja egy olyan magyar nyelvű chat AI rendszer tervezésének és megvalósításának bemutatása, amely a RAG technológiát alkalmazza a nagy méretű, strukturálatlan szöveges adatbázisok hatékony kezelésére és kapcsolódó információk gyors lekérdezésére szabadon feltett kérdések segítségével. A rendszer intuitív hozzáférést biztosít a lényeges információkhoz forrásmegjelöléssel, ezzel megkímélve a felhasználókat a dokumentumhalmaz manuális átnézésétől.

## Célkitűzések

Ahogyan korábban már szó volt erről, a strukturálatlan szöveges adatok mennyisége folyamatosan, és gyorsuló tempóban növekszik, ami új feladatokat jelent az adatok feldolgozása és értelmezése terén. Kimondottan a magyar nyelvű szövegek esetében van szükség olyan innovatív megoldásokra, amelyek képesek optimalizáltan kezelni a nyelv sajátosságait. Mivel nem cél kizárólag a magyar nyelvű tudományos írásokra szűkíteni a fókuszt, így a publikációk többségére jellemző angol nyelvű dokumentumok feldolgozása, és ezek alapján angol vagy akár magyar nyelven feltett kérdésekre adott, a kérdéssel megegyező nyelvű válasz írása fontos szempont lett.

A célkitűzéseim a következők voltak:

* Pontos információkinyerés: A felhasználóknak szabadon megfogalmazott kérdésekkel kell gyorsan és pontosan hozzáférniük a strukturálatlan szöveges adatbázisok információihoz. Az kért információt lehetőleg egyértelműen, kell a rendszernek közölnie. A dolgozatom során szöveges PDF-ek feldolgozását tűztem ki célul, és valósítottam meg. Fontos kiemelnem, hogy a szövegfeldolgozásra fókuszál a feladat, így a dokumentumban felmerülő ábrák figyelmen kívül hagyása mellett döntöttem.
* Magyar nyelv támogatása: Olyan megoldás kidolgozása a cél, amely a magyar nyelv sajátosságaira jól optimalizált, figyelembe véve a nyelv morfológiai és szintaktikai jellegzetességeit. A cél az, hogy nem csak a magyar, de az angol nyelvű dokumentumból is akár angol, de akár magyar nyelven is képesnek kell lennie közölni a megkeresett információt. Ez a nyelvi kritérium a modellválasztás és a rendszer tervezésének az egyik fő szempontja.
* Felhasználóbarát interfész: Egy intuitív, könnyen használható chat alapú felület kialakítása a cél a természetes, felhasználói interakciókhoz. A forrásokat egyértelműen kell közölnie a hitelesség miatt. Amennyiben nem talált olyan információt a rendszer, ami alapján kellő megalapozottsággal tudja visszaadni az információt, úgy bár generálnia kell választ a megtalált szövegekből, de fel kell hívnia a figyelmet arra, hogy a visszaadott szöveg megalapozottsága gyenge. Amennyiben ez a megalapozottság olyan mértékű, hogy már nem volna biztonságos válaszolnia, a hallucinációk csökkentése és kizárása miatt egyszerűen közölnie kell a rendszernek, hogy nem tudja a választ megadni.
* Technológiai innováció: Összehasonlítani és értékelni a különböző technológiákat és módszereket (többek között a felhasznált modellek, vektortárolók, fordítási mechanizmusok), és ezek alapján megalapozott döntéseket hozni a rendszer tervezése során. Az előzetes értékelések után elkészült rendszer értékeléséhez módszertant kell kialakítani és el kell végezni a megfelelő vizsgálatokat.
* Hozzájárulás a tudományos közösséghez: Olyan módon kell bemutatni a RAG technológia alkalmazhatóságát a magyar nyelvű szövegek feldolgozásában, hogy az minél szélesebb körben érthető legyen. Ennek tükrében pontos leírásokat kell adni a szakkifejezések jelentéséről, adott esetben a dolgozat során használt kifejezések egyedi, a dolgozatban használt értelmezéséről.

A dolgozat során részletesen bemutatom a kapcsolódó témák és a dolgozatot megelőző fejlesztések szakirodalmát. Ezekből következtetéseket vonok le, valamint implementálom az ezekből kinyerhető, a rendszer készítéséhez kapcsolódó építő tudásanyagot. Részletesen bemutatom a rendszer tervezésének és implementációjának folyamatát, beleértve az adatfeldolgozást, a modellválasztást és a technológiai döntések indoklását. Megvizsgálom az esetleges alternatívákat és a döntési pontokat a rendszerhez kapcsolódó kérdések kapcsán. A megalkotott rendszert több szempontból vizsgálom. A vizsgálat módszertani korlátait és lehetőségeit részletesen bemutatom. Az eredményeit összegzem és a tapasztalatokat levonva értékelést teszek. Az előző munkáimra alapozva összehasonlítom a korábbi és a jelenlegi megközelítéseket, kiemelve az előnyöket és hátrányokat, valamint az elért eredményeket.

A dolgozat végső célja egy olyan prototípus létrehozása, amely demonstrálja a rendszer működőképességét, és alapot szolgáltat további fejlesztésekhez és kutatásokhoz a magyar nyelvű természetes nyelvfeldolgozás területén.

Különös figyelmet érdemel, hogy a dolgozatban vizsgált technológiai környezet, azon belül is a generatív nyelvi modellek és a RAG rendszerek nagyon dinamikusan fejlődő szakterületet képez. Már a dolgozati feladatkiírás időpontjától kezdődően is sok jelentős módszertani és technológiai előrelépés történt, amely jól tükrözi a kutatási eredmények intenzív ütemű gyarapodását. A mesterséges intelligencia ezen ágaiban az új modellek, algoritmikus lehetőségek és az implementációs eszközök folyamatosan változnak, gyakran néhány hónapos időtávon belül is, ezzel új standardokat alkotva a tudományos közösség és az ipari alkalmazók körében egyaránt. Jelen dolgozat a 2025 májusáig elérhető szakirodalomra, technológiai megoldásokra és fejlesztői eszközkészletre épít. Éppen ezért indokolt hangsúlyozni, hogy a bemutatott módszerek, architektúrák és ajánlások érvényessége elsősorban ezen időszak kontextusában értelmezendőek. A RAG rendszerek jövőbeli alkalmazása és fejlesztése során fontos a legfrissebb kutatási eredmények és technológiai fejlesztések folyamatos követése, mivel bizonyos megközelítések az újabb innovációk fényében érdemben módosulhatnak. A diplomamunkában ismertetett megoldások így elsősorban egy adott időpillanat technológiai állapotát tükrözik, és nem feltétlenül garantálható, hogy azok hosszabb távon is változatlan formában a legkorszerűbb gyakorlatot fogják képviselni.

# Irodalmi áttekintés

## Természetes nyelvfeldolgozás és LLM-ek

Tapasztalhatóan 2025-ben a természetes nyelvfeldolgozás (Natural Language Processing, NLP) a számítástechnika, azon belül a mesterséges intelligencia egyik legdinamikusabban fejlődő területe, amely az emberi nyelv gépi megértésével és szövegek generálásával foglalkozik. A természetes nyelvfeldolgozás célja, hogy a számítógépek képesek legyenek az emberi nyelv feldolgozására, ezáltal megvalósítva az emberi kommunikációhoz hasonló, azt imitáló interakciókat (Chowdhury, 2003).

Az utóbbi években a mélytanulási módszerek fejlődése nagy áttörést hozott a természetes nyelvfeldolgozás területén. A hagyományos, szabályalapú és kézzel tervezett jellemzőkön alapuló módszerek helyett a gépi tanulás, azon belül is a neurális hálók és a mélytanulás, lehetővé tette a komplex nyelvi mintázatok automatikus tanulását nagy adathalmazokból (Young *et al.*, 2018). Ennek eredményeként mindenki számára tapasztalható módon számos, akár nyíltan elérhető alkalmazásban, szolgáltatásban tapasztalhatóan számottevő a teljesítményjavulás, például a gépi fordításban, a beszédfelismerésben, a szövegértésben, vagy akár hangulatelemzésben.

A nagy nyelvi modellek (Large Language Models, LLM-ek) megjelenése további lendületet adott a természetes nyelvfeldolgozás fejlődésének. Az olyan modellek, mint az OpenAI GPT-3 (Brown et al., 2020) és a Google BERT (Reimers and Gurevych, 2019), hatalmas méretű neurális hálózatok, amelyek több milliárd paraméterrel rendelkeznek, és óriási mennyiségű szöveges adaton lettek betanítva. Tapasztalhatóan ezek a modellek képesek általános nyelvi reprezentációk tanulására, amelyek sokféle feladatra alkalmazhatók minimális további tanításával (fine-tuning), vagy akár anélkül is.

Az LLM-ek egyik legfontosabb tulajdonsága, hogy méretüknek megfelelően képesek megérteni a szövegek kontextusát és összefüggéseit, így koherens és kontextusérzékeny válaszokat generálnak. Ez leginkább olyan alkalmazásokban hasznos, mint a kérdés-válasz rendszerek, a szövegösszegzés és a tartalomgenerálás (Radford *et al.*, 2019).

A GPT-3 modell például 175 milliárd paraméterrel rendelkezik, és képes több nyelven is működni, bár a teljesítménye az angol nyelven a legjobb (Brown et al., 2020). A BERT modelleknél pedig kétirányú tanulást alkalmaz, ami javítja a kontextus megértését és a szövegbeli összefüggések felismerését. Az egyszerűbb nyelvi modellek jellemzően egyirányúak (unidirectional) voltak. Ez azt jelentette, hogy a szövegek kontextusát egyetlen irányból, például balról jobbra haladva értelmezték. Ez korlátozta a modellek képességét, mert nem tudták teljes mértékben kihasználni a szöveg kontextusát. A BERT modell kétirányú tanulása azt jelenti, hogy a szöveg egy adott szóhoz kapcsolódó kontextusát mindkét irányból egyszerre veszi figyelembe (Reimers and Gurevych, 2019).

A felsorolt előnyök mellett azonban a nagy nyelvi modellek számos problémát is felvetettek. A robosztus méretükből adódóan olykor hatalmas számítási erőforrásokat igényelnek, ami jelentősen korlátozza a használatukat bizonyos területeken. Az erőforrásigényes fázis rend szerint a nyelvi modell létrehozása, fejlesztése, vagy akár a meglévő modell finomhangolása, de az egyszerű generatív felhasználása is bizonyos esetekben korlátozott lehetőségekkel valósulhat meg, ahol a tárhelykapacitási korlátokat is figyelembe kell venni. Mindemellett pedig a nyelvi reprezentációik nagyon gyakran angol nyelvű adatokon alapulnak. Ez nem meglepő, hiszen akár iparágtól függetlenül is kijelenthető általánosságban, hogy a leggyakrabban használt nyelv az angol. Tapasztalható módon a tudományos publikációk, technikai leírások, dokumentációk, de az egyszerű hírek, könyvek jelentős része angol nyelvű. ami problémát jelent a kisebb nyelvek, például a magyar esetében (Farkas, Vincze and Schmid, 2012). Ez az állítás a kutatás elején is némileg igaznak látszódott. A kisebb, kifejezetten magyar nyelven tanított modellek természetesnek tűnő nyelvezetet imitáltak, például a HUN-REN fejlesztésű PULI nyelvmodell (Yang, Szlávik and Ligeti-Nagy, 2024). Azonban a legnagyobb, többnyelvű modellek nyelvi támogatottsága a magyar nyelv esetében még kísérleti fázisban volt. A fejlesztési fázis végére a nagy nyelvi modellek dinamikus fejlődése miatt a legnagyobb erőforrásokkal fejlesztett nyelvi modellek már képesek voltak a magyar nyelvet kellő mértékben használni a szöveggenerálás tekintetében, így a később olvasható fejlesztési irányok is ennek megfelelően módosultak.

## Retrieval-Augmented Generation (RAG) technológia

A Retrieval-Augmented Generation (RAG) technológia egy olyan lehetőség a természetes nyelvfeldolgozás területén, amely két terület, az információ-visszakeresési rendszerek és a generatív nyelvi modellek integrációján alapul (Lewis *et al.*, 2020). A RAG célja, hogy a nagy nyelvi modellek fogalmazásban kimagasló tudását kiegészítse külső tudásbázisokból származó információkkal, ezáltal garantálva a válaszok pontosságát, és a dokumentumok frissességétől függő aktualitást. Ezek az adatbázisok lehetnek speciálisan erre a célre létrahozott technológia alapúak, amelyek a strukturálatlan szövegeket tárolják valamilyen struktúra szerint rendszerezve, de lehetnek hagyományosak is, tehát akár SQL adatbázisokkal is implementálható a technológiai megvalósítás (Guu *et al.*, 2020).

A továbbiakban, ha nem említi a szöveg kifejezetten, akkor a dokumentumok eltárolásával kapcsolatos technológiákról lesz szó. Az ilyen típusú RAG rendszer két fő részből áll: egy visszakeresőből (Retrieval), amely a dokumentumok bizonyos részeit azonosítja a felhasználói kérdések alapján, és egy generátorból, amely ezek felhasználásával választ készít. Ehhez gyakran használnak neurális beágyazási technikákat, amelyek a lekérdezéseket és a dokumentumokat egy közös vektortérben reprezentálják, ezzel megoldva a hasonlóság alapú gyors keresést (Karpukhin, Oguz, *et al.*, 2020).

A generatív rész ezután a visszakeresett információk és a felhasználói lekérdezés kombinációjával állít elő egy mondat vagy bekezdés szerűén megfogalmazott választ az erre a célra használt LLM segítségével. Ez tehát úgy működik, hogy a generatív modell megkapja a válaszhoz szükséges információdarabokat és ezekből kreál értelmes nyelvezettel, emberi olvasásra, befogadásra alkalmas mondatokat  (Lewis *et al.*, 2020). A generatív modell így képes kontextusba helyezni a visszakeresett adatokat, és koherens, informatív válaszokat adni a felhasználónak. Fontos továbbá, hogy lehetőség szerint a modell a válaszokat ilyenkor nem egészíti ki a saját tudásával, hanem csak a nyelvtani, fogalmazással kapcsolatos funkcióit használjuk.

A RAG technológia előnyei közé tartozik:

* Aktuális és pontos információk biztosítása: Mivel a modell külső adatbázisokból származó információkat használ, képes friss és aktuális válaszokat adni, ellentétben a statikus tudással rendelkező nagy nyelvi modellekkel. (Borgeaud *et al.*, 2022). Ez akkor lehet igazán hasznos, ha olyan adatbázisra kapcsoljuk, ami valós időben sűrűn frissül. Ilyen lehet egy csomagszállító cég naplózást vezető adatbázisa, ahonnan szöveges formában lehet információkat kérni, vagy akár egy légitársaság is integrálhatja ezt a technológiát egy olyan rendszerbe, ahol valós időben frissülő járatinformációkról lehet kérdezni a rendszert. Ezen dokumentumok begyűjtése automatizálható, azonban nagyon fontos a megbízhatóság ebben a tekintetben, ugyanis a RAG rendszereknek pontosan az a céljuk, hogy kontrollált környezetből származnak a hiteles információk.
* Pontosság és megbízhatóság növelése: A visszakeresett dokumentumokban található információk felhasználása csökkenti a hallucinációk és pontatlan válaszok előfordulását (Kwiatkowski *et al.*, 2019). Tapasztalhatóan sűrűn fordul elő, hogy az LLM szolgáltatásokkal működő, hasonló feladatokat ellátó rendszereknél abban az esetben, ha a kért információ nem elérhető, a rendszer valamilyen hamis információt szolgáltat, ami akár hitelesnek is tűnhet. Ez egy olyan jelenség, amit a RAG rendszerek esetében helyes működés mellett nem kéne tapasztalni.
* Kisebb modellméret mellett is megfelelő működés: A külső tudásbázis használatával a generatív modell kisebb paraméterszámmal is nyújthat kiváló teljesítményt, mivel nem szükséges minden információt a paramétereiben tárolnia.Ehelyett a szükséges adatokat igény szerint egy külső tudásbázisból hívja elő  (Borgeaud *et al.*, 2022). Borgeaud és társai (2022)tanulmányukban bemutatták ezt a koncepciót a RETRO (Retrieval-Enhanced Transformer) modell kapcsán, amely bebizonyította, hogy 25-ször kisebb paraméterszámmal, de külső adatforrások elérésével is képes versenyképes teljesítményt nyújtani nagyobb, GPT-3 modellel szemben.

Az alkalmazási területek kapcsán a szakirodalom ismerete alapján a következők emelendőek ki:

* Nyílt doménű kérdés-válasz rendszerek: A RAG technológia segítségével a rendszer az adatbázis jellegétől függően, akár széles körű témákban is válaszolhat a felhasználói kérdésekre, miközben garantálja a relevanciát és alacsonyan tartja a hibaarányt (Izacard and Grave, 2021). Természetesen mindez az adatbázis méretén és széles spektrumú tematikáján múlik.
* Dokumentumösszegzés és információkinyerés: A RAG technológia alkalmazható hosszú dokumentumok összegzésére és a lényeges információk kiemelésére (Liu *et al.*, 2019). Ezek mellett akár több dokumentumból is képes szintetizálni információkat, amik új dimenziókat nyitnak meg a dokumentumokból kinyert információk felhasználhatósága kapcsán.
* Chatbotok: A technológia javítja a chatbotok képességét, hogy kontextusban megfelelő és pontos válaszokat adjanak, növelve ezzel a felhasználói elégedettséget. Sajnos a hallucináció, ami a köznyelvben és a tudományos világban a téves információk hiteles bemutatását jelenti a kritikus rendszereknél nagy veszélyeket rejt magában. Ezzel a technológiával azonban kellő odafigyeléssel biztonságosan lehet chat alapon információt szolgáltatni a kritikus rendszerek esetében is akár (Thoppilan *et al.*, 2022).

A fenti előnyök mellett azonban a RAG technológia kapcsán kihívások is felmerülnek. A retrieval (visszakereső) komponens megfelelően pontos működése nagyon fontos kritérium, mivel a nagy méretű tudásbázisokban gyorsan és pontosan kell célzott információkat találni (Lee, Chang and Toutanova, 2019). Ehhez fejlett indexelési és beágyazási módszerek szükségesek, mint pl. a sűrű vektoros reprezentációk és az Approximate Nearest Neighbor (ANN) keresési algoritmusok alkalmazása (Johnson, Douze and Jegou, 2021). Ezekről később bővebben lesz szó.

