Kodolányi János Egyetem

Szakdolgozat

Kovács Bálint

Üzemmérnök-Informatikus Alapképzési szak

Budapest

2025.

Kodolányi János Egyetem
Informatika Tanszék

AiFusion – Többmodellű LLM-kollaboráció alaprendszerének tervezése és megvalósítása

Rendszerleírás, architektúra és működésbemutatás

Konzulens: Dr. Pitlik László

Készítette: Kovács Bálint
Üzemmérnök-Informatikus
Alapképzési szak

Budapest
2025.

Kodolányi János University
Department of Informatics

AiFusion: Design and Implementation of a Core Framework for Multi-Model LLM Collaboration

System Description, Architecture, and Operational Demonstration

Supervisor: Dr. László Pitlik

Author: Bálint Kovács
Computer Science Operational Engineering
Bachelor's Degree,

Budapest
2025.

Tartalomjegyzék

Kivonat

A dolgozat az AiFusion platformot mutatja be, amely több nagy nyelvi modell együttműködését (master–slave–judge) használja annak vizsgálatára, hogy a kollektív válaszadás miként hat a megbízhatóságra, költségre és késleltetésre. A prototípus Python/Flask alapú backendből, React webes felületből és konzolos batch-runnerből áll; token-alapú hozzáférés-védelemmel és részletes token/költség-naplózással. A fókusz a rendszerarchitektúra és a megvalósítás; a bemutatott kismintás mérések demonstrációs célúak. Általános, statisztikailag megalapozott teljesítményjavulásra nem teszünk állítást; ugyanakkor előzetes megfigyeléseink szerint a bíró-logika képes saját korábbi válaszával szemben is elfogulatlanul dönteni, és a kollaboráció költség-/késleltetés-kompromisszumokkal jár. Hozzájárulások: moduláris orkesztráció, bíró-prompting munkafolyamat, Pareto-alapú triádválasztás, valamint reprodukálható kötegelt futtatás mérési adatrögzítéssel. Jövőbeli munka: backend-oldali orkesztráció, gazdagabb mérőszámok és a hallucináció/bias mérséklésének célzott vizsgálata.

Abstract

This thesis presents the AiFusion platform, which orchestrates the collaboration of multiple large language models (master–slave–judge) to examine how collective answering affects reliability, cost, and latency. The prototype comprises a Python/Flask backend, a React-based web interface, and a console batch runner, and includes token-based access control with detailed token/cost logging. The focus is on system architecture and implementation; the reported small-sample measurements are demonstrative. We do not make a general, statistically substantiated claim of performance improvement; however, preliminary observations indicate that the judge logic can adjudicate impartially even against its own prior answer, and that collaboration entails cost/latency trade-offs. Contributions include modular orchestration, a judge-prompting workflow, Pareto-based triad selection, and reproducible batch execution with measurement logging. Future work includes backend-side orchestration, richer evaluation metrics, and a targeted investigation into mitigating hallucination and bias.

1. Bevezetés

1.1. A szakdolgozat témájának felvezetése

A mesterségesintelligencia-kutatásban az utóbbi években kiemelt figyelmet kaptak a nagy nyelvi modellek (Large Language Model – a továbbiakban: LLM). E modellek – mint például az OpenAI által fejlesztett ChatGPT vagy az Anthropic Claude – jelentős előrelépést hoztak a különböző, korábban ember által végzett feladatok *„felgyorsításában”*, automatizálásában (Noy & Zhang, 2023), (Dell'Acqua, és mtsai., 2023). Ez a trend a hazai szakirodalomban is megjelent, ahol a modelleket már specifikus, magyar nyelvű *„szakértői rendszerek”* (pl. tűzvédelmi) létrehozására is alkalmazzák (Karsa, 2024)

Az első gondolat a nagy nyelvi modellek kollaborációs vagy konzíliumos alkalmazásáról egy, a konzulensem és tanszékvezetőm Dr. Pitlik László által írt levél értelmezése során vetődött fel. A levélben a szakdolgozat leadási határidejét kellett megadni a válaszlevél tárgyában, pontosan, megadott szabályok szerint. A levelet a ChatGPT és a Claude aktuális legfrissebb, webes felületen elérhető verziójával is megválaszoltattam. A ChatGPT hibásan értelmezte a levelet, a Claude azonban pontos választ adott.

A második eset szintén Dr. Pitlik Lászlóhoz köthető: az egyik órai feladat során valahogy szóba került egy találós kérdés, amelyet hibásan értelmezett valamelyik nagy nyelvi modell. A találós kérdés így szólt: „Hogyan ejtsünk le egy friss tojást a szilárd betonra úgy, hogy az ne törjön össze?”. A kérdésben az „az” szó értelmezése félrevezető, mivel az utolsó tárgy, amelyre a szó rámutat a beton, tehát nem a tojás összetörésére irányul a kérdés, hanem a betonéra. Ebből következően a helyes válasz: a tojás bárhogyan leejthető, a beton nem fog összetörni a tojástól. Ezen találós kérdést több elérhető nyelvi modellnek is feltettem, viszont újra csak egy részük volt képes helyesen megválaszolni azt. Innentől kezdve elkezdtem gyűjteni az ismert válasszal rendelkező fejtörőket és találós kérdéseket, majd azokat több nyelvi modellel is megválaszoltattam és felfigyeltem arra, hogy általában legalább egy LLM mindig helyesen oldotta meg a feladatokat. Innen következett a feladat: összeköttetést kell teremtenem a nyelvi modellek között, hogy azok erősségeit kihasználva a lehető legmagasabb helyes válaszarányt kaphassam.

Az első kísérletek során kézi összeköttetést hoztam létre a ChatGPT, a Claude és a DeepSeek rendszerek között. Az összeköttetés a kezdeti promptok megadása után, a LLM-ek válaszainak kézi továbbítását jelentette meghatározott szabály szerint a soron következő LLM részére, így gyakorlatilag a LLM-ek egymással kommunikálni tudtak. A kísérlet során különböző logikai fejtörőket kellett megoldaniuk a nyelvi modelleknek összefogással. A rövid kísérletezés során a LLM kollaboráció akkor is helyes választ adott, ha egy vagy több, a kérdés megválaszolásában résztvevő LLM válasza hibás volt. A LLM kollaborációk egyszerűbb tanulmányozásának céljából elkezdtem építeni a platformot, melynek neve **AiFusion** lett. A konzulensem által felügyelt, mesterséges intelligenciával és annak alkalmazásaival is foglalkozó „*Magyar Internetes Agrár/Alkalmazott Informatikai Újság (MIAÚ) kutatási portál*" (MIAÚ, 2025) biztosította azt a hátteret, amelyben a felvetődött ötletek kidolgozásra kerülhetettek.

Az AiFusion egy olyan platform, amely több különböző mesterségesintelligencia-modell együttműködésével ad választ a felhasználó kérdéseire. A rendszer jellemzően egyszerre párhuzamosan több nagy nyelvi modellt von be a válaszadásba, majd a kapott válaszokat kiértékeli. A kapott válaszok egymásra épülnek, az AiFusion rendszerben meghatározott struktúra szerint, így a nagy nyelvi modellek válaszai egymásra hatással vannak, ezzel egyfajta kollektív (kollaborációs), AI-válaszadást valósítva meg. Az AiFusion platform célja, hogy az alábbi kérdésekre segítse a válaszadást:

- Több LLM-től származó válasz egyesítésével jobb minőségű és megbízhatóbb választ kapunk, mintha csupán egyetlen modellre támaszkodnánk?
- Több nyelvi modell összesített válaszköltsége és válaszminősége milyen arányban áll egy nagy válaszköltségű LLM-hez viszonyítva?
- Több nyelvi modell összesített válaszideje és válaszminősége milyen arányban áll egy nagy válaszidejű LLM-hez viszonyítva?
- A különböző LLM kollaborációk válaszidejei és válaszminőségei milyen arányban állnak egymáshoz képest?

Az AiFusion alapötlete párhuzamba állítható a *"wisdom of crowds"* (tömegek bölcsessége) elvével, miszerint több független vélemény kombinálása vezethet pontosabb eredményhez (Surowiecki, 2004.) . Hasonló elv érvényesül a gépi tanulásban az *„ensemble*

módszereknél" is, ahol több modell együttes döntése javíthatja az előrejelzések pontosságát (Dietterich T. G., 2000) .

Az AiFusion rendszer jelenlegi prototípusa három fő komponensből áll: egy szerveroldali háttérrendszerből, egy webes frontendből az egyedi kérdések és LLM konfigurációk teszteléséhez, valamint egy batch feladatok futtatására alkalmas konzolos alkalmazásból. A backend egy Python nyelven, Flask keretrendszerrel készült Alkalmazásprogramozási Felület (Application Programming Interface – a továbbiakban: API) szerver, amely kezeli a beérkező kérdéseket és továbbítja azokat a megfelelő AI modellhez, majd a kapott választ visszaküldi a frontendre. A webes frontend egy React (egy JavaScript programkönyvtár felhasználói felületek építéséhez) alapú felület, amely lehetővé teszi a felhasználónak a modellek kiválasztását, a paraméterek (pl. kreativitás, maximális token-hossz) beállítását, a kérdés beküldését, valamint a párhuzamosan futó modellek válaszainak áttekintését. A konzolos alkalmazás a backendhez hasonlóan Pythonban íródott és Flask keretrendszerrel készült, feladata a kérdésadatbázisok automatizált kollaborációs megválaszolása egy kimeneti fájlba. A rendszer jelenlegi implementációjában a "bíró" logika kísérleti jelleggel a frontenden valósul meg: a kliens program gyűjti össze a kiválasztott modellek válaszait, és egy újabb API hívással egy kiválasztott bíró-LLM-nek küldi el azokat kiértékelésre. E koncepció későbbi verzióiban már egy általánosabb, backend oldali orkesztrációs modul is kialakítható a modellek együttműködésének kezelésére.

1.2. Problémafelvetés és célkitűzés

A nagy nyelvi modellek egyedi használatának komoly korlátai vannak, amelyek korlátozzák a rájuk épülő rendszerek megbízhatóságát. Az egyik ismert jelenség a „*hallucináció"*, amikor a modell valósnak tűnő, de tényszerűen hibás vagy kitalált információt generál (Ji Z. , és mtsai., [25.] Survey of Hallucination in Natural Language Generation, 2023). Ezen túlmenően gyakori probléma a következetlenség is: a modell válaszai eltérhetnek egymástól azonos kérdés többszöri lekérdezésekor, vagy érzékenyen függhetnek a kérdés megfogalmazásától. A LLM-ek zárt doboz jellegű működése miatt a felhasználó sokszor nem kap visszajelzést a válasz bizonyosságáról sem – a generált szöveg magabiztos stílusa ellenére a háttérben *„nincs garancia arra, hogy az állítás igaz"* (Bang, és mtsai., 2023). Ezek a kihívások együttesen azt eredményezik, hogy egyetlen

modell kimenetére hagyatkozva a felhasználó nehezen tudja megítélni a kapott információ hitelességét és pontosságát.

Az AiFusion rendszeren keresztül ezen problémák megoldását kívánom kutatni, a kollektív válaszadás segítségével. Az az elképzelés, hogy több különböző LLM egyidejű alkalmazásával és a válaszaik egyesítésével a modellek komplementer tudását és erősségeit ki lehet aknázni. Különböző modellek eltérő tréningadatokon és architektúrával tanulnak, így hajlamosak különféle hibákra és tévedésekre - egy adott kérdésre egyik modell pontosabb, míg egy másik kreatívabb választ adhat. Ha ezen válaszokat egy magasabb szintű algoritmus (jelen esetben egy bíró LLM) értékeli és kombinálja, az remélhetőleg kiszűri az egyedi modellek tévedéseit és növeli a végső válasz megbízhatóságát. E szakdolgozat központi célkitűzése ennek a koncepciónak a megvalósítása: egy olyan rendszer tervezése és implementálása, amely több LLM együttes használatával megbízhatóbb választ ad a felhasználói kérdésekre, mint bármelyik résztvevő modell önmagában.

1.3. A kutatás részfeladatokra történő tagolása

A kollaborációs válaszadás kutatását a három fő egységre tagoltam:

1. A teszteléshez szükséges platform alapjainak megteremtése.
2. A kollaborációs válaszadás tesztelése, a kollaborációs-struktúrák finomhangolása, lehetőleg külső felhasználók (egyetemek, tanszékek) bevonásával, majd a válaszok alapszintű kiértékelése.
3. A kollaborációs válaszok mintázatának tudományos kiértékelése, amennyiben a mintázatok arra érdemesnek tekinthetőek.

A jelenlegi dolgozatban a kutatás első fázisát taglaljuk, vagyis a dolgozat a platform működésére fókuszál. A dolgozatban található kismintás tesztelés nem a kollaboráció minőségének megítélését szolgálja, hanem a platform működését hivatott demonstrálni.

1.4. Célcsoport

A jelen szakdolgozat elsősorban az alábbi csoportok számára lehet releváns és hasznos:

1.4.1. Mesterséges intelligenciával foglalkozó kutatók és hallgatók

Azok számára, akiket érdekel a nagy nyelvi modellek (LLM-ek) kollaborációjának lehetősége, az ensemble módszerek alkalmazása a megbízhatóság növelésére, valamint az ilyen rendszerek tesztelésének módszertana. Az AiFusion platform bemutatása betekintést nyújt egy konkrét megvalósításba és a felmerülő kihívásokba (pl. modell-összehasonlíthatóság, referenciaadatok minősége). Mivel a rendszer tömeges „LLM ranking”-ként is funkcionál, a különböző LLM-ek gyors összehasonlíthatóságát is segítheti a platform.

1.4.2. Szoftverfejlesztők és üzemmérnök-informatikusok

Azok számára, akik gyakorlati példát keresnek modern webes architektúrák (Python/Flask backend, React frontend), API integráció (több LLM szolgáltató egységes kezelése) és automatizált tesztelési keretrendszerek (batch runner) megvalósítására. A dolgozat részletesen bemutatja az architektúrát (3. fejezet) és az implementáció során felmerült gyakorlati szempontokat (pl. biztonság, nevezéktan).

1.4.3. Szakértők és döntéshozók

Az AiFusion platform hasznos lehet bármely területen, ahol precíz, megbízható válaszokra van szükség komplex kérdésekben. A rendszer segíthet eldönteni, hogy egy adott szakterület kérdéskorpuszára melyik LLM vagy LLM-kollaboráció adja a legalkalmasabb válaszokat, lehetővé téve a legmegfelelőbb eszköz kiválasztását az adott domain specifikus feladataira. Ez különösen értékes lehet olyan területeken, mint a jog, az orvostudomány, a pénzügy vagy a mérnöki tudományok.

1.4.4. IT döntéshozók és projektmenedzserek

Bár a dolgozat jelen fázisban nem tesz állítást a teljesítményjavulásról, betekintést nyújthat a többmodellű megközelítésekben rejlő potenciálba és az ezzel járó költség- és

komplexitásbeli kompromisszumokba. A „4. Vita” fejezetben tárgyalt módszertani korlátok tanulságosak lehetnek hasonló rendszerek bevezetésekor.

A dolgozat technikai mélysége miatt a szélesebb laikus közönség valószínűleg nem tartozik a közvetlen célcsoportba.

1.5. Hasznosság

Az AiFusion platform és a jelen szakdolgozatban bemutatott kutatás több szempontból is hasznos lehet a „1.4 Célcsoport fejezetben” azonosított szereplők számára:

1.5.1. Megbízhatóság növelése

A platform alapvető célja annak vizsgálata, hogy a kollektív válaszadás („2.4 Kollektív (ensemble) AI-megközelítés”) révén csökkenthetők-e az egyes LLM-ekre jellemző korlátok, mint a hallucináció és a bias („2.5 Etikai és megbízhatósági kihívások... „). Ha ez igazolható, az AiFusion (vagy hasonló rendszerek) hozzájárulhatnak a megbízhatóbb MI-alapú válaszok generálásához kritikus területeken is.

1.5.2. Objektív modellválasztás támogatása

A platform lehetővé teszi különböző LLM-ek és kollaborációs sémák szisztematikus tesztelését és összehasonlítását („3.3. Kismintás tesztelés”). Ez segítheti a szakértőket és döntéshozókat abban, hogy adatvezérelt módon válasszák ki a saját specifikus kérdéskörükre legalkalmasabb modellt vagy modellkombinációt, ahelyett, hogy csupán egyetlen, általános célú modellre hagyatkoznának.

1.5.3. Kutatási és oktatási platform

Az AiFusion nyílt architektúrája (3. fejezet) és automatizált tesztelési képességei („3.2.7. Batch Runner modul”) kísérleti platformként szolgálhatnak további MI (Mesterséges Intelligencia)-kutatásokhoz (pl. új kollaborációs stratégiák, prompt engineering technikák tesztelése) és az oktatásban is felhasználható a modern LLM technológiák gyakorlati bemutatására.

1.5.4. Költség- és teljesítményelemzés

A rendszer részletes naplózása („3.2.12. Naplózás és adatvédelem”, „3.3.5. A kimeneti eredmények tárolása, kiértékelése„) lehetővé teszi a különböző modellek és kollaborációk költségének és válaszidejének elemzését. Ez segíthet az optimális költség-teljesítmény arány megtalálásában különböző felhasználási esetekre.

A projekt eddigi fejlesztése során az alábbi idő és pénzeszközök kerültek felhasználásra:

- API hívások költségei: 35 amerikai dollár (United States Dollar – a továbbiakban: USD)
- ChatGPT előfizetés díja: 20 USD / hó, durván 6 hónapnyi előfizetés $\Rightarrow$ 120 USD
- aifusion.hu domain bejegyzésének díja: 2 286,- magyar forint (Hungarian Forint, a továbbiakban: HUF)
- Fejlesztői MacBook Air M1 megvásárlása: 170 000.- HUF
- Durván 150-200 gép előtt töltött „munkaóra”

Az AiFusion rendszer fejlesztésével nem a közvetlen profitszerzés a cél, hanem a kifejlesztett technológia egyéb, saját projektjeimbe való beépítése és a technológia fejlesztése során felhalmozott tudásbázis későbbi hasznosítása.

A platform a következő fejlesztési fázisában nyilvánosan elérhetővé kell váljon kutatási és oktatási célból. A nyilvános eléréshez az alábbi költségbecslést végzem:

    - Az AiFusion platform ezen szakaszához nem keletkezik pénzügyi tranzakció az „ügyfelek” és a platform között. Vagy a platform saját API kulcsait használja a rendszer (vagyis a platform üzemeltetője finanszírozza az API hívásokat a LLM szolgáltatók felé), vagy a felhasználóknak rendelkezniük kell saját API kulcsokkal közvetlenül a LLM szolgáltatók felé. Tehát az AiFusion platform sem közvetlen szolgáltatóként sem szolgáltatásközvetítőként nincs jelen a piacon (lásd: „3.2.13.4. Jogi háttér szerkezete”).
- Ebben az esetben az „ügyfelek” felé pusztán adminisztratív kötelezettségei vannak az üzemeltetőnek, tehát leginkább az adatkezelési és felhasználási

feltételekkel kell foglalkozni. Korábbi munkatapasztalatom szerint egy nemzetközi ügyvédi irodával egy ilyen jellegű szerződést meg lehet íratni 1 000 EUR környékén.

- Mivel a platform részéről ezen a ponton már meg kell nevezni azt a célszerűen jogi személyt, aki a szolgáltatást nyújtja, valamelyik megfelelő TEÁOR-ral (Tevékenységek Egységes Ágazati Osztályozási Rendszere) rendelkező cégem portfoliójához adnám az AiFusion platform üzemeltetését.
- A rendszer ezen a ponton már publikus eléréssel rendelkezik. Vagy saját szervert és biztonsági protokollt alkalmazunk, vagy egy felhőszolgáltatóba költöztetjük a rendszert (pl. Amazon Web Services – a továbbiakban: AWS). Mivel a biztonsági rések foltozása és a hackertámadások elhárítása állandó feladatot adna saját szerver esetén, érdemesebbnek gondolom a szolgáltatást pl. AWS-be költöztetni. Az AWS áráról nincs pontos kalkulációm.
- A szükséges frontendi fejlesztés szubjektív becslésem alapján 200-250 munkaóra.
- A backend fejlesztése szubjektív becslésem szerint legalább 100 munkaóra.
- Ezen a ponton szükségesnek látom legalább egy magyarországi védjegy bejegyzése az esetleges jogviták, téves Facebook tiltások gördülékenyebb kezelése céljából. A magyar védjegy bejegyzésének költsége egy áruosztály esetén jelenleg *„81 000.- Ft + ügyvédi költség”* (az ügyvédi költség korábbi védjegybejegyzéseim alapján plusz 60-80 ezer forint) ([27.] Magyar, uniós és nemzetközi védjegy díjak, 2025).

A megfelelő publikus tesztelést követően lehetne megállapítani a rendszer működésének tényleges hasznosságát (amely még nem bizonyított!), a rendszer piacképes szolgáltatásait leírni és azok értékét felbecsülni. Mivel szerintem (megfelelő hasznosság esetén) a rendszert legfeljebb egy szűk réteg használná, amennyiben a közvetlen profitszerzés lenne a cél, reálisabbnak látom a kifejlesztett technológiák (algoritmusok, módszertanok) értékesítését más, nagy nyelvi modellekkel foglalkozó piaci szereplők felé.

1.5.5. Referenciaadatok validálása és a váratlan hasznosság

Ahogy a „4. Vita” fejezetben is kiemelésre került, a kollaboratív válaszok konzisztenciájának elemzése potenciálisan segíthet a meglévő tesztadatbázisok hibáinak vagy kétértelműségeinek feltárásában.

1.6. A dolgozat kialakításának áttekintése

Az alábbiakban először a szakdolgozat szerkezeti felépítését ismertetem, majd bemutatom azokat a stílus- és formázási elveket, illetve fókuszpontokat és elhatárolásokat, amelyek a dolgozat további fejezeteinek kialakítását meghatározzák.

1.6.1. Szerkezeti felépítés

A dolgozat hátralévő része az alábbiak szerint épül fel:

Az **2. fejezet** áttekinti a témához kapcsolódó elméleti és szakirodalmi hátteret. Ebben kitérek a nagy nyelvi modellek működésének alapjaira, a szöveggenerálás és a paraméter-beállítások (pl. *temperature*) szerepére, valamint bemutatom a több modell együttes alkalmazásának (*ensemble*) koncepcióját és a benne rejlő lehetőségeket.

A **3.1. fejezet** konkretizálja az AiFusion rendszerrel szemben támasztott követelményeket, és bemutat egy tipikus use-case forgatókönyvet, amely szemlélteti a platform gyakorlati használatát.

A **3.2. fejezet** bemutatja a rendszer részletes tervezését és architektúráját, bemutatva a backend és frontend fő komponenseit, azok kapcsolatait és a kommunikációs folyamatokat, fókuszálva a megvalósítás részleteire, kitérve a fejlesztés során alkalmazott technológiákra, eszközökre, valamint a kritikus implementációs kihívásokra és azok megoldásaira.

A **3.3. fejezet** egy validációs tesztet és egy próbaüzemet mutat be, a teszten elért eredmények kiértékelése nélkül.

Végül a **4. fejezet** igyekszik kritikusan elemezni és összefoglalni a szakdolgozatban tárgyalt témákat, módszertanokat.

1.6.2. Stílusjegyek, formázás

A dolgozat formázása során több, tudatosan felvállalt szerkesztési elvet követtem:

- Akadémiai és Intézményi Megfelelés: Az *„alapvető formázás”* (mint a címsorok hierarchiája, tartalomjegyzék, ábrajegyzék) a Kodolányi János Egyetem szakdolgozati követelményeit (Kodolányi János Egyetem, 2021), valamint az általános tudományos publikációs gyakorlatot követi. A *„helyesírási szabályok”* alkalmazása során a magyar helyesírás szabályainak 12. kiadása az iránymutató (MTA, 2025). Az irodalmi hivatkozások rendszere az Amerikai Pszichológiai Társaság (American Psychological Association – a továbbiakban: APA) *„szabványon”* alapul (APA, 2024).
- Konzulensi Iránymutatás (Explicit Hivatkozás): A konzulensi visszajelzések alapján a dolgozat egyedi formázási szabályt alkalmaz a forráshivatkozások egyértelműsítésére. Minden külső forrásmegjelölés (pl. (Szerző, 2024)) egy, a szövegtörzsben „dőlt betűvel és idézőjellel” kiemelt kulcsfogalomhoz vagy szó szerinti idézethez kapcsolódik. Ennek a felvállalt formázásnak a célja az olvasó segítése: egyértelműen és visszakereshetően jelöli, hogy a hivatkozás pontosan melyik állítást, definíciót vagy adatot támasztja alá, elkerülve a bekezdés végén elhelyezett, többértelmű hivatkozás-csomagokat.
- Technikai Tartalom Érthetősége: Mivel a dolgozat egy szoftverarchitektúrát és annak implementációját mutatja be, a jobb érthetőség érdekében a szöveg olyan vizuális elemeket is tartalmaz, mint kódrészletek („10. ábra”), parancssori kimenetek („2. ábra”, „11. ábra”) és felhasználói felületi képernyőképek („6-9. ábrák”). Ezeket a releváns szövegkörnyezetbe ágyazva, ábrajegyzékkel és forrásmegjelöléssel ellátva használom.

