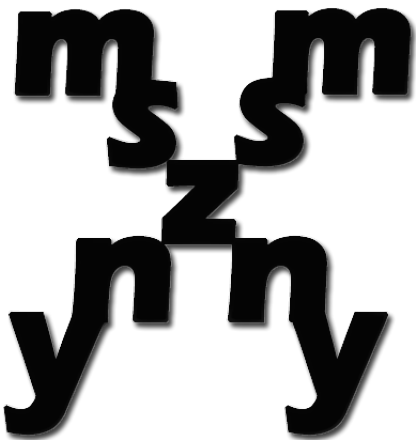
XXII. Magyar Számítógépes Nyelvészeti Konferencia



Szerkesztette:
Berend Gábor
Gosztolya Gábor
Vincze Veronika

Szeged, 2026. január 29-30.

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

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

ISBN: 978-963-688-100-9

Szeged, 2026. február

Az MSZNY 2026 konferencia szervezője:
HUN-REN–SZTE Mesterséges Intelligencia Kutatócsoport

Támogatók:
Neumann János Számítógéptudományi Társaság
SZTE Informatikai Intézet

Előszó

2026. január 29–30-án immáron huszonkettedik alkalommal kerül sor a Magyar Számítógépes Nyelvészeti Konferencia megrendezésére. A konferencia fő célkitűzése a kezdetek óta állandó: lehetőséget biztosítani a nyelv- és beszédtechnológia területén végzett kutatások eredményeinek ismertetésére és megvitatására, ezen felül pedig a különféle hallgatói projektek, illetve ipari alkalmazások bemutatására.

Az idei évben a 18 beküldött cikkből gondos mérlegelést követően 15 cikk került elfogadásra, melyek témája a nyelv- és beszédtechnológia számos szakterületét lefedi a legújabb nyelvi modellek bemutatásától kezdve a beszédtechnológia eredményein keresztül egészen a magyarra adaptált nagy multimodális modellek erőforráshatékony létrehozásáig.

A konferencia programját két plenáris előadás is gazdagítja, amelyeket Vincze Veronika és Török Balázs fog megtartani. Vincze Veronika meghívásának apropóját az adta, hogy a közelmúltban kihirdetett, HTE Top 50 Nők a Mesterséges Intelligencia Területén elnevezésű díj 1. helyezettje lett. Előadásának címe *Szólj, s ki vagy, elmondom - A nyelvhasználati jellegzetességek szerepe osztályozási feladatokban.* A konferencia második napján Török Balázs plenáris előadását hallgathatjuk meg a MOZAIK Kiadó képviseletében *Nagy nyelvi modellek alkalmazása számonkérésben és digitális tananyagfejlesztésben* címmel.

A korábbi szokásoknak megfelelően díjazzuk a konferencia legjobb cikkét, mely a legjelentősebb eredményekkel járul hozzá a magyarországi nyelv- és beszédtechnológiai kutatásokhoz. Ezen felül immár nyolcadik alkalommal osztjuk ki a legjobb bírálónak járó díját, amellyel a bírálók fáradságos, ugyanakkor nélkülözhetetlen munkáját kívánjuk elismerni.

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

Tartalomjegyzék

Ágens-alapú chatbot architektúra valós idejű vezetői Big Data lekérdezésekhez

Vándor Péter[1] és Csáki Csaba[1]

[1] Budapesti Corvinus Egyetem, Fővám tér 8,
H-1093 Budapest, Magyarország
peter.vandor@stud.uni-corvinus.hu
csaki.csaba@uni-corvinus.hu