Emellett a generatív modell és a visszakeresett információk kombinálása is komplex feladat. Fontos biztosítani, hogy a modell megfelelően súlyozza a visszakeresett adatokat, és azokat helyesen építse be a válaszba (Lewis *et al.*, 2020). Tematikus RAG rendszerek esetén a modell finomhangolása és a megfelelő tanítási stratégia kiválasztása is fontos lehet a rendszer teljesítménye szempontjából (Guu *et al.*, 2020). Általános, tehát nem tematizált rendszerek esetében ezen lépés helyett inkább az elérhető, és a feladat szempontjából megfelelő újabb modellek közötti teljesítmények mérése vezethet jobb eredményhez.

Asai és munkatársai (2019) munkája új előrelépést ért el a retrieval (visszakeresési) rendszerek terén a HotpotQA feladatban, amely összetett kérdések megválaszolását igényli a Wikipedia adatbázisán. Kutatásuk során a Wikipedia oldalai közötti kapcsolatokból létrehozott gráfok segítségével modellezték az érvelési utakat, és ezekre alapozva végeztek eredményes visszakeresést. Az általuk bemutatott módszer több mint 14 százalékpontos javulást eredményezett a pontosságban a korábbi visszakeresési technikákhoz képest. Bár tanulmányuk nem RAG-alapú rendszerrel dolgozott, eredményeik rávilágítanak a retrieval komponensek fejlesztésének fontosságára, amely a RAG rendszerek egyik fő eleme.

A RAG technológia továbbfejlesztése és alkalmazása aktív kutatási terület. A jövőbeni munkák fókuszálhatnak a retrieval (visszakereső) komponensek hatékonyságának növelésére, a többnyelvű támogatás kiterjesztésére és a különböző szakterületekre, témakörökre, alkalmazási területekre való specializációra. A dolgozatban bemutatott rendszer egyik lehetséges továbbgondolt alkalmazása a magyar nyelvű wikipédia feldolgozásán alapulhat. Ehhez nem szükségesek különösebb fejlesztési lépések, azonban a dokumentumállomány feldolgozása meglehetősen erőforrásigényes feladat. A szöveges adatok folyamatos változása miatt azonban folyamatos utókövetés szükséges, ami a régi szövegek beágyazott reprezentánsait eldobja, és az újakat bevonja. Ennek a lehetőségnek a további tárgyalása nem célja a dolgozatnak, valamint a wikipédia kompatibilitás megoldása sem, azonban az ilyen rendszerhez szükséges, központi RAG technológia részletes bemutatása igen.

## Vektortárolási megoldások és dokumentum-visszakeresés

Az LLM-eket használó információ-visszakereső rendszerek konzisztenciája nagy mértékben függ a dokumentumok reprezentációjától és a keresési algoritmusoktól. A vektortárolási megoldások alapvetően meghatározzák, hogy a rendszerek milyen gyorsan és pontosan képesek megfelelő információkat visszakeresni nagy méretű adatbázisokból. Például Manning, Raghavan és Schütze (2008) szerint a vektortér modellben a dokumentumok és a lekérdezések vektorként történő kezelése lehetővé teszi a koszinusz hasonlóság számítását, amely egy különösen jó módszer az érdemleges találatok rangsorolására. Az ilyen megközelítések főként akkor fontosak, ha a rendszernek milliós nagyságrendű dokumentumkorpuszban kell megfelelően működnie.

A dokumentumok vektorizációja során a szövegeket magas dimenziójú vektortérben reprezentálja a rendszer, ahol a hasonló jelentésű szövegek vektorai közel helyezkednek el egymáshoz. A beágyazási (embedding) technikák, mint például a Word2Vec (Mikolov *et al.*, 2013), a GloVe (Pennington, Socher and Manning, 2014) alkalmasak a szövegek szemantikai információik megragadására és azok numerikus reprezentációvá alakítására. Az általam használt embeddingnnem Word2Vec vagy GloVe típusú, hanem egy modern modell, az OpenAI text-embedding-3-large alapján készül, amely kontextusfüggő, mondat- vagy chunk-szintű beágyazást biztosít GPT-alapokon, hasonlóan a Sentence-BERT koncepciójához. Ez nem klasszikus token embedding (mint BERT tokenenként), hanem mondat- vagy szövegdarab-szintű embedding. Erről később több szó esik a dolgozatban.

A vektorizálás folyamata jellemzően a következő lépésekből áll:

1. Tokenizáció: A szöveges dokumentumok tokenekre bontása megfelelő nyelvi modell segítségével.
2. Beágyazások hozzárendelése: A tokenekhez kontextusfüggő beágyazások rendelése előtanított nyelvi modellek, például BERT vagy GPT, felhasználásával, amelyek figyelembe veszik a tokenek környezetét, ezáltal igyekeznek a jelentést is figyelembe venni.
3. Aggregálás dokumentumszinten: A token-beágyazások aggregálása dokumentumszintű reprezentációvá, amely történhet egyszerű átlagolással vagy speciálisabb műveletekkel.
4. Normalizálás: A dokumentum-vektorok normalizálása, dimenziócsökkentése vagy más jellemző-előkészítési technikák alkalmazása, amelyek elősegítik a pontos indexelést és visszakeresést.

Az így előállított vektorizált dokumentum-reprezentációk egy megfelelő indexelő és visszakereső rendszer segítségével tárolhatók és lekérdezhetők.

Az LLM-ek, mint például a GPT-3, képesek előállítani ún. sentence embeddingeket, amelyek a teljes mondatok jelentését sűrítik vektorokba (Reimers and Gurevych, 2019). Ezek a beágyazások nagyon fontosak a visszakeresési folyamatban, mivel támogatják a szemantikus hasonlóságon alapuló keresést a szintaktikai egyezés helyett. A későbbiekben kitérek az általam használt embedding modellre.

Az én esetemben a dokumentumok beágyazása nem token-, hanem chunk-szinten történik. A dokumentumok darabokra vannak bontva, és ezek a darabok kerülnek embeddingelésre az említett modern modell által. Ez a megközelítés egy sokkal modernebb megoldást nyújt. A bemutatott rendszernél az aggregálás és a dimenziócsökkentés ilyen formában nem célravezető, mert a visszakeresés chunk-szintű. Ennek azért van jelentősége, mert nem csak az egyszerű válaszadás a cél, hanem a korábban említett oldalszintű hivatkozások hozzárendelése és közlése is. Ennek a részletes bemutatása később olvasható.

A vektorizált dokumentumok megfelelő tárolása és visszakeresése speciális adatstruktúrákat és algoritmusokat igényel. A nagy méretű vektorkészletekben való kereséshez gyakran használnak Approximate Nearest Neighbor (ANN) algoritmusokat, amelyek a pontosság enyhe feláldozásával számottevő gyorsulást érnek el (Jégou, Douze and Schmid, 2011).

Az egyik legelterjedtebb vektortárolási megoldás a Facebook AI Similarity Search (FAISS) könyvtár, amely GPU-gyorsított keresést tesz lehetővé milliárdnyi vektor között (Johnson, Douze and Jegou, 2021). A FAISS számos indexelési módszert kínál, amelyek az optimalizált memóriahasználatot és a keresési sebesség növelését szolgálják.

Egy másik ígéretes vektortároló a Chroma, ami bizonyos szempontokból jobb, míg más szempontokból gyengébbnek bizonyult a tapasztalataim és a szakirodalmi kutatás alapján. Erről később részletesebben lesz szó. A FAISS és a Chroma összehasonlítása során több szempontot vettem figyelembe, mint például a teljesítmény, a funkcionalitás, az integráció, jövőbeli fejlődési potenciál és a könnyű használat.

A FAISS kifejezetten az ANN algoritmusokra épül, és gyors keresést biztosít a nagy méretű vektorterekben. Különböző indexelési módszereket kínál, mint például az IVF (Inverted File Index) és a PQ (Product Quantization), amelyek segítségével finomhangolható a sebesség és a hibaarány közötti kompromisszum (Laangchain, 2024).

A Chroma a Hierarchical Navigable Small World (HNSW) algoritmust használja, amely egy Approximate Nearest Neighbor (ANN) módszer. Single-node üzemmódban minden gyűjteményhez egyetlen HNSW indexet hoz létre, amelyet a gyűjtemény UUID alapú alkönyvtárában tárol. Az új vektorok először Brute Force indexbe kerülnek, amely kimerítő, azaz minden vektort összehasonlító keresést végez, mielőtt azokat az HNSW indexbe helyeznék át. Ez a megközelítés kis méretű adathalmazok esetén kellően pontos, nagy adathalmazok esetében azonban némileg időigényes lehet (Amikos Tech LTD, 2024).

A FAISS képes több milliárd vektor kezelésére és gyors keresésre GPUgyorsítással. Johnson és társai (2021) tanulmányukban bemutatják, hogy a FAISS képes másodpercenként több millió lekérdezést kezelni milliárdnyi vektor esetén, miközben a keresési pontosság 90% feletti marad.

Egy 50 kérdéses tesztelés során a FAISS és a Chroma a következő módon teljesített (Stepkurniawan, 2024):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f-érték | keresési idő (mp) |
| FAISS | 0,96 | 0,93 | 0,95 | 1,81 |
| Chroma | 0,94 | 0,89 | 0,91 | 2,18 |

1. táblázat FAISS és Chroma vektortárolók összehasonlítása (Stepkurniawan, 2024)

Látható, hogy minimális eltérések vannak és minkét technológia kellően pontos és kellően gyors is.

A FAISS erőteljes és skálázható megoldás, de integrációja összetettebb, főként akkor, ha metaadatokat is szeretnék kezelni, mint a dolgozatban bemutatott rendszer esetében. A FAISS használatához gyakran szükség lehet mélyebb technikai ismeretekre (Johnson, Douze and Jegou, 2021).

A Chroma és a FAISS előnye, hogy könnyen integrálhatóak modern természetes nyelvfeldolgozást implementáló keretrendszerekkel, mint például a LangChain (Laangchain, 2024). A Chroma natív integrációval rendelkezik a LangChain keretrendszerben, míg a FAISS esetében ehhez külön adapter szükséges, és a metaadatkezelést manuálisan kell megoldani. A Chroma által nyújtott lehetőségek ezáltal megkönnyítik a beágyazások tárolását és visszakeresését, valamint a metaadatok kezelését. Ráadásul a Chroma nemcsak tárolja a metaadatokat, de lehetőséget biztosít azok igény szerinti szűrésre is, így támogatva az összetettebb keresési igényeket. Ez előnyt jelenthet az átláthatóságban és a rendszer rugalmasságának növelésében.

A projekt követelményei mindkét rendszert kipróbáltam, és a tapasztalataim, illetve a szakirodalom alapján bár a FAISS kissé jobb eredményt mutatott néhány szempont alapján, a Chroma mellett döntöttem a rugalmassága és a jövőbeli fejlesztési potenciáljai miatt. Beépített embedding modellje automatizálja az adatok vektorokká alakítását, így nincs szükség külön pipeline-ra (MyScale, 2024). Jelen projektben nyelvi szempontok miatt nem a beépített embedding modellel készülnek az embeddingek, hanem az OpenAI egyik legmodernebb, magyar nyelvet jól kiszolgáló modelljével, amelyeket a Chroma vektortároló rendkívül jól kezel. Véleményem szerint kellően jól támogatja az összetett tartománykereséseket és vektor attribútumok kombinálását, miközben a metadatokkal optimalizált eredményeket nyújt gyorsan és pontosan. A választás indoklását alátámasztó érvek a rendszer komplexebb bemutatását igénylik, így később további érvek olvashatóak.

## 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 (Tai *et al.*, 2025). 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 (Gehman *et al.*, 2020).

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. Ez súlyos problémákat okozhat olyan kritikus területeken, mint az egészségügy vagy a jogrendszer információellátása (Tai *et al.*, 2025).

További etikai dilemmát jelent a nagy modellek betanítására használt óriási adathalmazok eredete, tisztasága és a velük kapcsolatos szerzői jogi kérdések. Az adatvédelmi aggályok is felmerülnek, különösképpen személyes információk modellbe kerülésével kapcsolatban. Ilyen aggályok alapján például Olaszországban egy időre betiltásra került az OpenAI ChatGPT szolgáltatása (Euronews, 2023).

A nagy nyelvi modellek biztonsági aggályai 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(Goodfellow, Shlens and Szegedy, 2015). Egy rosszindulatú támadás során például egy STOP táblát olyan módon lehet módosítani, hogy azt a rendszer tévesen "Szabad az út" jelzésként értelmezze, ami súlyos közlekedésbiztonsági kockázatokat jelenthet például az önvezető járművek esetében. 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 (Owasp, 2025). 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.

## Az internetes dokumentumalapú kérdezési funkciók és a RAG rendszerek kapcsolata

A felhasználók számára már több vállalat is nyújt generatív AI szolgáltatásokat. Ezek többnyire kombinált szolgáltatások, melyek között több funkció is megjelenik, mint például a képgenerálás, képek elemzése. A legszélesebb körben ismertek a chatbot alkalmazások piaci részesedése alább látható.

1. ábra Generatív AI chatbotok piaci részesedései 2025 áprilisában (Bailyn, 2025)

Mint az látható az 1. ábrán, a ChatGPT szolgáltatása rendelkezik a legnagyobb piaci részesedéssel. ChatGPT szolgáltatásai közé tartozik egy olyan funkció, ami a felhasználó számára PDF, illetve más formátumú szöveges fájlok feltöltését teszi lehetővé, és ez alapján a chatbot képes a dokumentum alapján válaszolni (OpenAI, 2025). A fentiek miatt érdemes összehasonlítani a ChatGPT által használt technológiát (ami a legtöbb esetben megegyezik más chatbot szolgáltatásokkal az alapelveket tekintve) és a RAG rendszert, mivel más módszerrel történik a felhasznált dokumentum alapon történő válaszadás.

Fontos megérteni, hogy a ChatGPT (GPT-4 modell) nem rendelkezik „hosszú távú emlékezettel” a feltöltött dokumentumokat illetően. A modell a beszélgetés alkalmával kizárólag az aktuális kontextusablakban lévő üzenetekből dolgozik, és nincs tartós belső memóriája az azon kívüli információkra (Pena, 2024).

A gyakorlatban ez azt jelenti, hogy amikor egy PDF szöveget feltöltünk (vagy bemásoljuk a tartalmát), akkor csak addig „emlékezik” a dokumentumra, ameddig az a beszélgetés része. Minden új felhasználói üzenet esetén a ChatGPT az addigi beszélgetés előzményeit (beleértve a dokumentum tartalmát is) figyelembe veszi a kontextusablakán belül, de ha a párbeszéd túlságosan elhúzódik vagy a dokumentum szövege túllépi a modell maximális kontextusméretét, akkor a korábbi részek kiszorulhatnak a kontextusablakból. A kontextusablak mérete az egyszerre kezelhető tokenek számát jelenti(Pena, 2024). A GPT-4 esetében a kibővített modellverziók kontextusablaka akár 8 192 vagy 32 768 token is lehet (Sharma, 2025). Egy token általánosságban 4 betűt jelent, ami alapján durva becsléssel az mondható, hogy egy szó hozzávetőlegesen 1,5 token. Ez azt jelenti, hogy a kibővített kontextusablakkal rendelkező modell nagyjából 21845 szót tartalmaz. Ezen az oldalon 391 szó található. Ezek alapján általánosítva egy átlagos oldalon nagyjából 300-500 szó lehet, ami alapján kiszámolható, hogy mindez 44-73 oldalt jelent a szavak számától, oldalak elrendezésétől függően. Ez lehetővé teszi, hogy egy rövidebb tanulmány vagy dokumentum teljes egésze egyszerre a modell figyelmében legyen. Mindazonáltal egy hosszabb dokumentum (például egy 100 oldalas PDF) teljes tartalma még a 32 ezer tokenes GPT-4 esetén sem fér be egyszerre a kontextusba. Ilyenkor vagy össze kell foglalni a dokumentumot, vagy szakaszokra bontva, több körben kell feldolgozni, majd a külön körökből származó eredményekre ismételten egy hívást kell alkalmazni, ami összegzi az eredményeket. Bár ezen kiegészítő lépések használatosak, de érezhető, hogy ezek a folyamatok belsőleg több interakciót igényelnek, összetettebbé válik a folyamat.

Amikor egy PDF dokumentumot egyszerre adunk be, azt nevezi a szakirodalom contect stuffing módszernek. Ez azonban nem minden esetben jó megoldás. Kutatások rámutattak, hogy a túl hosszú, nagyrészt irreleváns szöveg egyetlen kontextusként való átadása csökkenti a válasz minőségét és növeli a hallucináció esélyét (Catav, 2023). Emellett nem szabad elfelejteni, hogy a nagy mennyiségű szövegek bevitele költséges is, mivel a nyelvi modell működése tokenalapon számolható, és a több token feldolgozása arányosan drágább. A legjobb eredményt az LLM-ek jellemzően akkor érik el, ha csak a legrélevánsabb dokumentumrészletek kerülnek a kontextusba, nem pedig az összes nyers adat (Catav, 2023). Ez tehát azt jelenti, hogy amikor egy nagyon hosszú szöveget adunk be, a modell nem mindig találja meg benne könnyen a választ, vagy figyelmen kívül hagy bizonyos részeket, elveszik a részletekben, ami a válasz pontosságának valószínűségét rontja. Amennyiben ez történik, és nem találja meg a helyes választ, úgy a ChatGPT hajlamos lehet magabiztosan választ adni ekkor is. Ilyenkor a saját tréningadatbázisára támaszkodva, a modell tanítása során keletkezett ismeretek alapján kitalálhat egy választ, ami téves is lehet. Az ilyen hallucinációk kockázata miatt több lépéses ellenőrzési folyamatot lehet integrálni, ami minden egyes beszélgetésnél lefut, de ez meglehetősen költséges megoldás.

Az is problémát jelent az ilyen rendszereknél, hogy a beszélgetések között nincs áthidalható memória a PDF-ek számára. Hiába került be egyszer egy dokumentum a rendszerbe, az több szálon, több beszélgetésben nem felhasználható. Bár a ChatGPT funkciói között létezik egy memória nevű tárhely, ami a beszélgetések kisebb, relevánsabb részeit tartalmazza (például a felhasználó neve), de a PDF-ek tárolására ez messze nem alkalmas. Minden egyes beszélgetés izolált ilyen szempontból. Ez nem hiányosságként értelmezendő, mivel ez egy szándékos adatbiztonsági döntés eredménye (OpenAI, 2025).