1.6.3. A szakdolgozat fókuszpontjai és elhatárolása

A dolgozat fókusza tudatosan az AiFusion platform alaprendszerének megtervezésére, implementálására („3.2. Rendszerfelépítés és implementáció”) és működőképességének demonstrálására („3.3. Kismintás tesztelés”) helyeződött. A terjedelmi és időbeli korlátok miatt azonban számos érdekes, kapcsolódó kutatási irányt csak érintőlegesen, vagy egyáltalán nem tárgyalok:

- **Részletes kvalitatív elemzés:** A dolgozat nem végez mélyreható kvalitatív elemzést arról, hogy a „Bíró” modell milyen nyelvi, logikai vagy szemantikai mintázatok alapján hozza meg döntéseit.
- **Prompt Engineering mélyvizsgálata:** Bár a platform lehetővé tenné, a dolgozat nem tér ki egy nagyszabású kísérletre, amely azt vizsgálná, hogy a „Bíró” modellnek adott különböző instrukciók (pl. „válaszd a legjobbat”, „fésüld össze a válaszokat”, „elemezd kritikusan és írd újra”) hogyan befolyásolják a végeredmény minőségét, költségét és sebességét.
- **Célzott etikai és bias-mérés:** A 2.5 Etikai és megbízhatósági kihívások... fejezet felveti a témát, de a dolgozat nem végez célzott kísérleteket a kollaboráció bias-csökkentő hatásainak mérésére (pl. specifikus társadalmi torzításokra fókuszáló tesztkészletekkel).
- **Alternatív kollaborációs architektúrák:** Az AiFusion a „master-slave-judge” (post-inference) modellt valósítja meg. A dolgozat terjedelmi okokból nem tér ki más, a szakirodalomban említett modellek (pl. „router” alapú vagy valós idejű, token-szintű „inference-közbeni” kollaborációk) gyakorlati implementálására és összehasonlítására.

Ezek a területek mindegyike önmagában is elegendő témát szolgáltatna egy különálló kutatáshoz, amely a jövőben az AiFusion platformra épülhet. A fejlesztések további lehetőséget a „3.2.13. Jövőbeli fejlesztési irányok” fejezet tárgyalja.

2. Szakirodalmi háttér

A szakdolgozat témájának alapját adó mesterséges intelligencia kutatása Magyarországon sem új keletű. A hazai tudományos közösség már a 2000-es évek közepén aktívan foglalkozott a haladó *„számítástechnikai kutatás”* és a *„mikroprocesszor-vezérelt alkalmazások”* területeivel, amelyek megalapozták a későbbi, modernebb modellek kutatását is (Magyar Tudomány, 2005). Nemzetközi szinten a téma *„tudásbázisának gyűjtése”* szintén korán elkezdődött, olyan meghatározó intézmények portáljain, mint például az Association for the Advancement of Artificial Intelligence (rövidítve: AAAI) (AAAI, 2009). Ezzel párhuzamosan a téma a hazai felsőoktatásban is megjelent; a mesterséges intelligencia és a neurális hálók már 2010 előtt is a magyar *„egyetemi tantervek részét képezték”,* például az ELTE programtervező matematikus képzésében (ELTE TTK, 2008). Maga a KJE/SZIE kutatási környezet is régóta foglalkozik a témával a magyar nyelvű „*MIAÚ portálon*” keresztül (MIAÚ, 2014).

2.1. Nagy nyelvi modellek működése és az embedding alapelvei

A nagy nyelvi modellek (LLM-ek) a bemenetként kapott szöveget belső numerikus reprezentációkká alakítják, hogy ezen keresztül sajátítsák el a nyelv mintázatait. Ennek kulcsa a „*szavak vektoralapú megjelenítése”*, azaz az „*embedding”* minden szó vagy token egy magas dimenziójú vektortér egy pontjaként jelenik meg (Pennington, Socher, & Manning, 2014). E pontok elrendezése úgy alakul, hogy a „*hasonló jelentésű*” vagy használatú szavak vektora közel kerüljön egymáshoz, (Turtogtokh, Pitlik, & Pitlik, 2025.), (Kollár, 2011), míg *„eltérő jelentésű szavak távolabb”* legyenek (Pennington, Socher, & Manning, 2014). Ez a szemlélet, amely a döntéshozatalt a *„hasonlóság-elemzésre”* alapozza, már korábbi intézményi kutatásokban is megjelent (Pitlik L. , [5.] MY-X: CONT-ROLLING-STONES, 2011). A LLM nem konkrét szavakat tárol, hanem a szavak közötti kapcsolatok sokdimenziós mintázatait: numerikusan kifejezhető a jelentésbeli hasonlóság azáltal, hogy két szó vektorának távolsága vagy szöge kicsi (pl. *“*kutya*”* és *“*macska*”* közel esnek, míg a *“*autó*”* messze helyezkedik el tőlük az embedding térben). Az ilyen „*word embedding”* technikák alapját olyan algoritmusok teremtették meg, mint a Word2Vec vagy a GloVe, melyek nagy szövegkorpuszokban figyelik meg, mely szavak

fordulnak elő gyakran hasonló kontextusban (Pennington, Socher, & Manning, 2014). Pennington és munkatársai megfogalmazása szerint a GloVe modell mögötti fő intuíció az az egyszerű megfigyelés, hogy *"ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning"* (Pennington, Socher, & Manning, 2014), vagyis: „a szavak együttes előfordulási valószínűségeinek arányaiban megvan a potenciál, hogy a jelentés valamilyen formáját kódolják". Ennek eredményeként a LLM a nyelvi tudást efféle szemantikus térben tárolja, ami lehetővé teszi számára, hogy a *„jelentéssel bíró hasonlóságokat"* matematikailag kezelje, miközben „a magas dimenzió" (gyakran 100-nál is több) gondoskodik arról, hogy a nyelv finom árnyalatai is reprezentálhatók legyenek, és a modellek *„megfelelő általánosító képességgel"* rendelkezzenek (Pennington, Socher, & Manning, 2014).

A LLM-ek belső működésének másik alappillére az ún. *„transformer architektúra"*, amely *„figyelem (attention) mechanizmust"* használ a szövegösszefüggések kezelésére. Ennek részletei meghaladják e fejezet kereteit, de lényeges, hogy a modell képes hosszú tartalmakban is felismerni a releváns összefüggéseket (Vaswani, és mtsai., 2017). A transformer alapú LLM először beágyazott vektorok sorozataként értelmezi a bemenetet (tokenenként egy vektorral), majd több rétegen keresztül – figyelem fejtők alkalmazásával – egyre absztraktabb reprezentációkat épít. Végül ezen *„reprezentációk segítségével"* hozza létre a kimenetet, összességében tehát a LLM-ek a nyelvet *„statisztikai mintázatok"* formájában ragadják meg: a tanulás során súlyokat hangolnak be úgy, hogy a lehető legjobban modellezzék, milyen szavak és kifejezések hogyan kapcsolódnak egymáshoz a természetes nyelvben (Vaswani, és mtsai., 2017).

2.2. Tokenizáció és a szöveggenerálás logikája

Mielőtt a modell generálni kezdene bármilyen választ, a bemeneti szöveget először fel kell dolgoznia. Ennek első lépése a tokenizáció, amelynek során a nyers szöveget kisebb elemekre – tokenekre – bontja a rendszer. A token lehet egy teljes szó, szótő, szótag vagy akár egyetlen karakter, a használt tokenizáló algoritmustól függően. A modern LLM-ek gyakran ún. *„subword tokenizálást alkalmaznak (pl. Byte-Pair Encoding, WordPiece)*", amely biztosítja, hogy a gyakori szavak egy tokenként jelennek meg, míg a ritkább vagy összetett szavak több tokenre bonthatók (Microsoft, 2025.). A tokenizáció célja, hogy a szöveg megfelelő formátumú numerikus sorozattá (token indexek sorává) alakuljon,

amelyen a modell már tud műveleteket végezni. A Microsoft dokumentációja szerint a tokenek *"pieces of words, syllables, or even characters that text is broken down into for the model to process."* (Microsoft, 2025.), vagyis: „szavak darabjai, szótagok vagy akár karakterek, amelyekre a szöveget felbontják, hogy a modell fel tudja dolgozni". Például a *"*Megyek haza*"* mondat tokenjei lehetnek ["Meg", "yek", " haza"] egy Bájtpár Kódolás (Byte-Pair Encoding – a továbbiakban: BPE) alapú tokenizálás esetén, ahol a szóvégi rag külön tokenként jelenik meg. Minden tokenhez a modell egy *„egyedi azonosítót és egy hozzárendelt vektor-(embedding) reprezentációt"* tart nyilván (Microsoft, 2025.).

Miután a bemenet tokenek formájában a modellhez került, megindul a szöveggenerálás folyamata. A LLM a dekóder (szövegkibocsátó) részében iteratív módon állítja elő a kimenő mondatot tokenről tokenre. Minden lépésben kiszámít egy „*valószínűségi eloszlást*" a lehetséges következő tokenekre – azaz megjósolja, hogy az addigi szövegkörnyezet alapján mi lehet a következő szó vagy részlet (Sybrandwildeboer, 2025). Ezt tekinthetjük úgy, hogy a modell minden pillanatban rangsorolja a szókincs összes lehetséges folytatását egy valószínűségi értékkel. Például egy megfelelően betanított modellnél, ha a bemenet a "A macska leugrott a …", akkor a következő tokenre olyan valószínűségi eloszlást adhat, hogy "kanapéról" 80%, "falról" 15%, "tégláról" 0.5% stb. – attól függően, a *„tréningkor mely folytatások voltak jellemzőek"* (Sybrandwildeboer, 2025). Alapértelmezésben a modell a legmagasabb valószínűségű token(eke)t választja ki következőnek, és hozzáfűzi a szöveghez. Ezt követően az így kibővült szöveg végét veszi figyelembe *„új kontextusként"* (Sybrandwildeboer, 2025), és *„megismétli a predikciós lépést"* a soron következő tokenre, (Microsoft, 2025.). Ez a folyamat iteratívan halad előre mindaddig, amíg a modell le nem zárja a választ (például egy Szekvencia Vége (End of Sequence – a továbbiakban: EOS*)* vagyis a válasz vége token kiadásával, vagy amíg el nem éri a megengedett maximális hosszt). Fontos kiemelni, hogy a modell *„a teljes eddigi kontextust figyelembe veszi"* minden egyes lépésnél – nem csupán az utolsó néhány szót – , így képes a hosszabb távú nyelvtani és szemantikai összefüggésekre is reagálni (Sybrandwildeboer, 2025). Ennek köszönhető, hogy egy LLM nagyobb szövegablak esetén is következetes maradhat és visszautalásokat kezelhet.

A tokenizáció és a szekvenciális szövegépítés kombinációja azt eredményezi, hogy a LLM-ek meghökkentően koherens mondatokat tudnak létrehozni pusztán a statisztikai mintázatok követésével. Ugyanakkor e működési logikából fakadnak a korlátai is: mivel a

modell mindig a legvalószínűbb (vagy majdnem legvalószínűbb) folytatást igyekszik adni, nincsen beépített mechanizmusa az igazságtartalom ellenőrzésére – erről még lesz szó a „2.5.2 Hallucináció" alfejezetben az ún. hallucináció kapcsán. Mindenesetre, a tokenizáció precíz megválasztása és a generálási algoritmus (dekódolási stratégia) nagyban befolyásolja a modell válaszainak minőségét és jellegét. A *„dekódolási stratégia"* (Ji Z. , és mtsai., [25.] Survey of Hallucination in Natural Language Generation, 2023) lehet például *„greedy search"* (mindig a top1 tokent választva) (Brown, és mtsai., 34. Language models are few-shot learners), *„beam search"* (egyszerre több lehetséges folytatást követve nyomon) (Sennrich, Haddow, & Birch, 2016), vagy *„random sampling"* (Hugging Face, 2025) különféle paraméterekkel (pl. *„nucleus sampling"* top-p szerint), (Holtzman, Buys, Du, Forbes, & Choi, 2019).

2.3. Kreativitás vs. determinizmus: a temperature paraméter szerepe

A szöveggenerálásnál megismert valószínűségi eloszlásokból fakadóan a modellek kimenete többé-kevésbé randomizálható. Ennek szabályozására szolgál a **temperature (hőmérséklet)** nevű paraméter, amely közvetlenül befolyásolja, mennyire legyen „kreatív" vagy éppen kiszámítható a modell válasza. Matematikailag *a „temperature a kimeneti valószínűségi eloszlás simítását vagy élesítését végzi*": alacsony értéknél az eloszlás csúcsosabbá válik (a legvalószínűbb tokenek dominálnak), míg magasabb értéknél laposabb (a kisebb valószínűségű tokenek is nagyobb eséllyel bekerülhetnek a mintavételbe) (Microsoft, 2025.).

Gyakorlati szempontból ez azt jelenti, hogy egy nagyon alacsony temperature (pl. 0.0–0.2) esetén a modell szinte mindig a legvalószínűbb következő szót választja – ezzel determinisztikus és jellemzően pontos, de olykor *„száraz"* vagy sablonos választ ad (Sybrandwildeboer, 2025). Shelar ezt úgy foglalja össze: *"Low-temperature values make the output more deterministic, favoring the most likely words."* (Shelar M. , 2025), vagyis: „Az alacsony hőmérsékleti értékek determinisztikusabbá (kiszámíthatóbbá) teszik a kimenetet, előnyben részesítve a legvalószínűbb szavakat.". Ilyenkor a kreativitás minimális: ha kétszer tesszük fel ugyanazt a kérdést a modellnek azonos paraméterekkel, jó eséllyel szó szerint ugyanazt a választ kapjuk vissza. **Ezzel szemben**, ha magasabb a temperature (pl. 1.0–1.2, vagy extrém esetben még magasabb), a modell a lehetséges

folytatások közül arányosabban válogat, beleértve a kevésbé valószínű opciókat is. Ennek eredménye egy **változatosabb, kreatívabb szöveg**, amely viszont kevésbé kiszámítható, és nagyobb eséllyel tartalmaz kisebb pontatlanságokat vagy stílusbeli meglepetéseket. Például a "A macska leugrott a …" mondat folytatása temperature=0.2 mellett nagy valószínűséggel "kanapéról." lesz, míg temperature=1.2 esetén akár "napfényben úszó kerítés tetejéről, egyenest az udvar puha füvére." jellegű kreatív folytatást is kaphatunk. Látható, hogy utóbbi jóval *„színesebb és részletgazdagabb, noha az információ szempontjából nem feltétlen pontosabb"* (Sybrandwildeboer, 2025).

A temperature paramétert gyakran a **0 és 1,5 közötti tartományban** állítják be a gyakorlatban. A 0 közeli értékek a maximum valószínűségre törő (argmax) szövegkibocsátást közelítik – ilyenkor a modell determinisztikusan működik, mintha egy *„autokomplet"* funkció mindig a legkézenfekvőbb folytatást fűzné hozzá a mondathoz (Shelar M. , 2025). A ~1.5 érték körüli temperature ezzel szemben a teljes tanult eloszlást kihasználja, ami kreatív, néha meglepő fordulatokat hozó válaszokat eredményez. Fontos azonban megjegyezni, hogy magas temperature esetén a modellek hajlamosak lehetnek koherenciavesztésre: a szöveg összefüggéstelenebbé válhat, mert túlságosan laposssá tesszük az eloszlást (szinte véletlenszerű választást engedélyezve).

A **kísérleti beállításoknál** a temperature 0,0. Érdekes megközelítés, amely a következő fejezetben tárgyalt kollektív modell használatra is kihat, hogy különböző LLM-eket vagy akár ugyanazon LLM több példányát eltérő temperature értékekkel futtatjuk párhuzamosan. Például egy kísérletben az egyik modell nagyon alacsony kreativitással (0,0), a másik nagyon magas kreativitással (pl. 1,5) generál választ, majd egy meta-szintű algoritmus vagy egy bíró modell kombinálja ezeket – így egyszerre van egy „ultraracionális" és egy „ultrakreatív" nézőpontunk, amelyekből egyensúlyozott végső válasz születhet. Ez a fajta kontrasztív beállítás növelheti a válaszok diverzitását, ugyanakkor biztosítja, hogy a végső outputban a megbízhatóság is megmaradjon.

2.4. Kollektív AI-megközelítés és előnyei

A mesterséges intelligencia és gépi tanulás területén régóta ismert módszer, hogy *„több modell együttes alkalmazásával (ensemble)"* javítható az előrejelzések pontossága és megbízhatósága (Breiman, 1996). Ennek hátterében az a statisztikai jelenség áll, hogy a különböző modellek különböző hibákat vétenek, és ha okosan kombináljuk a

kimeneteiket, a hibák egy része kiegyenlítheti egymást. Dietterich definíciója szerint az ensemble módszerek *"construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions"* (Dietterich T. G., 2000). Példák erre a bagging, boosting, stacking algoritmusok a gépi tanulásban, illetve az olyan meta-modellek, mint a random forest (ami sok döntési fát átlagol) vagy az **átlagos szakértő** módszere (ami több szakértő véleményéből képez konszenzust). A **kollektív intelligencia** elve – miszerint több független vélemény vagy forrás kombinációja jobb megoldást adhat, mint bármelyik önmagában – a mesterséges intelligenciában is kamatoztatható.

Mivel a LLM-ek eltérő szövegkorpuszon tanultak és más fejlesztésű algoritmusokat alkalmaznak a válaszadás során, ezért erősségeik és gyengeségeik különbözhetnek, adódik az ötlet, hogy ne egyetlen modelltől várjunk minden kérdésre tökéletes választ, hanem *„több modell válaszát kombinálva"* próbáljunk meg jobb eredményt elérni (Chen, és mtsai.). Egy friss kutatási áttekintés rámutat, hogy a különböző LLM-ek jelentősen eltérhetnek egymástól a felépítésük (architektúra, paraméterszám), a tokenizációs módszerük, a tanítási adatuk és egyéb tényezők miatt – emiatt *„ugyanarra a kérdésre is eltérő válaszokat adhatnak"* (Chen, és mtsai.). Ha ezeket az eltérő perspektívákat kombináljuk, esély van rá, hogy *„összességében jobb teljesítményt kapunk",* hiszen az egyik modell korrigálhatja a másik pontatlanságát, vagy kiegészítheti annak válaszát (Chen, és mtsai.). Ez hasonló ahhoz, mint amikor egy orvosi diagnózisnál több orvos véleményét kérjük ki (*második vélemény*): a végső diagnózis megbízhatósága nő.

A szakirodalomban megjelentek már konkrét módszerek a LLM-ek együttműködésére. Chen *„taxonómiája három fő kategóriába sorolja a LLM ensemble technikákat"* aszerint, hogy a kombináció mikor történik a folyamatban (Chen, és mtsai.): (1) *Inference előtt* – azaz a kérdés alapján előre kiválasztjuk, melyik modell valószínűleg a legjobb (úgynevezett router vagy szakosodott szakértők); (2) *Inference közben* – több modell párhuzamosan generálja a választ token szinten egymással kommunikálva, kvázi valós időben összeegyeztetve a kimenetet; (3) *Inference után* – minden modell megadja a teljes válaszát, majd egy utófeldolgozó mechanizmus (akár egy másik modell) dönt arról, hogy ezekből melyik(ek) a legjobb(ak), vagy hogyan lehet őket összeolvasztani. Az AiFusion megközelítése – ahogy a következő alfejezetben látni fogjuk – főként ebbe az utóbbi kategóriába esik: a külön modell-válaszok utólagos kombinálásával igyekszik a legjobb elemeket egyesíteni. A lényeg, hogy bármely stratégiát is választjuk, a *„több modell*

kollektív döntése” (Chen, és mtsai.) elméletben megbízhatóbb lehet, *„mint egy magányos modellé”*, (Kovács & Pitlik, 2025).

Az ensemble módszer előnyei tehát összefoglalva: növelheti az **pontosságot** (pl. kérdés-meg-válasz feladatoknál jobb találati arány), javíthatja a **robosztusságot** (egy-egy modell hibája kevésbé befolyásolja a végeredményt), és **csökkentheti a torzításokat** (ha az eltérő modellek más-más bias-szal rendelkeznek, azok részben kiegyenlíthetik egymást). Ugyanakkor megvannak a kihívásai is: a modellek együttműködésének összehangolása nem triviális (pl. konfliktusos válaszok esetén dönteni kell), növeli a számítási költséget, és a válasz magyarázhatóságát is bonyolíthatja, ha egy meta-szintű döntéshozó algoritmus dolgozik a háttérben. Ezen problémák egy részére az AiFusion egy sajátos megoldást kínál egy *bíró modell* formájában.

2.5. Etikai és megbízhatósági kihívások a LLM-eknél

A nagy nyelvi modellek alkalmazása kapcsán komoly etikai és felhasználói élménybeli aggályok merültek fel az utóbbi években. Ezek közé tartozik egyrészt a **bias** (torzítás, elfogultság) problémája, másrészt az úgynevezett **hallucináció**, azaz amikor a modell magabiztosan állít valótlanságokat.

2.5.1. Bias

A **bias** alatt azt értjük, hogy a modell tréningadatában meglévő társadalmi előítéletek, sztereotípiák, egyenlőtlenségek beépülnek a modell viselkedésébe, és így annak *„kimeneteiben is megjelennek”* (Kovács & Pitlik, 2025). Mivel a LLM-ek rendkívül nagy korpuszokon tanulnak (tipikusan internetes szövegek milliárdjain), óhatatlanul magukba szívják az emberi nyelvben meglévő történelmi és kulturális torzításokat (például nemi vagy faji sztereotípiákat). Caliskan és munkatársai (Caliskan, 2017) tanulmányukban kimutatták, hogy a szóbeágyazási modellek is *„emberi jellegű előítéleteket”* hordoznak: például a *“nő szóhoz közelebb asszociálnak bizonyos család vagy művészet tematikájú szavakat, míg a férfi szóhoz a „tudomány vagy matematika szavait”* (Guo, és mtsai.). Hasonlóképpen a nyelvi modellek hajlamosak lehetnek folytatni a szöveget olyan módon, ami megerősíti a társadalmi sztereotípiákat – ha például a

tanítóadatban gyakoribb egy bizonyos csoporttal kapcsolatos negatív kontextus, ezt a mintázatot a modell továbbviheti a válaszaiba. Az etikai kockázat nyilvánvaló: egy bias-t tartalmazó modell félrevezető vagy diszkriminatív outputot adhat, ami sértheti egyes felhasználói csoportok érdekeit, és *„alááshatja a rendszer pártatlanságába vetett bizalmat”* (Guo, és mtsai.). A szakirodalom folyamatosan keresi a megoldásokat e problémára: egyrészt adat-szintű beavatkozásokkal (kiegyensúlyozottabb, megtisztított tanító adatkészletek használatával), másrészt modell-szintű technikákkal (pl. utólagos finomhangolás explicit fairness célfüggvénnyel, utófeldolgozó szűrés a kimeneteken) igyekeznek mérsékelni a LLM-ek torzításait. Emellett a jogi-szabályozási oldal is megjelent: például az Európai Únió készülő „AI Act” (European Parliament, 2025) rendelete előírja a *„magas kockázatú AI rendszerek esetén a humán felügyeletet és a fairness biztosítását”* (Kuriakose, 2024). Mindez mutatja, hogy a bias kezelése nem csupán technikai, hanem társadalmi-kulturális feladat is.

2.5.2. Hallucináció

A **hallucináció** jelensége alatt azt értjük, amikor a nyelvi modell meggyőzően hangzó, de valótlan vagy alaptalan információt ad ki. Ji és munkatársai átfogó felmérésükben úgy definiálják a ténybeli hallucinációt, mint amikor "*the generated output contains factual inconsistencies or fabrication, which cannot be inferred from [...] the world knowledge or the provided source information*" (Ji Z. , és mtsai., [25.] Survey of Hallucination in Natural Language Generation, 2023), vagyis: „a generált kimenet ténybeli következetlenségeket vagy kitalációt tartalmaz, amely nem vezethető le [...] a világ ismeretéből vagy a biztosított forrásinformációból”. A LLM egyszerűen “kitalál” valamit – például egy nem létező tényt, hamis hivatkozást, valótlan eseményt – anélkül, hogy tudatában lenne tévedésének. Ez a viselkedés abból fakad, hogy a modellnek nincs valódi ismeretbázisa vagy valóság-ellenőrző mechanizmusa: a tanulás során nem épít ki tényadatbázist, csupán a szövegmintázatokat tanulja meg. Így amikor egy kérdésre a helyes választ nem „emlékszik” a tanultakból, akkor is megpróbál valami hihetőt generálni, hiszen erre van beprogramozva – a legvalószínűbb következő szavakat fogja leírni még akkor is, ha azok együtt valótlan állítást alkotnak. Az ilyen hallucináció lehet viszonylag ártalmatlan (pl. egy vicces anekdotát tulajdonít rossz történelmi személynek), de lehet nagyon is veszélyes: *„félrevezető orvosi tanács, nem létező jogi precedens citálása, üzleti adatok pontatlan közlése stb.”* is előfordult már a gyakorlatban (Nirdiamant, 2025).

A megbízhatóság különösen kritikus az olyan magas kockázatú területeken, mint a védelem-egészségügy, ahol a modern technológiák (pl. VR/AR, szimuláció) már a *„képzésben és a gyakorlatban is megjelennek”* (Fejes, Pitlik, Rikk, Szűcs, & Túri, 2024/1-2). Az esetek többségében a modell nem szándékosan „hazudik”, inkább egy túlságosan magabiztos autokomplettőrként viselkedik, aki nem tudja azt mondani, hogy “nem tudom”, hanem *„mindig válaszol valamit”* (Nirdiamant, 2025). Ez komoly UX-megbízhatósági kihívás: a felhasználó számára nehezen megkülönböztethető, mikor pontos a válasz és mikor koholt – hiszen a stílus mindkét esetben magabiztos.

2.5.3. A hallucináció és a bias enyhítése

A hallucináció és a bias jelensége egyaránt aláássa a LLM alapú rendszerekbe vetett bizalmat, ezért a szakirodalom több megoldást is javasol ezek enyhítésére. Az egyik út a modellek kimeneteinek utóellenőrzése: például egy második modell vagy egy speciális algoritmus ellenőrizheti a tényállításokat (pl. források keresésével), illetve kiszűrheti a nyilvánvaló ellentmondásokat a válaszban. Egy másik megközelítés a **kollektív válaszadás** – éppen az ensemble módszerek alkalmazása. Ha több modell válaszol ugyanarra a kérdésre, utána pedig összevetjük ezeket, akkor az inkonzisztens vagy kilógó elemek gyanúként azonosíthatók. A szakirodalomban ezt néha self-consistency néven említik: egy modell saját több mintáját vagy több modell változatot futtatva többszörös válaszokat gyűjtünk, majd a konszenzust keressük közöttük (pl. többségi szavazással vagy konszenzusos összevonással). *„Humán felügyelet bevonása”* is gyakran javasolt: fontos döntési helyzetekben (pl. orvosi diagnózis, pénzügyi döntés) a LLM-ek által generált tartalmat mindig át kell néznie egy emberi szakértőnek, mielőtt azt véglegesnek tekintenénk (Nirdiamant, 2025), (Schiller, 2024). Ez megfelel a human-in-the-loop paradigma követésének, ami számos mesterséges intelligencia alkalmazásnál bevett biztonsági háló.