Kivonat: A vállalatok, szervezetek működése során a különböző területek – értékesítés, marketing, ügyfélszolgálat, ellátási láncok, IoT-eszközök, felhasználói interakciók – hatalmas mennyiségű adatot generálnak folyamatosan. Az ennek eredményeként létrejövő, sokszor több milliárd rekordot tartalmazó Big Data adattárházak, OLAP kockák, dashboardok, elemzések nehezen áttekinthető és sokszor csak bonyolultan lekérdezhető környezetet teremtenek a vezetők számára, akiknek kulcsfontosságú az azonnali, pontos, valós idejű információ a stratégiai döntések meghozatalához. A nagy nyelvi modellek (LLM) megjelenésével lehetővé vált a természetes nyelvi lekérdezés, de a pontos adatok szolgáltatása és a hallucinációk kiküszöbölése továbbra is kihívást jelent. Erre nyújt megoldást az intelligens ágensre épülő, folyamat-vezérelt chatbot fejlesztés, melynek célja, hogy a vezetők természetes nyelven, technikai nehézségektől mentesen kérdezhessenek és megbízható információkat kapjanak. A bot saját MCP (Model Context Protocol) eszközök biztosításával és széleskörű paraméterezésével lehetővé teszi a valós idejű, részletes adat lekérdezést, grafikon rajzolást és adatelemzést. A komplex lekérdezés-paraméterezés egy kontrollált háttérfolyamat részeként valósul meg. E folyamat a végén egy könnyen értelmezhető, mondatokba foglalt válasszal szolgál. Az információszerzés során az ágens önállóan paraméterezi az adattárház lekérdezéseket, szükség esetén visszakérdez vagy több lekérdezést is futtat. Ez a módszer jelentősen csökkenti az adat lekérdezéshez és értelmezéshez szükséges speciális szakértelmet és elősegíti az adat-vezérelt működést.

Kulcsszavak: LLM, ágens, chatbot, MCP, RAG, Big Data

1 Bevezetés

A nagy nyelvi modellek (LLM) térnyerését követően a mesterséges intelligencia (MI) technológiai fejlesztések egy újabb hulláma is elindult, terjedni kezdtek az intelligens

ágensek és az ezekre épülő megoldások. Habár az OpenAI 2025-öt kikiáltotta az ágensek évének (Altman, 2025), ennek ellenére a technológia üzleti adoptálása lassan halad. Egy szervezeti környezetben kifejezetten erős szempont a pontosság az ágens rendszerek bevezetésekor, mivel a nem megfelelően szabályozott, pontatlanul kontrollált ágensek üzletkritikus hibákat is elkövethetnek.

Az MI ágensek olyan rendszerek, melyek bizonyos fokú autonómiával rendelkeznek, intelligens döntéseket hoznak a rendelkezésre álló adatok alapján egy előre meghatározott cél elérése érdekében és az adott környezettel proaktív módon interakcióba lépnek (Deng et al., 2024).

A legtöbb cég működése során rengeteg adat keletkezik folyamatosan, melyek feldolgozása, gyors és pontos lekérdezhetősége kritikus a felsővezetői stratégiai döntéshozatal szempontjából. A méret, keletkezési sebesség és eltérő formátumok miatt az esetek nagy részében speciális eszközökkel kezelhető Big Data adattárházakról beszélhetünk. Az információk lekérdezését követően elemzés, dashboard, riport vagy más típusú kimenet is készülhet az adatok átlátható összegzésére és a jelentéstartalom megvilágítására. A nyelvi modellek lehetővé teszik egy újfajta interfész kialakítását, amellyel természetes nyelven megfogalmazott kérések is kiszolgálhatóak, ezáltal ledöntve a gyakran adatelemzőkre szabott, nem intuitív kezelőfelületek által jelentett akadályokat a vezetői pozíciókat betöltő személyek számára.