### ChatGPT szolgáltatás és a RAG rendszerek összehasonlítása

A RAG rendszer tulajdonságai a dolgozatban részletesen bemutatásra kerülnek, de a ChatGPT-s PDF feldolgozással szemben kiemelendő, hogy a RAG rendszerek skálázhatóak nagy adathalmazokra, és nem korlátozódnak le a kontextusablakok méreteire a felhasználható dokumentumok. A dokumentumok teljes gyűjteménye indexelve van egy vektortárban, így szinte korlátlan mennyiségű szöveg között lehet keresni. A lekérdezés pillanatában csak a legrelevánsabb néhány szövegrész kerül át a modellhez, ami hatékonyan kihasználja a modell kontextusát és csökkenti a fölösleges tokenek miatti költséget, a túlzsúfolt kontextusablak miatti bizonytalanságot (Catav, 2023). A modellnek nem kell a saját tanítási tudását használnia az információk visszagenerálásához, hanem rendelkezésére áll egy megbízható külső adatforrás, ahonnan a válaszok átemelhetőek. A ChatGPT megoldásával szemben a dokumentumok vektortára egyfajta külső memóriaként működik, amely megmarad a beszélgetések lezártával is. Így a felhasználó bármikor új kérdést tehet fel a korábban betöltött PDF-ekre vonatkozóan anélkül, hogy újból be kellene adnia a teljes dokumentumot. Hátrányként azonban kiemelendő, hogy a szövegek feldolgozása, vektorizálása összetett folyamat, ami nagyobb erőforrásokat igényel a ChatGPT-s megoldásnál, azonban ez csak akkor hátrány, ha egy kisebb méretű dokumentumot csak egyszer akarnánk használni. A vektortárolóban ezek a dokumentumok mivel megmaradnak, ezért hosszabb távon ez költséghatékonyabb megoldás. A rendszer továbbá érzékeny lehet olyan problémára, ami azt jelenti, hogy a szükséges információ eltárolásra került a vektortárolóban, mégsem találja meg. Mindenki számára ismerős a jelenség, amikor az emberi kommunikáció alkalmával mindkét fél alapvetően birtokában van az ismereteknek, a rossz nyelvi kifejezések miatt mégsem tudják beazonosítani az üzenet tárgyát. Második próbálkozásként azonban máshogyan megfogalmazva a kérdést, már a kérdés érthetővé válik. Ez a jelenség előfordulhat a RAG rendszereknél is. Éppen ezért a felhasználónak érdeke, hogy minél tisztábban és pontosabban fogalmazza meg a kérdést. A dolgozat témáját alkotó elkészített rendszer erre a hibára több megoldást is használ. Használ egy history tárolót, ami alapján a kérdés kontextusát az előző kérdések alapján képes jobban megérteni és biztosabb választ adni, valamint amennyiben csak nagyon gyenge minőségű válaszok szerepelnek a kikeresett információdarabok között, úgy a rendszer nem ad forrásokat, helyette felhívja a figyelmet arra, hogy konkrétabb, jobban körülírt formában kell feltenni a kérdést, mert az túl általános. Extra erős pontatlanság mellett a hallucinációk elkerülése érdekében pedig egyszerűen a rendszer nem enged választ adni, és közli, hogy nem tudja a választ. Ez a típusú önkorlátozás egyaránt erény, mert növeli a hitelességet, de korlát is, mert a modell kevésbé “kreatív”, nem egészíti ki a hiányokat olyan információkkal, amiket esetleg maga a nyelvi modell ismerne.

A RAG rendszer hátránya, hogy minden egyes kérdés megválaszolása előtt a rendszernek a vektoros keresést le kell futtatnia, majd ki kell választania a leginkább a kérdéshez illeszkedő válaszdarabokat. Az én rendszeremben a hitelesség növelése érdekében extra ellenőrzési folyamat is lefut ilyenkor, amiről később lesz szó. A RAG rendszer ezen lépései mindenképp addicionális lépéseknek számítanak az egyszerű generatív alapon történő megoldáshoz képest, így mindenképp hosszabb lesz a válaszidő. Ez könnyen belátható, hiszen rendszer önmaga tartalmazza a generatív komponenst. Ezek a műveletek a futtatókörnyezet függvényében változnak, de általánosságban elmondható, hogy gyorsak, néhány másodpercet vesznek igénybe (Stepkurniawan, 2024). Ezek mellett az is nyereség az időt tekintve, hogy a generatív modellnek a válaszok megfogalmazásánál nem kell egész dokumentumnyi tokenen dolgoznia, így a tokenek mennyisége általában kevesebb.

Felhasználói oldalról az mondható el összességében, hogy kisebb dokumentumok esetében a ChatGPT használata kielégítő lehet, azonban nagy mennyiségű szövegekben történő keresés esetében bonyolultabb megoldásokat, hosszabb pipelineokat érdemes alkalmazni a dokumentumok feldolgozására. Ezekben az esetekben előnyösebb a RAG technológia. A dolgozat írásakor még nem volt elérhető, azonban szeparált módon a RAG technológia hivatalos integrálása az OpenAI részéről egy jó fejlesztési irány lehet.

### Egyéb források használata

A RAG technológia némiképp már megjelent a széles körben alkalmazott online szolgáltatások között. A Perplexity AI a keresőmotorjában egy RAG rendszerhez hasonló technológiát alkalmaz (xfunnel.ai, 2024)

A rendszer szabad szöveges kérdések által képes hiteles források felhasználásával válaszolni a felhasználónak. Azon alapul a működése, hogy egy böngészőmotor internetes forrásokat gyűjt, majd azokat egy gyors RAG-ra épülő rendszerbe betölti, amiből az LLM a választ megfogalmazza. A rendszere azonban nem teszi lehetővé, hogy saját, ellenőrzött dokumentumokra épülő adatbázisból történjen a keresés. Az ellenőrizhetőség pedig egy kritikus szempont jelen esetben.

Ez a technológia lehetőséget ad az összetettebb keresésekre és a válaszok pontosságát is nagyban javítja, azonban számos kockázatot is rejt magában. A jelenleg elérhető elemzések szerint a Perplexity rendszere gyakran hibás, vagy nem pontosan azonosítható forrásokat adott vissza válaszként, holott a RAG rendszereknek ez az alapvető feladatuk. A tanulmány szerint majdnem 40 százalékban helytelen forráshivatkozások szerepeltek, vagy nem tükrözte a válasz megfelelően a forrásban szereplő információkat (Quesada, 2025). A pontatlan idézetek és a rendszer által nagy magabiztossággal megfogalmazott válaszok nagy kockázatot rejtenek magukban. Az említett cikk összesen 1600 lekérdezést végzett, nyolc chatbot szolgáltatásnál a hitelességet vizsgálva. Az eredmények alapján a válaszok 60 százalékban helytelennek bizonyultak, bár a hibaarány platformonként eltérő volt. A legalacsonyabb hibaarányt a perplexity érte el az említett aránnyal, míg a legrosszabb eredményt az Elon Musk vállalata által fejlesztett Grok3 produkálta 94 százalékos hibaaránnyal (Quesada, 2025).

A rendszer használatával továbbá szerzői jogi aggályok is felmerültek a Reuters beszámolója szerint a források rossz kezelése kapcsán (Chmielewski and Paul, 2024).

Az ilyen keresőmotorokra épített rendszerek sajnos nem tudják garantálni a hitelességet még tökéletes technológiai kivitelezés esetén sem, mivel az interneten való keresés során nem tudják garantálni, hogy az ott leírt információk helyesek. Egy zárt, ellenőrzött tudományos adatbázison alapuló, pontosan kivitelezett RAG rendszer esetében, ahol a dokumentumok megbízható forrásokból származnak, és azok megfelelő mélységben vannak feldolgozva, ilyen aggályok jellemzően nem merülnek fel.

# Módszertan

## Tervezési szempontok és követelmények

A rendszer tervezése során több szempontot és követelményt vettem figyelembe annak érdekében, hogy a végleges megoldás hatékonyan és megbízhatóan működjön. Korábban már szó esett részben alábbi pontokról, amelyek most részletesen bemutatják azokat a területeket, amikre a tervezés során nagy figyelmet kellett helyezni.

**Természetes nyelvű interakció támogatása**

Elsődleges célom volt, hogy a rendszer képes legyen a felhasználók által megfogalmazott kérdések megértésére és megválaszolására. Ez nagy jelentőséggel bír a magyar nyelv esetében, mivel agglutináló jellege miatt a szavak számos toldalékot kaphatnak, amelyek megváltoztatják azok jelentését vagy mondatbeli szerepét. Emellett a magyar nyelv szabad szórendje miatt ugyanaz a mondanivaló többféleképpen is kifejezhető, ami tovább nehezíti a nyelv feldolgozását és megértését a programok számára. (Rehm and Uszkoreit, 2012). Ennek érdekében olyan nyelvi modelleket kellett integrálnom, amelyek megfelelően kezelik ezeket a sajátosságokat.

**Folyamatos párbeszéd érzetének biztosítása**

A fejlesztés során nagy figyelmet fordítottam arra, hogy rendszer a lehető legtermészetesebben kommunikáljon. Ennek érdekében implementáltam egy előzménykövető- history komponenst, amely tárolja a beszélgetések során elhangzott kérdéseket és válaszokat. Ez lehetővé teszi, hogy a rendszer az új felhasználói inputok feldolgozásakor figyelembe vegye a párbeszéd addigi kontextusát.

A tesztelések során megfigyeltem egy nagyon fontos összefüggést: az előzménytárolásból ki kellett zárnom a pontos forrásokra mutató hivatkozásokat. Egyszerűen fogalmazva a figyelmét el lehetne terelni a rendszernek különféle súlyozásokkal a korábbi hivatkozásoknál, azonban ez a megközelítés nem eredményes. Amikor ugyanis a dokumentumcímeket és oldalszámokat is eltároltam, a RAG rendszer túlzottan ezekhez a konkrét forráshelyekhez kezdett kötődni. Ennek következményeként az új kérdésekre is ugyanazokban a dokumentumrészekben próbált választ keresni. Minimális súlyozás mellett is hajlamosító tényezőként hat a problémára a pontos hivatkozás megtartása, ami nem helyes.

Ez a viselkedés ellentétes volt a célkitűzéseimmel, hiszen a rendszernek kizárólag a szemantikai kontextust kellett volna megőriznie. A túlspecifikus forrásinformációk tárolása beszűkítette a keresési teret, és a rendszer egy-egy dokumentumszegmensre korlátozta magát, ahelyett hogy rugalmasan kezelte volna a teljes tudásbázist. A válaszként érkezett információk eltárolása tematikusan szűkítette a keresési teret, nem pedig dokumentumspecifikusan, ami a szándékaimmal megegyező volt. A módosított architektúrában a rendszer már csak a beszélgetés nyelvi elemeit őrzi meg, így biztosítva a továbbra is megfelelő információ-visszakeresést.

**Magas szintű nyelvi megértés és válaszminőség**

A kutatás egyik legfontosabb pontja volt a természetes nyelvű kérdések feldolgozásának és a válaszgenerálásnak az optimalizálása. A vizsgálat több nagyméretű nyelvi modellre terjedt ki, beleértve az OpenAI GPT családját, a Meta LLAMA modelljeit és a Mistral változatait. Az elemzés fókuszában a modellek kérdésértelmezési képessége és válaszgenerálási minősége állt.

A később tárgyalt tapasztalatok igazolták, hogy a modern nyelvi modellek integrációja számottevően növeli a RAG rendszer pontosságát. A modellek képesek a kérdések szemantikai struktúrájának azonosítására, a többértelmű megfogalmazások kezelésére, valamint kontextuálisan megfelelő válaszok előállítására. A modellszelekció során meghatározó szempontként szerepelt a szemantikai értelmezés precizitása, a generált válaszok koherenciája és a rendszer válaszideje.

A részletes teljesítményméréseket és a modellek összehasonlító elemzését a dolgozat következő fejezetében mutatom be. Ott térek ki arra is, hogy a különböző modellek erősségei és gyengeségei hogyan befolyásolták a végső architektúra kialakítását.

**Retrieval-Augmented Generation (RAG) használata**

A RAG technológia alkalmazásának fontosságáról és lényegéről korábban már esett szó. Ennek megvalósításához szükség volt egy megfelelő RAG-ot támogató keretrendszerhez, amely képes kezelni a tematikusan kapcsolódó dokumentumok gyors és pontos azonosítását a felhasználói kérdések alapján.

**Hatékony vektortárolási megoldás**

A dokumentumok beágyazásait gyorsan és hatékonyan kellett kezelni, ezért a vektortárolók kiválasztásánál a sebességre, a könnyű integrációra és a skálázhatóságra összpontosítottam. Korábban már esett némi szó általánosságban a vektortárolókról, azonban a továbbiakban bővebben kifejtésre kerül a választás indoklása, ami bemutatja, hogy miért a Chroma vektortároló mellett döntöttem, miért illeszkedik rendkívül jól a projekt igényeihez a Chroma funkcionalitása.

**Nyelvi konzisztencia és többnyelvű támogatás**

A rendszernek biztosítania kellett a nyelvi konzisztenciát a felhasználói interakciók során. Mivel a felhasználók magyar és angol nyelven is kérdezhetnek, fontos volt a kérdés nyelvének automatikus felismerése és a válaszok megfelelő nyelven történő generálása. Ez hozzájárul a jobb felhasználói élményhez és a rendszer rugalmasságához.

**Felhasználói élmény és interfész**

Mindamellett, hogy a kérdés nyelvének megfelelően kell a választ megadnia a rendszernek, azt is vizsgálnia kell, hogy a kérdés a dokumentumokból szándékozik-e információkat kinyerni, vagy egyszerűen társalgási kategóriába eső kérdésről van szó. Ennek felismerése esetén ugyanis felesleges volna a vektortárolóban információk keresése.

A felhasználói élmény javítása érdekében egy intuitív és felhasználóbarát interfészt kívántam kialakítani. A Gradio keretrendszerrel egy webalapú felületet terveztem, amely egyszerű interakciót biztosít a rendszerrel (Gradio, 2024).

## Technológiák és eszközök kiválasztása

A rendszerelemek kiválasztásánál több szempontot vizsgáltam, elsősorban a teljesítményjellemzőkre, a bővíthetőségi mutatókra és az erőforrásigényekre. A végleges felépítés tervezésekor elemeztem a várható rendszerterhelést, a válaszidőre vonatkozó követelményeket, valamint a jövőbeli fejlesztési lehetőségeket. A következő szakaszban részletesen bemutatom az alkalmazott technológiai megoldásokat, ezek rendszeren belüli funkcióit, és áttekintem a kiválasztási folyamat során mérlegelt szempontokat. A komponensek integrálásának technikai részleteit és a megvalósítás során felmerült problémákat a dolgozat későbbi fejezete tárgyalja.

### Modellek

A kezdeti fejlesztések során a Mistral 7B modell és a Meta Llama 2 változatait alkalmaztam. A Mistral 7B egy 7,3 milliárd paraméteres nyelvi modell, amelyet a Mistral AI fejlesztett ki. Teljesítménye felülmúlja a Meta által készített Llama 2 13B modellt számos benchmark teszten, és bizonyos esetekben a Llama 1 34B modellt is meghaladja (Mistral, 2023).

Az előbb említett Llama 2 pedig a Meta AI által fejlesztett, többnyelvű támogatással rendelkező modell, amely alapvetően kielégítően jó teljesítményt nyújtott több természetes nyelvfeldolgozási feladatban (Touvron et al., 2023).

Bár a Mistral 7B és a Llama 2 modellek többnyelvű támogatással rendelkeznek, a magyar nyelv feldolgozásában a Mistral igen korlátozott teljesítményt mutatott. A kezdeti megoldások során azonban a kontextusablak méretei megkívánták a Mistral 7B alkalmazását annak ellenére is, hogy a magyar nyelv ragozásos jellege és összetett grammatikai szerkezete nehézségeket okozott a modell számára. Így tehát ez egy kompromisszumos megoldás volt a számomra elérhető modellek között. A generált válaszok gyakran magyar nyelven hiányosak vagy pontatlanok voltak, ami rontotta a felhasználói élményt. Sok esetben a ragozás teljesen értelmetlen volt, és komoly nyelvtani hibákkal válaszolt a rendszer a magyar kérdésekre. A modellek gyenge magyar nyelvi képességei miatt egy külön fordítási lépést kellett beiktatnom, amely a felhasználói kérdéseket angolra fordította, és a válaszokat vissza magyarra. Ehhez egy külön fordítási modult integráltam a rendszerbe a Meta Llama 2 modell felhasználásával, ami viszont jól használta a magyar nyelvet.

A fordítási lépés beiktatása növelte a rendszer összetettségét, ami több hibalehetőséget és karbantartási nehézséget eredményezett, valamint a további feldolgozási lépések miatt a rendszer válaszideje megnőtt. A fordítási folyamat során ráadásul olykor jelentős információvesztés és pontatlanság lépett fel, ami a végső válaszok minőségét rontotta.

A fenti korlátok felismerése, és újabb modellek megjelenése után érdemes volt megvizsgálni az OpenAI GPT-4 modelleket. Ezek a modellek nagy előrelépést jelentenek, érezhetően jobb teljesítményt nyújtanak több nyelven, beleértve a magyart. Nagyobb kontextusablakuk révén (akár 8192 token) képesek hosszabb és összetettebb párbeszédek kezelésére (OpenAI, 2023). Összességében tehát a generált válaszok minősége lényegesen javul ezen modell alkalmazásával. Az kontextusablak méretéből eredő fordítási lépések immáron nem voltak szükségesek. A rendszert a GPT modellekre építve terveztem, de olyan módon, hogy a GPT modellek frissítése újabb GPT modellekre ne okozzon nehézséget a jövőben.