Az AiFusion rendszer tervezése során ezen kihívások tudatosan kezelendők. A platform abból az alapötletből indul ki, hogy a *bíró modell* révén csökkenthető a hallucináció és bias hatása: ha több különböző LLM válaszát egy magasabb szintű modell értékeli, akkor esély van rá, hogy *„kiszűri az egymásnak ellentmondó vagy nyilvánvalóan hibás állításokat”*, mielőtt a választ a felhasználó megkapja (Kovács & Pitlik, 2025). Például, ha egy modell hallucinált egy téves “tényt”, de a többiek nem, a bíró ezt

detektálhatja és figyelmen kívül hagyhatja. Ugyanígy a bias mérséklésére is lehet esély: egy diverz modellt együttesen, ha egy adott torzítás csak az egyik modellnél erős, a bíró a többiek válaszai alapján korrigálhat. Természetesen mindez nem garancia a tökéletességre – a bíró modell is LLM, így maga is hibázhat vagy torz lehet –, de egy extra védelmi vonalat képez a rendszerben a megbízhatóság javítására. Emellett az AiFusion fejlesztése során is tervezem statikus szabályok és prompt-szintű korlátozások bevezetését (például a bíró modell utasításában kikötni, hogy csak akkor fogadjon el választ, ha az forrásokkal alátámasztható stb.). Az ilyen kombinált módszerek – technikai és emberi kontroll – együttes alkalmazása felel meg leginkább a jelenlegi legjobb gyakorlatnak a generatív modellek felelős használatában.

2.6. Az AiFusion architektúra a kollektív AI kontextusában

Az AiFusion platform architektúrája a fent tárgyalt elveket egy konkrét rendszerben valósítja meg. Felépítése a “fordított piramis” modellt követi: több párhuzamosan futó **slave** (szolga) LLM ad választ egy adott kérdésre, majd egy magasabb szintű **master** (bíró) modell értékeli és egyesíti ezen válaszokat. Praktikusan ez egy többlépcsős folyamatot jelent, ahol például az első szinten négy különböző nyelvi modell generál négy választ, a második szinten két (vagy akár több) modell összehasonlítja ezeket és kiválasztja a legjobb(ak)at, végül a harmadik szinten egy utolsó bíró modell dönt a fennmaradó válaszok között, vagy összefésüli őket egyetlen koherens válasszá. A folyamat biztosítja, hogy a végső output **több modell perspektíváját is figyelembe veszi**, kihasználva a kollektív intelligencia előnyeit a megbízhatóbb eredmény érdekében. Ahogy a kapcsolódó konferenciacikkben megfogalmaztuk, a cél egy olyan platform létrehozása, amely *"enables collaborative AI testing and the emergence of AI self-criticism"* (Kovács & Pitlik, 2025.) Ez a szervezeti felépítés szorosan kapcsolódik a *„hasonlóságelemzés”* (similarity analysis) és az *„automatizált intuíció-generálás”* (automated intuition-generation) alapelveihez, amelyek a KJE/SZIE kutatási körében már 2014-ben megjelentek (Pitlik L. , [1.] My-X Team, an innovative idea-breeding farm, 2014).

2.6.1. A lekérdezés menete, frontend

A rendszer adatútja ennek megfelelően alakul: a felhasználó egy kérdést intéz az AiFusionhoz, ami aztán a backend oldalon egyszerre továbbítja a kérést több kiválasztott LLM API-nak. Az egyes modellek megfogalmazzák a saját válaszukat, amelyeket ezután a *frontend* vagy a backend logika egyesít egy végső outputtá. Jelen implementációban – prototípus lévén – a bírói logika főként a frontenden került megvalósításra kísérleti jelleggel. A webes kliens program (JavaScript/React alapon) biztosít külön nézetet, például a "3 LLM + Judge" módot, ahol a felhasználó egyszerre három modellt futtathat párhuzamosan, majd a rendszer automatikusan meghív egy negyedik, bíró szerepű modellt, hogy feldolgozza a három választ. Ezt úgy valósítottam meg, hogy a frontenden a **ThreeLLMJudge.jsx** komponens egy miniatűr workflow-t implementál: a három *slave* modell API-hívását párhuzamosan indítja el, megvárja mindegyik eredményét, majd ezeket egy formázott prompt részeként beküldi a bíró modellnek értékelésre. A bíró modell promptja tartalmazza az eredeti felhasználói kérdést és a három kapott választ, kiegészítve előre definiált utasításokkal (pl. utasítás a bíró számára: döntse el, melyik válasz a legjobb, indokolással). Így a bíró LLM kvázi metamodellként működik: nem közvetlenül a felhasználói kérdésre válaszol, hanem a modell-válaszok alapján hoz ítéletet vagy összegzést.

2.6.2. Backend

A *backend* egy Flask alapú Python szerver, amely az **AIInterface** modulon keresztül valósítja meg a több modell együttes kezelését: paraméterei között szerepel, mely szolgáltatók (OpenAI, Anthropic, Google, DeepSeek) mely modelljeit hívja meg, mekkora max. token hosszúsággal, milyen kreativitással stb. E modul a hívások eredményeit összegyűjti és visszaadja a frontendnek. A frontenden aztán a ThreeLLM Judge komponens gondoskodik róla, hogy ha szükséges, a bíró modellt is bevonja. A jelenlegi prototípusban az ensemble logika tehát félig manuális (gombokkal aktiválható módok formájában), a jövőbeli fejlesztési tervek közt szerepel ennek automatizálása a backend oldalon egy általános orkeszráló modul formájában. Továbbá folyamatban van a bírói promptok finomhangolása és esetleg a bíró modell továbbképzése is, hogy minél megbízhatóbban ismerje fel a legjobb választ vagy a kombinálandó elemeket.

2.7. A tantárgyak és a dolgozat témájának összefüggései

A szakdolgozatban tárgyalt AiFusion platform tervezése és megvalósítása során nemcsak a közvetlen szakirodalmi háttérben tárgyalt technológiai alapelvek voltak relevánsak, hanem a képzés során elsajátított tágabb ismeretkörök is. Az alábbiakban ezeket a kapcsolatokat részletezem.

2.7.1. Matematikai alapok

A „Matematikai alapok” tantárgy ismeretanyaga több területen releváns. Az embeddingek (2.1) megértéséhez szükséges lineáris algebrai háttér mellett a tárgy további pontokon is kapcsolódik a dolgozat témájához:

- Az egyszerű valószínűségszámítás segít megérteni a modellek válaszainak minőségét, bizonytalanságát és a temperature paraméter működését (lásd: „2.3 Kreativitás vs. determinizmus: a temperature paraméter szerepe”).
- A modellek teljesítményének méréséhez statisztikát (pontosság, átlag, szórás) alkalmazunk („3.3. Kismintás tesztelés”).

2.7.2. Adatszerkezetek és algoritmusok

Az „Adatszerkezetek és algoritmusok” tantárgy keretében tanultak több ponton is kapcsolódnak a szakdolgozat témájához. Közvetlenül az embedding alapelvekhez (2.1) kötődnek a mátrixok és vektorok, amelyekkel a modellek a szemantikai kapcsolatokat reprezentálják. Emellett a tantárgy további elemei a dolgozat más részeihez kapcsolódnak:

- A kísérletek kiértékelése során kombinatorikai kérdések merültek fel a lehetséges triádok számának meghatározása során az „3.3.4. A kollaborációs partnerek és a JUDGE kiválasztása” fejezetben.
- A Pareto-front számítás algoritmikus alapjai relevánsak voltak az optimális modellkombinációk kiválasztásához „3.3.4.1. A Pareto-triádok kiválasztása”.

2.7.3. A jog szerepe a modern társadalmakban

Bár a dolgozat fókusza a rendszer technikai megvalósításán van, a platform tervezése és potenciális jövőbeli üzemeltetése kapcsán felmerülnek jogi és adatbiztonsági kérdések is. „A jog szerepe a modern társadalmakban" tantárgy keretében tanultak relevánsak lehetnek például a platform nevének és arculatának védjegyoltalma szempontjából, amelyre a „3.2.1. A platform elnevezése, logó, domain, védjegy" fejezet utal. A rendszer hozzáférési tokenekkel dolgozik (API-kulcsok, Bearer token), amelyek kezelése adatbiztonsági és jogi felelősség (hozzáférés-naplózás, jogosultságkezelés); ennek technikai implementációját és biztonsági vonatkozásait a „3.2.10. Biztonság és kulcskezelés" fejezet részletezi. Az adatvédelem kérdése a naplózás kapcsán is érintőlegesen megjelenik „3.2.12. Naplózás és adatvédelem".

2.7.4. Adatbázisok

A kísérleti fázisban Excel-t használtam „mini adatbázisként" (ennek gyakorlati alkalmazása látható a „3.2.7. Batch Runner modul" és az „3.3. Kismintás tesztelés" fejezetekben), de a rendszer természetéből adódóan a későbbi fázisokban relációs adatbázisra kell váltani (lásd „3.2.13.5. Adatbázis kapcsolat" fejezet). Az „Adatbázisok" tantárgy keretében tanult adatszervezési, normalizálási, lekérdezés-optimalizálási és jogosultságkezelési alapelvek adják meg az elméleti hátteret ahhoz, hogy a prototípus hogyan nőhet ki egy valódi, skálázható adatbázis-háttérré.

2.7.5. Az Elektronika fizikai alapjai és az Elektronikus áramkörök

Az „Elektronika fizikai alapjai" és az „Elektronikus áramkörök" tantárgyak ismeretanyaga szorosan összefüggnek egymással, így kapcsolatukat együtt tárgyalom.

Bár az AiFusion platform egy szoftveres rendszer, működése fizikai hardvereken (CPU (Central Processing Unit) /GPU (Graphics Processing Unit)-kon, memórián, hálózati eszközökön) alapul. A képzés során tanultak – mint az áramfelvétel, hőtermelés, hűtés, memória-sávszélesség, jeltovábbítás fizikai korlátai, valamint a stabil tápellátás, feszültségstabilizálás (VRM (Voltage Regulator Module)), szűrés és az EMI

(electromagnetic interference)-zaj minimalizálása – adják meg azt a kontextust, amely meghatározza a rendszer valós teljesítményének és skálázhatóságának felső határait. A megbízható áramköri működés közvetlenül befolyásolja a számítások sebességét és hibamentességét; a nem megfelelő működés például teljesítménycsökkenést (throttling) okozhat, ami torzíthatja az „3.3. Kismintás tesztelés” során mért futási időket és ronthatja a rendszer általános megbízhatóságát.

2.7.6. Felhasználói interfészek és vizualizáció

Az AiFusion platform fontos része a webes frontendből (lásd „3.2.6. Frontend architektúra”). Az itt alkalmazott megoldásokhoz – több modell válaszának érthető megjelenítése, az eredmények összehasonlítását segítő táblázatok és vizualizációk – az elméleti alapot a „Felhasználói interfészek és vizualizáció” tantárgy biztosította. A cél a könnyű használhatóság és az eredmények gyors áttekinthetősége volt, ami az „3.3.5. A kimeneti eredmények tárolása, kiértékelése” fejezetben bemutatott Excel táblákban is tükröződik.

2.7.7. Hálózatok és számítógép architektúrák

A „Hálózatok és számítógép-architektúrák” tantárgy ismeretanyaga szintén alapvető a rendszer megvalósításához. Az AiFusion platform több gépen, hálózaton keresztül hív külső AI-szolgáltatásokat (API-kat), így a sávszélesség és a hálózati késleltetés közvetlenül befolyásolja az API-hívások sebességét és a rendszer válaszidejét, ami a „3.3. Kismintás tesztelés” során mért időeredményekben is megmutatkozik. Emellett a rendszer futtatásához használt CPU architektúra és a memória-sávszélesség adják meg a helyi számítások (pl. a backend vagy a batch runner futtatása) teljesítménykorlátait.

2.7.8. Informatikai védelem és biztonság

Az „Informatikai védelem és biztonság” tantárgy alapelvei szintén megjelennek az AiFusion platform tervezésében és implementációjában. A rendszer hozzáférési tokeneket (Bearer token, API-kulcsok) használ az illetéktelen hozzáférés megakadályozására. Bár a prototípus jelenleg nem használja, a jövőbeli fejlesztések során elengedhetetlen a titkosított HTTPS (Hypertext Transfer Protocol Secure) kapcsolat

alkalmazása. A kérések és válaszok naplózása és auditálása, valamint az érzékeny adatok (pl. API kulcsok) maszkolása vagy biztonságos kezelése szintén kulcsfontosságú szempontok. Ezek technikai megvalósítására a „3.2.10. Biztonság és kulcskezelés” és a „3.2.12. Naplózás és adatvédelem” fejezetek térnek ki részletesebben.

2.7.9. Operációs rendszerek

Az „Operációs rendszerek” tantárgy adja meg azt az alapvető kontextust, amelyben az AiFusion platform fut. A rendszer komponensei (backend, frontend, batch runner) nem natívan futnak a hardveren, hanem operációs rendszereken. A fejlesztés során konkrétan Ubuntu Server és macOS operációs rendszereket használtam, ahogy azt a „3.2.2. Fejlesztői környezet” fejezet részletezi. Az Operációs Rendszer (Operating System – a továbbiakban: OS) által biztosított szolgáltatások (folyamatkezelés, memóriakezelés, hálózatkezelés) alapvetőek a „3.2. Rendszerfelépítés és implementáció” során leírt szoftverkomponensek működéséhez.

2.7.10. Programozás

A „Programozás" tantárgy alapvető ismereteket tárgyal a projekt konkrét megvalósításához: a Pythonos Flask backend („3.2.5. Backend architektúra”), a React frontend („3.2.6. Frontend architektúra”) és a batch-runner szkript („3.2.7. Batch Runner modul”) elkészítéséhez. A projekt jól mutatja az API-hívások („3.2.5.2. API végpontok és hitelesítés”, „3.2.6.1. Kommunikáció a backenddel”), a hibakezelés, a naplózás („3.2.12. Naplózás és adatvédelem”) és az automatizált futtatás („3.2.7. Batch Runner modul...”) gyakorlati megvalósítását.

2.7.11. Programozási alapelvek és módszertanok

A „Programozási alapelvek és módszertanok" segítettek a rendszer átlátható és bővíthető kialakításában. Ennek elemei a moduláris felépítés, a rétegezés és a separation of concerns elve („3.2.3. Architektúra áttekintése...”). Következetes nevezéktant alkalmaztam („3.2.8. Nevezéktani konvenciók”), és figyeltem a naplózásra („3.2.12. Naplózás és adatvédelem”) és hibakezelésre. A verziókezelést Git rendszerrel végeztem.

Az automatizált futtatást és mérést a batch-runner („3.2.7. Batch Runner modul...”) biztosítja, a skálázhatóságot és tesztelhetőséget pedig a tiszta interfészek és adapterek („3.2.5.1. Modulstruktúra és adapterek") segítik

2.7.12. Rendszermodellezés

A „Rendszermodellezés" eszköztára segített a rendszer működésének vizualizálásában és megértésében. A rendszer működése adat- és vezérlésáramlási diagramokkal ábrázolható (4. ábra), hasznos lehet a szekvenciadiagram a kérés–válasz lépésekhez („3.2.6.1. Kommunikáció a backenddel”) és az állapotgép a futások státuszaira. A komponensdiagram (adapterek, API-interfész, batch-runner) (4. ábra) segít a felelősségek tiszta szétválasztásában és a bővíthetőség megtervezésében.

2.7.13. Rendszertervezés

A „Rendszertervezés" elvei alapján a megoldás rétegzett felépítésű (frontend, backend, adapterek, judge-logika), egyértelmű felelősségi körökkel („3.2.3. Architektúra áttekintése...”). A követelményekből („3.1. Követelmények és Use-case”) indultam, majd ezekhez terveztem az interfészeket („3.2.5.1. Modulstruktúra és adapterek”), az adatáramlást („3.2.4. Adatáramlás és komponensek együttműködése”) és a hibakezelést. A skálázhatóságot bővíthető modulokkal és a jövőben microservice-irányú szétbontással biztosíthatjuk („3.2.13.3. Infrastruktúra és skálázhatóság”).

2.7.14. Szoftverarchitektúrák

A „Szoftverarchitektúrák" tantárgy ismeretei tükröződnek a rendszer rétegzett felépítésében (frontend–backend–adapterek) („3.2.3. Architektúra áttekintése”). A külső LLM-szolgáltatókhoz az adapter minta illeszkedik („3.2.5.1. Modulstruktúra és adapterek”), a bírói lépés pedig orchestrator szerepet tölt be (jelenleg a frontenden, „3.2.6.2. Párhuzamos lekérdezések...”). A skálázásnál a microservice irány is felmerül („3.2.13.3. Infrastruktúra és skálázhatóság”). A megbízhatóságot erősítő elemek (pl. observability) a jövőbeli fejlesztések részei lehetnek.

2.7.15. Szoftvertesztelés

A „Szoftvertesztelés" fontosságát mutatja a batch runner („3.2.7. Batch Runner modul...”) alkalmazása a tömeges vizsgálatokhoz („3.3. Kismintás tesztelés”). Ez lehetővé teszi a pontosság, idő és költség mérését, valamint a regressziós tesztek futtatását új verziók bevezetésekor. A működés alapvető ellenőrzése (endpoint hívások, hibakezelés, jogosultság) manuális és implicit tesztekkel történt a fejlesztés során.

2.7.16. Szoftverüzemeltetés

Bár a projekt jelenleg prototípus fázisban van, a „Szoftverüzemeltetés" szempontjai már most relevánsak a jövőre nézve. A stabil működéshez és folyamatos felügyelethez monitoring (idő, hiba, költség), logolás („3.2.12. Naplózás és adatvédelem”) és riasztás szükséges. Fontos a rendszeres frissítés, a biztonságos kulcskezelés („3.2.10. Biztonság és kulcskezelés”), a skálázás és az automatizált deploy (Folyamatos integráció / Folyamatos szállítás (Continuous Integration / Continuous Deployment – a továbbiakban: CI/CD)), amelyek a „3.2.13. Jövőbeli fejlesztési irányok” között is szerepelnek.

2.7.17. Komplex társadalomtudományi ismeretek

A rendszer többcélú optimalizációt kezel (pontosság–költség–idő), amelyet Pareto-szemlélettel és triád-alapú aggregálással tesz mérhetővé; a triádok sikerrátáját a „legalább egy helyes” (parallel_or) szabály alapján számítjuk, a bírói (JUDGE) modul pedig az ellentmondó kimenetek feloldására szolgál. Ez a megközelítés lehetővé teszi, hogy a technikai eredményeket társadalomtudományi szempontból is értelmezzük: a kompromisszumok (trade-offok) számszerűsíthetők, a döntési elvek átláthatóvá válnak, és reprodukálható módon összevethetők.

Szubjektív meglátásom szerint a társadalomban - a jelenleg is tapasztalható kezdeti lelkesedés mellett - kialakulóban van egyfajta bizalmatlanság és bizonytalanság a nagy nyelvi modellekkel (köznyelvben: „MI”) szemben. Mivel a LLM-rendszerek átláthatósága és megértése még a szakértők számára is kihívás, a hozzáértő társadalmi réteg felelőssége, hogy közérthetően tájékoztassa a többségi társadalmat a működésről és a megbízhatóságról. Ezt a közérthető tájékoztatást és a megértést kifejezetten segíthetik az olyan rendszerek által generált összehasonlítások és statisztikák, mint az AiFusion: a

platform standardizált protokollok mentén képes több LLM összehasonlító értékelésére, rangsorolására és a torzítások/hibák feltárására. Rendkívül fontosnak tartom a LLM-ek vizsgálatát megbízhatóság és előítéletesség szempontjából; a több-modellű, összehasonlító elemzések mintázatokat és statisztikákat szolgáltatnak ehhez, így adat-alapú beszélgetést tesznek lehetővé a társadalmi kockázatokról és előnyökről.

A fenti szempontok a rendszer társadalmi beágyazottságát és költség–haszon értelmezhetőségét erősítik: az átlátható mérőszámok és a reprodukálható módszertan elszámoltathatóbbá teszik a döntéseket, és alapot adnak a felelős, tájékozott közbeszédhez a generatív MI (Mesterséges Intelligencia) alkalmazásáról.

2.7.18. Európai civilizáció és identitás

A rendszertervezésben következetesen érvényesülnek az európai alapelvek: adatvédelem, átláthatóság, felelősségre vonhatóság. Az AiFusion ennek megfelelően reprodukálható futtatásokat, visszakövethető döntési lépéseket (napló, JSON/Excel kimenetek) és ellenőrizhető bírói értékelést biztosít. A többnézőpontos mérlegelés és a konszenzusra törekvés intézményes európai döntéshozatali mintázatokkal rokon (testületi mérlegelés, vitakultúra, indokolt állásfoglalás).

Közvetlen szinergia mutatkozik a 4.5. fejezetben tárgyalt jogi rendszerek LLM-alapú tesztelésével: a bírói modul és a triád-logika panel-szerű döntéstámogatást tesz lehetővé (különvélemények kezelése, többségi/döntési szabályok, indoklás), ami a jogi döntések MI-támogatásában különösen fontos. A platform így nemcsak technikailag, hanem értékalapon is illeszkedik az európai elvárásokhoz: az automatizált értékelés átlátható, auditálható és elszámoltatható keretben történik.

2.7.19. Emberi viselkedés és kommunikáció

Az AiFusion többmodellű működése (triád + bíró/JUDGE) egy többszereplős kommunikációs helyzetet modellez: különböző „beszélők” (LLM-ek) egymástól független válaszai kerülnek összevetésre, majd egy bírói modul visszacsatolással és konszenzuskereséssel egységes, indokolható kimenetet állít elő. Ez a folyamat jól rímel az emberi kommunikáció kulcselemeire: a félreértések kezelése, a torzítások (bias) csökkentése, a bizalomépítés és a közös jelentésalkotás.

Szubjektív megítélésem szerint a társadalmi bizalmatlanság csökkentéséhez nem elég az egyes LLM-ek teljesítményét állítani; szükség van közérthető, összehasonlítható eredményekre és magyarázható döntési folyamatokra. A platform naplózása és a bíró által készített, lépésenként rekonstruálható összegzés ezt a kommunikációs átláthatóságot szolgálja - emberi helyzetekhez hasonlóan, ahol a vitában végül egy moderált, indokolt álláspont születik.

2.7.20. Vállalati gazdaságtan

Az „Vállalati gazdaságtan" tantárgy keretében tanultak adnak alapot ahhoz, hogy az AiFusion rendszert ne csak technikai, hanem üzleti szempontból is értékeljük. A rendszer építése során felmerült annak potenciális piaci hasznosíthatósága, ami a brandépítés („3.2.1. A platform elnevezése...") és egy lehetséges üzleti modell kialakításának fontosságát veti fel a jövőre nézve („3.2.13.4. Jogi háttér szerkezete").

2.7.21. Vezetési és vállalkozási ismeretek

A „Vezetési és vállalkozási ismeretek" tantárgy a projekt fejlesztési megközelítéséhez kapcsolódik. A rendszer fejlesztése az Minimálisan életképes termék (Minimum Viable Product – a továbbiakban: MVP) elvét követte (kezdetben Excel alapú kiértékelés, majd prototípus fejlesztése), amelyet iteratív módon, a mért eredmények alapján lehet tovább skálázni („3.2.13. Jövőbeli fejlesztési irányok"). A projekt megvalósítása során fontos szempont volt a prioritáskezelés, a kockázatkezelés (költség, idő, minőség) és az eredmények érthető kommunikációja, például egyszerű üzleti mutatók (költség/helyes válasz) segítségével, amelyeket az „3.3. Kismintás tesztelés" és a „4. Vita" fejezetek érintenek.

2.7.22. Innovatív információs és kommunikációs technológiák az IT-biztonság kapcsán

Ez a tantárgy rávilágít azokra a modern eszközökre, amelyeket a rendszer biztonságának növelésére lehetne használni a jövőben. Ilyen a titkosított API-kapcsolat (HTTPS), a biztonságos secret-kezelés, a naplózás és riasztás („3.2.12. Naplózás és adatvédelem"), valamint olyan újabb technikák, mint a rate-limit, az anomaly detection

(szokatlan API-hívások kiszűrése) vagy a kulcsrotáció („3.2.11. A jelenlegi megoldás biztonsági korlátai”). A cél a szolgáltatás biztonságának fenntartása a rendszer fejlődése és skálázódása során.

2.7.23. IT-biztonsági fejlesztések minőség- és projektmenedzsmentje

A projekt fejlesztése során - bár nem formális keretek között - törekedtem a biztonságos Szoftverfejlesztési életciklus (Software Development Lifecycle – a továbbiakban: SDLC) alapelveinek követésére: a követelmények meghatározásától („3.1. Követelmények és Use-case”) a tervezésen („3.2. Rendszerfelépítés és implementáció”) és fejlesztésen át a tesztelésig („3.3. Kismintás tesztelés”). A minőség biztosítását a kód áttekinthetősége („3.2.8. Nevezéktani konvenciók”), a tesztelési logika („3.2.7. Batch Runner modul...”) és az alapvető biztonsági megfontolások („3.2.10. Biztonság...”) jelentették. A projektet a prioritások és a kockázatok (pl. költségkeret túllépése, API hibák) figyelembevétele vezérelte.

2.7.24. Mesterséges intelligenciák az IT-biztonság területén

Bár az AiFusion jelenleg nem alkalmaz MI-t önmaga biztonságának növelésére, ez a tantárgy rámutat a jövőbeli lehetőségekre. Gépi tanulással lehetne anomáliákat észlelni a rendszer használatában (pl. szokatlan API-hívások, költség-tüskék a naplók alapján – „3.2.12. Naplózás és adatvédelem”). Érdekes lehetőség, hogy az AiFusion kollaboratív modellje akár a biztonsági események verifikálására is használható lenne: ha több (akár biztonsági fókuszú) modell elemez egy logot vagy hálózati forgalmat, az eredmények összevetése csökkenthetné a téves riasztások (false positive) számát.