Egy ilyen új típusú felület esetében azonban kritikus a szolgáltatott adatok és válaszok minősége. Az LLM által közvetlenül, az adatforrás részletes ismerete nélkül írt függvények szuboptimális lekérdezési sebességet okoznának. A vállalatok feltehetően már rendelkeznek az adatbázis lekérdezésére szolgáló függvényekkel, kódokkal. Nem ajánlott a nyelvi modellre bízni a lekérdezések megírását a beérkező kérésekre, még az egyre fejlődő kódolási képességeket figyelembe véve sem. Az LLM-alapú modern MI ágensek számára ezeket az interakciókat az elérhető eszközök teszik lehetővé, melyek API-k vagy függvények közvetlen meghívásával kibővítik a lehetőségeket: legyen szó valós idejű keresésről, adatlekérdezésről, számítási módszerekről, vagy bármilyen a feladat megoldását támogató, az ágensnek új adatokat szolgáltató kódról vagy külső alkalmazásról (Sapkota et al., 2025). Mivel az üzleti szférában a megbízhatóság egyértelmű elvárás, ezért jelen kutatás fő célját a pontos válaszok biztosítása jelenti. További cél a tisztán LLM-re alapozó és általános ágens megoldásokhoz képest a hallucinációk elkerülése a valós idejű Big Data lekérdezések területén. Javaslatunk középpontjában az integrált eszközhasználat jelenik meg, mint célra vezető út: a már meglévő szervezeti tudást egy LLM ágens által felhasználható eszközzé konvertálni, ezáltal megőrizve a korábbi determinisztikus folyamatok biztonságát. Lehetővé válik a felhalmozott tudásbázis természetes nyelven történő lekérdezése és a válaszok is így jelenhetnek meg, ezáltal téve közvetlenül is elérhetőbbé az összetett adatbázisok tartalmát a nem-technikai vezetők számára is.

2 Módszertan

Az üzleti folyamatok során felmerülhetnek olyan kérdések, amelyek megoldásához domain-specifikus tudásra van szükség. Minden eshetőséget szinte lehetetlen az

utasítások közt megfogalmazni, a munkavállalók fejében létező tudást kell átadni az ágensnek ilyen szituációkban. A felhasználói beavatkozás az ágensek működésébe az eddigi munkákban ritkán vizsgált terület (Händler, 2023). Ez egyben a valós használat során bővülő memória és tanulási mechanizmus bevezetését jelenti. A tapasztalatokból tanulásra képes ágensek kiváló eredményeket demonstrálnak a szakirodalomban az egyszerű ágensekhez képest a többlépéses döntéshozatali feladatokban (Zhao et al., 2023). Az érvelés és akció összekapcsolásával a modell a környezetből szerzett információkra reflektálva képes terveket készíteni és döntéseket hozni. Egy ezt teljesíteni képes architektúra és folyamat megvalósításához számos meglévő technikai megoldásra építettünk, ezeket tekintjük át ebben a fejezetben.

A vállalati tudásbázis természetes nyelven lekérdezéséhez egy általános ágens architektúra megtervezése és fejlesztése képezte a kutatási projekt magját. A konkrét példát a Magyar Turisztikai Ügynökség (MTÜ) Statisztikai Adattárház jelentette. Ez a több milliárd rekordos, naponta frissülő adattárház elsősorban jól strukturált adatokat tartalmaz, de ezen felül több száz elkészült riport jelenti a nem strukturált adatforrást az ágens számára – mindkét forrástípus kezelése elengedtethetetlen a teljes vállalati tudásbázis kiaknázásához. A naponta változó belső adattárházon túl a havi bontásban publikusan elérhető KSH adatokkal is szükséges dolgozni. Az eredeti lekérdező felület egy komplex megoldás volt, melynek hatékony kezelése mélyebb ismereteket várt el az adattárház felépítéséről és a szélsőséges esetekről. Hibás használata gyakran az adatelemző és fejlesztő kollégák idejébe került, mivel nekik kellett végrehajtani a megfelelő lekérdezést, majd a vezetőségnek elmagyarázni az eredményeket. E felület kiváltása érdekében merült fel egy természetes nyelven alapuló megoldás kidolgozása.