Fontos megemlíteni, hogy nem kizárólag ez a nyelvi modell végez feladatokat a rendszer működése során. A dokumentumok feldolgozásakor a szövegek beágyazására egy külön, más típusú modellt kell alkalmazni. Több különböző modell került kipróbálásra a végső választás előtt. A text-embedding-ada-002 modell (openai, 2022) alacsony dimenziós vektorokat állít elő, de a magyar szövegek esetén a vektortérbeli eloszlása érezhetően sokszor nem tükrözte jól a szemantikai hasonlóságot, így a hasonló szegmensek vektorai nem kerültek kellően közel egymáshoz. A distiluse-base-multilingual-cased-v2 embedding modell (huggingface, 2019) többnyelvű támogatással rendelkezik, de hosszabb magyar mondatok és bonyolultabb bekezdések esetében nem alkotott kielégítően jól vektorreprezentációkat. A Meta LASER embeddings megoldása (LangChain, 2024) hasonló problémát mutatott, főleg az elméleti és a később tárgyalt archaikus stílusú szövegek esetén, ahol az absztrakt fogalmak közötti kapcsolatok vektortérbeli leképezése kifejezetten gyenge volt. A népszerű modellek egyszerű keresgélése helyett HuggingFace Massive Text Embedding Benchmark interaktív összehasonlításához jó reményekkel fordultam a fejlesztés során. A megfelelő szűrési opciók beállítása után a legelőkelőbb helyen a Voyage AI legújabb embedding modellje állt a keresett időpontban (huggingface, 2025). Ezen modellt kipróbálva, és a rendszert erre optimalizálva (batch méretek konfigurálása, LangChain integrációs hiányosságainak áthidalása) az eredmények meglepően jók voltak. A tesztelt modellek közül azonban a text-embedding-3-large adta a legkifejezőbb és legjobban szeparálható vektorokat magyar nyelvű dokumentumok feldolgozásához, így ezt használtam.

### Vektortárolók

A vektortárolók bemutatásáról és összehasonlításáról korábban némiképp esett szó, azonban a feladatspecifikus legfontosabb érv alább olvasható.

Mint ahogyan korábban, a vektortárolók bemutatásánál is szó volt róla, kezdetben a FAISS (Facebook AI Similarity Search) rendszer alkalmazása tűnt a legígéretesebbnek. A FAISS jól kezeli a nagy mennyiségű beágyazást, és gyors keresést tesz lehetővé (MyScale, 2024). A FAISS integrálása bonyolultabb, és nem támogatja natívan a metaadatok kezelését, ami fontos a dokumentumokhoz kapcsolódó információk tárolásához. Ez egy nagy hátránya ennek a technológiának. Ezeket a metaadatokat egy külön elkészített adatstruktúrában kellett így tárolni. Erre több megoldás is kínálkozik, például adatbázisban, JSON-ban, de én egyszerűen egy Python-szótárban kezeltem ezeket. A visszakapott indexek alapján lehet lekérdezni ilyen módon a megfelelő metaadatokat. A probléma ebben az esetben az, hogy egy dokumentumhoz több metaadat is kapcsolódhat, aminek a kezelése erőforrás-, ezáltal időigényes folyamat. Ezért áttértem a Chroma vektortároló használatára, amely könnyebben integrálható a LangChain keretrendszerrel (Amikos Tech LTD, 2024), és támogatja a metaadatok kezelését. Utóbbi funkció adta lehetőségek miatt beépítettem egy funkciót, ami egy szöveges fájlban a felhasználónak pontosan listázza az éppen betöltött dokumentumok címeit, ezzel aktuális képet adva a szöveges erőforrásokról. Ez elsősorban azért fontos, mert a PDF dokumentumok címei nem minden esetben egyeznek meg a dokumentumok címeivel, az általam készített rendszer pedig automatikusan kinyeri a feldolgozás során ezeket a metainformációkat.

### A válasz nyelvi korlátjainak feloldása

Az OpenAI GPT modellekre való áttérés megszünteti a fordítási lépés szükségességét. Az új GPT modellek kiváló magyar nyelvi képességei révén a rendszer közvetlenül képes a felhasználói kérdések megértésére és a megfelelő nyelvű válaszok generálására. A fordítási modul eltávolításával a rendszer egyértelműen egyszerűbbé és könnyebben karbantarthatóvá válik. A feldolgozási lépések számának csökkenése javítja a rendszer „reakcióidejét”, valamint a közvetlen feldolgozás révén a válaszok természetesebbé és pontosabbá válnak. Amikor a dokumentum nyelve és a kérdés nyelve nem egyezik meg, akkor fordítás is a GPT modellre van bízva, de ezt egy lépésben teszi meg, és nincs külön célnyelvre, majd a válasznyelvre való külön fordítási lépés. Ennek kiértékelése később olvasható. Az egyszerűsítési lépések miatt azonban továbbra is szükséges volt a stabil működés miatt a kérdés, és így a szükséges válasz nyelvének aktív detektálása, amit egy külön modul lát el.

## A LangChain keretrendszer

A tudásalapú rendszerek és információ-visszakeresés terén a LangChain egy fontos keretrendszer, amely jól integrálja az LLM-eket, adatbázisokat és más komponenseket AI-vezérelt alkalmazásokba. A Chroma natívan kezeli a LangChain bizonyos funkcióit.

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. A LangChain modularitásának köszönhetően a fejlesztők egyszerűen kombinálhatják és cserélhetik az egyes összetevőket, így az alkalmazások rugalmasabbá és könnyebben skálázhatóvá válnak (Laangchain, 2024).

A LangChain keretrendszer szerepe nem az egyes RAG komponensek (például embedding készítés, dokumentumbetöltés vagy vektortárolás) közvetlen megvalósítása, hanem azok hatékony összekapcsolása és menedzselése. Adaptereket biztosít különböző LLM-ekhez, embedding modellekhez, dokumentum-betöltőkhöz, szövegdarabolókhoz és vektortárolókhoz, mint például a Chroma vagy a FAISS.  
A RAG folyamat során a következő lépésekben van szerepe (Laangchain, 2024):

1. Dokumentumbetöltés: A dokumentumok betöltése dedikált betöltőosztályokkal (pl. PyPDFLoader), amiket a LangChain támogat, de maga a betöltés nem a LangChain alapfunkciója.
2. Szövegfeldarabolás: A szövegek feldarabolása például a RecursiveCharacterTextSplitter vagy más splitter segítségével, amelyeket a LangChain integrál, de a feldarabolás logikája szintén különálló komponens.
3. Beágyazás (embedding): Az embedding modellek (pl. OpenAI text-embedding-3-large, Hugging Face modellek stb.) használatát a LangChain megkönnyíti adapterekkel, de az embeddinget maga az adott modell végzi.
4. Vektortárolás: A Chroma, FAISS, Weaviate, Pinecone és más vektortárolók integrációját támogatja a LangChain, de ezek működése szintén nem a LangChain része.
5. Lekérdezés és válaszgenerálás: A LangChain biztosítja a láncolási logikát, például a ConversationalRetrievalChain vagy más chain-ek segítségével, amelyek koordinálják a lekérdezés, a visszakeresés és a válaszgenerálás lépéseit.

Látható, hogy ezek majdnem pontosan lefedik legalapvetőbb lépéseket a RAG folyamatban, ezért a Langchain egy nagyon fontos kiegészítő eleme a lépések közötti koordinációnak.

Ez az ilyen módon jól összekapcsolt folyamat lehetővé teszi a rendszer számára, hogy gazdag, kontextusfüggő válaszokat adjon a felhasználói kérdésekre. A LangChain rugalmassága miatt ez a folyamat testre szabható és finomhangolható a különböző alkalmazási igényeknek megfelelően.

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.

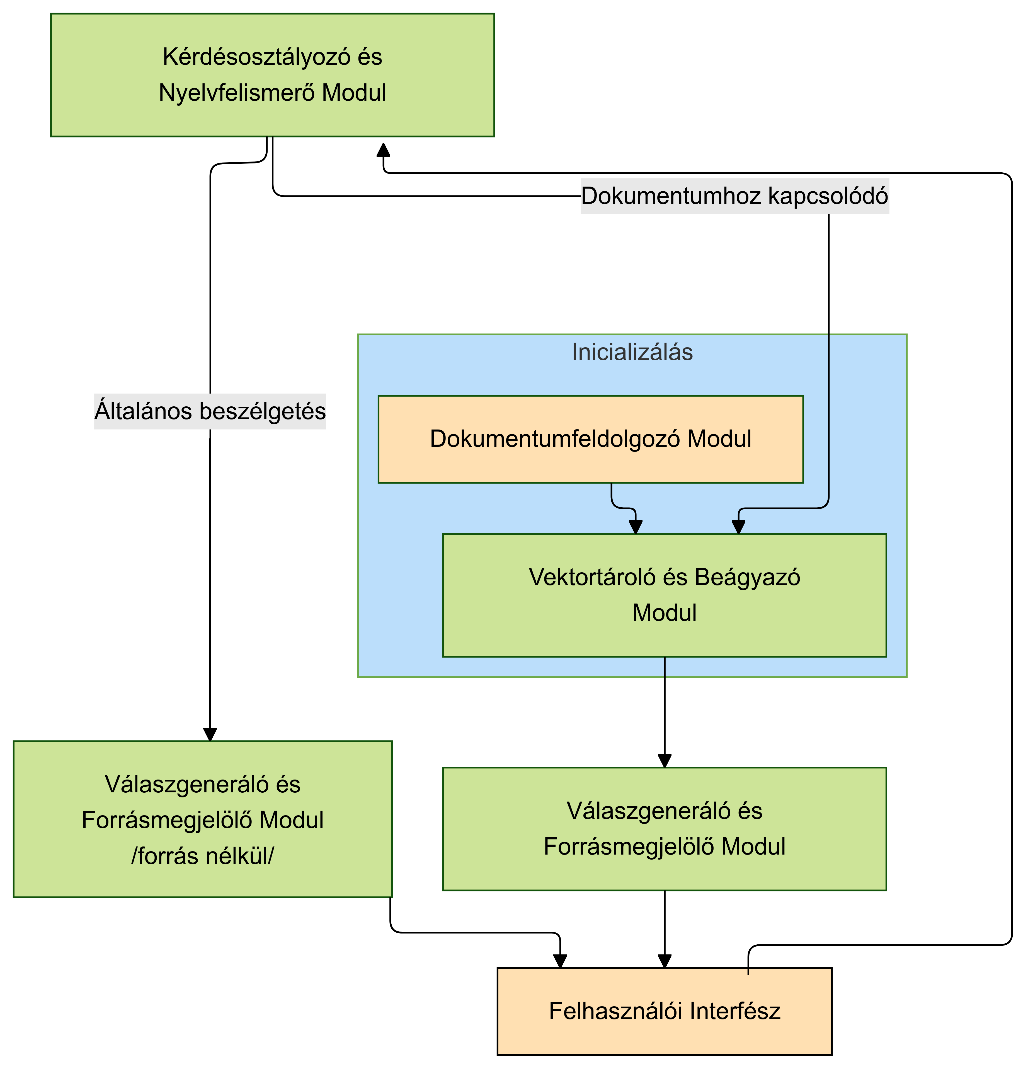
## Rendszerarchitektúra

A rendszer architektúrája komponens-alapú, és több fő komponensből áll, amelyek együttműködése biztosítja a funkcionális követelmények teljesítését. Ezek szorosan együttműködésben állnak, azonban jól elválaszthatóak a funkcionalitásuk szerint. Az alábbiakban bemutatom a rendszer főbb komponenseit és azok kapcsolatait.

A rendszer főbb komponense a 2. ábrán láthatóak, ezek a következők:

* **Dokumentumfeldolgozó modul:** A dokumentumok betöltéséért, előfeldolgozásáért, metaadatok kinyeréséért, és a beágyazások generálásáért felelős egy OpenAI embedding model segítségével.
* **Vektortároló és beágyazó modul:** A dokumentumok beágyazásait és metaadatait tárolja, és előkészíti az információ-visszakeresést.
* **Kérdésosztályozó és nyelvfelismerő modul:** Meghatározza a felhasználói kérdések nyelvét és típusát (dokumentumhoz kapcsolódó vagy általános beszélgetés).
* **Válaszgeneráló és forrásmegjelölő modul:** A felhasználói kérdések feldolgozását és a kérdésekre adott válaszokat állítja elő, szükség esetén a témához kapcsolódó dokumentumok alapján. Ennek a modulnak része a források pontos megjelölésének megkeresése és azok hozzáfűzése is, továbbá feladata a beszélgetési előzmények kezelése is.
* **Felhasználói interfész (Gradio) modul:** Böngészőből elérhető felület, amely biztosítja a felhasználók számára a rendszerrel való interakciót.

A fenti modulok részletes bemutatása később olvasható, a rendszer magas szintű blokkdiagramját az alábbi ábra szemlélteti.



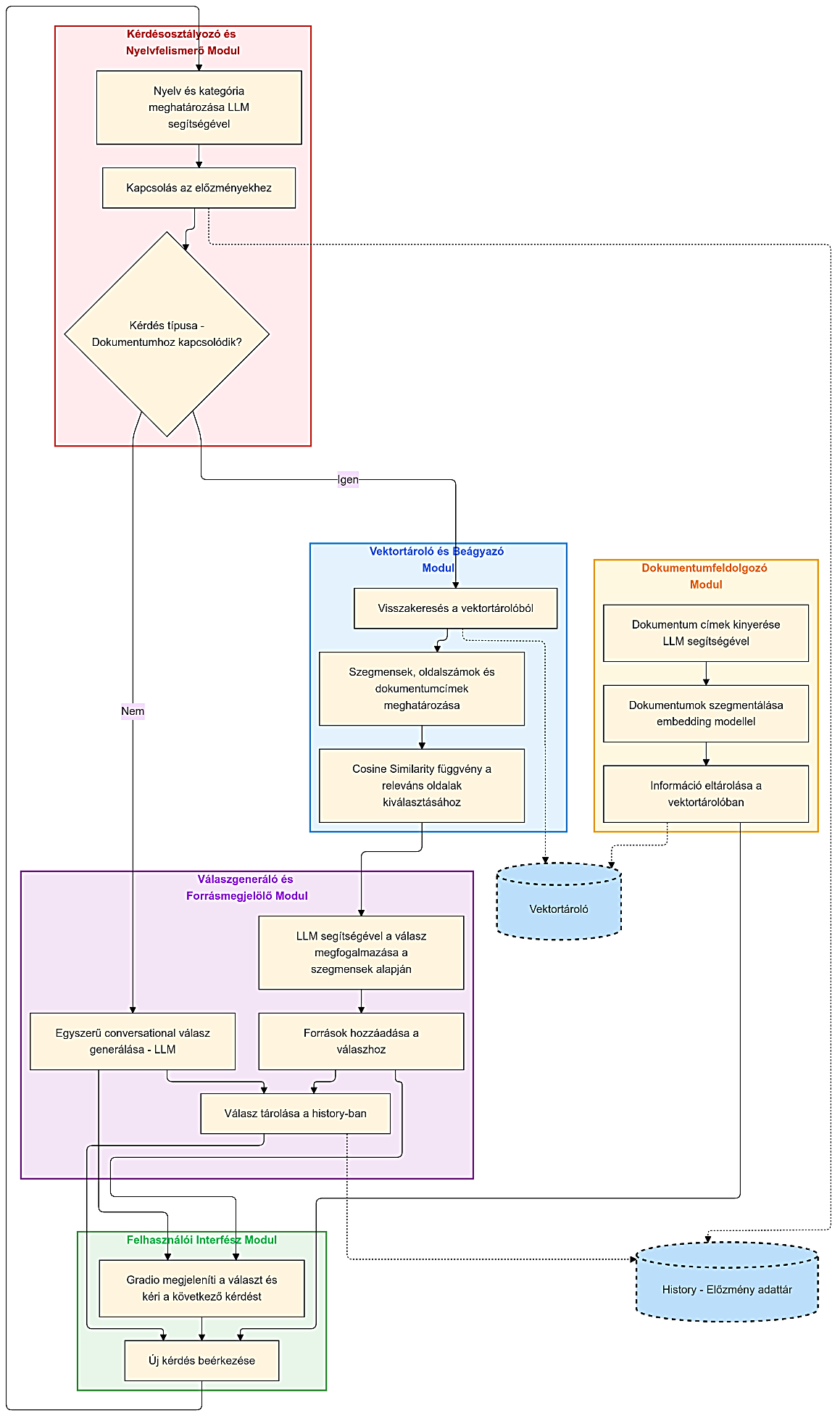
2. ábra Magas szintű blokkdiagram

A magas szintű blokkdiagram célja, hogy átfogó képet nyújtson a rendszer felépítéséről és az egyes modulok közötti kapcsolódási pontokról. A diagramot követően egy táblázatos ismertetés olvasható az egyes modulok működéséről.

A rendszer működésének részletesebb ábrázolása érdekében egy kiegészítő blokkdiagram is szerepel, amely a modulok közötti interakciókat és a rendszer működésének logikai összefüggéseit szemlélteti. Ez a részletes diagram pontosabb betekintést nyújt a rendszer technikai felépítésébe és folyamataiba.