2.7.25. Tudásmenedzsment az IT-biztonság területén

Ez a tantárgy arra ösztönöz, hogy a rendszer működése során gyűjtött adatokat (logok, futási eredmények – „3.2.12. Naplózás és adatvédelem.”, „3.3.5. A kimeneti eredmények tárolás, kiértékelése”) ne csak tároljuk, hanem tudásbázisként használjuk. A tapasztalatokból (pl. melyik modell hajlamos hibázni, milyen promptok működnek jól/rosszul) leszűrt szabályok és best practice-ek visszacsatolhatók a rendszerbe (pl. a

promptok finomításával, a bíró modell utasításainak pontosításával – „3.2.13.1. Működési logikák kibővítése”), így a rendszer idővel hatékonyabbá és megbízhatóbbá válhat.

2.8. A szakirodalmi háttér összegzése

Összességében az AiFusion architektúra a 2.1–2.7 alfejezetekben ismertetett elméleti fogalmak integrált alkalmazása: használja a LLM-ek belső embedding-alapú tudását, a tokenizációs és generálási mechanizmusokra épít, szabályozható kreativitási szintet enged (pl. a párhuzamos modellek eltérő temperature értékei révén), és az ensemble megközelítést egy bíró modell formájában valósítja meg. Célja, hogy a *kollektív AI-válaszadás* révén mérsékelje az egyes modellek korlátait (hallucináció, bias), és **jobb minőségű, megbízhatóbb választ** adjon a felhasználóknak, mintha bármely egy modellre hagyatkozna.

3. Saját fejlesztés

Ebben a fejezetben az AiFusion platform saját fejlesztésű megoldását mutatom be, kiindulva a rendszerrel szemben támasztott követelményekből és egy tipikus felhasználási forgatókönyvből. Ezek adják a későbbi architekturális és implementációs döntések értelmezési keretét.

3.1. Követelmények és Use-case

Ebben a fejezetben bemutatom az AiFusion platform legfontosabb követelményeit, valamint ismertetek egy tipikus use-case forgatókönyvet, amely szemlélteti a rendszer működését a gyakorlatban. A követelmények között szerepelnek a főbb funkcionális elvárások – így több nagy nyelvi modell egyidejű használata és integrációja –, továbbá biztonsági és felhasználói interfészre vonatkozó igények, valamint a rendszer egyedi, kutatást támogató funkciói. Ezt követően egy jellemző felhasználási eset leírása következik, amely lépésről lépésre bemutatja, hogyan zajlik egy kérdés megválaszolása az AiFusion segítségével.

3.1.1. Rendszerkövetelmények

A rendszerkövetelmények részletes bemutatása során először azokra a funkcionális elvárásokra térek ki, amelyek a több nagy nyelvi modell egyidejű bevonására és a kollektív válaszadás megvalósítására vonatkoznak.

3.1.1.1. Több modell egyidejű bevonása

Alapvető funkcionális követelmény, hogy a rendszer egy felhasználói kérdés megválaszolásához egyszerre több különböző nagy nyelvi modellt (LLM) tudjon igénybe venni, majd az eredményeket egyesítse. Ennek érdekében szükséges egy bíró komponens alkalmazása, amely értékeli a párhuzamosan futó modellek válaszait, és kiválasztja vagy összevonja a legjobb elemeket egy végső válaszba. Ez a kollektív AI-válaszadás koncepciója, amelytől jobb minőségű és megbízhatóbb eredményeket remélek. (A megoldás lényege tehát egy master–slave modellstruktúra alkalmazása: több modell generál válaszokat, egy magasabb szintű “bíró” modell pedig eldönti, melyik legyen a végső output.)

3.1.1.2. Integráció különböző AI szolgáltatókkal

A backendnek támogatnia kell több külső AI szolgáltató integrációját is, egységes interfészen keresztül. Jelenlegi prototípusban elvárás volt, hogy az **OpenAI ChatGPT**, az **Anthropic Claude**, a **DeepSeek** és a **Google Gemini** API egyaránt elérhető legyen a rendszeren keresztül. A különböző szolgáltatók illesztését moduláris módon, egységes felületen keresztül kell megoldani, hogy a kliensoldali használat a modelltől független maradjon. Ez biztosítja a rendszer bővíthetőségét is – új modellek viszonylag kis ráfordítással hozzáadhatók a jövőben, amennyiben az egységes interfészen keresztül követik a meglévő adapterek mintáját.

Biztonság és jogosultságkezelés

Követelmény egy egyszerű, de hatékony token alapú autentikációs mechanizmus bevezetése a rendszer illetéktelen használatának megakadályozására. Ennek biztosítania kell, hogy kizárólag a jogosult (a megfelelő tokennel rendelkező) felhasználók férhessenek hozzá az API végpontokhoz. A megvalósítás technikai részleteit, beleértve a token generálását és ellenőrzését, a „3.2.10. Biztonság és kulcskezelés” fejezet tárgyalja.

3.1.1.3. Erőforrás- és költségfelügyelet

Mivel a külső AI szolgáltatások használata költséges (általában token-alapú díjszabással), a rendszernek figyelemmel kell kísérnie a lekérdezések erőforrásigényét és becsült költségét. Követelmény, hogy minden egyes API válasz esetén rögzítésre kerüljön a felhasznált tokenek száma (beviteli, kimeneti és összesen), így elemezhető a használat költsége és hatékonysága. Célszerű továbbá lehetőséget biztosítani bizonyos költségkeret vagy kvóta beállítására, és figyelmeztetést adni, ha egy felhasználó adott időszakban túllépi a meghatározott keretet. Ezzel garantálható, hogy a rendszer használata anyagi szempontból kontrollálható maradjon, ami különösen fontos lehet, ha a szolgáltatást hosszabb távon, online formában üzemeltetik. (Megjegyzendő, hogy a jelenlegi implementációban nincs még automatikus korlát vagy leállító mechanizmus beépítve – a token felhasználási adatok gyűjtése azonban az első lépés egy későbbi monitorozó modulhoz.)

3.1.1.4. Felhasználói felület és UX követelmények

A kliensoldali alkalmazásnak áttekinthető, könnyen használható felületet kell nyújtania. A felhasználó számára lehetővé kell tenni a lekérdezés paramétereinek testreszabását, többek között: a használandó AI modellek kiválasztását (pl. legördülő listából ChatGPT, Claude, stb.), a generált válasz hosszát és stílusát befolyásoló paraméterek beállítását (mint a maximális token-szám, kreativitás/temperature érték), illetve opcionális módosítók megadását a kéréshez (pl. "Answer shortly" a rövidebb válaszért). Fontos továbbá, hogy a párhuzamosan futó modellek válaszai egymás mellé állíthatóan összehasonlíthatók legyenek a felületen – az alkalmazás jelenlegi változata lehetővé teszi, hogy a felhasználó egy nézetben lássa az összes kapott választ, megkönnyítve ezzel az eredmények összevetését. A kezelőfelületnek reszponzívnak kell lennie (különböző eszközökön, pl. laptopon és mobileszközön egyaránt megfelelően működjön), és a visszajelzések alapján érdemes vizuális segítséget nyújtani az eltérések kiemelésére a modellek válaszai között (pl. kiemelni a különbségeket színnel vagy formázással). Ezek a felhasználói élmény (User Experience – a továbbiakban: UX) - szempontok növelik a felhasználói élményt és a rendszer átláthatóságát, és már a tervezés során figyelembe lettek véve (bár bizonyos elemek – pl. eltérések kiemelése – még további fejlesztést igényelnek a prototípusban).

3.1.1.5. Kutatástámogató funkció, a batch feldolgozás

Alapvető követelmény volt, hogy a rendszer támogassa a tömeges lekérdezések automatizált futtatását is, kísérleti vagy elemzési célokból. Ez lehetővé teszi nagyszámú kérdés szisztematikus végrehajtását különböző konfigurációk mellett, ami elengedhetetlen a modellek teljesítményének tudományos igényű kiértékeléséhez. A funkció részletes megvalósítását a „3.2.7. Batch Runner modul" fejezet ismerteti.

3.1.2. Use-case, vagyis Felhasználási eset

Use-case forgatókönyv – Kérdés megválaszolása több modellel: Egy tipikus használati esetben a felhasználó először bejelentkezik a webalkalmazásba (a demó rendszerben fix teszt felhasználónév/jelszó párossal). Sikeres autentikáció után a felhasználó a lekérdezési felületen beír egy kérdést a rendszerbe. Itt lehetősége van kiválasztani, hogy **mely AI modelleket** vegye igénybe a válaszadáshoz – például

kiválaszthatja, hogy a kérdést egyszerre küldje el **ChatGPT-nek, Claude-nak és DeepSeek-nek** – továbbá beállíthatja a választ befolyásoló paramétereket, mint a maximálisan felhasználható tokenek száma, a válasz kreativitása (temperature érték), illetve egyéb opciók (pl. "Answer shortly" a rövid válaszért, "Answer randomly" a véletlenszerűbb stílusért). Miután a felhasználó konfigurálta a kérést, a „Küldés" gombra kattintva elindítja a folyamatot.

A kérdés elküldése után a háttérrendszer először autentikációt végez a megadott API token alapján. A lekérdezés csak akkor folytatódik, ha a felhasználó által megadott Bearer token érvényes, ellenkező esetben a backend hibaüzenetet küld és a folyamat megszakad. (A prototípusban a felhasználó a bejelentkezés után egy külön mezőben adja meg a szerver konzolján előzőleg generált token értékét. Ha ez hiányzik vagy hibás, a szerver 401-es hibával azonnal visszautasítja a kérést – biztosítva, hogy csak jogosult felhasználó indíthasson lekérdezést a továbbiakban. Amennyiben a token rendben van, a **backend párhuzamos API-hívásokat** indít a kiválasztott modellek felé. Technikailag a rendszer egyszerre küldi ki ugyanazt a kérdést az összes megjelölt LLM-nek, kihasználva a párhuzamosítás adta sebességelőnyt. Ennek eredményeként mindegyik modell megkezdi a válasz generálását a saját környezetében.

Miután a külön futó AI modellek egyedi válaszai beérkeznek a szerverhez, az AiFusion rendszer egy külön bíró modellhez fordul. A bíró modell bemenetéül szolgál az **eredeti kérdés**, valamint **az összes begyűjtött válasz**, méghozzá egy speciális prompt formájában összeállítva. Ez a prompt tartalmazza a kérdést és például felsorolásszerűen a modellválaszokat, és utasítást ad a bíró szerepű AI-nak, hogy döntsön vagy készítsen összefoglaló választ. A bíró AI a kapott információk alapján kiértékeli a válaszokat: kiválasztja közülük a **legjobbnak ítélt választ**, vagy pedig (a koncepció szerint) több válaszból **konszolidáltan összeállít egy újat**. A jelenlegi implementáció prototípusában a bíró logika egyszerűen kiválasztja a számára legoptimálisabb választ és azt adja vissza, de elviekben megvan a lehetőség arra is, hogy a bíró összefésülje a több modelltől érkező információkat egy kombinált válaszba. Miután a bíró modell meghozta a döntést és előállt a végső válasszal, ez a konszolidált eredmény visszajut a frontendhez.

Az alkalmazás kliensfelületén ezután **megjelenik a végső válasz**, amelyet a bíró modell ítélete alapján választott ki a rendszer. A felület a use-case forgatókönyv szerint lehetőséget ad arra is, hogy a felhasználó megtekintse az egyes modellek válaszait külön

(például egy másik nézetben egymás alatt felsorolva), így átláthatóvá válik, melyik modell milyen választ adott, és közülük melyiket (vagy mely részeket) emelte ki a bíró komponens. (A jelenlegi prototípusban az összes válasz megtekintése egy közös nézetben valósul meg, és az eltérések vizuális kiemelése még fejlesztés alatt áll, de a tervezett funkció szerint a lényeges különbségek később jól láthatóan jelölhetők lesznek.) A felhasználó a választ áttekintve dönthet úgy, hogy módosít bizonyos beállításokon – például növeli a kreativitás paraméter értékét, vagy kicseréli az egyik használt modellt egy másikra – és újra futtatja a lekérdezést a jobb eredmény reményében. Ily módon a rendszer interaktív, iteratív folyamatot biztosít: a felhasználó kísérletezhet a különböző modellekkel és beállításokkal, ami hozzájárul a jobb végeredmény eléréséhez és egyben növeli a felhasználói élményt.

3.2. Rendszerfelépítés és implementáció

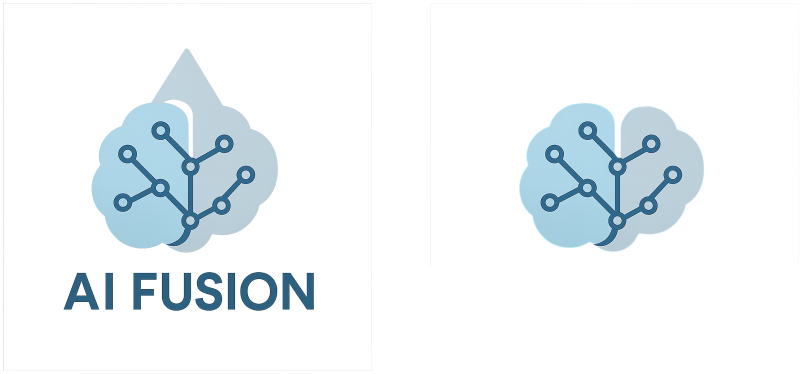
Ebben a fejezetben bemutatom az AiFusion platform architektúráját és megvalósítását. Ismertetem a master–slave–judge modellű felépítést, a komponensek (backend és frontend) együttműködését és az adatok áramlását a rendszeren belül. Részletesen kitérek a backend és frontend implementáció főbb elemeire – beleértve az osztályokat, modulokat és azok szerepét –, valamint bemutatom a batch-runner nevű modul működését, amely a rendszer tesztelését és a válaszok kiértékelését segíti. Végül összefoglalom a projekt során alkalmazott nevezéktani konvenciókat a kód olvashatóságának és karbantarthatóságának érdekében.

3.2.1. A platform elnevezése, logó, domain, védjegy

Minden vízióm kezdeti szakaszában fontos szempont a név kitalálása és a logó megtervezése. A név-logó páros egyfajta konténerként szolgál gondolataimnak, segítve azok rendszerezését.

A platform nevét rövid, pár napig tartó gondolkodást követően „megtaláltam”. Fontosnak tartom kiemelni, hogy a névválasztásnál soha nem használok gépi segítséget, így jelen esetben sem használtam. A névnek tisztán gondolati alapon kell „keletkeznie”, mert tapasztalatom szerint ilyenkor tudok vele a legnagyobb mértékben azonosulni.

A logó megtervezéséhez már az OpenAI Dall-E rendszerét használtam. Az eredetileg létrehozott logót Photoshop segítségével módosítottam (vö. 1. ábra).



1. ábra: Az AiFusion platform eredeti generált logója (b) és a módosított, végső variáns (j)
Forrás: chatgpt.com / saját szerkesztés

A platform nevének megfelelő „aifusion.hu” domain megvásárlásra került, jelenleg nincs tartalom a domain mögött.

A logót érdemes ellátni védjegyoltalommal. Részletekért lásd: „1.5.4 Költség- és teljesítményelemzés”.

3.2.2. Fejlesztői környezet

Ebben az alfejezetben röviden bemutatom a fejlesztés során használt eszközöket.

3.2.2.1. Ubuntu Server

A rendszer fejlesztését egy külön az AiFusion-nek fenntartott Lenovo X280 laptopon, Windows 11 alatt futó VirtualBoxban futtatott VM-ben, Ubuntu Server op. rendszer alatt kezdtem el. A Python kódokat először egyszerű szövegszerkesztőben írtam. A fejlesztés ebben a környezetben megterhelő és improduktív volt. Sokszor egyhuzamban csak 15-30 percet tudtam foglalkozni a fejlesztéssel és a rendszerindítások és leállítások minden egyes alkalommal szükséges elvégzése súlyos perceket vett el a szűkös időkeretemből. Ezen felül a VM futtatása miatt a laptop akkumulátoros üzemideje is gyenge volt (maximum 2-3 óra).

3.2.2.2. macOS

A fejlesztés durván 1/3-ánál vásároltam külön a projekt fejlesztésének céljára egy MacBook Air M1 laptopot, amelyen már lényeges komfortosabb környezetben tudtam a munkámat elvégezni. A Python kódokat az Ubuntu Serverrel azonos módon, terminálban tudtam futtatni, a kódolást kényelmesen, Visual Studio Code-ban folytattam. A hatékonyságom jelentősen nőtt, ugyanis a képernyő le- és felhajtásával kiváltható lett a teljes rendszerindítási és leállítási protokoll, a munkámhoz azonnal vissza tudtam térni és azt azonnal abba tudtam hagyni, így akár a pár perces holtidőimben is vissza tudtam térni a fejlesztéshez. Az MacBook akkumulátoros üzemideje a fejlesztési terhelés mellett 10+ óra.

3.2.3. Architektúra áttekintése, a Master–Slave–Judge modell

Az AiFusion platform jelenlegi architektúrája a 2.6 fejezetben részletezett master–slave–judge elvet követi, amely egyfajta „fordított piramis” struktúrát eredményez: az

alsó szinten több különálló modell dolgozik párhuzamosan, míg a csúcson egyetlen modell hoz döntést a legjobb válasz kiválasztásáról. Ennek köszönhetően a végső output több modell perspektíváját is figyelembe veszi, növelhetve a válasz megbízhatóságát és minőségét. (Például egy lehetséges forgatókönyvben első körben több különböző LLM generál választ, majd egy bíró LLM ezek alapján meghozza az ítéletet.) A bíró modell tulajdonképpen egy meta-LLM, amely a többi modell válaszát összesítve hoz döntést – hasonlóan ahhoz, mintha egy ember több szakértő véleményét meghallgatva választaná ki a legjobb megoldást. Ez a többlépcsős döntéshozó architektúra tudományos szempontból is érdekes, mert vizsgálható, hogy a bíró modell mennyire tudja kiszűrni az esetleges hibákat vagy ellentmondásokat a válaszok között, javítva a végeredmény minőségét.

3.2.4. Adatáramlás és komponensek együttműködése

A felhasználó kérdése a webes frontend felületen vagy a kötegelt feladatok futtatására létrehozott Batch Runner alkalmazáson keresztül jut el a backend szerverhez, amely a kérést párhuzamosan továbbítja az összes kiválasztott AI modell API-jához (külső szolgáltatókhoz). A különböző modellektől kapott eredményekből végül a rendszer egy egyesített választ állít elő egy bírói logika segítségével. A jelenlegi implementációban ez az összevonás a frontenden történik: a kliensoldali alkalmazás megvárja, amíg az első körben beérkeznek a párhuzamos modellválaszok, majd ezeket egy új promptba foglalva küldi el egy kiválasztott bíró modellnek értékelésre. Így a backend szempontjából minden részfeladat külön API-hívásként valósul meg, de a felhasználó egy összefüggő folyamatként érzékeli, hogy a kérdésére több modell együtt ad választ, egy utólagos bírói döntéssel kiegészítve. Az architektúra három fő komponense tehát a backend (szerveroldal), a webes frontend (kliensoldal) és a kötegelt feladatok automatizált futtatására szakosodott Batch Runner, melyek szorosan együttműködve valósítják meg a fent leírt folyamatot.

Az architektúrát egy áttekintő diagram is szemlélteti (vö. 5. Ábra), amelyen látható a frontend és a backend kapcsolata, a backend komponensei (Flask alapú API-szerver, AIInterface osztály, adapter modulok), valamint a külső AI szolgáltatások. Az ábrán nyilak jelzik a kérés–válasz folyamat irányát a rendszerben (frontend → backend → külső API → backend → frontend). Ezzel a felépítéssel a rendszer biztosítja, hogy a felhasználói

kérdések megválaszolása több modell együttes eredményére épüljön, növelve a válaszok hitelességét, miközben a folyamat a felhasználó számára egységes élményként jelenik meg.

3.2.5. Backend architektúra

A backend egy Python nyelven írt, Flask (egy Python alapú web keretrendszer) keretrendszerben futtatott program. Ez a szerveroldali komponens fogadja a kliens (frontend) HTTP (Hypertext Transfer Protocol) kéréseit, hitelesítés után továbbítja azokat a megfelelő AI szolgáltatónak, majd az eredményt visszaadja a kliensnek válaszként. A jelenlegi implementáció több különböző AI szolgáltató integrációját támogatja – beépítésre került az OpenAI ChatGPT, az Anthropic Claude, a DeepSeek és a Google Gemini API is –, és ezeket egységes módon, egy köztes interfészen keresztül kezeli. A kliens számára így az API használata egységes marad a választott modelltől függetlenül.

3.2.5.1. Modulstruktúra és adapterek

A backend kód felépítése moduláris. A fő futtatható modul az aifusion_backend.py (vö. 2. ábra), amely elindítja a Flask szervert (fejlesztés közben jellemzően a 127.0.0.1:5005 címen fut, hogy ne ütközzön más webszolgáltatással). Az aifusion_backend.py hívási paraméterei az 1. táblázatban, a kötelező HTTP-fejlécmezők a 2. táblázatban találhatók.

Paraméter neve	Típus	Alapértelmezett érték	Leírás
message	str	""	A felhasználó promptja, vagyis a kérdés / utasítás, amelyet a modellnek küldünk.
ai_model	str	"ChatGPT"	A választott szolgáltató neve: ChatGPT, Claude, DeepSeek, Gemini.
model	str	"gpt-4o"	A konkrét modell neve az adott szolgáltatón belül (pl. gpt-4o-mini, claude-3-sonnet).
max_tokens	int	1024	A válaszban engedélyezett maximális tokenmennyiség.
temperature	float	0.7	A kreativitási szint, amely befolyásolja a válasz változatosságát.

1. táblázat: Az aifusion_backend.py bemeneti paraméterei
Forrás: saját táblázat

Fejléc	Példa érték	Leírás
Authorization	Bearer 79c3914f09c83cccc5b89dffec	A biztonsági token, amely igazolja, hogy a kliens jogosult a kérés küldésére.
Content-Type	application/json	A kérés formátuma (mindig JSON).

2. táblázat: Az aifusion_backend.py kötelező HTTP-fejlécmezői
Forrás: saját táblázat

```
(.venv) kovacsbalint@MacBookAir aifusion-backend % python aifusion_backend.py
2025-10-20 22:49:16 INFO chatgpt_module [rid=- path=-] ChatGPT client initialized
2025-10-20 22:49:16 INFO claude_module [rid=- path=-] Claude client initialized
2025-10-20 22:49:16 INFO deepseek_module [rid=- path=-] DeepSeek client initialized
2025-10-20 22:49:16 INFO gemini_module [rid=- path=-] Gemini client initialized
2025-10-20 22:49:16 INFO ai_interface [rid=- path=-] AIInterface initialized
 * Serving Flask app 'aifusion_backend'
 * Debug mode: on
2025-10-20 22:49:16 INFO werkzeug [rid=- path=-] WARNING: This is a development server. Do not
use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5005
 * Running on http://192.168.1.106:5005
2025-10-20 22:49:16 INFO werkzeug [rid=- path=-] Press CTRL+C to quit
2025-10-20 22:49:16 INFO werkzeug [rid=- path=-]  * Restarting with stat
2025-10-20 22:49:16 INFO chatgpt_module [rid=- path=-] ChatGPT client initialized
2025-10-20 22:49:16 INFO claude_module [rid=- path=-] Claude client initialized
2025-10-20 22:49:16 INFO deepseek_module [rid=- path=-] DeepSeek client initialized
2025-10-20 22:49:16 INFO gemini_module [rid=- path=-] Gemini client initialized
2025-10-20 22:49:16 INFO ai_interface [rid=- path=-] AIInterface initialized
2025-10-20 22:49:16 WARNING werkzeug [rid=- path=-]  * Debugger is active!
2025-10-20 22:49:16 INFO werkzeug [rid=- path=-]  * Debugger PIN: 407-222-044
```

2. ábra: Példa a backend szolgáltatás indítására
Forrás: saját ábra

A backend magját az **AIInterface** osztály adja (definiálva az ai_interface.py fájlban), amely a különböző AI modellek elérését egységesíti. Az AIInterface a konstruktorában példányosítja a négy támogatott modellkliens osztályt (ChatGPTModule, ClaudeModule, DeepSeekModule, GeminiModule), amennyiben rendelkezésre állnak a szükséges API kulcsaik az adott szolgáltatókhoz. A 3. ábra szemlélteti az ai_interface modul használatát.

```
from ai_interface import AIInterface

iface = AIInterface(
   iChatGptApiKey_str="...",
   iClaudeApiKey_str="...",
   iDeepseekApiKey_str="...",
   iGeminiApiKey_str="..."
)
res = iface.funGetResponseFromChatGPT_dict("Hello!", "gpt-4o", 512, 0.7)
```

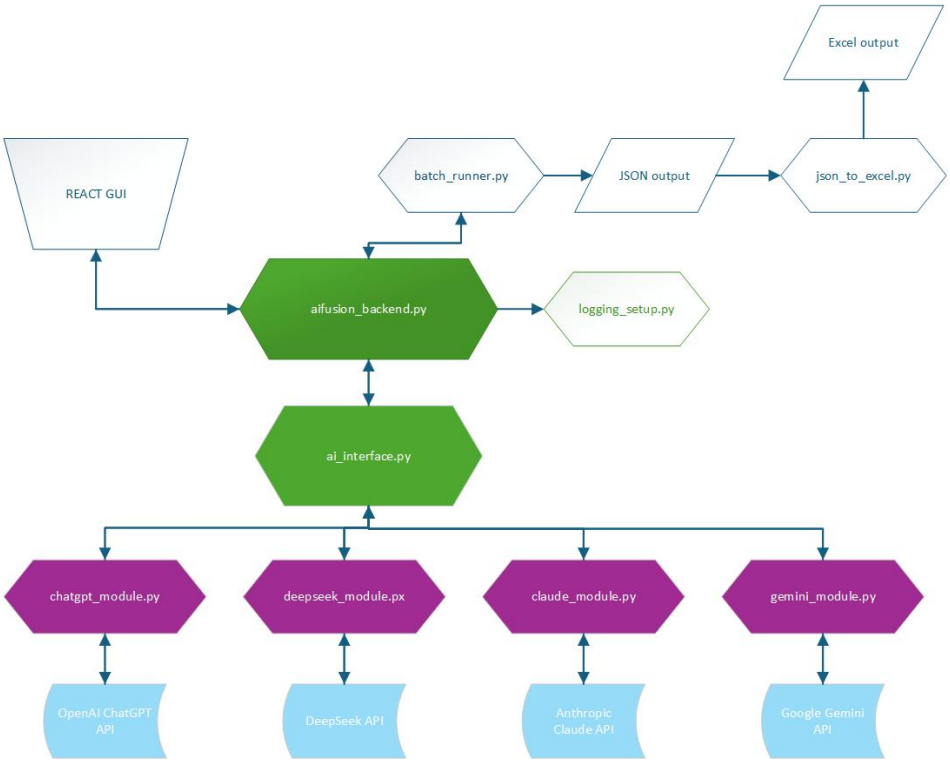
3. ábra: Példa az ai_interface importálására és futtatására
Forrás: saját ábra

Minden egyes külső modellhez tartozik egy adapter modul (pl. chatgpt_module.py, claude_module.py, stb.), amelyek egy egységes metódust (például def getResponse_str()) valósítanak meg a válasz lekérdezésére. Ez az adapterréteg elrejti a különböző API-hívások részleteit - például a ChatGPT, a DeepSeek és a Gemini modulok az OpenAI Python SDK-ját (Software Development Kit) használják a hívásokhoz, míg a Claude modul az Anthropic Claude SDK-jával hívja az Anthropic API-t - és minden modul a válaszból egységes, strukturált adatot ad vissza (pl. response_text, token statisztikák, használt modell neve, befejezés oka), hogy a felsőbb rétegek egységes formátumban dolgozhassák fel. Az AIInterface így magas szintű interfészt nyújt a backend többi része számára: a komponens expozíción keresztül elérhetők például olyan metódusok, mint getResponseFromChatGPT(), getResponseFromClaude() stb., amelyek belsőleg a megfelelő adapter objektum getResponse_str függvényét hívják meg. Ennek köszönhetően a Flask végpontok kódja nem tartalmaz modell-specifikus logikát, csak annyit tesz, hogy a kérést továbbítja a kiválasztott modell felé az egységes interfészen át. A különböző modellek eredményei egységes formában jelennek meg, mivel az adapter modulok standardizáltan adnak vissza minden fontos információt (a generált szöveget és metaadatokat, pl. token felhasználás, modell neve, befejezés oka). Ez a rétegzett megoldás átláthatóvá teszi a kódot és megkönnyíti a karbantartását.