2. táblázat Modulok feladatainak táblázatos bemutatása

|  |  |
| --- | --- |
| **Modul neve** | **Funkciók és feladatok** |
| **Dokumentumfeldolgozó Modul** | Ez a komponens felelős a dokumentumok előfeldolgozásáért, hogy azok használható formában álljanak rendelkezésre a rendszer számára. |
| A PDF formátumú dokumentumok betöltése, a szöveges tartalom kinyerése képek és ábrák eltávolításával. |
| A dokumentumok címének automatikus kinyerése LLM segítségével. |
| A szöveges tartalom szegmentálása kisebb egységekre a további feldolgozás érdekében. |
| Beágyazások (embedding) létrehozása a szegmensekhez, metaadatokkal (pl. dokumentumcím, oldalszám) kiegészítve. |
| A generált beágyazások és metaadatok eltárolása egy vektortárolóban (pl. Chroma). |
| **Felhasználói Interfész Modul** | A komponens a felhasználói interakciókat kezeli, biztosítva az információk be és kimenetét a rendszer számára. |
| A felhasználói kérdés fogadása a Gradio felületen keresztül. |
| Az új kérdés továbbítása a megfelelő feldolgozó modulokhoz. |
| A generált válasz megjelenítése a felhasználói felületen. |
| Az új kérdések feldolgozásának indítása, biztosítva a folyamat ismétlődését. |
| **Kérdésosztályozó és Nyelvfelismerő Modul** | Ez a komponens elemzi a beérkező kérdéseket és meghatározza azok típusát, valamint a kérdés feldolgozásához szükséges nyelvi információkat. |
| A kérdés nyelvének azonosítása LLM segítségével. |
| A kérdés típusának meghatározása: dokumentumhoz kapcsolódó vagy általános beszélgetési kérdés. |
| Az előzményekkel való kapcsolódás biztosítása, hogy a kontextus folyamatos maradjon. |
| **Vektortároló és Beágyazó Modul** | Dokumentumhoz kapcsolódó kérdések esetén ez a komponens felelős a releváns szegmensek gyors és pontos visszakereséséért. |
| A dokumentumtárolóban található szegmensek visszakeresése a kérdés beágyazásának hasonlósága alapján. |
| A tematikusan kapcsolódó szegmensek kiválasztása. |
| **Válaszgeneráló és Forrásmegjelölő Modul** | A válaszok generálását és a források hivatkozásának biztosítását végzi, a dokumentumokból nyert adatok alapján. |
| Dokumentumhoz kapcsolódó kérdés esetén:  A megtalált szegmensek felhasználása a válasz generálásához LLM segítségével.  Források (dokumentumcímek, oldalszámok) hozzáadása a válaszhoz. |
| Általános kérdés esetén:  Közvetlen válaszgenerálás LLM segítségével, a beszélgetési előzmények figyelembevételével.  Az elkészült válasz tárolása a beszélgetési előzmények frissítése. |



3. ábra Részletezett blokkdiagram

A technológiai kivitelezés során Python 3.11 környezetben dolgoztam, és az alábbi keretrendszereket, könyvtárakat, komponenseket használtam fel:

**Keretrendszerek:**

* LangChain: Lehetővé teszi összetett LLM-alapú alkalmazások építését különböző komponensek segítségével (Laangchain, 2024).
* Gradio: Egyszerű webes felület létrehozásához használtam. Ez az eszköz kimondottan az AI-t használó programokhoz nagyon hasznos (Gradio, 2024).

**Felhasznált könyvtárak:**

* langchain\_community.document\_loaders.PyPDFLoader: PDF dokumentumok beolvasására és szövegének kinyerésére használtam.
* langchain.text\_splitter.RecursiveCharacterTextSplitter: A betöltött szövegek karakteralapú feldarabolására használt eszköz, amely chunk-okra osztja a dokumentumokat a könnyebb feldolgozás érdekében.
* langchain.text\_splitter.RecursiveCharacterTextSplitter: A dokumentumok szövegének felosztására.
* langchain\_community.embeddings.OpenAIEmbeddings: OpenAI modell használatával szöveg-beágyazások (embeddingek) előállítására. Így állnak elő a dokumentumrészekből leképződő numerikus reprezentánsok.
* langchain\_community.vectorstores.Chroma: A vektortároló adatbázis használatára szolgáló könyvtár. Ezáltal tárolható továbbá perzisztensen a vektoradatbázis.
* langchain\_openai.ChatOpenAI: Az OpenAI GPT-4o-mini modell adaptálásához.
* langchain.chains.ConversationalRetrievalChain: A válaszgeneráláshoz szükséges komponens. Összeköti a nyelvi modellt a vektortárolóból érkező dokumentumrészekkel.
* dotenv: Környezeti változók kezeléséhez (API kulcsok tárolására alkalmas).
* os: Operációs rendszer függvényeinek használatához. (pl. fájlok kezelése, elérési utak összeállítása, könyvtárak létrehozása és törlése)
* shutil: Szintén fájlrendszer műveletekhez használtam. Ez egy standard Python könytár.
* gradio: A webes felhasználói környezet létrehozásához, konfigurálásához használt könyvtár.

# Implementáció részletei

## Kódstruktúra és komponensek

Az alábbiakban részletesen bemutatom az egyes komponensek működését és azok kölcsönhatását.

### Dokumentumfeldolgozó modul

A dokumentumfeldolgozó modul feladata a PDF formátumú dokumentumok betöltése, szöveges tartalmuk kinyerése, előfeldolgozása és a dokumentumok címeinek azonosítása.

Munkám során a nagyobb adatbázisok bevezetése előtt a proof of concept megvalósításához három merőben különböző PDF formátumú dokumentum feldolgozását végeztem el, amelyek különböző témában, nyelven és terjedelemben íródtak. Az első dokumentum, a Grimm testvérek mesegyűjteménye, archaikus nyelvezetével és irodalmi műfajával nagyon jó példája a narratív szövegek feldolgozásának, főként a magyar nyelv gazdag ragozási rendszerére, melyek jelen esetben valóban kimagaslóan változatosak, és a szabad szórend is erőteljesen jelenik meg ebben a régi könyvben. A szövegben olyan szóhasználat fordul elő például a nevek tekintetében, például a „Hófehérke”, vagy „Riki-roki-rokka Pál” amik nem valódi nevek, így komplex feladatként álltak elő a program számára ezek helyes értelmezése (Grimm and Grimm, 1892). A dokumentum régi, de változatos nyelvezetét, a benne fellelhető metaforákat, hasonlatokat külön kihívásnak szántam. A második dokumentum tudományos szövegként strukturált, saját magyar nyelvű tanulmány. Témáját tekintve egyháztörténelmi, közgazdasági, ami azért fontos, mert egy dokumentumon belül interdiszciplináris, ritkán használt szakkifejezések jelennek meg benne. A szöveg bonyolult történelmi és gazdasági összefüggéseket tárgyal (Szladek, 2019). A harmadik dokumentum szintén tudományos műfajba tartozik, de témában jelentősen más az előzőnél. Erősen akadémiai felépítettségű szakszöveg. Adatintenzív, mutatószámok és néhány helyen ukrán és orosz mondatrészek is előfordulnak benne. Az ok-okozati kapcsolatok sok esetben számtani összefüggések (Nagy, 2019; openai, 2022). Ez a kombináció biztosítja a különböző nyelvek, nyelvi szerkezetek, stílusok és szakterületek előfordulását, miközben az ezekre történő optimalizáció biztosítja, hogy a rendszer rugalmasan és pontosan képes legyen kezelni a különféle szövegtípusokat.

A modul a következő lépéseket tartalmazza:

**PDF betöltése és szövegkivonás**: A PyPDFLoader osztályt használtam a PDF dokumentumok beolvasásához, és a szöveg kinyeréséhez. A load\_and\_split() függvény segítségével a dokumentumokat oldalakra bontottam. Document objektummá alakítottam minden oldalt, ahonnan a szövegtartalom és az oldalszám metaadat kinyerhető. Fontos, hogy a split itt még csak oldalszintű!

**Dokumentumcímek kinyerése**: A dokumentumok címének azonosítása nagyon fontos a metaadatok kezeléséhez és a források hivatkozásához. Sajnos a legtöbb általam is kezelt PDF fájlattribútumai között a dokumentum címe címmező nem, vagy nem helyesen volt kitöltve, így nem lehetett arra támaszkodni. Ehelyett egy AI megoldást alkalmaztam, amely a dokumentum első két oldalának tartalma alapján kinyeri a címet. A get\_document\_title() függvényben egy előre megírt promptot használtam (1. kódrészlet), amely a modell segítségével azonosítja a címet magyar vagy angol nyelvű dokumentumok esetén is. Ha nem található cím, akkor egy fallback megoldásként a fájlnév kerül felhasználásra.



1. Kódrészlet: Dokumentum címének kinyerése LLM segítségével

**A szöveg feldarabolása és szegmentálása**: A nagy mennyiségű szöveg strukturált feldolgozása érdekében a dokumentumokat kisebb szegmensekre kellett bontani a *RecursiveCharacterTextSplitter* segítségével. A paraméterek beállításakor a felhasznált modell által nyújtott lehetőségeken belül kellett maradnom. A chunk\_size 1000 karakter, a chunk\_overlap pedig 200 karakter lett (2. kódrészlet). Ez utóbbi azt állítja be, hogy a szomszédeos szeletek 200 karakternyi átfedésben vannak egymással. Ez az átfedés biztosítja, hogy egy-egy mondat vagy gondolat ne szakadjon szét két chunk határán, és retrieval során is növeli annak esélyét, hogy a releváns információ egyben maradjon. A korábban említett RAG-gal kapcsolatos szakirodalmak egy része alapvetően kisebb 300-500 karakteres chunk méretet javasol, kb 10-15 százalékos overlap értékkel. Túl nagy méretnél ugyanis előfordulhat, hogy feleslegesen sok információ kerülhet egyszerre a kontextusba, túl általános lesz, vagy a konkrét témáról, eltereli más előforduló rész a hangsúlyt. Tapasztalataim alapján viszont nem a túl nagy, hanem éppen a túl kicsi chunk\_size okozott problémát. Irreleváns források, nem releváns oldalak és pontatlan válaszok jelentek meg kisebb chunk méretek mellett, mivel a kisebb szegmensek szétaprózták a logikai egységeket. A 1000 karakteres chunk méret bizonyult optimálisnak, mert így a gondolatmenetek összefüggőek maradtak, és csökkent az irreleváns találatok aránya. A tapasztalataimat az Nvidia egyik online kurzusa is megerősítette, ami pontosan ezeket az értékeket javasolta (Nvidia, 2023).

Minden létrehozott szövegszegmens metaadatai közé bekerült a dokumentum címe is (metadata['title']), amelyet a címkinyerési folyamat eredményeként határoztam meg. Ez a metaadat lehetővé teszi, hogy a későbbiekben pontosan azonosítható legyen, hogy melyik dokumentumból származik egy adott válasz. Az oldalszámot már az oldalakra történő bontás alkalmával attribútumként megkapta minden elem, amit helyesen beállítva a chunkolás után is örököl. Így ezt külön nem kell már hozzárendelni.



2. Kódrészlet: A szöveg feldarabolása

### Vektortároló és beágyazó modul

A dokumentumszegmensek numerikus reprezentációjának előállítása és ezek metaadatokkal együtt történő strukturált tárolása nagyon fontos a gyors és szemantikusan helyes információ-visszakereséshez. A numerikus vektorok lehetővé teszik, hogy jelentésbeli hasonlóságon alapuló keresést végezzünk a dokumentumok között.

**Beágyazások generálása**: Az OpenAIEmbeddings adapter osztályt használtam a szövegszegmensek beágyazásainak előállításához, a text-embedding-3-large modellt alkalmazva, amelyet alkalmasnak találtam a magyar nyelvű szövegek kezelésére is. A vektorok és a hozzájuk tartozó metaadatok (dokumentumcím és oldalszám) tárolásához a Chroma tárolót alkalmaztam, ahogyan erről korábban már esett szó. A szegmentált dokumentumok embeddingjeit a from\_documents() metódus segítségével hoztam létre és tároltam el. Ebben a lépésben a szövegrészekhez tartozó metaadatok is tárolásra kerültek csatolt módon, hogy a vektorokkal együtt legyenek kezelve. A keresési és válaszgenerálási folyamatot a ConversationalRetrievalChain segítségével valósítottam meg, ami összeköti a nyelvi modellt és a retrievert. A return\_source\_documents=True beállítás biztosítja, hogy a válasz mellett a felhasznált szövegrészek forráshivatkozásai (dokumentumcím és oldalszám) is elérhetőek legyenek.

### Kérdésosztályozó és nyelvfelismerő modul

A kérdés nyelvének és típusának (dokumentumhoz kapcsolódó vagy általános beszélgetés) osztályozását egyetlen LLM-hívással végzem el, a classify\_question\_and\_language() függvény segítségével. A modell megállapítja a kérdés nyelvét és azt, hogy a kérdés dokumentumhoz kapcsolódik-e vagy csak egy általános beszélgetés.

A kérdésosztályozást prompt-alapú instrukció-követéssel oldottam meg, amely zero-shot jellegű megközelítés. A modell nem volt előzetesen betanítva a nyelv és a kérdés típusa szerinti osztályozásra. Egy szöveges prompt útján kap a modell instrukciókat arra vonatkozóan, hogy hajtsa végre az osztályozási feladatot. Így nincs szükség külön tanító adathalmazra vagy finomhangolt osztályozóra. Ez a prompt alább olvasható:



4. Kódrészlet: A kérdés típusának és nyelvének osztályozásához szükséges prompt

A megadott prompt világosan és a tapasztalataim alapján hatékonyan, biztonságosan utasítja a modellt a feladat elvégzésére. A két feladat egyetlen hívásban történő megoldása csökkenti az API-hívások számát és javítja a rendszer válaszidejét, ugyanakkor ez a megközelítés némileg növeli a promptértelmezés komplexitását. Amennyiben az LLM eltérne a várt formátumtól, akkor biztonsági szempontok miatt automatikusan conversational működésként megy tovább a folyamat.

A képen szöveg, Betűtípus, képernyőkép, fehér látható

Automatikusan generált leírás

4. ábra: conversational típusú válasz

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

5. ábra: Documentum related típusú válasz

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

6. ábra: Angol nyelvű kérdésre adott válasz

A helyes működési logika meghatározásával a rendszer képes arra, hogy különböző nyelven írt szövegeket közösen tudjon kezelni. Ennek több feltétele van. Az egyik a megfelelő embedding modell kiválasztása, és annak helyes használata. Ezáltal a különböző nyelvű dokumentumok közös vektortérben kerülnek eltárolásra. Ez azt jelenti, hogy a magyarul és angolul leírt, ugyanazon jelentéssel bíró mondat nagyon hasonló embedding vektort fog eredményezni. Ez azért van, mert nem kulcsszavas keresésen alapul a technológia, hanem a kérdés és a válasz közötti embeddinget hasonlítja össze a rendszer, ami a jelentés, szemantika és nem a nyelvi forma alapján kerül kiszámításra, és ezt azonosítja a rendszer.

### Válaszgeneráló és forrásmegjelölő modul

A válaszgenerálás során a rendszer figyelembe veszi a kérdés típusát és a témához tartozó információkat az előző alfejezetben tárgyaltak szerint, a classify\_question\_and\_language() függvény segítségével. Ezt követően a kérdés típusa határozza meg, hogy történik-e dokumentumalapú visszakeresés A chatbot\_response() függvény felelős a válasz generálásáért. Ha a kérdés dokumentumhoz kapcsolódik, akkor a rendszer a kérdés vektorképét összehasonlítja a dokumentumszegmensek embeddingjével, tehát visszakeresi a témához szorosan kapcsolódó szegmenseket a vektortárolóból, és azokat felhasználva generál választ a nyelvi modell. Ha a kérdés általános, akkor a modell közvetlenül generál választ a beszélgetési előzmények alapján.

A format\_sources() függvény csoportosítja és formázza a forrásokat a dokumentum címek és oldalszámok alapján. A cosine similarityből kalkulált cosine distance függvény segítségével rangsorolom a dokumentumrészeket, és a legjobban illeszkedőek oldalszámát jelenítem meg. A cosine simmilarity függvény két bemenetet, két vektort használ. A kérdésből és a dokumentum egy szegmenséből vektorok formájában. A metódust a Chroma a korábban bemutatott HNSW indexeken keresztül számítja. A kiértékelés módja a következő:

1. Egyenlet: Cosine similarity függvény

Ahol a kérdés vektorképét jelöli, a dokumentum szegmensének vektorképét, a számlálóban a két vektor skaláris szorzatát, a nevezőben pedig a két vektor hosszait. Az eredményt úgy kell értékelni, mintha korrelációról beszélnénk. Az 1 érték teljes egyezőséget jelent (párhuzamosság), a 0 a két vektor függetlenségét (merőlegesek egymásra) a -1 pedig azt jelenti, hogy a két vektor teljesen ellentétes. Ez utóbbi szövegek esetében nagyon ritkán fordul elő azonban. A Cosine simmilarity alapján előáll a cosine distance, ami 1-cosine simmilarity képlettel számolható.

A válaszok generálásakor biztosítom, hogy a válasz nyelve megegyezzen a kérdés nyelvével, és hogy a dokumentumokból származó információk megfelelően legyenek hivatkozva. Explicit módon utasítom az LLM-et a megfelelő nyelv használatára, esetlegesen a fordításra. A hivatkozások és a rendszerüzenetek külön kerülnek az üzenet végén hozzáfűzésre. A rendszer képes több dokumentumból visszanyerhető információkat szintetizálni, megjelenítve a kapcsolódó forrásokat:

A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.7. ábra: Több forrást felhasználó válasz

Abban az esetben, ha a rendszer felismeri, hogy dokumentumból kell információt kinyernie a kérdés megválaszolására, de nem talál megfelelő információt, vagy az LLM nem tud értelmes választ adni, mert a dokumentumrészletek túl zavarosak, a rendszer figyelmezteti a felhasználót erre, és arra kéri, hogy pontosítsa a kérdést, hátha akkor sikerrel jár.

8. ábra: Nem megtalálható információ esetén adott üzenet.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Abban az esetben, ha a szövegekből képzett embeddingek és a kérdés vektorképe között a cosine distance értéke 0,87 felett van, a qa\_chain lefut ugyan, de egy védelmi trashold funkció automatikusan figyelmeztetést ad a források helyett. Ekkora értéknél általánosságban ugyanis a válasz relevanciáját bár általában jónak, de néhány esetben már kétesnek találtam. Ezt a küszöbértéket empirikus úton határoztam meg.

A képen szöveg, képernyőkép, Betűtípus, dokumentum látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

9. ábra: Nem kellően releváns források esetén adott válasz

Elméletben előfordulhat az, hogy a visszaadott dokumentumrészekből az LLM nem tud értelmes választ adni, mert túl zavarosnak találja a szegmenseket, azonban a cosine distance érték alapján mégis válaszként visszaadásra kerül valamilyen forrás. Ebben az esetben a válasz elemzésével le lehet tiltani egyszerűen a forráshivatkozást, azonban mivel a kiértékelés és a fejlesztés alatt sem találkoztam ezzel az esettel, így az számítási lépések minimalizálása miatt ez a funkció kikapcsolásra került.

### Felhasználói interfész modul

A felhasználói interfész a Gradio keretrendszerrel készült, ami webalapú interaktív chatfelületet biztosít a rendszerhez. A gr.ChatInterface osztályt, egy magas szintű komponenst használtam a chatfelület megvalósításához. A ChatInterface példányosítása során a válaszgenerálásért felelős chatbot\_response függvényt adtam meg, amely a felhasználói kérdést és a korábbi beszélgetési előzményeket felhasználva állítja elő a választ.

A képen szöveg, képernyőkép, dokumentum, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

10. ábra: A felhasználói interfész

Ahogyan a 10. ábrán látszik, a felhasználói interfész tetején egy lenyitható listában meg lehet tekinteni, hogy éppen milyen dokumentumok vannak betöltve, feldolgozva.

A beszélgetési előzményeket ChatInterface kezeli. A Gradio list of lists formátumban tárolja a kérdés-válasz párokat, míg a LangChain ConversationalRetrievalChain list of tuples formátumot vár. Ennek átalakítására külön függvényt készítettem (convert\_history\_for\_chain()), ami megoldja a kompatibilitást. A rendszerben emellett a history hosszát is korlátoztam a process\_history() függvénnyel, ami mindig csak az utolsó 3 kérdés-válasz párt adja át a láncnak. Ez csökkenti a felesleges kontextus terjedelmét, és segít elkerülni, hogy régi válaszok torzítsák az újabb kérdések megválaszolását. Korábban említett fontos szempont volt, hogy a forráshivatkozások ne kerüljenek be az előzmények közé. A history-ban ezért kizárólag a válasz szövegének tartalmi része szerepel, a források csak a megjelenített válaszban láthatóak. Az interfész elindítása a iface.launch() metódussal történik, ami alapértelmezetten a lokális gépen futó szerveren biztosít webes elérhetőséget. A share=True opcióval a Gradio tunneling szolgáltatását használva lehetőség van ideiglenes, publikus URL létrehozására, amelyen keresztül a rendszer külső felhasználók számára is elérhető. A szolgáltatás ideiglenes jellege miatt ez az elérés korábban legfeljebb 72 óráig volt biztosított, a dolgozat elkészülésekor azonban már egy hétig, így ez a Gradio aktuális szolgáltatási feltételeitől függ. A limit kiterjesztésére vannak opciók, például a gradio támogatja bizonyos programok webes futtatását (Gradio, 2024).

# Kiértékelés

A RAG, de egyszerűen az információ visszakereső rendszerek esetében a rendszer teljesítményének az értékelése, mérése alapvető fontosságú. A keresett találatok hatékony azonosítása nem csupán technikai kérdés, hiszen maga az összetett folyamat önmagában befolyásolja a felhasználó elégedettségét is. Ez az aspektus azért emelendő ki, mert, egy olyan problémakörről van szó, ami nem egy egyszerű adatbázisról keres elő értékeket, hanem látszólag sem a kérdések, sem a dokumentumok sem strukturáltak a felhasználó szemszögéből. Technikai megközelítéssel nézve persze a strukturáltság triviális, azonban a felhasználónak nem célja ezt a struktúrát megérteni. Számára egy feketedobozként működő rendszerről van szó csupán. Ezt az aspektust figyelembe kell venni a működés értékelésekor is.

A rendszerértékelés módszertani kihívásai között szerepel, hogy a relevancia fogalma nem mindig egyértelműen meghatározott az ilyen esetekben. A különböző felhasználói igények, kérdéstípusok, nyelvezet, tartalmi kontextusok eltérő elvárásokat támasztanak azzal kapcsolatban, hogy mi tekinthető helyes válasznak, és mi nem. A kettő közötti határ könnyen belátható mindenki számára, hogy nem éles. Emiatt nagyon fontos az olyan kiértékelési megközelítések alkalmazása, amelyek figyelembe veszik a felhasználói szempontokat és a tartalmi megfelelést, nem csupán az automatizált algoritmusok által mért szintaktikai vagy szótávolságon alapuló hasonlóságot.

A kézi, manuális relevanciaértékelés is az egyik legpontosabb módszer a rendszer teljesítményének vizsgálatára. Az LLM szolgáltatások széles választéka kísértheti a fejlesztőt azzal kapcsolatban, hogy a kiértékelést és a nyelvezeti pontosságot, relevanciák mérését rábízza egy AI algoritmusra, azonban kritikus esetekben, amikor a tapasztalatokat kell vizsgálni, akkor mindenképp jobb, ha az ember végzi a megfigyeléseket, hiszen a tapasztalatok levonásakor olyan feladatok is felmerülnek, amikor a rendszer emberi kepésekéit imitáló funkcióját mérjük. Tudományos források is alátámasztják, hogy a gépi és az emberi tapasztalatok közötti eltérések esetében az emberi megfigyelés és döntéshozatal értékesebb (InterSystems, 2024).

A következőkben bemutatásra kerül az alkalmazott manuális tesztelési módszertan részletes leírása, valamint az értékelés során alkalmazott mérőszámok és azok indoklása.

## A kiértékelési módszertan kialakítása

### A kérdésállomány összeállításának szempontjai

A tesztelés egy kellően nagy elemszámú PDF dokumentumokból ellátott poolal, illetve egy kérdéslistával történt. A kérdéskészlet kialakításánál fontos szempont volt a kérdések tematikai és komplexitás szerinti sokszínűségének biztosítása. Ennek a célja egyrészt az volt, hogy a rendszer ne csak egy szűk tematikus fókuszú kérdéskörben legyen képes a megfelelő dokumentumokat kiválasztani, hanem különböző tartalmi területeken is megőrizze teljesítményét. A teszteléshez a rendszerbe 51 tudományos publikáció került betöltésre. Ezek egy részét hivatalos kutatóintézetek oldaláról kerültek letöltésre, de akadt közöttük egyetemi TDK dolgozat is, illetve az MNB elemzéseiből is kerültek dokumentumok az adathalmazba. A publikációk tematikájukat tekintve széles spektrumot fedtek le, de fontos szempont volt az is, hogy csoportosíthatóak legyenek, és minden területhez legyenek egymáshoz közel álló tematikával rendelkező publikációk is. A betöltöt PDF-ek az alábbi tematika alapján csoportosíthatóak:

* Természettudományok
* Energetika és klíma
* Munkaerőpiac, oktatás, gazdaság
* Társadalmi kérdések, vallás, történelem

Mindegyik csoportban legalább tíz dokumentum szerepelt, ugyanakkor tematikában voltak még ezen csoportok között is interdiszciplináris művek, amelyek a csoportjukon túl, még más csoporthoz tartozó elemeket is tartalmaztak.

A kiértékelés az utolsó csoport dokumentumait célozta, mivel azokban szerepelt a legtöbb olyan dokumentum, ami más csoportokkal átfedhet olyan módon, hogy bár a főtematika egyedi, de más csoportok szakkifejezései sűrűn előfordultak. Ilyen például a vallás kapcsán az evolúciós kérdéskörrel foglalkozó dokumentum, ami gyakran használt természettudományos kifejezéseket, vagy olyan történelemmel foglalkozó publikációk, amelyek gazdasági szakkifejezésekben bővelkedtek.

A kérdések megalkotásánál figyelembe kellett venni a különböző kérdéstípusokból legyen mindnek reprezentációja: szerepeltek egyszerűbb, factoid jellegű kérdések (pl. meghatározások, alapinformációk), valamint komplexebb, érvelést vagy összehasonlítást igénylő kérdések is (pl. különböző nézőpontok viszonyának elemzése). Utóbbinál az is fontos szempont volt, hogy néhány összehasonlítás két különböző dokumentumban szereplő tételre vonatkozzon. Ezen diverzitás biztosítja, hogy a mérési eredmények ne torzuljanak egyetlen kérdéstípus felé, és a rendszer általános képességeiről adjon képet. A kérdések a függelékben olvashatóak.

A kérdésállomány méretének meghatározásánál a statisztikai mintavétel elveiből kiindulva legalább 30 kérdés került alkalmazásra. Ez a mintaelemszám elfogadott a legtöbb empirikus vizsgálatban, amikor az eredmények normalitásközeliségére és a variancia stabilitására törekszünk (Groves et al., 2009). A jelen vizsgálatban alkalmazott 30 kérdés ennek megfelelően biztosítja a metrikák átlagértékeinek értelmezhetőségét és az eredmények általánosíthatóságát (Ketskeméty and Pintér, 1999).

A válasz relevanciájára a következő definíciót vezetem be: A dokumentum információtartalma nem pusztán érinti a kérdés témáját, hanem konkrét, a kérdés szempontjából értelmezhető, szakmailag megalapozott állításokat, magyarázatokat vagy érveket közöl. Az ilyen dokumentumok képesek ténylegesen elősegíteni a kérdés megválaszolását, nem csupán periférikusan kapcsolódnak a témához.

Bár ilyen módon a relevancia vizsgálata szubjektív alapokon mérhető, az erre épített kvalitatív megközelítés bevezetése biztosítja a metrikák egyértelmű és reprodukálható számítását.

## Használt metrikák és számítási módjuk

A relevanciaértékelés során alkalmazott metrikák kiválasztása és számítása nem triviális kérdés. Az információ-visszakeresési rendszerek hatékonyságának mérésére a szakirodalom több mutatót alkalmaz, amelyek közül a Precision, Recall, Mean Reciprocal Rank (MRR). (Manning and Nayak, 2013; restack, 2025). Bár a rendszer hatékonyságát nem méri, de megvizsgáltam a Score of First Relevant Document értékét is. Ez ugyanis válaszol arra a kérdésre, hogy a feltett kérdés milyen nehézségű volt a rendszer számára a betöltött dokumentumok függvényében. A magasabb érték nehezebb feladatot jelentett. Ennek tükrében vizsgáltam a rendszer hatékonyságát mérő metrikákat. A vizsgálat célja az volt, hogy ezen metrikák segítségével számszerűsíthető képet kapjunk arról, hogy képes-e a rendszer a kérdésekre vonatkozóan releváns dokumentumokat a találati listák elején megjeleníteni.

### Precision és Recall definíciója és jelentősége

A Precision (pontosság) azt fejezi ki, hogy az adott kérdésre visszaadott találatok közül mekkora arányban szerepeltek valóban releváns dokumentumok. A Recall (teljesség) pedig azt méri, hogy az összes releváns dokumentumból mennyit sikerült visszanyerni. A két metrika közötti különbség jól illusztrálja az információ-visszakeresési rendszerek egyensúlyi dilemmáját, mert míg a Precision a téves pozitív találatok minimalizálására törekszik, addig a Recall a kihagyott, viszont releváns találatokat méri.

Matematikai definíciójuk a következő:

2. egyenlet. A Precision képlete

3. egyenlet. A Recall képlete

Egy adott kérdés esetén, ha például 2 releváns dokumentum található a teljes dokumentumhalmazban, és a rendszer a top-5 találat között két relevánsat adott vissza, akkor a Precision értéke 2/5 = 0,4, a Recall pedig 2/2 = 1. A vizsgálat során a K = 5 értéket alkalmaztam, vagyis a rendszer első öt találatát vettem figyelembe a metrikák számításakor. A szakirodalom szerint a k=5 egy gyakran használt és elfogadott benchmark érték. Elég dokumentumot ad a generátornak a releváns információkkal, ugyanakkor nem terheli túl a modellt felesleges vagy irreleváns adatokkal, amelyek ronthatnák a válasz minőségét (Yu *et al.*, 2024). A mérési hibák elkerülése miatt olyan kérdéseket kellett előállítani, amelyek legfeljebb 5 dokumentumból szerezhetnek releváns információkat. Ennek a recall érték számításakor van szerepe. Ezen listában a találatok a rendszer által rangsorolva sorba vannak rendezve a cosine distance érték alapján olyan módon, hogy a rendszer által legnagyobb relevanciájúnak vélt elem van az első helyen, és csökkenő sorrendben követi a többi.

### Mean Reciprocal Rank (MRR) és a Score of First Relevant Document

A Mean Reciprocal Rank (MRR) metrika a releváns találatok listán elfoglalt helyét veszi figyelembe. Az MRR azt fejezi ki, hogy az első releváns találat hányadik helyen jelent meg a visszaadott listában, és ennek reciprokát használja az értékelés során.

Egyetlen kérdésre vonatkozó Reciprocal Rank (RR) számítása:

4. egyenlet. Az RR képlete

Az MRR képlete az összes kérdésre számítva pedig:

5. Egyenlet Az MRR képlete

ahol a Q a kérdések száma, a ranki pedig az első releváns találat helyezése az i-edik kérdés esetén.

A rendszer teljesítményének értékelésére bár nem alkalmas, de megvizsgáltam a Score of First Relevant Document metrikát is, ami azt méri, hogy a találati listában elsőként előforduló releváns dokumentum milyen pontszámot ért el a rendszer ranking-függvénye szerint.

A bemutatott rendszer esetében az alacsony score érték azt jelzi, hogy a rendszer nagy bizonyossággal sorolta előre a releváns dokumentumot, míg a magas score alacsony biztonságú rangsorolást feltételez. Ez persze a numerikus hasonlósági módszertől függ. Az euklideszi távolságok mérésénél a kisebb érték a jobb, de egy koszinusz azonosság esetében az 1-hez közeli, és a 0-tól távoli.

A fentiek mellett ki kell emelni, hogy a cosine distance elfogadhatóságának a határértékeinek vizsgálata, meghatározása önmagában nem lehet cél, és értelme sincs sok esetben, ugyanis ez nem a rendszerre vonatkozó mertrika, hanem a nagyban függ inkább a betöltött dokumentumoktól, azok zajosságától, heterogenitásuktól. Önmagában nem alkalmas a rendszer hatékonyságának mérésére. Ezekre a Precision, Recall és az MRR mutatók szolgálnak. Bár más aspektusokra vonatkoztatva a szakirodalom ad határértékekre iránymutatásokat, de kiemeli, hogy ezek önmagukban a rendszer hatékonyságát nem tudják mérni. Sokkal inkább ad képet arról, hogy a kérdések milyen nehézségűek voltak, illetve a szövegek mennyire voltak nehezen kereshetőek. Ezen értékek célja az, hogy egy sorrend ezek alapján kialakuljon a találatok között, ami alapján a válasz megfogalmazható. Tehát nem egy teljesítménymutatóról van szó, hanem egy módszer a hasonlóságok kiszámítására, amely után a tényleges teljesítményt külön, az említett metrikával szükséges mérni (Karpukhin, O˘ Guz, *et al.*, 2020; Thakur *et al.*, 2021).

## Eredmények bemutatása

A relevanciaértékelés során kapott kvalitatív metrikák a korábban ismertetett módszertan alapján a következőkben kerülnek bemutatásra:

|  |  |  |
| --- | --- | --- |
|  | **Átlag** | **Szórás** |
| **Score of First Relevant Document** | 0,7947 | 0,06 |
| **Precision** | 0,987 | 0,05 |
| **Recall** | 0,91 | 0,19 |
| **Mean Reciprocal Rank (MRR)** | 1 | 0 |

3. táblázat. Az értékelés eredményei

Az adatokból jól látható, hogy a Precision rendkívül magas (közel 99%), ami azt jelzi, hogy a rendszer által visszaadott dokumentumok majdnem minden esetben relevánsak. A Recall értéke szintén magas, de ennél a metrikánál nagyobb szórást tapasztaltam. Ennek oka az volt, hogy bizonyos kérdések esetében előfordult, hogy nem minden releváns dokumentum került a top-k találatok közé. Ez elsősorban azoknál a kérdéseknél jelentkezett, ahol egynél több releváns dokumentum volt kijelölve. Ennél a metrikánál ki kell emelni azt, hogy a rendszernek nem célja az összes dokumentumból forrásokat bevonnia a válaszokhoz. Előfordulhat ugyanis az, hogy egy adott kérdésben két dokumentum is tartalmazza a válaszokat, de az egyik nyelvezete és tematikája jobban illik a kérdéshez, részletesebb információkat tartalmaz, míg a másik általánosabban fedi le a kérdés tárgyát. Ilyen esetben mindkét dokumentumból érkezhetnének releváns válaszok, de amennyiben mindkettő betöltésre került a vektortárolóba, úgy előfordulhat, hogy a részletesebb dokumentum részei felülreprezentáltak lesznek a jobb illeszkedés miatt. Ez nem hiba, hanem előny, ugyanakkor a recall metrika számításakor módszertanilag az összes releváns dokumentumot fel kell sorolni az arány kiszámításához, ami miatt ez a torzulás előállhat. Az eredmények értékelésekor ezt a tényezőt is figyelembe kell venni.

Az MRR értéke 1, amely azt mutatja, hogy minden esetben az első találat releváns volt. Ez egy nagyon fontos indikátora a rendszer rangsorolási hatékonyságának. Ugyanakkor azt is jelezheti, hogy a kérdések nem voltak elég nehezek a rendszer számára. A nehézséget a Score of First Relevant Document tudja mérni, ami kellően magas volt ahhoz, hogy a kérdéseket nehéznek értékeljük. A kérdéslista tartalmazott számos félrevezető szándékkal megfogalmazottokat is. Ilyen nehezítő tényező például az, amikor egy történelemmel kapcsolatos szövegben modern nyelvezettel teszünk fel kérdéseket.

A kérdéslistából egy példát kiemelve: *Milyen gazdasági ösztönzőket alkalmazott a spártai állam a társadalmi stabilitás fenntartására, és hogyan különbözött ez a modern humán erőforrás menedzsment eszközeitől?*

A Spártáról szóló szövegforrás nem tartalmazott olyan modern kifejezéseket, mint a *humán erőforrás menedzsment* vagy a *gazdasági ösztönző*, ugyanakkor több modern témákkal foglalkozó tanulmány is használta ezeket a kifejezéseket. Ezek a tanulmányok szintén betöltésre kerültek előzetesen. A célja az ilyen kérdéseknek az volt, hogy félrevezesse a rendszert, hiszen a szakkifejezések többsége nem a kérdést helyesen megválaszolandó dokumentumban vannak. A rendszer azonban ezekben az esetekben is jól felismerte a kérdés lényegét és el tudott vonatkoztatni a szándékosan félrevezető nyelvezettől, pontos választ adott a megfelelő hivatkozásokkal. Az MRR kapcsán felvetődhet a kérdés, hogy módszertani hiba miatt lett 1 az eredmény, azonban a mérés a korábban ismertetett szakirodalmak pontos követése alapján készült. Feltehetően nagyobb minta, vagy jóval nehezebb kérdések szükségesek a stabil MMR kibillentéséhez, azonban a korábban bemutatott módszertani elvek és szakirodalmi források alapján a rendszer értékelhetősége szempontjából érvényes elemszámokkal történt a vizsgálat.

Bár nem a rendszer, de a vizsgálat értékelésében nemcsak a találatok sorrendje, hanem a rangsorolást megalapozó hasonlósági értékek vizsgálata is fontos. A vizsgálat során alkalmazott metrika a Score of First Relevant Document, a releváns dokumentum és a kérdés embeddingjei közötti cosine distance-en alapul. A vizsgálatban a cosine distance mérése a first relevant document vonatkozásában történt, vagyis mindig az első releváns dokumentum embeddingjének távolságát vizsgáltam a kérdés embeddingjéhez képest.