3.2.5.2. API végpontok és hitelesítés

A Flask alapú webszerver két fő API végpontot kínál a kliens számára. Az /api/ai végpont (POST metódussal) szolgál a tényleges AI lekérdezések kiszolgálására: ide küldi a frontend a felhasználó kérdését és a beállított paramétereket JSON formátumban, pl. az AI szolgáltató típusát (ai_model), a konkrét modell nevét (model), a maximálisan felhasználható tokenek számát (max_tokens), a kreativitási szintet (temperature), stb. A backend ezen végpontja a kérés feldolgozása után JSON formátumban adja vissza a generált választ és a kapcsolódó metaadatokat (pl. token számok, válaszidő). A másik végpont a /api/test (GET), amely egy egyszerű egészség-ellenőrző (health check) funkció: hitelesítés után egy rövid üzenetet ad vissza ("API is working correctly"), jelezve, hogy a szolgáltatás elérhető és megfelelően működik. Minden érzékeny végpont – így különösen az /api/ai – megköveteli érvényes Bearer token megadását az **Authorization** fejlécben a hívás során (a prototípusban egy globális statikus token használatos a védelemhez, lásd Biztonság alfejezet). A végpontok működésének logikája röviden tehát a következő: az

/api/ai kérés beérkezésekor a backend ellenőrzi a hozzáférési tokent, majd a kapott paraméterek alapján kiválasztja, melyik AI modult kell meghívni. Ezt az AIInterface segítségével végzi: pl., ha az ai_model értéke "OpenAI", akkor a kód meghívja az AIInterface.getResponseFromChatGPT() metódust, ami végső soron a ChatGPT adapter modulhoz irányítja a lekérdezést Hasonlóképpen, más érték esetén a megfelelő modulhoz delegálódik a kérés (Anthropic esetén Claude, DeepSeek esetén a saját modulja, stb.). Az így kapott választ (valamint a metaadatokat) a backend visszaadja a kliensnek HTTP válaszként. A modulok kapcsolatát a 4. ábra szemlélteti.

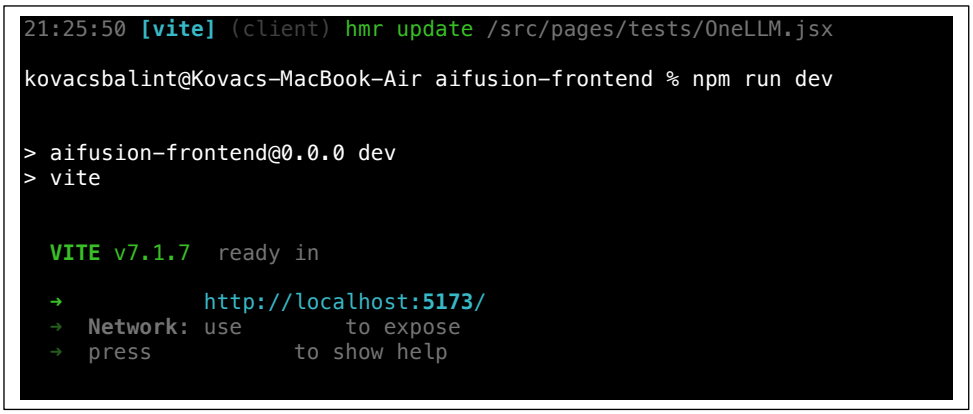


2. ábra: Az AiFusion rendszer felépítése, főbb szerkezeti elemei
Forrás: saját ábra

A Flask végpont implementációja tehát nem tartalmazza a különböző szolgáltatók sajátos kezelési logikáját – azt az adapter réteg végzi –, csak a kérések továbbítását és az eredmények összegyűjtését, formázását. Ennek köszönhetően az API könnyen bővíthető és karbantartható, hiszen új modell integrálásakor elegendő egy új adapter modult írni és az ai_interface osztályt kiegészíteni, a meglévő kódbázis minimális módosítása mellett.

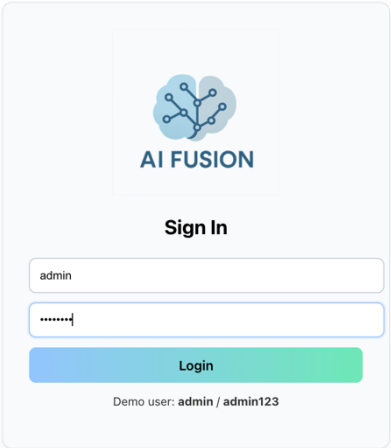
3.2.6. Frontend architektúra

Az AiFusion frontend egy **egyoldalas webalkalmazás** (Single Page Application) a React keretrendszer felhasználásával, Vite build rendszerrel. **Az 5. ábra szemlélteti a Vite futtatását.**



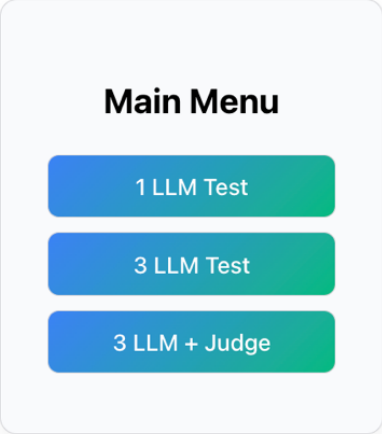
3. ábra: Példa a frontend futtatására
Forrás: saját ábra

A kliensoldali alkalmazás biztosítja a felhasználói felületet a rendszerhez: itt történik a felhasználó bejelentkezése, a lekérdezési paraméterek megadása, a kérdések elküldése a backend felé, valamint a kapott válaszok megjelenítése. A prototípusban egy egyszerű bejelentkezési mechanizmus működik: a Login oldal (Login.jsx) ellenőrzi a felhasználónév/jelszó párost egy beépített demó felhasználó (admin/admin123) ellenében (vö. 6. ábra), és sikeres belépés esetén eltárol egy loggedIn flaget a böngésző Local Storage-ében (az auth.js modulban) a session jelzésére.



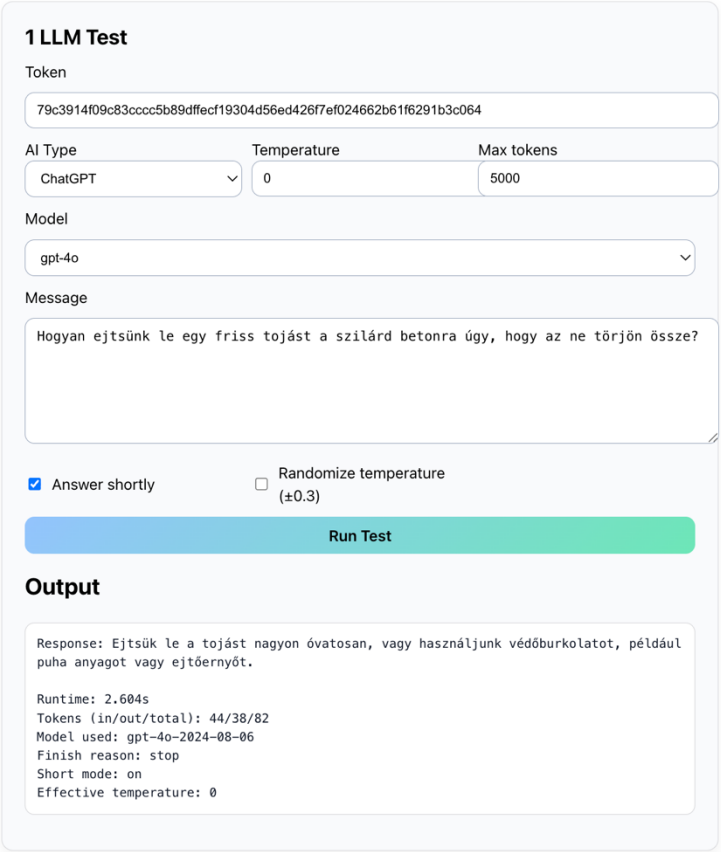
4. ábra: A login modul
Forrás: saját ábra

Bejelentkezés után a főmenü oldal (Menu.jsx) jelenik meg (vö. 7. ábra), ahonnan három tesztmód választható: 1 LLM Test, 3 LLM Test és 3 LLM + Judge. Ezek külön React komponensként vannak megvalósítva – rendre a OneLLM, ThreeLLM és ThreeLLMJudge oldalak –, melyek a menüből választva dinamikusan töltődnek be. Mindegyik teszt nézet saját űrlappal és vezérlőelemekkel rendelkezik a lekérdezés futtatásához.



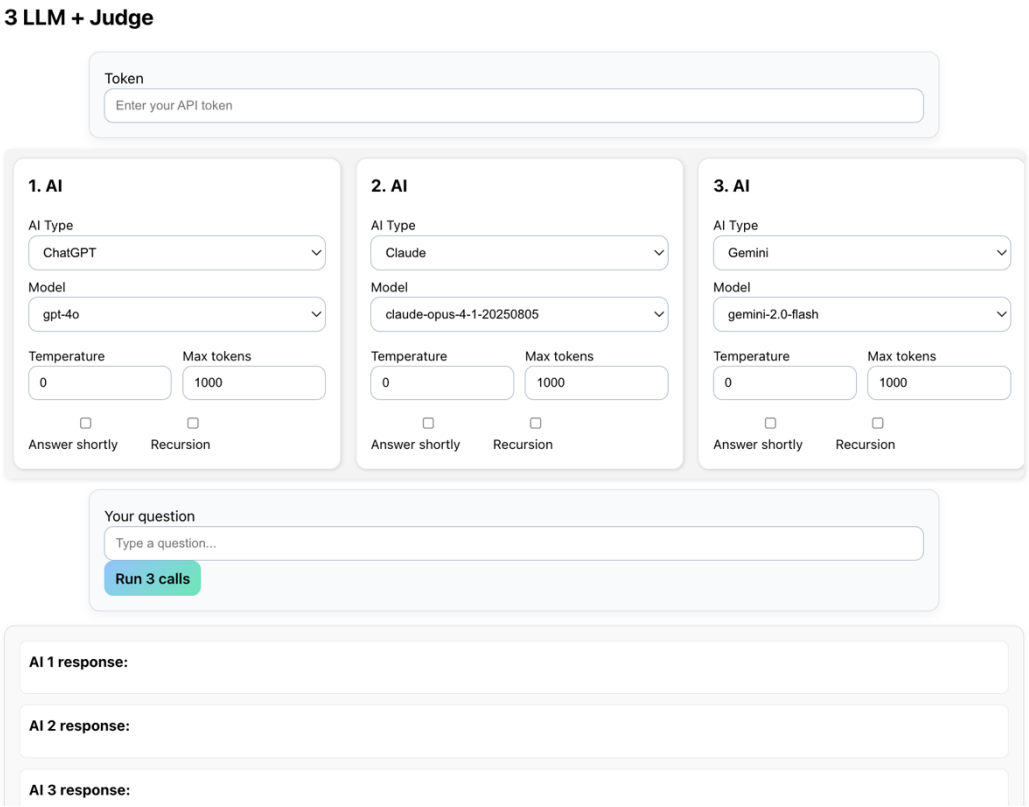
5. ábra: Egyszerű menü
Forrás: Saját ábra

A felhasználói felület (vö. 8. és 9. ábra) lehetővé teszi a lekérdezések paraméterezését. Például minden tesztoldalon található egy legördülő lista, amelyből kiválasztható az AI szolgáltató és annak konkrét modellje (ez a lista a backend által támogatott modellek neveit tartalmazza), egy számbemeneti mező a maximálisan felhasználható tokenek számának megadására, egy szöveges mező a kreativitást befolyásoló temperature érték állítására, valamint opcionális jelölőnégyzetek a válasz stílusának módosításához (pl. "Answer shortly" a rövid válaszért, "Answer randomly" a véletlenszerű fogalmazásért).



6. ábra: Egy LLM lekérdezés
Forrás: saját ábra

Ezen beállítások mind a felhasználó kezébe adják a működés finomhangolását és a kísérletezést a modellekkel. (A jelölőnégyzetek hatása technikailag abban nyilvánul meg, hogy a frontend a felhasználó kéréséhez fűz egy instrukciót a kiválasztott opcióknak megfelelően – például, ha be van pipálva a "Válaszolj röviden" opció, akkor a prompt elejéhez hozzáad egy utasítást a tömör válaszadásra.



7. ábra: Három LLM párhuzamos lekérdezés
Forrás: saját ábra

3.2.6.1. Kommunikáció a backenddel

A frontend és a backend között HTTP-alapú kommunikáció zajlik, JSON formátumú adatokkal. Amikor a felhasználó elindít egy lekérdezést (pl. rákattint a "Küldés" gombra), a frontenden egy JavaScript objektum áll össze a kérés paramétereiből – tartalmazza többek között az ai_model (AI szolgáltató típusa), a model (a konkrét modell neve), a message (a felhasználó kérdése), a max_tokens és temperature értékeket stb. Ezt követően a kód egy HTTP hívást indít a backend felé a Fetch API segítségével. A hívás a helyi fejlesztői szerveren futó végpontot éri el (http://127.0.0.1:5005/api/ai), és tartalmazza a szükséges fejlécet is az autentikációhoz: az **Authorization** fejlécben a

felhasználó által megadott Bearer tokent. Az 10. ábra szemlélteti a frontend oldali hívás logikáját JavaScript-ben:

```
const payload = { ai_model, model, message, max_tokens, temperature };
fetch("http://127.0.0.1:5005/api/ai",

{
 method: "POST",
 headers: { "Authorization": "Bearer " + token, "Content-Type": "application/json" },
 body: JSON.stringify(payload)
})
 .then(response => response.json())
 .then(data => {
  setOutput(data.response_text); // megjeleníti a kapott választ
 });
```

8. ábra: Lekérdezés küldése a backend /api/ai végpontra (egyszerűsített példa)
Forrás: saját ábra

A teljes frontend forráskód listája és a komponensek implementációja a mellékletben található.

A backend JSON formátumú választ küld vissza, amely tartalmazza a generált szöveges választ és kiegészítő metaadatokat. A frontend JavaScript kódja feldolgozza ezt a JSON választ: kiemeli belőle a response_text mezőt (és igény szerint egyéb adatokat, pl. token statisztikát), majd megjeleníti a felületen. A React komponensek állapotkezelését (state) felhasználva a kapott válasz bekerül a komponens állapotába, ami újrarenderelést eredményez, így a válasz szövege azonnal látható lesz a felhasználó számára egy <pre> vagy elemben.

3.2.6.2. Párhuzamos lekérdezések és bírói logika

A 3 LLM Test nézet kódja annyiban különbözik, hogy itt egyszerre három lekérdezést indít a frontend párhuzamosan. A program három különböző konfigurációs objektumot készít elő (három külön kiválasztott modell számára), majd a JavaScript Promise.all() metódusával egyszerre küldi el mindhárom fetch kérést a backend felé. Ezzel a megoldással a három válasz nagyjából párhuzamosan érkezik meg, jelentősen

csökkentve a teljes válaszidőt egyenkénti lekérdezéshez képest. A 3 LLM + Judge nézet ennél egy fokkal összetettebb folyamatot valósít meg. Ebben az üzemmódban először hasonló módon párhuzamosan lefut három slave modell lekérdezése (az aktuális beállítástól függően), majd, amikor mindegyik modellválasz beérkezett, a frontend egy új üzenetet állít össze. Ez az üzenet tartalmazza az eredeti felhasználói kérdést, valamint strukturált formában a kapott modellválaszokat is (pl. felsorolva vagy idézve őket), és egy második körben a program ezt az új promptot küldi el egy kiválasztott bíró modellnek egy újabb /api/ai hívással. A bíró modelltől kapott választ tekintjük a végső, egyesített eredménynek, amit a felhasználó a felületen lát. Ezt a több-lépcsős folyamatot teljes egészében a frontend orchestrálja, azaz a kliensoldali kód felügyeli a folyamat lépéseit (Promise-ok és callbackek segítségével). A backend számára mindez csupán több egymást követő API hívás sorozata – először a párhuzamos slave lekérdezések, majd a bíró lekérdezés –, de a felhasználó egyetlen összefüggő folyamatnak érzékeli a kérdésére adott kollektív választ.

(Megjegyzés: A prototípus jelenlegi UI-megvalósítása egyszerű, de működőképes. A válaszok nyers szövegként jelennek meg egymás alatt. A rendszer azonban elő van készítve a későbbi UI fejlesztésekre – pl. a kód strukturált felépítése lehetővé teszi a komplexebb megjelenítést, mint táblázatos összehasonlítás vagy eltéréskiemelés a modellek válaszai között.)

3.2.7. Batch Runner modul

Az AiFusion rendszer működésének tesztelését és a válaszok kiértékelését nemcsak manuális próbák segítik, hanem rendelkezésre áll egy Batch Runner nevű eszköz is a tömeges automatikus futtatáshoz. A batch-runner modul (a projekt batch-runner/ könyvtárában) célja, hogy nagy mennyiségű kérdést tudjunk lefuttatni a rendszeren és az eredményeket rögzíteni további elemzéshez. Például egy CSV vagy Excel fájlban megadott kérdéssorozat esetén a Batch Runner képes sorban végighívni az összes kérdést a kiválasztott modellekkel, majd a válaszokat automatikusan elmenti egy kimeneti JSON fájlba. Ez az eszköz Python nyelven, parancssori szkript(ek) formájában valósul meg, és konfigurálható paraméterekkel futtatható. A batch_runner.py szkript futásakor egy futási azonosítót vagy bemeneti fájlnevet vár paraméterként, amely alapján beolvassa a kérdéseket, majd a futás során egy, az azonosítóval és időbélyeggel ellátott JSON fájlban

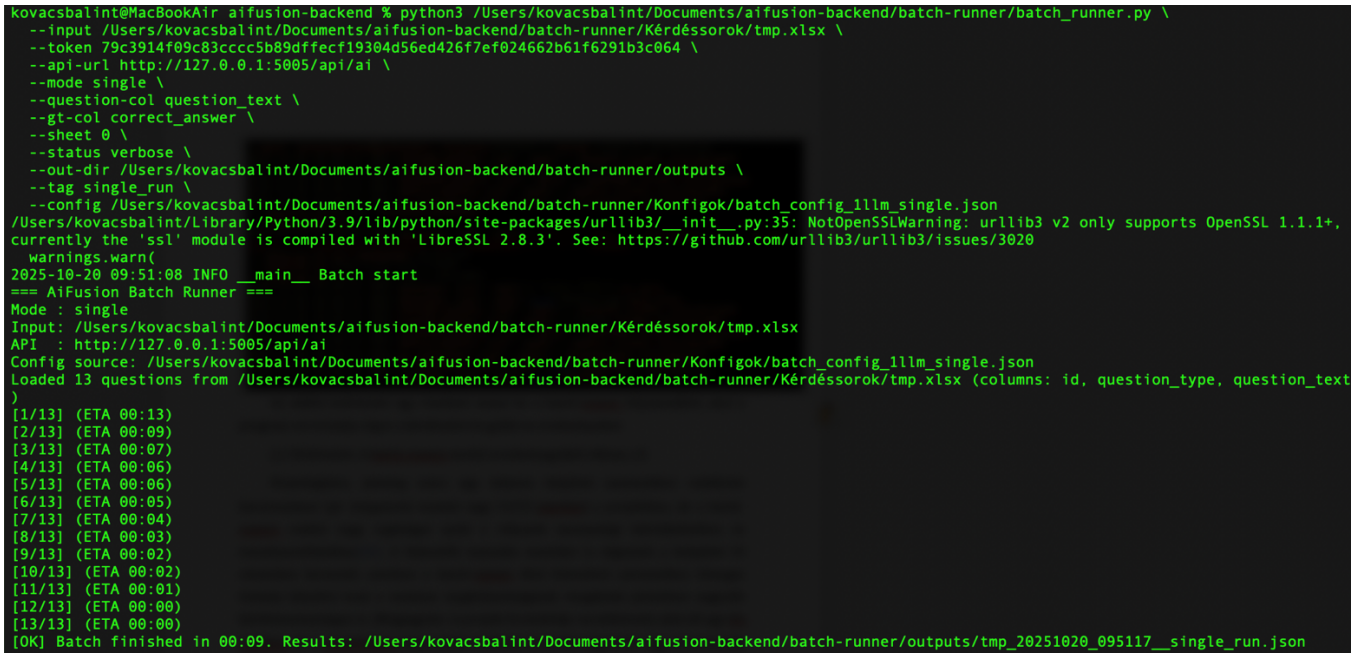
gyűjti össze a kapott eredményeket. Így a tömeges tesztek kimenetei struktúrált formában menthetők, ami megkönnyíti azok elemzését és összehasonlítását. A mentett eredményfájl minden kérdéshez tartalmazza a generált választ, valamint a kapcsolódó metaadatokat (pl. tokenfelhasználási statisztikák, válaszidő, a használt modell neve, befejezés oka, esetleges hibaüzenet), lehetővé téve a válaszok kvantitatív kiértékelését és a modellek összehasonlítását a különböző szempontok mentén.

A batch-runner modul bemeneti paraméterei:

- **--input** (kötelező) – Bemeneti fájl elérési útja. Vesszővel tagolt értékek (Comma-Separated Values – a továbbiakban: CSV) vagy Excel: .csv / .xlsx / .xls.
- **--token** (kötelező) – A Flask API-hoz szükséges Bearer token.
- **--api-url** *(alapértelmezés: http://127.0.0.1:5005/api/ai)* – A Flask API végpontja.
- **--config** *(opcionális)* – JSON konfigurációs fájl útvonala; ha nincs megadva, a script megpróbálja a batch_config.default.json-t megtalálni (script könyvtár / aktuális mappa), ellenkező esetben beépített defaultokat használ.
- **--mode** *(alapértelmezés: collab)* – Futási mód: collab (3 válaszadó LLM + 1 judge) vagy single (csak per-modell futások). Más érték hibát dob.
- **--method** *(opcionális, egész szám)* – A judge prompt-módszer indexe (0/1/2). Csak collab módban releváns; tartományon kívül hiba.
- **--question-col** *(opcionális)* – A kérdés oszlopnév felülírása (pl. question_text).
- **--gt-col** *(opcionális)* – A helyes válasz / ground-truth oszlopnév felülírása (pl. correct_answer).
- **--options-col** *(opcionális)* – Az opciók/alternatívák oszlopnév felülírása (pl. options/choices).
- **--qtype-col** *(opcionális)* – A kérdéstípus oszlopnév felülírása (pl. question_type).
- **--sheet** *(opcionális)* – Excel munkalap indexe (0,1,…) vagy neve (pl. Sheet1). CSV esetén ignorálva van.
- **--status** *(alapértelmezés: normal)* – Konzolos kiírás szintje: quiet | normal | verbose.
- **--log-file** *(opcionális)* – Ha megadjuk, ide is naplóz (append módban).
- **--out-dir** *(opcionális)* – Kimeneti mappa; ha nincs megadva, az input fájl mappája lesz. A kimeneti fájlnév időbélyegzett .json.

- **--tag** *(opcionális)* – Opcionális címke, amely bekerül a kimeneti fájlnévbe (pl. method2).

A 11. ábra egy kódrészletet mutat be a batch-runner folyamatából, ahol a program sorra hajtja végre a kérdéseket és gyűjti az eredményeket:



9. ábra: Kódrészlet: A batch_runner modul eredménygyűjtő ciklusa
Forrás: saját ábra

Összefoglalva, jelenleg nincs egy teljesen kiépített automatikus validációs keretrendszer (pl. integrációs tesztek vagy CI/CD pipeline) a projektben, de a Batch Runner eszköz nagy segítséget nyújt a válaszok mennyiségi kiértékelésében és összehasonlításában. A fejlesztők manuális teszteket is végeznek a beépített UI nézeteken keresztül, azonban a Batch Runner által biztosított automatikus tömeges futtatás lehetővé teszi a rendszer megbízhatóságának vizsgálatát jelentősen nagyobb kérdésmennyiségen is. (Megjegyzés: A projekt forráskódja verziókövetés alatt áll egy privát Git (elosztott verziókezelő rendszer) repóban, de a commitokhoz kapcsolódó részletes tesztelési jegyzetek vagy automatizált futtatások jelenleg nem dokumentáltak.)

3.2.8. Nevezéktani konvenciók

A kódolás során következetes nevezéktani konvenciókat alkalmaztam annak érdekében, hogy a forráskód olvasható, karbantartható és önmagát dokumentáló legyen. A régi, VB6-ból eredő szokásaimból eredezik, hogy a kódrészletnek önmagában is

értelmezhetőnek kell lennie, tehát a változó és függvényneveknek szélsőségesen beszédesnek kell lenniük, a kódrészlet olvasását minél gördülékenyebbé és gyorsabbá kell tenni. Ez a hozzáállás segíti a nagy tömegű kódok hatékony átlátását és bennük a hibakeresést.

3.2.9. A Python kódban alkalmazott nevezéktan

- A változónevek végén típusjelölő szuffixumot használok (pl. _str, _int, _float, _bool, _list, _dict, _val, stb.), például userName_str, retryCount_int.
- A függvények bemeneti paraméterei “i” előtagot kapnak, a kimeneti (visszatérő) értékek pedig “o” előtagot (pl. iUserName_str, oResponse_str).
- A függvénynevek „fun” prefixszel kezdődnek, utána CamelCase alakban leírom a funkciót, és a végére kerül a visszatérési érték típusának szuffixuma – például “funBuildPrompt_str” vagy “funMergeScores_dict”.
- A logikai (bool) változók is/has/should kezdetű nevet kapnak és _bool végződést, pl. “isValid_bool”, “hasToken_bool”.
- Az osztályok neve PascalCase formátumú (pl. AiFusionInterface). A konstansokat nagybetűs snake_case szintaxissal nevezzük el (szükség esetén típusszuffixummal), pl. DEFAULT_PORT_INT. A modulfájlok neve kisbetűs snake_case (pl. logging_setup.py).

3.2.10. Biztonság és kulcskezelés

A rendszer biztonsági architektúrájának központi eleme egy egyszerű token alapú autentikáció. A backend szerver indulásakor generál egy véletlenszerű, 64 karakter hosszú hexadecimális API_SECRET_KEY tokent, és kiírja ennek értékét a konzolra (ezt a fejlesztő kimásolja és betölti a kliensoldali alkalmazásba teszteléskor). Minden beérkező API hívásnál a szerver ellenőrzi az **Authorization** fejlécet: egy CheckAuth() segédfüggvény vizsgálja, hogy a header Bearer token formátumú-e, és megegyezik-e a szerver által ismert titkos kulccsal. Ha nem, a szerver HTTP 401 Unauthorized hibával visszautasítja a kérést, meggátolva ezzel az illetéktelen hozzáférést. Ez a megoldás

egyszerű, de a prototípus számára elegendő védelmet nyújt: biztosítja, hogy csak az férjen hozzá a rendszer API-jához, aki ismeri a szerver által generált titkos kulcsot.

A külső AI szolgáltatókhoz tartozó API kulcsok kezelése szintén kritikus biztonsági kérdés. Jelenleg az OpenAI, Anthropic stb. API kulcsai a kód egy konfigurációs szakaszában, gyakorlatilag hardcode-olt módon vannak tárolva a backendben. Ez nyilvánvalóan nem ideális megoldás biztonsági szempontból. A szakdolgozat dokumentációja is kiemeli, hogy ezt a módszert csak a demó fázisban tartottam elfogadhatónak; a későbbi fejlesztések során át kell térni a kulcsok biztonságosabb kezelésére. Ennek egyik módja a kulcsok egy környezeti változókat tartalmazó .env fájlban történő tárolása, és annak kihagyása a forráskódból (felvétele a .gitignore listába). A projektben ennek előkészítése részben már megtörtént: a Python oldalon fel van véve a python-dotenv csomag a függőségek közé, és a repository konfigurációban a .env fájl már ignorálva van. Általános elv, hogy nyilvános kódtárházban soha nem szabad titkos API kulcsokat tárolni –a jelenlegi prototípus egy privát repóban készült, de egy éles rendszerben mindenképp gondoskodni kell a kulcsok titkos kezeléséről. A kulcskezelési architektúra része az is, hogy a frontenden egyáltalán nem tárolunk érzékeny API kulcsokat – a front csak egy hozzáférési tokent kezel, azt is a felhasználó adja meg számára indításkor, és a tényleges titkos API kulcsok csak a backend konfigurációban léteznek.

3.2.11. A jelenlegi megoldás biztonsági korlátai

Érdemes kitérni néhány további biztonsági megfontolásra. Jelenleg minden szerver-indításkor új token generálódik, azonban ennek a tokennek nincs lejárati ideje vagy egyéb érvényességi korlátja. A jövőben ajánlott lehet időkorlátos (expire) tokeneket bevezetni, vagy – amennyiben a rendszer valódi felhasználókezeléssel bővül – felhasználónként egyedi tokeneket kezelni a finomabb jogosultságkezelés érdekében. További fontos lépés a biztonságos hipertext átviteli protokoll (Hypertext Transfer Protocol Secure – a továbbiakban: **HTTPS)** használata, ha a rendszert interneten keresztül érjük el: jelenleg (fejlesztői környezetben, localhoston) a token nyílt szövegként utazik a hálózaton, ami helyben nem jelent veszélyt, de egy éles telepítésnél TLS (Transport Layer Security) titkosítással kell védeni a kommunikációt, hogy a token ne legyen lehallgatható. Hosszabb távon felvetődik egy korszerűbb autentikációs mechanizmus integrálása is – például JSON web tokenek (JSON Web Token – a továbbiakban: JWT) tokenek vagy OAuth2 alapú

bejelentkezés –, különösen akkor, ha a szolgáltatást több felhasználóval, publikus formában tervezem üzemeltetni.

3.2.12. Naplózás és adatvédelem

Az architektúra tervezésekor szempont volt, hogy érzékeny adatok ne kerüljenek naplózásra. A backend naplói (logjai) jelenleg minimális információt tartalmaznak – induláskor kiírják a generált titkos tokent, illetve a Flask keretrendszer alapvető kimenetei látszanak. A kódban ugyanakkor megvannak a helyek további adatok naplózására (például egy lekérdezés azonosítója, feldolgozási ideje, token felhasználás mértéke, a használt modell neve stb.), de fontos, hogy sem a felhasználó által küldött üzenet, sem az API kulcsok ne jelenjenek meg a logokban. Ezt a fejlesztés során szem előtt tartottam: a szerver oldali logolás nem írja ki a prompt szövegét vagy a kulcsokat, a kliensoldali fejlesztői konzolon pedig bár látható a fetch hívások URL-je és headerje, a token értékét a felület nem jeleníti meg nyilvánosan (az a böngésző memóriájában marad, és a konzol amúgy is csak a fejlesztő számára érhető el). A jövőben érdemes lehet egy összetettebb monitorozó és napló-elemző megoldást bevezetni a rendszerhez – például audit logokat, metrika-gyűjtést és riasztásokat tartalmazó monitoring rendszert –, hogy a használatot és esetleges rendellenességeket nyomon lehessen követni anélkül, hogy a felhasználói adatok veszélybe kerülnének.

3.2.13. Jövőbeli fejlesztési irányok

Az AiFusion platform prototípusa számos továbbfejlesztési lehetőséget vet fel, amelyek a rendszer jövőbeni kibővítését és éles bevetésre alkalmassá tételét célozzák. Az alábbiakban összefoglalom a legfontosabb lehetséges fejlesztési irányokat.

3.2.13.1. Működési logikák kibővítése

A rendszer fejlesztése során sok, előre nem látott lehetőség felmerült a platform hasznosíthatóságával kapcsolatban, pl. a platform alkalmas AI ranking feladatok elvégzésére, hibabecslésekre. Adott tesztsorozat könnyen lefuttatható egy újonnan kiadott LLM esetében és a kapott végeredmény gyorsan illeszthető kiértékelésre a már meglévő eredmények közé. A több LLM által generált válaszokon módszereit bővíteni lehetne. A jelenleg TÖBB LLM + JUDGE módszer mellett számos dinamikusabb

együttműködési lehetőséget lehetne implementálni (pl. a LLM-ek egymással való vitatkozása megadott körön keresztül, majd közös álláspont kikényszerítése és az esetleges különvélemény kezelése).

3.2.13.2. Felhasználói felület és UX fejlesztések

A jelenlegi frontend funkcionalitása alapvetően demonstrációs célokra készült, így számos lehetőség van a továbbfejlesztésére. Javítani lehetne a megjelenést és a használhatóságot: például a generált válaszok jelenleg egyszerűen egymás alatt jelennek meg, formázás nélkül – a jövőben érdemesebb lenne a válaszokat áttekinthetőbb formában prezentálni (akár táblázatosan vagy külön panelekben). A felületet reszponzívvá kell tenni, hogy mobil eszközökön is megfelelően használható legyen. Emellett a jelenlegi statikus, egyfelhasználós autentikációt (fix demó login) le kell cserélni egy valódi többfelhasználós bejelentkezési rendszerre. Ez azt jelentené, hogy regisztrációs és bejelentkezési funkciókat kell bevezetni, felhasználónként külön azonosítóval és jogosultságokkal. Így minden felhasználó saját API tokent kaphatna, és a token kezelését is automatizálhatná a rendszer (pl. bejelentkezéskor a szerver generál egy JWT tokent, amit a frontend tárol és használ minden kéréshez). Mindezt természetesen biztonságos módon, HTTPS-en keresztül érdemes megvalósítani, hogy a felhasználói adatok és tokenek ne kerülhessenek illetéktelen kézbe. A jobb UX érdekében továbbá érdemes olyan kényelmi funkciókat is bevezetni, mint a lekérdezés közbeni státuszkijelzés (pl. "Loading..." jelzés, amíg a modellek válaszolnak) vagy a korábbi kérdések és válaszok megjelenítése (esetleges jövőbeni chat funkció alapjaként).

3.2.13.3. Infrastruktúra és skálázhatóság

Ahhoz, hogy az AiFusion a prototípus fázisból produkciós környezetbe léphessen, néhány infrastruktúrális fejlesztést is érdemes mérlegelni. Az alkalmazás **konténerizálása** (Docker használatával) megkönnyítené a telepítést és a hordozhatóságot, hiszen a teljes stacket egy konténerbe zárva egyszerűbben futtatható különböző környezetekben. A skálázhatóság érdekében szóba jöhet a microservice architektúra irányába való elmozdulás is – azaz a frontend, a backend, sőt akár az egyes adapterek vagy a bírói logika külön szolgáltatásként, skálázható módon futtatása. Így nagyobb terhelés esetén a komponensek külön-külön skálázhatók lennének (pl. több párhuzamos backend worker vagy külön microservice a bírói döntésekhez).

3.2.13.4. Jogi háttér szerkezete

Amennyiben a platform továbbfejlesztésre kerül, célszerű lehet a rendszer logóját és nevét védjegyoltalommal ellátni. Korábbi tapasztalataim alapján a védjegyre való hivatkozás megkönnyíti az esetleges jogviták rendezését, vagy pl. téves Facebook tiltások kezelését. Továbbá a portfóliók keveredésének megakadályozása érdekében, a platform kezelésére érdemes lehet önálló céget bejegyezni akár Magyarországon, akár Magyarországon kívül.

Fontos a megfelelő felhasználói licencek és szabályzatok létrehozása. Tanulmányozni kell, hogy a platform megfelel-e a hatályos jogi környezetnek. A megfelelés egyik fontos eleme, annak tisztázása, hogy az AiFusion platform szolgáltatásközvetítőként van-e jelen a LLM szolgáltatók és a felhasználók, az AiFusion platformon keresztül végzett interakciójában. Véleményem szerint a rendszer jogi státuszát illetően fontos megjegyezni, hogy az AiFusion a jelenlegi prototípus-terv alapján nem minősül a *„2001. évi CVIII. törvény (Eker. tv.)”* szerinti *„közvetítő szolgáltatónak”* (Wolters Kluwer, 2001). Bár a rendszer végez adattovábbítást a felhasználó és a külső LLM szolgáltatók (pl. OpenAI, Google) között, nem passzív közvetítőként működik. A platform aktív, hozzáadott értéket teremtő funkciót tölt be (pl. orkesztráció, bírói modell általi kiértékelés). Ezen felül, a tervezett működési modellben a felhasználó a saját API kulcsait használja, így a pénzügyi és szolgáltatási jogviszony közvetlenül a felhasználó és a LLM szolgáltató között jön létre, az AiFusion platform pedig inkább egy, a felhasználó által birtokolt erőforrásokat kezelő intelligens szoftvereszközként funkcionál.

3.2.13.5. Adatbázis kapcsolat

A rendszer fejlesztésének következő szakaszában adatbázis-kezelő rendszer bevezetése válik szükségessé az adatok strukturált, biztonságos és hatékony tárolása érdekében. Az adatbázis többféle információ kezelését fogja ellátni, például a kérdés-adatbázist (feladatok, válaszopciók, helyes megoldások), a modellválaszok és eredmények tárolását, valamint a tesztfutások metaadatait (időbélyegek, paraméterek, sikerességi arányok). A legvalószínűbb jelöltek közé tartozik egy relációs adatbázis (pl. PostgreSQL vagy MySQL), amely jól illeszkedik a rendszer Python alapú backendjéhez, illetve megfontolható egy NoSQL (amely gyakran a "Not only SQL", azaz "nem csak SQL" rövidítése) megoldás (pl. MongoDB) is a nagyobb rugalmasság érdekében. Jelenleg

azonban a rendszer még a finomhangolási és kísérleti szakaszban van, ezért az adatrekordok viszonylag alacsony száma (néhány ezer tétel nagyságrendjében) és a vizuális átláthatóság igénye miatt az adatok kezelése Excel-fájlokon keresztül történik. Ez a megoldás rövid távon megkönnyíti az adatok manuális ellenőrzését és korrekcióját, de nem jelent hosszú távú megoldást: a rendszer jövőbeli bővítése és a tömeges adatfeldolgozás igénye elkerülhetetlenné teszi az adatbázis-integráció megvalósítását.

Összességében elmondható, hogy az AiFusion jelenlegi architektúrája jól szemlélteti a több modell együttműködésén alapuló válaszadási koncepciót, ugyanakkor a fenti fejlesztési irányok mentén továbbhaladva a prototípus egy ipari környezetben is helytálló, kiforrott rendszerré fejleszthető. A javasolt bővítések és optimalizációk megvalósításával az AiFusion platform a jövőben még hatékonyabban és biztonságosabban szolgálhatná a felhasználókat, illetve további kísérleti lehetőségeket nyújthat a kollektív mesterséges intelligencia területén.

3.3. Kismintás tesztelés

Az előző alfejezetekben bemutatott architekturális és implementációs megoldások gyakorlati értékeléséhez kismintás tesztelést végeztem. A következő alfejezetek ezt a tesztkörnyezetet, a felhasznált eszközöket és a vizsgálat főbb lépéseit ismertetik.

3.3.1. Áttekintés, bevezetés

Az AiFusion platform **technikai működésének** demonstrálásához/teszteléséhez egy 20 kérdésből álló feladatsort állítottam össze, az Interneten fellelhető olyan *„fejtörőkből”,* amelyekhez ismert helyes válasz is tartozik (50. 7 Math Riddles Only the Smartest Can Get Right, 2025), ([51.] 15 Challenging Logic Puzzles and Their Answers, 2025), ([52.] Difficult Mathematical Puzzles - 5, 2025). A feladatsort 22 LLM-nek adtam át megválaszolásra (összesen 440 db lekérdezés). A JSON kimenetek .xlsx formátumra lettek konvertálva, majd Excelben kerültek letárolásra és kiértékelésre (lásd: 1. sz. melléklet). A kérdések és modellek listája szintén az 1. sz. mellékletben található.

3.3.2. A Batch Runner használata

A tesztelés a rendszer „Batch Runner” moduljának felhasználásával történt. A Batch Runner (batch_runner.py) képes kérdéssorok kötegelt lekérdezésére. A program bemenete rugalmas, a kérdéseket Excel-táblában, CSV-ben, vagy JSON-ban is képes fogadni. A teszt során a Batch Runner számára két bemeneti fájl került átadásra:

1. A kérdéseket tartalmazó Excel-tábla, amelynek az alábbi két oszlopot tartalmaznia kell:
    a. **question_type**: a kérdés típusa. Két féle lehet: open_ended -› azon kérdésekhez, amelyek válaszát szabadon kell megfogalmazni. multiple_choice -› a feleletválasztós kérdésekhez.
    b. **question_text**: kérdések szövegtörzse.
    c. **(opcionális) options**: multiple_choice esetén a választható feleletek
2. A kérdések lekérdezéséhez szükséges konfigurációt tartalmazó JSON fájlt. A JSON fájl tartalmazza a használni kívánt nagy nyelvi modelleket, azok paraméterezésével együtt (vö. 12. ábra).

```
{
      "ai": "Gemini",
      "model": "gemini-2.0-flash-
lite",
      "temperature": 0.0,
      "max_tokens": 20000,
      "answerShortly": false,
      "recursion": false
    }
```

10. ábra: Részlet a bemeneti JSON fájl struktúrájából
Forrás: saját ábra

A Batch Runner a bemeneti Excel táblában található összes kérdést felteszi a bemeneti JSON fájlban található összes nagy nyelvi modellnek. A tesztben 20 kérdés kerül megválaszolásra 22 LLM által (összesen 440 kombináció). A tesztelésre használt kérdések megtalálhatóak az 1. sz. melléklet „Q&A” lapján, a használt nagy nyelvi modellek listája szintén az 1. sz. melléklet mellékletben, a „Models and pricing” oldalon találhatóak. A batch runner futtatását a 13. ábra szemlélteti.

```
kovacsbalint@MacBookAir aifusion-backend % python3 /Users/kovacsbalint/Documents/aifusion-backend/batch-
runner/batch_runner.py \
  --input /Users/kovacsbalint/Documents/aifusion-backend/batch-runner/Kérdéssorok/tmp.xlsx \
  --token 79c3914f09c83cccc5b89dffecf19304d56ed426f7ef024662b61f6291b3c064 \
  --api-url http://127.0.0.1:5005/api/ai \
  --mode single \
  --question-col question_text \
  --gt-col correct_answer \
  --sheet 0 \
  --status verbose \
  --out-dir /Users/kovacsbalint/Documents/aifusion-backend/batch-runner/outputs \
  --tag single_run \
  --config /Users/kovacsbalint/Documents/aifusion-backend/batch-runner/Konfigok/batch_config_1llm_single.json
```

11. ábra: Példa Batch Runner hívására
Forrás: saját ábra

3.3.3. A tesztkérdések és azok lekérdezése

A teszt során a **3 LLM + 1 JUDGE** modellt vizsgáltam. A kollaborációs partnerek megtalálásához először ki kellett értékelnem a modellek egyéni teljesítményét, majd az egyéni teljesítményekből következtettem a modellek elméletileg lehetséges összeadott teljesítményére. A 440 válasz helyességét az ismert helyes válaszok vonatkozásában manuálisan, kézzel validáltam.

A kiértékelés Excel táblában történt. Ehhez a kimeneti JSON fájlt át kellett konvertálnom xlsx-be, amelyhez írtam egy segédprogramot (json_to_excel.py). A segédprogram használatát a 14. ábra szemlélteti.

12. ábra: A json_to_excel.py használata
Forrás: saját ábra

3.3.4. A kollaborációs partnerek és a JUDGE kiválasztása

Első lépésként az egy kollaborációban résztvevő partnerek számát határoztam meg. A lehetséges LLM párosok elméleti maximális sikerrátáját szemléltető segédmátrixon (**17. ábra**) látszik, hogy már két kollaboráló LLM is képes lehet 100%-os sikerrátát nyújtani a kérdéskorpuszon, azonban jelen esetben nem a sikerráta maximalizálása volt

a cél, hanem az AiFusion rendszer tesztelése, így fokozva az összetettséget, az egy csoportban kollaboráló LLM-ek számát önkényesen háromban határoztam meg.

A 22 db LLM-ből többféleképpen ki tudunk választani hármat. A legegyszerűbb módszer szerint minden lehetséges triádot megvizsgálunk, tehát meg kell vizsgálnunk, a 22-ből hányféleképpen tudunk kiválasztani hármat. A helyes számítás a

$$\binom{22}{3} = 1540$$

tehát összesen 1540 egyedi kombinációja létezik a 22 LLM-nek. Ha ezt az 1540 kombinációt a 20 kérdésre vetítve, mind a 22 LLM-et JUDGE-ként is kipróbálva szeretnénk számolni, az alábbi számítást kapjuk:

$$\binom{22}{3} \times 20 \times 22 = 677\,600$$

Mivel jelen esetben nem a teljeskörű kiértékelés a cél, hanem a rendszer technikai működésének demonstrálása, a teszteléshez Pareto-szűréssel választottam ki a kollaboráló triád-csoportokat (vö. 3.3.4.1. A Pareto-triádok kiválasztása), valamint önkényesen 7 db LLM-t választottam ki JUDGE modellként. A kiválasztott JUDGE modellek a gpt-4.1-nano, a gpt-5-nano, a gemini-2.0-flash-lite, a gemini-2.5-flash-lite, a gpt-4.1-mini, a deepseek-chat és a claude-3-5-haiku-20241022.

3.3.4.1. A Pareto-triádok kiválasztása

A triádok kiválasztása **többcélú optimalizálás** eredménye: a sikerrátát maximalizáljuk, miközben a költséget és a futásidőt minimalizáljuk. A jelölteket a *„Pareto-front”* (Wikipedia, [21.] Pareto front, 2025), vagy másképpen *„Pareto-hatékonyság”* (Wikipedia, [18.] Pareto-hatékonyság, 2025), alapján választottam ki: ide olyan triádok kerülnek, amelyeket egyetlen más triád sem dominál a (siker, költség, idő) hármas dimenzióban. Ez a többcélú optimalizálási szemlélet, amely a hatékonyságra fókuszál, már korábbi intézményi hallgatói munkákban is megjelent, például *„gyártástervezési folyamatok kapcsán”* (Kerepesi, 53. Gyártástervezés és termelésirányítás folyamatainak optimalizálása az Országos Villamostávvezeték Zrt.-nél, 2013). Minden kiválasztott triád **ésszerű kompromisszum** a célok között – a döntéshozó preferenciái (pl. költségérzékenység, időkorlát) szerint a front bármely pontja indokolható.

A 22 db LLM-t 1540 egyedi kombinációjából a kiválasztott Pareto triádok száma: 11db. Értelmezés: egy triád nem dominált, ha nincs másik, ami ≥ sikert ad ÉS ≤ költség ÉS ≤ idő mellett (legalább az egyikben szigorúan jobb).

A triádok összevetésénél a következő aggregálási szabályokat alkalmaztam:

- **Sikerráta**: azon kérdések aránya, amelyeknél legalább egy triádtag helyeset adott (logikai OR).
- **Költség**: kérdésenként a triádtagok költségeinek összege, majd átlag a teljes kérdéskorpuszra.
- **Futásidő**: párhuzamos futtatás feltétele mellett kérdésenként a triádtagok futásidejének maximuma, majd átlag a teljes korpuszra.
- **Lefedettség** (coverage): a triád által ténylegesen lefutott kérdések aránya (a jelen mintában 100%).
- **A triádokat sikerráta-sávokba soroltam** (≥95%, 90–95%, 80–90%, 70–80%, 60–70%, 50–60%). Minden sávon belül három „legjobb" triádot emelek ki, eltérő preferenciákra optimalizálva:
    1. „cheapest" - a legalacsonyabb költségű
    2. „fastest" - a legalacsonyabb átlagos futásidejű
    3. „balanced" – a legjobb ár/futásidő arányú

Fontos megjegyezni, hogy az 1. számú melléklet elkészültekor a ChatGPT 5-ös verziója a temperature bementei érték hiánya miatt még nem került kizárásra a kollaborációs tesztekből (vö. Vita – 4.2), valamint idő közben kiderült, hogy a külső forrásból beszerzett kérdéssorokhoz tartozó helyes válaszok bizonyos kérdéseknél hibásak, vagy kétértelműek. Mivel a szakdolgozat célja a kollaborációs kutatás hátthereként szolgáló platform technikai működésének bemutatása, a hibás tesztadatok és a ChatGPT 5 kizárása a rendszer technikai értelemben vett működésének bizonyítását nem befolyásolta.

3.3.4.2. Példa a Pareto-dominanciára

Formálisan, legyen minden triádhoz két célfüggvény rendelve:

f_1: a lefedettség (coverage) maximalizálandó, f

f_2: a redundancia (redundancy) minimalizálandó.

Ekkor egy *A* triád dominál egy másik *B* triádot, ha az alábbi feltételek teljesülnek:

$$\mathrm{A} \prec \mathrm{B} \Leftrightarrow \begin{cases} f_1(A) \geq f_1(B), \\ f_2(A) \leq f_2(B), \\ \text{és legalább az egyik egyenlőtlenség szigorú.} \end{cases}$$

A Pareto-front az összes olyan triád halmaza, amelyet egyetlen másik triád sem dominál:

$$P = \{\mathrm{Ai} \mid \nexists \mathrm{Aj}: \mathrm{Aj} \prec \mathrm{Ai}\}$$

Megjegyzés 1.: a triádok elemzése önmagában még nem végeredmény, mert egy JUDGE modell beépítése is szükséges az ismeretlen válaszú kérdések megválaszolásához.