A vizsgálat során átlagoltam az eredményeket, azonban a mikro szinten keletkezett eredmények is fontos tanulságokat hordoztak.

Minden kérdés esetében a rendszer első helyen adta vissza valamelyik releváns dokumentumot (MRR = 1). Ez azt jelenti, hogy a rangsorolás sorrendisége a mérésben tökéletes. Ugyanakkor az első helyen lévő releváns dokumentumokhoz tartozó cosine distance score értékek eltérőek. Ez azt mutatja, hogy nem minden első helyezett találat volt azonos mértékben „közel” a kérdés embeddingjéhez a szemantikai térben. Összességében ez a kérdés nehézségét tükrözi. Például a Q16 kérdés esetében az első releváns dokumentum score-ja 0,934, ami magasabb távolságot jelez, míg más kérdéseknél (pl. Q11: 0,656) lényegesen kisebb a távolság. Ez arra utal, hogy bár mindkét esetben első helyen van a releváns dokumentum, a Q16-nál a rendszer kevésbé „biztos” a választásban. A cosine distance score fontos kiegészítő mutató, amely segít feltárni a ranking döntéseinek mögöttes minőségét, de csak a többi, korábban említett metrikákat kiegészítőleg vehető figyelembe. A score és a Recall közötti kapcsolat is jól megfigyelhető. Ahol több releváns dokumentumból kell szintetizálni a választ (pl. Q2, Q6, Q8), ott a Recall alacsonyabb, és a cosine distance score is gyakran magasabb. Ez arra utal, hogy ezeknél a kérdéseknél a rendszer nehezebben tudja mindegyik releváns dokumentumot a megfelelő pozíciókba sorolni, nehezen dönt arról, hogy mi egyértelműen a releváns. A score-értékek szórása (0,0607) viszonylag alacsony, de még így is jelzi, hogy bizonyos kérdéstípusoknál (főleg összetettebb, több dokumentumot igénylő kérdéseknél) nehezebb dolga van a rendszernek. Elméletileg a cosine distance 0 és 2 között vehet fel értéket, ahol a 0 teljes hasonlóságot, a 2 pedig teljes ellentétet jelent. Az NLP rendszerekben azonban az 1 feletti érték ritka. A mérés átlagos eredménye 0,7947 lett, ami arra utal, hogy a releváns dokumentumok embeddingjei szemantikailag kellően távol helyezkedtek el egymáshoz, de a válaszok minősége ennek ellenére is megfelelő volt. A kellő távolság azért fontos, mivel nem átlagos, hanem a szándékosan nehéz, félrevezető, vagy több dokumentumból szintetizálandó választ váró kérdések esetében nagyobb ez az érték. Ennek teljesülését ez a mutató bizonyította.

### A Voyage modell

Ahogyan korábban is szó volt erről, a RAG rendszerek hatékonysága nagymértékben függ az alkalmazott embedding modelltől, amik a dokumentumrészek és a kérdések vektoros reprezentációit biztosítják. A Voyage voyage-3-large modelljét azért vizsgáltam meg részletesen, mert a jelenleg elérhető nyílt embedding modellek közül ez tűnt az egyik legígéretesebb alternatívának a RAG rendszeremhez az OpenAI embeddingek mellett. Az összehasonlítást tíz véletlenszerűen kiválasztott kérdésen végeztem, amiket a korábban tárgyalt vizsgálatból gyűjtöttem. A mért metrikák továbbra is a korábban bemutatottak voltak.

Az OpenAI minden kérdésnél megtalálta legalább az egyik releváns dokumentumot az első helyen (MRR = 1), míg a Voyage esetén több kérdésnél is hátrébb sorolódott a találat.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Precision** | **Recall** | **MRR** | **Score of First Relevant Doc** |
| **OpenAI** | **Átlag** | 0,98 | 0,95 | 1 | 0,78 |
| **Szórás** | 0,06 | 0,14 | 0 | 0,06 |
| **Voyage** | **Átlag** | 0,87 | 0,86 | 0,95 | 0,71 |
| **Szórás** | 0,21 | 0,22 | 0,14 | 0,07 |

4. táblázat: Az OpenAI és a Voyage embedding modelljének összehasonlítása

A cosine distance alapú „Score of First Relevant Doc” metrikában a Voyage modell jobb átlagos eredményt ért el (0,71 szemben az OpenAI 0,78-cal), ami arra utal, hogy a kérdés és a releváns dokumentum embeddingjei közelebb esnek egymáshoz. Ennek ellenére a többi érték gyengébb lett. A precision, recall és MRR értékek mind alacsonyabbak voltak, mint az OpenAI esetében.

Ez jól érzékelteti, hogy az embedding tér numerikus közelsége önmagában nem elég. Egyszerűen fogalmazva nemcsak a releváns dokumentumokat kell közel hozni, hanem az irrelevánsakat távol kell tartani. A Voyage embedding tér valószínűleg kevésbé diszkriminatív, azaz a releváns és irreleváns dokumentumok túl közel helyezkednek el egymáshoz. Ezzel szemben az OpenAI modell embedding tere jobb szeparációt biztosít, így biztosabb a dokumentumok helyes rangsorolása.

Az egész hátterében az is állhat, hogy a Voyage modell főleg a nagyobb nyelvekre lett inkább tanítva, hiszen a HuggingFace elemzésében (huggingface, 2025) a legelőkelőbb helyen állt a beállított szűrési feltételekkel, ugyanakkor külön a magyar nyelvre nem lehetett szűrni. Az OpenAI embedding ezek alapján kijelenthető, hogy jobban kezeli a magyar nyelv szerkezeti sajátosságait.

## A módszer előnyei és korlátai

A manuális relevanciaértékelés pontos képet ad a rendszer tényleges teljesítményéről, főként az összetettebb, elemző jellegű kérdések esetén. A több metrika (Precision, Recall, MRR) együttes alkalmazása lehetővé teszi a rendszer különböző teljesítményszempontjainak átfogó vizsgálatát. A cosine distance score kiegészíti a rangsorolási metrikákat, és segít azonosítani azokat az eseteket, ahol a találati sorrend jó, de a szemantikai egyezés a kérdés nehézsége miatt gyenge. A relevancia megítélése a tesztelést végző szempontjából szubjektív. A 30 kérdés statisztikailag megalapozott, de tesztelés a konkrét kérdéskészletre vonatkozik, így más típusú kérdések esetén eltérő teljesítmény várható. Egy átfogóbb kép érdekében több felhasználó bevonása, akik más érdeklődési területtel, más nyelvezettel, felhasználói szándékkal rendelkeznek jó visszajelzéssel szolgálhatnak a jövőben.

## A fordításon alapuló információkinyerés értékelése

A rendszer teljesítményének értékelése szempontjából fontos vizsgálni azt a funkciót is, amikor a bemenetként érkező kérdések magyar nyelvűek, de a háttérdokumentumok angol nyelven állnak rendelkezésre, és a válaszokat szintén magyar nyelven kell megfogalmazni. Ez a feladat azért fontos, mert az ilyen nyelvközi információkinyerés és fordítás alapú válaszgenerálás nehezebb feladatot jelenthet a rendszer számára és más típusú hibákat is eredményezhet, mint az egynyelvű környezetben végzett keresés.

Mivel ebben az esetben nem a teljes rendszer általános működését, hanem egy speciális képességét kell vizsgálni, ezért méréshez kisebb mintanagyságot választottam. A kérdéskészletnek ebben az esetben nem feltétlenül kell 30 kérdést tartalmaznia. Már 10 jól megválasztott kérdés is képes lehet feltárni a legfontosabb hibalehetőségeket és a rendszer erősségeit ebben a részfunkcióban. A kérdések összeállításánál fontos szempont, hogy a fordítási nehézségek, szóhasználati eltérések és szinonimák hatását is teszteljük, így kellő figyelmet fordítottam ezekre a tényezőkre a kérdések megalkotásakor.

A kapott eredmények összehasonlításra kerültek a kontrollcsoportként működő angol kérdés–angol dokumentum kérdéssorozattal, ahol a kérdések megegyeztek, csak a nyelvük volt magyar helyett angol. A vizsgálat nem a több releváns dokumentum detektálását célozta, hanem kifejezetten a cross-lingual retrieval teljesítményét mérte egy-egy releváns dokumentum esetében. A Recall mutatót ezért a kiértékelésből kizártam. A K értéke továbbra is 5 volt. A kérdések megtalálhatóak a függelékben.

A mért értékek alább láthatóak:

|  |  |  |
| --- | --- | --- |
|  | **Magyar kérdés** | **Angol kérdés** |
| **Precision** | 0,86 | 0,98 |
| **MMR** | 0,9 | 1 |
| **Score of First Relevant Doc** | 0,8475 | 0,7348 |

5. Táblázat. A többnyelvű mérés eredményei

A magyar nyelvű kérdésekkel elért Precision (0,86) és MRR (0,9) értékek jó eredménynek számítanak, figyelembe véve, hogy a dokumentumok kizárólag angol nyelven álltak rendelkezésre. Az angol kérdéses kontrollkísérlet valamivel jobb eredményt hozott (Precision = 0,98, MRR = 1), amely a fordítás szükségtelenségével és a szaknyelvi egyezéssel magyarázható. A Score of First Relevant Document mutató esetében a magyar kérdések átlaga 0,8475, az angol kérdések esetén 0,7348. Ez a különbség a magyar nyelvről angolra történő fordítás torzítására utal, de a nehezített körülmények ellenére, amiről ez a mutató árulkodik a rendszer teljesítményét mérő értékek továbbra is kielégítőek voltak. A Precision csökkenése néhány magyar nyelvű kérdésnél (pl. AQ3, AQ9) azt jelzi, hogy a rendszer helyenként irreleváns szegmenseket is magas helyre sorolt a releváns mellett. Ez leginkább azoknál a kérdéseknél fordult elő, ahol a válasz összetettebb, vagy komolyabb szaknyelvi kifejezéseket tartalmazott, amiket a fordítás során másként értékelt.

Az eredmények azt mutatják, hogy a fordítási lépés nem okozott jelentős teljesítményvesztést a dokumentumszintű visszakeresés során, de nehezebb feladat volt a rendszer számára. Fordítás nélkül egyértelműen jobb a teljesítmény, mivel a fordítás során egyes szakkifejezések más alakot vehetnek fel, amik gyengítik a keresés hatékonyságát. Azok számára, akik más nyelvű dokumentumokban akarnak keresni, de nem értik az adott nyelvet, ez is egy igen hatékony módszernek bizonyult a mérések alapján.

# Összefoglalás

A jelen dolgozat célja egy magyar nyelvű, dokumentum-alapú chat AI rendszer megtervezése és megvalósítása, amely képes kérdések megválaszolására a rendelkezésre álló dokumentumok alapján. A munkában részletesen bemutattam a nagy nyelvi modellek (LLM-ek) és a Retrieval-Augmented Generation (RAG) technológia elméleti alapjait, valamint a vektortárolási megoldások és a dokumentum-visszakeresés fontosságát az ilyen rendszerekben.

A módszertani részben ismertettem a tervezési szempontokat és követelményeket, amelyek meghatározták a rendszer kialakítását. A technológiák és eszközök kiválasztásánál elemeztem az OpenAI GPT modellek, a Mistral 7B, valamint a Meta Llama 2 modellek előnyeit és korlátait a magyar és angol nyelvű feldolgozásra. A korai fejlesztési fázisban alkalmazott modellek nem nyújtottak megfelelő teljesítményt magyar nyelven, ezért áttértem az OpenAI GPT-4o-mini modell használatára, ami lényegesen javította a válaszok minőségét és a rendszer általános teljesítményét.

Az adatfeldolgozás és előkészítés során három különböző PDF dokumentumot dolgoztam fel, amelyek különböző témákat és szövegtípusokat képviseltek: két tudományos tanulmány és egy irodalmi mű. A dokumentumokat betöltöttem, szövegüket kinyertem, majd szegmentáltam és beágyazásokat generáltam a szövegszegmensekből. A beágyazásokat a Chroma vektortárolóban tároltam, amely lehetővé tette a gyors és pontos információ-visszakeresést a felhasználói lekérdezések alapján.

Az implementáció részleteinél bemutattam a rendszer architektúráját, amely több fő komponensből áll: dokumentumfeldolgozó modul, vektortároló és beágyazó modul, kérdésosztályozó és nyelvfelismerési modul, válaszgeneráló és forrásmegjelölő modul és felhasználói interfész. Részletesen ismertettem a kódstruktúrát és az egyes komponensek működését, kiemelve a dokumentumcímek automatikus kinyerését, a nyelv- és kérdéstípus osztályozását, valamint a válaszok generálását és a források megfelelő hivatkozását.

A felhasználói interfész a Gradio keretrendszerrel készült, amely egyszerű és felhasználóbarát webes felületet biztosít a rendszerhez. A tesztelések során a rendszer sikeresen igazolta képességeit a felhasználói kérdések megértésében, a kapcsolódó információk visszakeresésében és a koherens válaszok generálásában, mind magyar, mind angol nyelven.

A jelenlegi rendszer megvalósítása során néhány olyan területet azonosítottam, ahol további fejlesztések és kutatások nagyban javíthatják a rendszer teljesítményét és funkcionalitását. Az egyik fontos fejlesztési irány a vektortárolási megoldások, mint a Chroma és a FAISS, teljesítményének alapos összehasonlítása. Bár a szakirodalom számos tanulmányt közöl ezekről a technológiákról, a konkrét alkalmazási környezetben történő saját mérések és elemzések értékes információkkal szolgálhatnak. A másik fejlesztési lehetőség, hogy a rendszer hatékonyságának növelése érdekében érdemes lenne kidolgozni egy automatizált folyamatot, amely új dokumentumokat képes feldolgozni és integrálni a rendszerbe a felhasználói interfészbe integrált módon. Ez biztosítaná, hogy a rendszer mindig olyan információkkal rendelkezzen, amiket a felhasználó meghatároz, és képes legyen kezelni a folyamatosan bővülő adatforrásokat.

Irodalomjegyzék

Amikos Tech LTD (2024) *ChromaDB Cookbook*. Available at: https://cookbook.chromadb.dev/core/concepts/ (Accessed: 18 November 2024).

Asai, A. *et al.* (2019) ‘Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering’, in. Available at: https://doi.org/10.48550/arXiv.1911.10470.

Bailyn, E. (2025) *Top Generative AI Chatbots by Market Share*. Available at: https://firstpagesage.com/reports/top-generative-ai-chatbots/ (Accessed: 23 April 2025).

Borgeaud, S. *et al.* (2022) ‘Improving language models by retrieving from trillions of tokens’, in K. Chaudhuri et al. (eds) *Proceedings of the 39th International Conference on Machine Learning*. PMLR, pp. 2206–2240. Available at: https://doi.org/10.48550/arXiv.2112.04426.

Brown, T.B. *et al.* (2020a) ‘Language Models are Few-Shot Learners’, in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., pp. 1877–1901. Available at: https://proceedings.neurips.cc/paper\_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf (Accessed: 1 December 2024).

Brown, T.B. *et al.* (2020b) ‘Language Models are Few-Shot Learners’, *Advances in Neural Information Processing Systems*, 2020-December. Available at: https://arxiv.org/abs/2005.14165v4 (Accessed: 17 November 2024).

Catav, A. (2023) *Less is More: Why Use Retrieval Instead of Larger Context Windows | Pinecone*, *Pinecone*. Available at: https://www.pinecone.io/blog/why-use-retrieval-instead-of-larger-context/ (Accessed: 23 April 2025).

Chmielewski, D. and Paul, K. (2024) *Murdoch’s Dow Jones, New York Post sue Perplexity AI for ‘illegal’ copying of content | Reuters*, *Reuters*. Available at: https://www.reuters.com/legal/murdoch-firms-dow-jones-new-york-post-sue-perplexity-ai-2024-10-21/ (Accessed: 26 April 2025).

Chowdhury, G.G. (2003) ‘Natural language processing’, *Annual Review of Information Science and Technology*, 37(1), pp. 51–89. Available at: https://doi.org/10.1002/aris.1440370103.

Euronews (2023) *OpenAI’s ChatGPT chatbot blocked in Italy over privacy concerns | Euronews*. Available at: https://www.euronews.com/next/2023/03/31/openais-chatgpt-chatbot-banned-in-italy-by-watchdog-over-privacy-concerns (Accessed: 26 April 2025).

Farkas, R., Vincze, V. and Schmid, H. (2012) ‘Dependency Parsing of Hungarian: Baseline Results and Challenges’, in W. Daelemans (ed.) *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Avignon: Association for Computational Linguistics, pp. 55–65. Available at: https://aclanthology.org/E12-1007 (Accessed: 18 November 2024).

Gehman, S. *et al.* (2020) ‘RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models’, in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 3356–3369. Available at: https://doi.org/10.18653/v1/2020.findings-emnlp.301.

Goodfellow, I.J., Shlens, J. and Szegedy, C. (2015) ‘EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES’. Available at: https://github.com/lisa-lab/pylearn2/tree/master/pylearn2/scripts/ (Accessed: 26 April 2025).

Gradio (2024) *Quickstart*, *HuggingFace*. Available at: https://www.gradio.app/ (Accessed: 18 November 2024).

Grimm, J. and Grimm, W. (1892) *Grimm testvérek összegyűjtött meséi*. Translated by E. Benedek.

Guu, K. *et al.* (2020) ‘REALM: Retrieval-Augmented Language Model Pre-Training’, in *37th International Conference on Machine Learning, ICML 2020*. JMLR.org, pp. 3887–3896. Available at: https://doi.org/10.5555/3524938.3525306.

huggingface (2019) *sentence-transformers/distiluse-base-multilingual-cased-v2 · Hugging Face*. Available at: https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2 (Accessed: 27 April 2025).

huggingface (2025) *MTEB Leaderboard - a Hugging Face Space by mteb*. Available at: https://huggingface.co/spaces/mteb/leaderboard (Accessed: 27 April 2025).

InterSystems (2024) *Retrieval Augmented Generation (RAG): Mi ez és hogyan előzi meg a mesterséges intelligencia hibáit? | InterSystems*. Available at: https://www.intersystems.com/hu/resources/retrieval-augmented-generation-rag-mi-ez-es-hogyan-elozi-meg-a-mesterseges-intelligencia-hibait/ (Accessed: 24 April 2025).

Izacard, G. and Grave, E. (2021) ‘Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering’, in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 874–880. Available at: https://doi.org/10.18653/v1/2021.eacl-main.74.

Jégou, H., Douze, M. and Schmid, C. (2011) ‘Product quantization for nearest neighbor search’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1), pp. 117–128. Available at: https://doi.org/10.1109/TPAMI.2010.57.

Johnson, J., Douze, M. and Jegou, H. (2021) ‘Billion-Scale Similarity Search with GPUs’, *IEEE Transactions on Big Data*, 7(3), pp. 535–547. Available at: https://doi.org/10.1109/TBDATA.2019.2921572.

Karpukhin, V., Oguz, B., *et al.* (2020) ‘Dense Passage Retrieval for Open-Domain Question Answering’, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 6769–6781. Available at: https://doi.org/10.18653/v1/2020.emnlp-main.550.

Karpukhin, V., O˘ Guz, B., *et al.* (2020) ‘Dense Passage Retrieval for Open-Domain Question Answering’, pp. 6769–6781. Available at: https://github.com/facebookresearch/DPR. (Accessed: 25 April 2025).

Ketskeméty, L. and Pintér, M. (1999) *Bevezetés a matematikai statisztikába*. Available at: https://www.szit.bme.hu/~kela/stat.pdf (Accessed: 24 April 2025).

Kwiatkowski, T. *et al.* (2019) ‘Natural Questions: A Benchmark for Question Answering Research’, *Transactions of the Association for Computational Linguistics*, 7, pp. 453–466. Available at: https://doi.org/10.1162/tacl\_a\_00276.

Laangchain (2024) *Introduction | LangChain*. Available at: https://python.langchain.com/docs/introduction/ (Accessed: 18 November 2024).

LangChain (2024) *LASER Language-Agnostic SEntence Representations Embeddings by Meta AI | 🦜️🔗 LangChain*. Available at: https://python.langchain.com/docs/integrations/text\_embedding/laser/ (Accessed: 27 April 2025).

Lee, K., Chang, M.-W. and Toutanova, K. (2019) ‘Latent Retrieval for Weakly Supervised Open Domain Question Answering’, in K. Anna, T. David, and M. Lluís (eds) *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 6086–6096. Available at: https://doi.org/10.18653/v1/P19-1612.

Lewis, P. *et al.* (2020) ‘Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks’, *Advances in Neural Information Processing Systems*, 2020-December. Available at: https://doi.org/10.5555/3495724.3496517.

Liu, X. *et al.* (2019) ‘Multi-Task Deep Neural Networks for Natural Language Understanding’, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 4487–4496. Available at: https://doi.org/10.18653/v1/P19-1441.

Manning, C. and Nayak, P. (2013) *Introduction to Information Retrieval*. Stanford.

Manning, C.D., Raghavan, P. and Schütze, H. (2008) *Introduction to Information Retrieval*. Cambridge University Press. Available at: https://doi.org/10.1017/CBO9780511809071.

Mikolov, T. *et al.* (2013) ‘Efficient Estimation of Word Representations in Vector Space’, in *1st International Conference on Learning Representations*. Scottsdale, Arizona: ICLR, pp. 1301–1781. Available at: http://arxiv.org/abs/1301.3781.

Mistral (2023) *Mistral 7B | Mistral AI | Frontier AI in your hands*. Available at: https://mistral.ai/news/announcing-mistral-7b/ (Accessed: 18 November 2024).

MyScale (2024) *FAISS vs Chroma: Vector Storage Battle*. Available at: https://myscale.com/blog/faiss-vs-chroma-vector-storage-battle/ (Accessed: 18 November 2024).

Nagy, S. (2019) ‘Digital Economy And Society. A Cross Country Comparison Of Hungary And Ukraine’, *Economics of Networks eJournal* [Preprint]. Available at: http://arxiv.org/abs/1901.00283.

Nvidia (2023) *Deep Learning Institute | NVIDIA*. Available at: https://learn.nvidia.com/courses/course-detail?course\_id=course-v1:DLI+S-FX-15+V1 (Accessed: 20 November 2024).

openai (2022) *New and improved embedding model | OpenAI*. Available at: https://openai.com/index/new-and-improved-embedding-model/ (Accessed: 27 April 2025).

OpenAI *et al.* (2023) ‘GPT-4 Technical Report’. Available at: http://arxiv.org/abs/2303.08774.

OpenAI (2025) *ChatGPT — Release Notes*. Available at: https://help.openai.com/en/articles/6825453-chatgpt-release-notes?utm\_source=chatgpt.com (Accessed: 23 April 2025).

Owasp (2025) *LLM09:2025 Misinformation - OWASP Top 10 for LLM & Generative AI Security*. Available at: https://genai.owasp.org/llmrisk/llm092025-misinformation/ (Accessed: 26 April 2025).

Pena, D. (2024) *Understanding How ChatGPT Maintains Context — SitePoint*. Available at: https://www.sitepoint.com/understanding-how-chatgpt-maintains-context/ (Accessed: 23 April 2025).

Pennington, J., Socher, R. and Manning, C. (2014) ‘Glove: Global Vectors for Word Representation’, in M. Alessandro, P. Bo, and D. Walter (eds) *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1532–1543. Available at: https://doi.org/10.3115/v1/D14-1162.

Petroc, T. (2024) *Data growth worldwide 2010-2028 | Statista*, *Amount of data created, consumed, and stored 2010-2023, with forecasts to 2028*. Available at: https://www.statista.com/statistics/871513/worldwide-data-created/ (Accessed: 23 November 2024).

Quesada, A. (2025) *Citas falsas y respuestas “confiadamente incorrectas”: los chatbots son incapaces de atribuir las informaciones | Tendencias | Proyecto | EL PAÍS*. Available at: https://elpais.com/proyecto-tendencias/2025-04-25/citas-falsas-y-respuestas-confiadamente-incorrectas-los-chatbots-son-incapaces-de-atribuir-las-informaciones.html (Accessed: 26 April 2025).

Radford, A. *et al.* (2019) ‘Language Models are Unsupervised Multitask Learners’, *OpenAI blog*, 1.8, p. 9. Available at: https://cdn.openai.com/better-language-models/language\_models\_are\_unsupervised\_multitask\_learners.pdf (Accessed: 18 November 2024).

Rehm, G. and Uszkoreit, H. (2012) *The Hungarian Language in the Digital Age*. Berlin, Heidelberg: Springer Berlin Heidelberg (White Paper Series). Available at: https://doi.org/10.1007/978-3-642-30379-1.

Reimers, N. and Gurevych, I. (2019) ‘Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks’, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 3980–3990. Available at: https://doi.org/10.18653/v1/D19-1410.

Reinsel, D., Gantz, J. and Rydning, J. (2018) ‘The Digitization of the World From Edge to Core’, *Framingham: International Data Corporation*, 16, pp. 1–28. Available at: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (Accessed: 17 November 2024).

restack (2025) *Information Retrieval Effectiveness Metrics | Restackio*. Available at: https://www.restack.io/p/information-retrieval-answer-effectiveness-metrics-cat-ai (Accessed: 25 April 2025).

Sharma, M. (2025) *The Ultimate Guide to PDF Extraction using GPT-4*. Available at: https://www.docsumo.com/blog/pdf-reading-with-gpt4 (Accessed: 23 April 2025).

Stepkurniawan (2024) *Comparing RAG Part 2: Vector Stores; FAISS vs Chroma | by Stepkurniawan | Medium*. Available at: https://medium.com/@stepkurniawan/comparing-faiss-with-chroma-vector-stores-0953e1e619eb (Accessed: 18 November 2024).

Szladek, M. (2019) *A felvilágosodás korai árnyékai*. Budapest.

Tai, Y.C. *et al.* (2025) ‘GenAI vs. Human Fact-Checkers: Accurate Ratings, Flawed Rationales’, *Proceedings of the 17th ACM Web Science Conference 2025 (Websci ’25), May 20â•ﬁ24, 2025, New Brunswick, NJ, USA*, 1. Available at: https://doi.org/10.1145/3717867.

Thakur, N. *et al.* (2021) ‘BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models’. Available at: https://arxiv.org/pdf/2104.08663 (Accessed: 25 April 2025).

Thoppilan, R. *et al.* (2022) ‘LaMDA: Language Models for Dialog Applications’. Available at: http://arxiv.org/abs/2201.08239.

Touvron, H. *et al.* (2023) ‘Llama 2: Open Foundation and Fine-Tuned Chat Models’. Available at: https://doi.org/10.48550/arXiv.2307.09288.

xfunnel.ai (2024) *Inside Perplexity AI: How Their Revolutionary Search Engine Works, Its Base Model & RAG, and How They Pick Companies to Show You*. Available at: https://www.xfunnel.ai/blog/inside-perplexity-ai (Accessed: 26 April 2025).

Yang, Z.G., Szlávik, S. and Ligeti-Nagy, N. (2024) ‘Magyar nyelvű utasításkövető korpusz építése Stanford Alpaca promptok fordításával és lokalizálásával’, in *XX. Magyar Számítógépes Nyelvészeti Konferencia*. Szeged: Szegedi Tudományegyetem, pp. 243–255.

Young, T. *et al.* (2018) ‘Recent Trends in Deep Learning Based Natural Language Processing [Review Article]’, *IEEE Computational Intelligence Magazine*, 13(3), pp. 55–75. Available at: https://doi.org/10.1109/MCI.2018.2840738.

Yu, Y. *et al.* (2024) ‘RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs’. Available at: https://arxiv.org/pdf/2407.02485v1 (Accessed: 25 April 2025).

# Ábrajegyzék

## Ábrák

[1. ábra Generatív AI chatbotok piaci részesedései 2025 áprilisában 24](#_Toc196615406)

[2. ábra Magas szintű blokkdiagram 37](#_Toc196615407)

[3. ábra Részletezett blokkdiagram 39](#_Toc196615408)

[4. ábra: conversational típusú válasz 44](#_Toc196615409)

[5. ábra: Documentum related típusú válasz 45](#_Toc196615410)

[6. ábra: Angol nyelvű kérdésre adott válasz 45](#_Toc196615411)

[7. ábra: Több forrást felhasználó válasz 47](#_Toc196615412)

[8. ábra: Nem megtalálható információ esetén adott üzenet. 47](#_Toc196615413)

[9. ábra: Nem kellően releváns források esetén adott válasz 48](#_Toc196615414)

[10. ábra: A felhasználói interfész 49](#_Toc196615415)

## Képletek

[1. Egyenlet: Cosine similarity függvény 39](#_Toc196514276)

[2. egyenlet. A Precision képlete 45](#_Toc196514277)

[3. egyenlet. A Recall képlete 45](#_Toc196514278)

[4. egyenlet. Az RR képlete 46](#_Toc196514279)

[5. Egyenlet Az MRR képlete 46](#_Toc196514280)

## Táblázatok

[1. táblázat FAISS és Chroma vektortárolók összehasonlítása 12](#_Toc183971816)

[2. táblázat Modulok feladatainak táblázatos bemutatása 20](#_Toc183971817)

## Kódrészletek

[1. Kódrészlet: A szöveg feldarabolása 24](#_Toc183971818)

[2. Kódrészlet: Beágyazások kezelése 24](#_Toc183971819)

[3. Kódrészlet: A kérdés típusának és nyelvének osztályozásához szükséges prompt 25](#_Toc183971820)

[4. Kódrészlet: Felhasználói interfész 28](#_Toc183971821)

# Függelék

## A 30 elemszámú méréshez használt kérdések

Q1 Hogyan viszonyul a felvilágosodás filozófiája a keresztény istenérvek kritikájához és a természettudományos világmagyarázathoz?

Q2 Miként értelmezhető a szexualitás forradalma a felvilágosodás és a modern párkapcsolati kultúra fényében?

Q3 Mit mondanak a keresztény gondolkodók a teremtés és az evolúció összeegyeztethetőségéről, és milyen álláspontot képvisel e kérdésben Hetesi Zsolt?

Q4 Milyen kritikák érik az enciklopédizmust a felvilágosodás idején, és hogyan függ ez össze a pozitivizmus bírálatával a modern tudományban?

Q5 A spártai állam mely társadalmi csoportjai játszottak kiemelt szerepet a hadviselésben, és hogyan viszonyul ez a nők társadalmi szerepéhez?

Q6 Milyen kapcsolat van a muszlim dzsihád értelmezése és a keresztény térítési hagyomány között?

Q7 Hogyan jelenik meg a gyermekvállalás ösztönzése a családpolitikában és a párkapcsolati viselkedésben?

Q8 Hogyan magyarázza Hetesi Zsolt az anyagi világ keletkezését, és mennyiben ellentétes ez a naturalista világnézettel?

Q9 Milyen vallási érvek szólnak az emberi méltóság védelme mellett, és hol jelennek meg ezek a házasság, a család, illetve a társadalmi együttélés témáiban?

Q10 Hogyan értelmezhető a dzsihád fogalma a muszlim közösségek önvédelmi diskurzusaiban, és hogyan reagál erre a keresztény teológia?

Q11 Mi a szerepe az Encyclopédie-nek a felvilágosodás tudásfelfogásában, és milyen bírálatok érték a korabeli egyházi gondolkodás részéről?

Q12 Milyen konfliktusok adódhatnak a vallási alapú házasságkoncepciók és a modern párkapcsolati formák között?

Q13 Hogyan befolyásolja a házasságkötési hajlandóságot a társadalmi mobilitás és a karrierorientáció?

Q14 Milyen érvek szólnak a házasság hagyományos definíciójának megtartása mellett, és hogyan kapcsolódik ez a családpolitikai programokhoz?

Q15 Hogyan jelenik meg a felvilágosodás kritikája a vallási és politikai hatalmi viszonyok tükrében?

Q16 Mi a különbség a spártai helóták és reformáció között?

Q17 Milyen módon próbálta a felvilágosodás gyengíteni az egyház tekintélyét, és miként reagált erre a teológia?

Q18 Hogyan hat a szexuális szabadosság a demográfiai folyamatokra és a párkapcsolati elköteleződésre?

Q19 Milyen érvek szólnak az evolúció és a teremtés kiegyezése mellett a keresztény gondolkodásban?

Q20 Hogyan jelenik meg a vallási identitás szerepe a migrációs konfliktusokban?

Q21 Milyen filozófiai alapon tagadja Hetesi Zsolt a véletlenek szerepét a világegyetem keletkezésében?

Q22 Hogyan befolyásolta a kereszténység és az iszlám közötti párbeszédet a 2000-es évek terrortámadásai után kialakult politikai légkör?

Q23 Milyen érvek jelennek meg a felvilágosodás gondolkodóinál a vallási toleranciával kapcsolatban a világ keletkezése tükrében?

Q24 Miként kapcsolódik a demográfiai válság és a párkeresési kultúra problematikája a családpolitikai eszközökhöz?

Q25 Hogyan vélekedik Balásházy Imre a szexuális forradalom hatásairól a párkapcsolati struktúrákra?

Q26 Mi a spártai állam politikai berendezkedésének alapja, és hogyan biztosították az arisztokrácia hatalmát?

Q27 Milyen szerepet tulajdonítottak az önfegyelemnek a spártai nevelésben, és hogyan kapcsolható ez a párkeresési viselkedésre és a szexuális forradalom utáni társadalmi normákhoz?

Q28 A munkaerőpiaci viszonyok hogyan működtek Spártában?

Q29 Milyen gazdasági ösztönzőket alkalmazott a spártai állam a társadalmi stabilitás fenntartására, és hogyan különbözött ez a modern humán erőforrás menedzsment eszközeitől?

Q30 Hogyan jelent meg a jogegyenlőség és az emberi jogok gondolata a vallásos államszervezetben és a felvilágosodás kritikájában?

## A többnyelvű méréshez használt kérdések

### Magyar kérdés -Angol dokumentum

M-AQ1 Miért alacsony Magyarországon a mobil internet penetráció, noha a 4G lefedettség magas?

M-AQ2 Mely területeket vizsgálja az Európai Unió digitális gazdasági és társadalmi indexe, és mi ezek jelentősége a tagállamok összehasonlításában?

M-AQ3 Hogyan definiálta Tapscott és Williams a digitális gazdaság négy alapelvét?

M-AQ4 Milyen időpontban rögzítik leggyakrabban a műholdképeket, amikor a vidéki piacok tevékenységét vizsgálják a mezőgazdasági térségekben?

M-AQ5 Miért választották a heti egyszeri piacokat preferált célpontként a piaci tevékenység megfigyelésére?

M-AQ6 A „Dunkelflaute” időszakok kapcsán hány TWh hosszú távú tárolási kapacitást becsültek szükségesnek abban az esetben, ha az európai országok között teljes földrajzi kiegyenlítés lehetséges?

M-AQ7 Mekkora Magyarország digitalizáltsága?

M-AQ8 Milyen szerepe van a bizalomnak a mesterséges intelligenciával támogatott online vásárlás elfogadásában a fogyasztói szándék szempontjából?

M-AQ9 Milyen szerepet játszanak a glaciális tavak a magashegységi nemzeti parkokban a jégdugó áradások (GLOF) kialakulásában?

M-AQ10 Ukrajnában vagy Magyarországon jobb a közszolgáltatás digitalizációja?

### Angol kérdés- Angol dokumentum

A-AQ1 Why is mobile internet penetration low in Hungary despite the high 4G coverage?

A-AQ2 Which areas are assessed by the European Union's Digital Economy and Society Index, and what is their significance for comparing member states?

A-AQ3 How did Tapscott and Williams define the four key principles of the digital economy?

A-AQ4 At what time of day are satellite images most frequently captured when monitoring rural market activities in agricultural regions?

A-AQ5 Why were weekly markets selected as preferred targets for observing market activities?

A-AQ6 In the context of 'Dunkelflaute' periods, how many TWh of long-term storage capacity was estimated to be required under the assumption of full geographic balancing across European countries?

A-AQ7 What is the level of digitalization in Hungary according to cross-country comparisons?

A-AQ8 What role does trust play in consumer acceptance of online shopping supported by artificial intelligence?

A-AQ9 What role do glacial lakes play in the formation of glacial lake outburst floods (GLOF) in high mountain national parks?

A-AQ10 Is public service digitalization more advanced in Ukraine or in Hungary?