Megjegyzés 2.: Az elméleti 100%-os sikerráta két LLM kombinálásával is elérhető (Pareto-duók), azonban a teszthez triádokat használtam.

Fontos! A tesztelés közben kiderült, hogy a bemeneti kérdésekre a megadott helyes válaszok között szerepelt hibás, vagy kétértelmű válasz. A tévesen megadott „helyes" válaszokat a kollaborációs válaszok kiértékelése során a rendszer jelezte, ugyanis a kérdés-triád-JUDGE kimeneten rendkívül nagy szórás mutatkozott a kimeneti helyes választ illetően azokhoz a kérdésekhez képest, amelyek előre megadott helyes válasza nem volt hibásan megadva. Ezzel az AiFusion rendszer egy új vizsgálati területre mutatott rá: a végső válaszok konzisztenciája mintázatot és irányt mutathat egy ismert válasz helyességének elbírálásában. A hiba rávilágít arra a tényre, hogy a rendszer kezdeti tesztelése és finomhangolása terjedelmét és összetettségét tekintve akár egy külön szakdolgozat témáját adhatja, ezért a továbbiakban a hibás adatokkal dolgozom, ugyanis a cél a rendszer használhatóságának bemutatása, nem pedig kimeneti eredményeinek értékelése.

3.3.4.3. A Pareto-szűrő és annak használata

A Pareto-front megállapításához és a Pareto-triádok kiválasztásához létrehoztam egy Python kódban írt „Pareto-szűrő” programot (melléklet: pareto_select_triads.py).

A Batch Runner által előállított futási naplókat (per LLM: helyesség, token- és költségadatok, futásidő) a pareto_select_triads.py dolgozza fel. A bemenet egy Excel fájl (pareto_calculator_input.xlsx), „runs” munkalappal. Minden sor egy (kérdés, modell) kimenetet reprezentál. A pareto_select_triads.py összeállítja az összes lehetséges triádot, kiszámítja a ⟨success_rate, cost_usd, latency_s, coverage⟩ mutatókat, majd Pareto-szűrést végez, végül sávonként kiválasztja a cheapest / fastest / balanced triádot. (A Batch Runner és az Excel/JSON kimenetek használatát a 3.3.2. és 3.3.3. alfejezetek mutatják be.)

Fő konfigurációs mezők (pareto_config_triads.json):

- **input_xlsx_path_str, input_sheet_name_str:** bemeneti állomány és munkalap (alapértelmezés: pareto_calculator_input.xlsx / runs).
- **exclude_models_regex_list:** reguláris kifejezések listája a kizárandó modellekre (pl. ["^gpt-5.*"]).
- **bands_list:** sikerráta-sávok alsó határai 0-100-ig (pl. [95, 90, 80, 70, 60, 50]).
- **min_coverage_float:** minimális lefedettség (pl. 1.0).
- **select_roles_list:** mely kiválasztásokat kérjük sávonként (["cheapest", "fastest", "balanced"]).
- **out_json_bool, out_xlsx_bool, out_dir_str:** kimenetformátum(ok) és célkönyvtár.
- **round_digits_int:** kerekítés a riportolt mezőknél.

A program kimenetként JSON és opcionálisan Excel összegzést ad. A JSON struktúrában triádonként szerepel a modellek listája, a sáv, a szerep (cheapest/fastest/balanced) és a fő mutatók. Az Excel kimenet a kézi adatelemzéshez kényelmes.

A triádok kiválasztása reprodukálható. A futások során rögzítem a konfig fájlt, a bemeneti Excel verzióját és az időbélyeget, a triádok így megismételhetően előállíthatók ugyanazon beállítások mellett.

3.3.4.4. A Pareto-triádok felsorolása

Az itt felsorolt triádok kiválasztása a ChatGPT 5-ös verzióinak **kizárását követően** történt. Az **1. számú** mellékletben a **kizárás előtti** állapot látható.

≤95% sáv:

- Fastest: ['DeepSeek | deepseek-chat', 'Gemini | gemini-2.0-flash', 'Gemini | gemini-2.5-pro'], 95%, 16,95645s, 0,013973228 USD
- Cheapest: ['DeepSeek | deepseek-chat', 'DeepSeek | deepseek-reasoner', 'Gemini | gemini-2.0-flash-lite'], 95%, 80,9133s, 0,000928859 USD

90%-95% sáv

- Fastest: ['DeepSeek | deepseek-chat', 'Gemini | gemini-2.0-flash-lite', 'Gemini | gemini-2.5-pro'], 90%, 16,95645s, 0,013967823 USD
- Cheapest: ['ChatGPT | gpt-4.1', 'DeepSeek | deepseek-reasoner', 'Gemini | gemini-2.0-flash-lite'], 90%, 80,9133s, 0,001299934 USD

80%-90% sáv

- Fastest: ['Claude | claude-sonnet-4-5-20250929', 'DeepSeek | deepseek-chat', 'Gemini | gemini-2.0-flash-lite'], 80%, 3,23925 s, 0,00111466 USD
- Cheapest: ['DeepSeek | deepseek-reasoner', 'Gemini | gemini-2.0-flash-lite', 'Gemini | gemini-2.5-flash-lite'], 85%, 80,9133s, 0,000913194

70%-80% sáv

- Cheapest**:** ['DeepSeek | deepseek-chat', 'Gemini | gemini-2.0-flash-lite', 'Gemini | gemini-2.5-flash-lite'], 70%, 2,0921 s, 0,00006282 USD

60%-70% sáv

- Cheapest: ['ChatGPT | gpt-4.1-nano', 'Gemini | gemini-2.0-flash', 'Gemini | gemini-2.0-flash-lite'], 60%, 1,1061s, 0,000050625 USD

50%-60% sáv

- Cheapest**:** ['Gemini | gemini-2.0-flash', 'Gemini | gemini-2.0-flash-lite', 'Gemini | gemini-2.5-flash-lite'], 55%, 1,06975 s, 0,000048535 USD
- Fastest: ['ChatGPT | gpt-4.1-nano', 'Gemini | gemini-2.0-flash', 'Gemini | gemini-2.5-flash-lite'], 55%, 0,9436s, 0,00005465 USD
- Balanced: ['ChatGPT | gpt-4.1-nano', 'Gemini | gemini-2.0-flash-lite', 'Gemini | gemini-2.5-flash-lite'], 55%, 1,0465s, 0,000049245 USD

3.3.5. A kimeneti eredmények tárolása, kiértékelése

A Batch Runner kimeneti fájljait először külön Excel-fájlokban kezeltem, azonban a könnyebb átláthatóság érdekében egy nagy, az aktuális tesztelés összes eredményét magában foglaló, soklapos Excel dokumentum létrehozása mellett döntöttem (1. számú melléklet).

3.3.6. A teszt bemenetét és részeredményeit tartalmazó Excel tábla lapjai

A részeredményeket tartalmazó Excel-tábla megtalálható a szakdolgozat 1. számú mellékleteként.

- Summary: az eredeti JSON kimenet alapadatait tartalmazza.
- Meta: az eredeti JSON kimenetben található futási környezeti adatok.
- Q&A: a 20 darab kérdés és válasz páros.
- Items: a 20 darab kérdés a Batch Runner-nek átadott formátumban.
- Models and pricing: a 22 db LLM 1 000 000 db és 1 db input és output tokenre vetített költsége tételesen felsorolva.
    - ChatGPT tokenköltség: (OpenAI, 2025)
    - Claude tokenköltség: (Anthropic, 2025.)
    - Gemini tokenköltség: (Google, 2025)
    - DeepSeek tokenköltség: (DeepSeek, 2025.)

- Candidates: az eredeti JSON kimenetben található 22 db LLM 20 db kérdésre adott válasza (440 sor + fejléc) I/O tokenszámmal és futásidővel.
- Base datas: kézi feldolgozás táblázata a helyes/helytelen válaszok jelzésének helye, és a USD-re vetített, lekérdezésenkénti költségek számolása.
- Efficiency: összesítő táblázat (vö.15. ábra) a LLM-ek sikerrátájáról, költségéről és futásidejéről.

	A	B	C	D	E	F	G
1	api_name	display_name	runs_count	success_rate_%	avg_cost_usd	total_cost_usd	total_time_s
2	gpt-5-nano	gpt-5-nano	20	90,0	0,000953	0,019067	383,56
3	gpt-5-mini	gpt-5-mini	20	90,0	0,001508	0,030164	239,94
4	gpt-5	gpt-5	20	90,0	0,010193	0,203860	276,15
5	deepseek-reasoner	deepseek-reasoner	20	85,0	0,000883	0,017662	1618,27
6	gemini-2.5-pro	Gemini 2.5 Pro	20	80,0	0,013922	0,278441	339,13
7	gemini-2.5-flash	Gemini 2.5 Flash	20	80,0	0,020936	0,418716	346,75
8	deepseek-chat	deepseek-chat	20	65,0	0,000033	0,000655	41,84
9	claude-opus-4-1-202508	Claude Opus 4.1	20	70,0	0,010917	0,218340	118,66
10	claude-opus-4-2025051	Claude Opus 4	20	70,0	0,011071	0,221415	86,84
11	claude-sonnet-4-20250	Claude Sonnet 4	20	65,0	0,001963	0,039258	77,14
12	claude-3-7-sonnet-2025	Claude Sonnet 3.7	20	55,0	0,000741	0,014823	26,74
13	claude-sonnet-4-5-2025	Claude Sonnet 4.5	20	55,0	0,001069	0,021378	64,79
14	gemini-2.0-flash	Gemini 2.0 Flash	20	50,0	0,000018	0,000369	15,50
15	gpt-4.1	gpt-4.1	20	50,0	0,000404	0,008076	23,98
16	gemini-2.0-flash-lite	Gemini 2.0 Flash-Lite	20	45,0	0,000013	0,000261	19,12
17	gpt-4o	gpt-4o	20	45,0	0,000445	0,008905	33,29
18	gpt-4o-mini	gpt-4o-mini	20	40,0	0,000030	0,000610	29,49
19	gpt-4.1-mini	gpt-4.1-mini	20	40,0	0,000071	0,001420	18,97
20	claude-3-5-haiku-20241	Claude Haiku 3.5	20	40,0	0,000292	0,005833	33,88
21	gpt-4.1-nano	gpt-4.1-nano	20	35,0	0,000019	0,000383	15,84
22	gemini-2.5-flash-lite	Gemini 2.5 Flash-Lite	20	25,0	0,000017	0,000341	11,73
23	claude-3-haiku-202403(	Claude Haiku 3	20	10,0	0,000075	0,001498	15,90

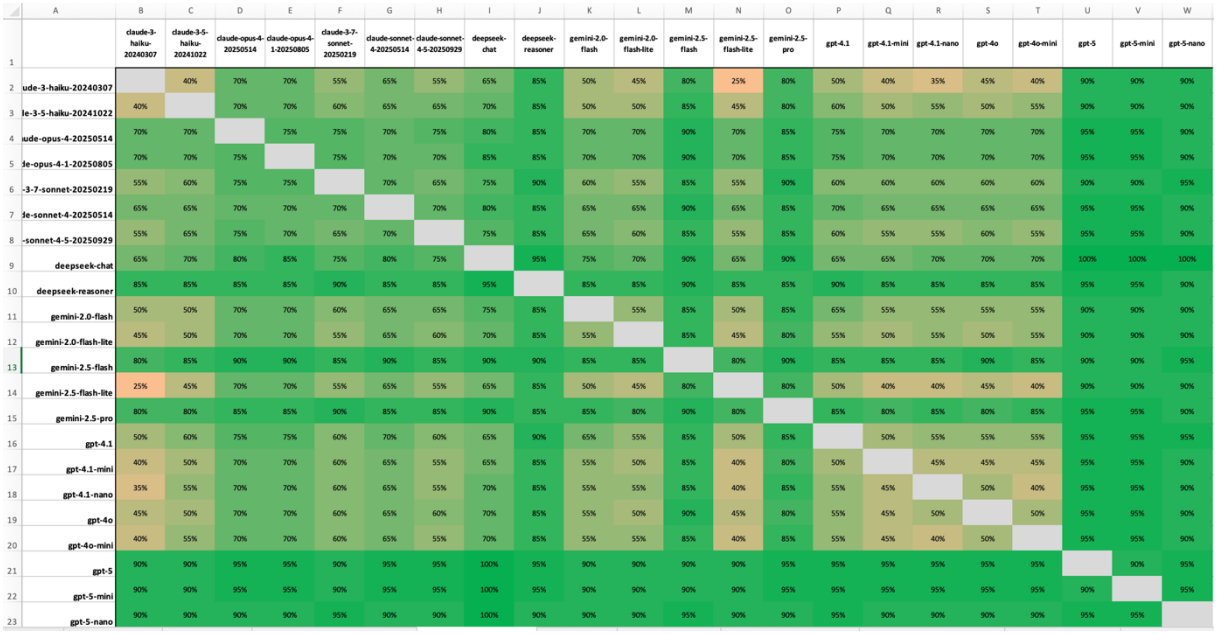
13. ábra: Az Efficiency oldal fő táblázata
Forrás: saját ábra

- Correct answer matrix: vizuális segédmátrix (vö. 16. ábra), a modellek (Y) kérdésekre (X) adott helyes válaszának mintázatáról. A helyes válaszok zöld színnel, a helytelenek piros színnel jelölve.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	MODELL /Q_ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	claude-3-haiku-20240307	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ
3	claude-3-5-haiku-20241022	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ
4	claude-opus-4-20250514	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	HAMIS	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
5	claude-opus-4-1-20250805	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
6	claude-3-7-sonnet-20250219	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	HAMIS	IGAZ
7	claude-sonnet-4-20250514	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
8	claude-sonnet-4-5-20250929	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
9	deepseek-chat	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ
10	deepseek-reasoner	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
11	gemini-2.0-flash	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ
12	gemini-2.0-flash-lite	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	IGAZ	IGAZ	HAMIS	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ
13	gemini-2.5-flash	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	HAMIS	IGAZ
14	gemini-2.5-flash-lite	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ
15	gemini-2.5-pro	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	HAMIS	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
16	gpt-4.1	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ
17	gpt-4.1-mini	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
18	gpt-4.1-nano	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ
19	gpt-4o	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
20	gpt-4o-mini	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	HAMIS	HAMIS	HAMIS	HAMIS	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	HAMIS	IGAZ
21	gpt-5	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ
22	gpt-5-mini	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ
23	gpt-5-nano	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ	HAMIS	IGAZ	IGAZ	IGAZ	IGAZ

14. ábra: Az egyéni modellek helyesválasz-mátrixa
Forrás: saját ábra

- Duo efficiency: vizuális segédmátrix (vö. 17. ábra), a lehetséges LLM párosok elméleti maximális sikerrátájáról, hőmérséklet segédvizualizációval. 0% (sárga)-100% (sötétzöld).



	claude-3-haiku-20240307	claude-3-5-haiku-20241022	claude-opus-4-20250514	claude-opus-4-1-20250805	claude-3-7-sonnet-20250219	claude-sonnet-4-20250514	claude-sonnet-4-5-20250929	deepseek-chat	deepseek-reasoner	gemini-2.0-flash	gemini-2.0-flash-lite	gemini-2.5-flash	gemini-2.5-flash-lite	gemini-2.5-pro	gpt-4.1	gpt-4.1-mini	gpt-4.1-nano	gpt-4o	gpt-4o-mini	gpt-5	gpt-5-mini	gpt-5-nano
ude-3-haiku-20240307		40%	70%	70%	55%	65%	55%	65%	85%	50%	45%	80%	25%	80%	50%	40%	35%	45%	40%	90%	90%	90%
le-3-5-haiku-20241022	40%		70%	70%	60%	65%	65%	70%	85%	50%	50%	85%	45%	80%	60%	50%	55%	50%	55%	90%	90%	90%
ude-opus-4-20250514	70%	70%		75%	75%	70%	75%	80%	85%	70%	70%	90%	70%	85%	75%	70%	70%	70%	70%	95%	95%	90%
de-opus-4-1-20250805	70%	70%	75%		75%	70%	70%	85%	85%	70%	70%	90%	70%	85%	75%	70%	70%	70%	70%	95%	95%	90%
-3-7-sonnet-20250219	55%	60%	75%	75%		70%	65%	75%	90%	60%	55%	85%	55%	90%	60%	60%	60%	60%	60%	90%	90%	95%
de-sonnet-4-20250514	65%	65%	70%	70%	70%		70%	80%	85%	65%	65%	90%	65%	85%	70%	65%	65%	65%	65%	95%	95%	90%
-sonnet-4-5-20250929	55%	65%	75%	70%	65%	70%		75%	85%	65%	60%	85%	55%	85%	60%	55%	55%	60%	55%	95%	95%	90%
deepseek-chat	65%	70%	80%	85%	75%	80%	75%		95%	75%	70%	90%	65%	90%	65%	65%	70%	70%	70%	100%	100%	100%
deepseek-reasoner	85%	85%	85%	85%	90%	85%	85%	95%		85%	85%	90%	85%	85%	90%	85%	85%	85%	85%	95%	95%	90%
gemini-2.0-flash	50%	50%	70%	70%	60%	65%	65%	75%	85%		55%	85%	50%	85%	65%	55%	55%	55%	55%	90%	90%	90%
gemini-2.0-flash-lite	45%	50%	70%	70%	55%	65%	60%	70%	85%	55%		85%	45%	80%	55%	50%	55%	50%	55%	90%	90%	90%
gemini-2.5-flash	80%	85%	90%	90%	85%	90%	85%	90%	90%	85%	85%		80%	90%	85%	85%	85%	90%	85%	90%	90%	95%
gemini-2.5-flash-lite	25%	45%	70%	70%	55%	65%	55%	65%	85%	50%	45%	80%		80%	50%	40%	40%	45%	40%	90%	90%	90%
gemini-2.5-pro	80%	80%	85%	85%	90%	85%	85%	90%	85%	85%	80%	90%	80%		85%	80%	85%	80%	85%	95%	95%	90%
gpt-4.1	50%	60%	75%	75%	60%	70%	60%	65%	90%	65%	55%	85%	50%	85%		50%	55%	55%	55%	95%	95%	95%
gpt-4.1-mini	40%	50%	70%	70%	60%	65%	55%	65%	85%	55%	50%	85%	40%	80%	50%		45%	45%	45%	95%	95%	90%
gpt-4.1-nano	35%	55%	70%	70%	60%	65%	55%	70%	85%	55%	55%	85%	40%	85%	55%	45%		50%	40%	95%	95%	90%
gpt-4o	45%	50%	70%	70%	60%	65%	60%	70%	85%	55%	50%	90%	45%	80%	55%	45%	50%		50%	95%	95%	90%
gpt-4o-mini	40%	55%	70%	70%	60%	65%	55%	70%	85%	55%	55%	85%	40%	85%	55%	45%	40%	50%		95%	95%	90%
gpt-5	90%	90%	95%	95%	90%	95%	95%	100%	95%	90%	90%	90%	90%	95%	95%	95%	95%	95%	95%		90%	95%
gpt-5-mini	90%	90%	95%	95%	90%	95%	95%	100%	95%	90%	90%	90%	90%	95%	95%	95%	95%	95%	95%	90%		95%
gpt-5-nano	90%	90%	90%	90%	95%	90%	90%	100%	90%	90%	90%	95%	90%	90%	95%	90%	90%	90%	90%	95%	95%	

15. ábra: A LLM párosok elméleti sikerrátája vizuálisan szemléltetve
Forrás: saját ábra

- Pareto triades: a Pareto triádok listája
- JUDGE models: a választott JUDGE LLM-ek listája
- JUDGE inputs: az összefűzött JUDGE bemeneti promptok (220 darab), az adott triád maximális futásidejével és összköltségével.
- Judged by GPT 4.1 nano, Judged by GPT 5 nano, Judged by GPT 4.1 mini, Judged by Gemini 2.0 flash lite, Judged by Gemini 2.5 flash lite, Judged by Deepseek Chat, Judged by Claude 3.5 Haiku: a judge modellek JSON kimenetéből konvertált Excel táblázatok, triád+JUDGE összesített futásidővel, összesített költséggel, sikerrátával, triád+JUDGE-onkénti bontásban (összesen 84 triád+JUDGE páros, 1680 db lekérdezéssel).
- Final conclusion: a triád+JUDGE eredményeket és az egyéni LLM eredményeket együttesen tartalmazó táblázat.

Az Excel táblában nyomon követhetőek a tesztelés részfázisainak kimenetei, a hozzájuk tartozó mátrixokkal, és a végleges eredmények áttekintésével. A tesztelést a rendszer sikeresen végrehajtotta, összesen 1 680 db automatizált lekérdezést végrehajtva. Az AiFusion platform **az aktuális célkitűzéseket teljesítette**, fejlesztése a következő szakaszba léphet.

4. Vita

A kismintás tesztelés elsődleges célja az AiFusion platform működésének demonstrációja volt, nem pedig a kollaborációs modellek abszolút teljesítményének igazolása. A tesztelés során azonban olyan tényezők merültek fel, amelyek alapvetően befolyásolják az eredmények minőségi kiértékelését.

4.1. A tesztelés módszertani korlátai

A „4. fejezetben” leírt tesztek során a „bírói” promptok összeállítása még részben manuálisan, Excel-táblában történt. A „batch-runner.py” modul azonban már automatizáltan kezelte ezen összeállított promptok futtatását a bírói körökben. Ez a félig automatizált folyamat igazolta, hogy a platform technikailag képes a többlépcsős (post-hoc) konszolidációra, de rávilágított, hogy a jövőbeli, teljes automatizáláshoz a prompt-generálást is a backendre kell helyezni.

4.2. A "GPT-5" modellek torzító hatása az összehasonlíthatóságra

A platform fejlesztése során igyekeztem az újonnan megjelenő LLM-eket implementálni. Az AiFusion-nel végzett kismintás validációs tesztek futtatása előtt nem sokkal jelentek meg a "GPT-5" gyűjtőnév alatt futó modellek („gpt-5”, „gpt-5-nano”, „gpt-5-mini”) amelyek a vizsgálat során jelentősen torzították a modellek összehasonlíthatóságát. Ahogy az „5. Kismintás tesztelés” során empirikusan megfigyelhető volt, ezen modellek:

1. Jelentősen hosszabb futásidővel dolgoztak;

2. Következetesen pontosabb válaszokat adtak a többi vizsgált modellnél;

3. Nem rendelkeztek állítható „temperature” (kreativitás) paraméterrel.

Feltételezhető (bár a dolgozat keretein belül nem bizonyítható), hogy ezen modellek már eleve egyfajta belső, többlépcsős verifikációs logikát alkalmaznak, ami megmagyarázza a magasabb pontosságot és a hosszabb futásidőt. Mivel ez a működés

alapvetően eltér a többi, paraméterezhető modelltől, az azonos feltételek melletti (költség, idő, minőség) összehasonlítás okafogyottá vált.

4.3. A referencia válaszok érvényességének problémája

A mérést torzító másik kulcstényező magából a tesztadatbázisból fakadt. Az „3.3.4.1." fejezetben részletezettek szerint a futtatások során kiderült, hogy az „ismert" helyes válaszok egy része hibás vagy kétértelmű volt.

Ez a jelenség – bár kezdetben a platform hibájának tűnt – végül az AiFusion koncepciójának egyik váratlan igazolásává vált. A rendszer éppen a kollaboratív modelljei (triád+JUDGE) révén képes volt kimutatni a referenciaadatok anomáliáit, jellemzően a válaszok magas szórásával. Ez rávilágított a rendszer egy lehetséges jövőbeli felhasználási módjára: a platform nemcsak LLM-ek, hanem maguk a tesztadatbázisok validálására is alkalmas lehet.

4.4. A használhatóság és a működőképesség értékelése

A fenti két torzító tényező (a „GPT-5" modellek és a hibás tesztadatok) miatt a jelen dolgozat tudatosan nem tesz kísérletet a modellek minőségi rangsorolására.

Ami viszont a szakdolgozat elsődleges célkitűzése volt és egyértelműen kiértékelhető: **az AiFusion platform rendszerszintű működése igazoltan stabil.**

- A rendszer képes volt nagy tömegű (összesen 1680+440) lekérdezést kezelni fagyás és adatvesztés nélkül.
- Az egyedi LLM-ek API-hibáit a rendszer sikeresen kezelte („graceful fallback") anélkül, hogy a teljes batch futás leállt volna.
- A naplózás és az eredmények JSON-kimenete hibátlanul működött, lehetővé téve az utólagos kiértékelést.
- Az architektúra (backend - frontend - batch-runner) bizonyította, hogy alkalmas LLM-ranking és LLM-kollaborációs kísérletek elvégzésére, és (az

adaptereken keresztül) könnyen bővíthető új modellekkel vagy új kollaborációs metódusokkal.

Az AiFusion platform tehát teljesítette az elsődleges célkitűzést: egy stabil, skálázható és bővíthető kísérleti környezetet biztosít a jövőbeli, tisztább tesztadatokon végzett LLM-kollaborációs mérésekhez.

4.5. Rokonkutatások és a platform fejlesztésének szinergiái

A „4. Vita" fejezet során azonosított kihívások – különösen a megbízhatóság, a modell-összehasonlítás és a referenciaadatok validálásának igénye – nem csak az AiFusion platform sajátjai, hanem a mesterséges intelligencia alkalmazott kutatásának központi kérdései. A szakdolgozatban hivatkozott, különböző szakterületeken végzett magyar kutatások, mint például a *„tűzvédelmi szakértői rendszerek fejlesztése"* (Karsa, 2024) vö. https://miau.my-x.hu/miau/319/tuzmunkakornyezet/chatgpt_tuz-munka-kornyezet--vedelmi_vizsga.docx?vagy a *„jogi rendszerek LLM-alapú tesztelése"* (Pitlik L. , 2025) rávilágítanak arra, hogy a domain-specifikus tudás és a LLM-ek kapcsolata kritikus fontosságú.

Ezek a rokonkutatások és az AiFusion platform között egyértelmű kétirányú együttműködési potenciál rejlik:

Az AiFusion mint tesztelési keretrendszer: Az olyan szakterületek kutatói, mint a jog vagy a tűzvédelem, gyakran egy-egy specifikus modellt (pl. COPILOT, ChatGPT) vizsgálnak. Az AiFusion platform hatékony és tömeges tesztelési (benchmarking) környezetet biztosíthat számukra, lehetővé téve, hogy a saját kérdéskorpuszukat gyorsan lefuttassák az összes integrált LLM-en és azok kollaborációin. Ezzel objektív választ kaphatnak arra, hogy az ő specifikus területükön melyik modell vagy modell-együttműködés nyújtja a legmegbízhatóbb eredményt.

A rokonkutatások mint tudásbázisok: Az AiFusion platform fejlődéséhez – különösen a „JUDGE" modell finomhangolásához – kritikus fontosságú a jó minőségű, szakterületi tudásbázisok (tesztkérdések és ismert helyes válaszok) beszerzése. Az ilyen együttműködések révén a platform specializált, validált adathalmazokhoz juthat, amelyek elengedhetetlenek az AiFusion domain-specifikus ágainak (pl. egy jövőbeli "AiFusion-

Legal" vagy "AiFusion-Engineering" modul) kifejlesztéséhez és pontosságának növeléséhez.

Ez a kooperáció tehát felgyorsíthatja a szakterületi kutatásokat azáltal, hogy egy kész, automatizált tesztelési platformot ad a kezükbe, miközben az AiFusion rendszert látja el azokkal a nélkülözhetetlen adatokkal, amelyek a platform intelligenciájának fejlődését biztosítják.

5. Összegzés, összefoglalás

A rendszer tesztelését a hibás tesztadatok által okozott bonyodalmak ellenére sikeresnek tekintem. Az AiFusion platform képes stabilan kezelni az egyedi, frontenden végrehajtott LLM műveleteket és képes a kötegelt, ezres nagyságrendű kimenettel bíró lekérdezések futtatására is. Ezzel gyakorlatilag a kezdeti célomat elértem: segítséget biztosítani abban, hogy nagy tömegű LLM hívásokat automatizáltan hajtsunk végre, több százezres vagy milliós rekordszámú adatbázisok építéséhez, amelyeket kutatási célokra lehet hasznosítani, mintázataikból következtetéseket lehet levonni.

A hibás tesztadatokat először kudarcként fogtam fel, azonban mivel utólag látom, miként jelezte a rendszer az anomáliát (az inkonzisztens válaszadással), tulajdonképpen még pozitívan is értékelem a tesztadatok hibáját. Ezzel a rendszer hibatűrésének és hibafelismerésének egyik módszere mutatkozott meg, amit egy későbbi továbbfejlesztés során figyelembe kell venni.

A rendszer üzemeltetését és fejlesztését rövidtávon meglévő cégeim valamelyikében kívánom folytatni. Amennyiben a fejlesztés során a platform olyan mérföldkőhöz érkezik, amely után a portfóliótisztítás mellett döntenék, az AiFusion rendszer üzemeltetésére és a tudásbázis licencelésére külön céget jegyeznék be. A cégbejegyzés során figyelembe kell venni a területi hatályosságot is, vagyis a régiónként eltérő szabadalmi és licencpolitikát. Emiatt elképzelhető, hogy külön cég bejegyzése szükséges az Európai Unió és az Amerikai Egyesült Államok területén is.

6. Jegyzékek, kiegészítések, megjegyzések

A záró fejezet a dolgozatot kísérő háttérelemeket - az irodalomjegyzéket, az elemző táblázatokat, az ábra- és táblázatjegyzéket, a rövidítések és a fájlmellékletek jegyzékét - foglalja össze egységes keretben. Ezek a kiegészítések biztosítják a hivatkozások átláthatóságát, a kutatás reprodukálhatóságát és a technikai megvalósítás részletes nyomon követhetőségét.

6.1. Irodalomjegyzék

Schiller, C. A. (2024. 03 13). *[12.] The Human Factor in Detecting Errors of Large Language Models: A Systematic Literature Review and Future Research Directions*. Forrás: arXiv: https://arxiv.org/abs/2403.09743

Caliskan, A. B. (2017). [40.] Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334), 183–186. Forrás: https://www.science.org/doi/10.1126/science.aal4230

Dell'Acqua, F., McFowland III, E., Mollick, E., Lifshitz-Assaf, H., Kellogg, K. C., Rajendran, S., . . . Lakhani, K. R. (2023). *[22.] Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality*. Forrás: https://www.hbs.edu/faculty/Pages/item.aspx?num=64700

Sennrich, R., Haddow, B., & Birch, A. (2016). [35.] Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (old.: 1715–1725). Berlin, Germany: Association for Computational Linguistics. Forrás: https://aclanthology.org/P16-1162/ aclanthology.org

DeepSeek. (2025.). *[48.] DeepSeek pricing*. Letöltés dátuma: 2025. 11 01, forrás: https://api-docs.deepseek.com/quick_start/pricing

Chen, Z., Li, J., Chen, P., Li, Z., Sun, K., Luo, Y., . . . Yu, P. S. (dátum nélk.). *[39.] Harnessing Multiple Large Language Models: A Survey on LLM Ensemble.* arXiv. Forrás: https://arxiv.org/html/2502.18036v1#:~:text=consider%20that%2C%20for%20each%20task,field%20of%20LLM%20Ensemble%20explores

Shelar, M. (2025). *[14.] Understanding OpenAi's Temperature Parameter.* Letöltés dátuma: 2025. 10 27, forrás: DEV Community:

https://dev.to/mrunmaylangdb/understanding-openais-temperature-parameter-2pj6

Dietterich, T. G. (2000). [24.] Ensemble methods in machine learning. In J. Kittler, & F. (. Roli, *Multiple Classifier Systems* (old.: 1-15). Springer. Forrás: https://doi.org/10.1007/3-540-45014-9_1

Nirdiamant. (2025. 03 04). *[44.] LLM hallucinations explained.* Forrás: Medium: https://medium.com/@nirdiamant21/llm-hallucinations-explained-8c76cdd82532

Google. (2025). *[47.] Gemini Developer API Pricing _ Gemini API _ Google AI for Developers*. Letöltés dátuma: 2025. 11 01, forrás: https://ai.google.dev/gemini-api/docs/pricing

Noy, S., & Zhang, W. (2023). [19.] Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 187–192. Forrás: Science: https://www.science.org/doi/10.1126/science.adh2586?

Guo, Y., Guo, M., Su, J., Yang, Z., Zhu, M., Li, H., . . . Liu, S. S. (dátum nélk.). *[41.] Bias in Large Language Models: Origin, Evaluation, and Mitigation.* arXiv. Forrás: https://arxiv.org/abs/2411.10915

Turtogtokh, S., Pitlik, L., & Pitlik, L. J. (2025.). *[13.] Abstract – Objective evaluation of performances in case of students based on similarity analyses and Moodle-logs*. Forrás: MIAU: https://miau.my-x.hu/miau/319/performances/Abstract_Shagai_Turtogtokh.docx

Surowiecki, J. (2004.). *[23.] The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations.* New York: Doubleday.

Sybrandwildeboer. (2025. 02 10). *[32.] How Large Language Models Predict the Next Word (and Why That's Powerful)*. Forrás: Medium: https://medium.com/@sybrandwildeboer/how-large-language-models-predict-the-next-word-and-why-thats-powerful-b1ac78995e74

[27.] Magyar, uniós és nemzetközi védjegy díjak. (2025). Letöltés dátuma: 2025. 10 29, forrás: https://keserubarna.hu/vedjegy_arak.html

[50.] 7 Math Riddles Only the Smartest Can Get Right. (2025. 10 25). Forrás: https://www.rd.com/list/math-riddles/

[51.] 15 Challenging Logic Puzzles and Their Answers. (2025. 10 25). Forrás: https://loquiz.com/2024/11/12/15-challenging-logic-puzzles-and-their-answers/

[52.] Difficult Mathematical Puzzles - 5. (2025. 10 25). Forrás: https://www.hitbullseye.com/puzzle/hard-math-puzzles.php

Anthropic. (2025.). *[46.] Claude pricing*. Letöltés dátuma: 2025. 11 01, forrás: https://docs.claude.com/en/docs/about-claude/pricing

AAAI. (2009). *[6.] AITopics*. Letöltés dátuma: 2025. október 26, forrás: https://aitopics.org/

APA. (2024. 02). *[28.] APA Style and Grammar Guidelines*. Letöltés dátuma: 2025. 11 01, forrás: https://apastyle.apa.org/style-grammar-guidelines: https://apastyle.apa.org/style-grammar-guidelines

Bang, Y., Cahyawijaya, S., Lee, N., Dai, W., Su, D., Wilie, B., . . . Fung, P. (2023). [26.] A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. *Association for Computational Linguistics (ACL)*, (old.: 675-718). Nusa Dua, Bali. Forrás: https://aclanthology.org/2023.ijcnlp-main.45/

Breiman, L. (1996). [4.] Bagging predictors. *Machine Learning*, old.: 123–140. Forrás: https://link.springer.com/article/10.1007/BF00058655.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... (2020). [34.] Language models are few-shot learners. *Advances in Neural Information Processing Systems 33 (NeurIPS 2020).* 2020: Curran Associates, Inc. Forrás: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

ELTE TTK. (2008). *[11.] Programtervező matematikus szak – választható tárgyak*. Letöltés dátuma: 2025. október 26, forrás: http://formed2008.inf.elte.hu/targyak/

European Parliament. (2025). *[43.] EU - AI Act*. Letöltés dátuma: 2025. 11 01, forrás: European Parliament: https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

Fejes, Z., Pitlik, L., Rikk, J., Szűcs, D., & Túri, P. (2024/1-2). [15.] E-volution a védelem-egészségügyben. *Honvédorvos*, 61-69.

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2019). *[37.] The Curious Case of Neural Text Degeneration.* arXiv. Forrás: https://arxiv.org/abs/1904.09751

Hugging Face. (2025. 10 25). *[36.] Generation strategies*. Forrás: Transformers documentation (Hugging Face): https://huggingface.co/docs/transformers/en/generation_strategies

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., . . . Fung, P. (2023). [25.] Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, old.: 1-38. Forrás: https://doi.org/10.1145/3571730

Karsa, R. (2024). [16.] Tűzvédelmi szakértői rendszer létrehozása nagy nyelvi modellek segítségével. *Rendvédelem Tudományos Folyóirat*.

Kerepesi, A. (2013). *[53.] Gyártástervezés és termelésirányítás folyamatainak optimalizálása az Országos Villamostávvezeték Zrt.-nél.* Budapest: Kodolányi János Egyetem (MIAU portál).

Kollár, Z. (2011). *[8.] Intelligens számítási módszerek alkalmazása logikai játékokban*. Forrás: MIAU: https://miau.my-x.hu/miau/160/machine_learning.pdf

Kodolányi János Egyetem. (2021. 03 17). *[7.] A KJE Egységes Szakdolgozati Szabályzata.* Letöltés dátuma: 2025. 10 31, forrás: A KJE Egységes Szakdolgozati Szabályzata: https://www.kodolanyi.hu/konyvtar/images/tartalom/File/kje_egyseges_szakdolgozati_szabalyzat_2021marc17_hatalyos.pdf

Kovács, B., & Pitlik, L. (2025). [2.] AIFUSION: A COLLABORATIVE AI TESTING PLATFORM AND THE EMERGENCE OF AI SELF-CRITICISM. In *5th AHI EVRAN International Conference on Scientific Research: Full Texts Book* (old.: 444-447). Dubai: IKSAD Institute. Forrás: https://www.ahievranconference.org/_files/ugd/614b1f_9e6f77dbf53a49dfb078554b387d1c9c.pdf

Kuriakose, A. A. (2024. 05 14). *[42.] The Role of Human Oversight in LLMOps*. Forrás: Algomox Blog: https://www.algomox.com/resources/blog/what_is_the_role_of_human_oversight_in_llmops/

MTA. (2025). *[49.] A magyar helyesírás szabályai, 12. kiadás*. Letöltés dátuma: 2025. 11 02, forrás: https://helyesiras.mta.hu/helyesiras/default/akh12

Magyar Tudomány. (2005. június/6). [9.] Megemlékezés. Dr. Vajda Ferenc. (1934-2004). *Magyar Tudomány*, 772.

Microsoft. (2025.. 05. 29.). *[31.] Understand tokens*. Letöltés dátuma: 2025. 10 27, forrás: Microsoft Learn (/.NET): https://learn.microsoft.com/en-us/dotnet/ai/conceptual/understanding-tokens

MIAÚ. (2014). *[10.] MIAU*. Letöltés dátuma: 2025. 11 01, forrás: https://miau.my-x.hu/miau2009/index.php3?x=miau128&where[indexkod]=miau03

MIAÚ. (2025). *[17.] MIAU - Magyar Internetes Agrár/Alkalmazott Informatikai Újság*. Letöltés dátuma: 2025. október 26, forrás: https://miau.my-x.hu/miau2009/index.php3

OpenAI. (2025). *[45.] ChatGPT Pricing*. Letöltés dátuma: 2025. 11 01, forrás: https://openai.com/api/pricing/

Pennington, J., Socher, R., & Manning, C. D. (2014). [29.] GloVe: Global vectors for word representation. (old.: 1532–1543). Doha, Qatar: Association for Computational Linguistics. Forrás: https://doi.org/10.3115/v1/D14-1162

Pitlik, L. (2011). *[5.] MY-X: CONT-ROLLING-STONES*. Letöltés dátuma: 2025. október 26, forrás: MIAU: https://miau.my-x.hu/myx-free/

Pitlik, L. (2014). *[1.] My-X Team, an innovative idea-breeding farm.* Innoreg Regional Innovation Agency of Central-Hungary Khe. Forrás: https://www.researchgate.net/profile/Laszlo-Pitlik

Pitlik, L. (2025). [20.] A kontinentális jogrendszer elemi szintjeinek tesztelése COPILOT támogatással. *MTA Gazdálkodástudományi Bizottság.* Veszprém. Forrás: https://miau.my-x.hu/miau/325/mta_gb_tm_copilot_pitlik.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *[30.] Attention Is All You Need.* arXiv. Forrás: https://arxiv.org/abs/1706.03762

Wikipedia. (2025. 10 25). *[18.] Pareto-hatékonyság*. Forrás: Wikipedia: https://hu.wikipedia.org/wiki/Pareto-hat%C3%A9konys%C3%A1g

Wikipedia. (2025. 10 25). *[21.] Pareto front*. Forrás: Wikipedia: https://en.wikipedia.org/wiki/Pareto_front

Wolters Kluwer. (2001). *[3.] Net Jogtár 2001. évi CVIII. törvény*. Letöltés dátuma: 2025. 10 30, forrás: https://net.jogtar.hu/jogszabaly?docid=A0100108.TV&searchUrl=/gyorskereso?keyword%3D2001.%2520%25C3%25A9vi%2520CVIII.%2520t%25C3%25B6rv%25C3%25A9ny%2520%28Eker.%2520tv.%29

6.2. A felhasznált irodalom attribútumainak elemzése

Ebben a fejezetben a felhasznált irodalmak **jellemzői** alapján végezzük az elemzést. A forrásokat **négy attribútum** (jellemző) szerint osztályozzuk, ahol minden attribútum **két lehetséges állapotot** vehet fel.

Az attribútumok tételes felsorolása:

1. **KJE (Kodolányi János Egyetem)-Van / KJE-Nincs**: azon felhasznált irodalmak, amelyek személyi, vagy intézményi értelemben KJE kötődéssel rendelkeznek, a KJE-Van kategóriába esnek. A KJE kötődéssel nem rendelkező források a KJE-Nincs kategóriába sorolódnak. Személyi kapcsolatnak veszem azt, ha olyan illető szerepel a forrásanyag szerzői között, aki a KJE-en tanult, vagy tanított bármikor. Intézményi kapcsolat a KJE-el a forrás kapcsán igazolhatóan kooperáló intézmény, vagy a KJE jogelődje.
2. **Új / Régi**: A 2014-es és korábbi anyagok a **Régi** kategóriába, a 2014 utáni források az **Új** kategóriába sorolódnak.
3. **Angol / Magyar**: A forrás eredeti nyelve szerinti besorolás. Amennyiben a forrás több nyelven is elérhető, pl. a weboldal nyelve a böngésző nyelvének megfelelően lehet angol vagy magyar is, a nyelvi besorolás szabadon választható.
4. **Szakcikk/Weblap**: Szakcikknek tekintem az ISBN (International Standard Book Number) számmal ellátott kiadványban megjelent forrásokat, valamint a szerzővel ellátott, konferenciaközlönynek formailag megfelelő, Interneten elérhető és önmagában letölthető anyagokat. Weboldalnak az Interneten elérhető, szerzőmegjelöléssel ellátott olyan oldalakat tekintem, amelyek teljes megismeréséhez elegendő a böngészőben megnyitni az adott weboldalt.

A **3. táblázat** a felhasznált irodalmak „újszerűség / KJE-kötődés / nyelv / minőség” szerinti besorolását mutatja be. A táblázatban szereplő sorszámok és a hivatkozások közötti kapcsolat a **6.3. fejezetben** („A 3. sz. táblázatban szereplő sorszámok hivatkozás-kapcsolata”) található. A források **kategóriák szerinti darabszám- és százalékos megoszlásá**t a **4. táblázat** szemlélteti. Az **egyes kategóriák** (régi / új; KJE-van / KJE-nincs; angol / magyar; szakcikk / weblap) **átfedéseit** az **5. táblázat** összegzi, míg a **6. táblázat** az egyes attribútumokhoz tartozó **összesített előfordulásszámokat** mutatja be. A táblázatokban alkalmazott színezés kizárólag a tájékozódást segíti.

	Angol / Szakcikk	**Angol / Weblap**	**Magyar / Szakcikk**	**Magyar / Weblap**
Régi / KJE-Van	1.	5.	8., 53.	10.
Régi / KJE-Nincs	4., 23., 24., 29.,	6.	9.	3., 11.
Új / KJE-Van	2.	13.	15., 20.	7., 17.
Új / KJE-Nincs	12., 19., 22., 25., 26., 30., 34., 35., 37., 39., 40., 41.	14., 21., 28., 31., 32., 36., 42., 43., 44., 45., 46., 47., 48., 50., 51., 52.	16.	18., 27., 49.

3. táblázat - A felhasznált irodalmak csoportosítása
Forrás: saját táblázat

Össz. darabszám: **51**	**Angol / Szakcikk**	**Angol / Weblap**	**Magyar / Szakcikk**	**Magyar / Weblap**
Régi / KJE-Van	1 db **1,92%**	1 db **1,92%**	2 db **3,85%**	1 db **1,92%**
Régi / KJE-Nincs	4db **7,69%**	1 db **1,92%**	1 db **1,92%**	2 db **3,85%**
Új / KJE-Van	1 db **1,92%**	1 db **1,92%**	2 db **3,85%**	2 db **3,85%**
Új / KJE-Nincs	12db **23,08%**	16db **30,77%**	1 db **1,92%**	3db **5,77%**

4. táblázat - A felhasznált irodalmak darabszám és százalék szerinti megoszlása
Forrás: saját táblázat

	KJE-van	**KJE-nincs**	**Régi**	**Új**	**Angol**	**Magyar**	**Szakcikk**	**Weblap**
KJE-van	-	-	5	6	4	7	6	5
KJE-nincs	-	-	8	32	33	7	18	22
Régi	5	8	-	-	7	6	8	5
Új	6	32	-	-	30	8	16	22
Angol	4	33	7	30	-	-	18	19
Magyar	7	7	6	8	-	-	6	8
Szakcikk	6	18	8	16	18	6	-	-
Weblap	5	22	5	22	19	8	-	-

5. táblázat - A felhasznált irodalmak attribútummátrixa (db)
Forrás: saját táblázat

Felhasznált irodalmak attribútuma	Darabszám
KJE-van:	11 db
KJE-nincs:	40 db
Új:	39 db
Régi:	12 db
Angol:	37 db
Magyar:	14 db
Weblap:	27 db
Szakcikk:	25 db

6. táblázat - Az egyes attribútumokhoz tartozó összesített előfordulásszámok
Forrás: saját táblázat

6.3. A 3.sz. táblázatban szereplő sorszámok hivatkozás-kapcsolata

A 33. és 38. sorszámú tétel utólag törlésre került, de a sorszám-forrás kapcsolatok szoros összefüggése miatt a sorszámozást nem módosítottam.

1. (Pitlik L. , [1.] My-X Team, an innovative idea-breeding farm, 2014)
2. (Kovács & Pitlik, 2025)
3. (Wolters Kluwer, 2001)
4. (Breiman, 1996)
5. (Pitlik L. , [5.] MY-X: CONT-ROLLING-STONES, 2011)
6. (AAAI, 2009)
7. (Kodolányi János Egyetem, 2021)
8. (Kollár, 2011)
9. (Magyar Tudomány, 2005)
10. (MIAÚ, 2014)
11. (ELTE TTK, 2008)
12. (Schiller, 2024)

13. (Turtogtokh, Pitlik, & Pitlik, 2025.)
14. (Shelar M. , 2025)
15. (Fejes, Pitlik, Rikk, Szűcs, & Túri, 2024/1-2)
16. (Karsa, 2024)
17. (MIAÚ, 2025)
18. (Wikipedia, [18.] Pareto-hatékonyság, 2025)
19. (Noy & Zhang, 2023)
20. (Pitlik L. , 2025)
21. (Wikipedia, [21.] Pareto front, 2025)
22. (Dell'Acqua, és mtsai., 2023)
23. (Surowiecki, 2004.)
24. (Dietterich T. G., 2000)
25. (Ji Z. , és mtsai., [25.] Survey of Hallucination in Natural Language Generation, 2023)
26. (Bang, és mtsai., 2023)
27. ([27.] Magyar, uniós és nemzetközi védjegy díjak, 2025)
28. (APA, 2024)
29. (Pennington, Socher, & Manning, 2014)
30. (Vaswani, és mtsai., 2017)
31. (Microsoft, 2025.)
32. (Sybrandwildeboer, 2025)
33. -
34. (Brown, és mtsai., [34.] Language models are few-shot learners)
35. (Sennrich, Haddow, & Birch, 2016)
36. (Hugging Face, 2025)
37. (Holtzman, Buys, Du, Forbes, & Choi, 2019)
38. -
39. (Chen, és mtsai.)
40. (Caliskan, 2017)
41. (Guo, és mtsai.)
42. (Kuriakose, 2024)
43. (European Parliament, 2025)
44. (Nirdiamant, 2025)

45. (OpenAI, 2025)
46. (Anthropic, 2025.)
47. (Google, 2025)
48. (DeepSeek, 2025.)
49. (MTA, 2025)
50. ([50.] 7 Math Riddles Only the Smartest Can Get Right, 2025)
51. ([51.] 15 Challenging Logic Puzzles and Their Answers, 2025)
52. ([52.] Difficult Mathematical Puzzles - 5, 2025)
53. (Kerepesi, [53.] Gyártástervezés és termelésirányítás folyamatainak optimalizálása az Országos Villamostávvezeték Zrt.-nél, 2013)

6.4. Ábrajegyzék

6.5. Táblázatjegyzék

6.6. Rövidítésjegyzék

1. **API** (Application Programming Interface) – Alkalmazásprogramozási felület
2. **APA** (American Psychological Association) – Amerikai Pszichológiai Társaság
3. **AWS** (Amazon Web Services) – Amazon Web Szolgáltatások
4. **AAAI** (Association for the Advancement of Artificial Intelligence) – Mesterséges Intelligencia Fejlesztéséért Egyesület
5. **BPE** (Byte-Pair Encoding) – Bájtpár kódolás
6. **CI/CD** (Continuous Integration / Continuous Deployment) – Folyamatos integráció és folyamatos szállítás (vagy telepítés)
7. **CPU** (Central Processing Unit) – központi feldolgozó egység
8. **CSV** (Comma-Separated Values) – Vesszővel tagolt értékek
9. **EMI** (electromagnetic interference) – elektromágneses interferencia
10. **EOS** (End of Sequence) – Szekvencia vége
11. **Flask** – Python web keretrendszer
12. **Git** – Elosztott verziókezelő rendszer
13. **GPU** (Graphics Processing Unit) – grafikus feldolgozó egység
14. **GUI** (Graphical User Interface) – Grafikus felhasználói felület
15. **HTTP** (Hypertext Transfer Protocol) - Hipertext átviteli protokoll
16. **HTTPS** (Hypertext Transfer Protocol Secure) - Biztonságos hipertext átviteli protokoll
17. **HUF** (Hungarian forint) – Magyar forint
18. **ISBN** (International Standard Book Number)
19. **JSON** (JavaScript Object Notation) – JavaScript objektumjelölés
20. **JWT** (JSON Web Token) – JSON web token
21. **KJE** (Kodolányi János Egyetem)
22. **LLM** (Large Language Model) – Nagy nyelvi modell

23. **MI** (Mesterséges Intelligencia)
24. **MIAÚ** (Magyar Internetes Agrár/Alkalmazott Informatikai Újság)
25. **MVP** (Minimum Viable Product) – Minimálisan életképes termék
26. **NoSQL** (Not only SQL) – Nem csak SQL (adatbázis típus)
27. **OS** (Operating System) – Operációs rendszer
28. **PII** (Personally Identifiable Information) – Személyazonosításra alkalmas információ
29. **React** – JavaScript programkönyvtár felhasználói felületek építéséhez
30. **SDK** (Software development kit) - szoftverfejlesztő készlet
31. **SDLC** (Software Development Lifecycle) – Szoftverfejlesztési életciklus
32. **SQL** (Structured Query Language) – Strukturált lekérdezőnyelv
33. **TEÁOR** - Tevékenységek Egységes Ágazati Osztályozási Rendszere
34. **TLS** (Transport Layer Security) – Szállítási réteg biztonsága
35. **USD** (United States Dollar) – Amerikai dollár
36. **UX** (User Experience) – Felhasználói élmény
37. **VM** (virtual machine) – virtuális gép
38. **VRM** (Voltage Regulator Module) - feszültségszabályozó modul

6.7. Fájlmellékletek jegyzéke

1. Kismintás validációs teszt.xlsx (473 KB)