Az itt bemutatott kutatásban az ágens architektúrájának kidolgozása során inspirációt merítettünk a ReAct (Yao et al., 2022) keretrendszer struktúrájából. A kivitelezés központi elemének az Anthropic által kifejlesztett Model Context Protocol (MCP) technológiát választottuk, mely egy nyílt forráskódú standard az AI applikációk külső rendszerekhez csatolásához. Az MCP gyorsan a modell-eszköz kommunikáció standard protokolljává vált (Anthropic, 2024) az elmúlt évben, még a versenytárs LLM fejlesztő cégek is bevezették és adoptálták az új modellagnosztikus protokollt. Az MCP eszközök könnyen, újra felhasználható módon csatlakoztatják az egyedi eszközöket bármely LLM-hez, sőt, akár több nyelvi modellhez is. A külső rendszerek lehetnek adatforrások, eszközök, vagy akár munkafolyamatok. Az MCP-t gyakran hasonlítják az USB-C csatlakozóhoz, mint egyszerű kapcsolódást biztosító megoldás a nyelvi modellek számára (Introduction - Model Context Protocol, 2025). A protokoll egy kliens-szerver architektúrát követ, ahol a kliens megszerzi a kontextust a szervertől. Ebből következik, hogy MCP szervert lokálisan vagy távoli eléréssel is futtathatunk. Az adatcsere egy JSON-RPC 2.0 üzenetstruktúrán keresztül valósul meg (Introduction - Model Context Protocol, 2025).

A legtöbb ágens eszközhasználattal interaktál a környezetével. Azonban az, hogy a nyelvi modell által végrehajtott tervezési lépésben meddig terjedhet az autonóm döntéshozatal, az adott ágens architektúrától függően változhat (Händler, 2023). A hatékonyságon túl átláthatósági és biztonsági szempontokból is fontos a kódból kezelt és a nyelvi modell által kezelt feladatrészek elkülönítése.

Az LLM-ek feladatának meghatározásához fontos technikai elem a rendszer prompt, ami egy átfogó utasításhalmaz, mely meghatározza az MI ágens viselkedését

valamennyi LLM interakció során. Az ágens-alapú rendszerek feladat- és hatáskörét a rendszer prompt szabja meg, ezen promptok ismerete jelentős információkat árul el az adott ágens működéséről (GitHub, 2025). A rendszerutasítások általában egyszer kerülnek beállításra, és változatlanok maradnak, konzisztenciát biztosítva az MI ágens általános szerepében, feladatkörében. Az alapvető működés szempontjából a rendszer promptban megfogalmazott utasítások hatása erősebb a felhasználói promptolásnál.

Az LLM-et irányító instrukciók pontos és részletekbe nyúló megfogalmazása kritikus pontja az ágens létrehozásának. A kontextus a legfontosabb az LLM-ek válaszminőségét tekintve, ezt támasztja alá a context engineering megjelenése az egyszerű promptoláson túl (Mei et al., 2025). További lehetőség az ún. docstring-ek használata, ami a forráskód adott szegmensének működését dokumentáló szöveg. Az ezekben megfogalmazott leírások a függvény funkcióját, a paraméterek lehetséges értékeit, vagy túl nagy értékkészlet esetén a keresési módszert, továbbá a default értékeket, a formátumokat és ha-akkor felépítésű szabályokat is tartalmaznak. A függvényekből a FastMCP könyvtár eszköz definíciókat készít, melyek legfontosabb része a függvényleírás tartalma.

Az ágenshez beérkező kérések komplexitásának elemzésének egy praktikus módja a feladatok lépésszámának meghatározása, mivel ez határozza meg a válaszidőt és növeli a hibázási lehetőséget is.

A tesztelés alatt zárt és lokálisan futtatható nyílt modelleket is alkalmaztunk: GPT-5 és GPT-5 Mini (OpenAI, 2025), Gemini-2.5-flash és Gemini-2.5-flash-lite (Comanici et al., 2025), Llama 4 Maverick (Meta, 2025), Qwen3 80B és Qwen3 235B (Yang et al., 2025), valamint gpt-oss-20b és gpt-oss-120b (Agarwal et al., 2025).

3 Az ágens-alapú chatbot architektúra és működése

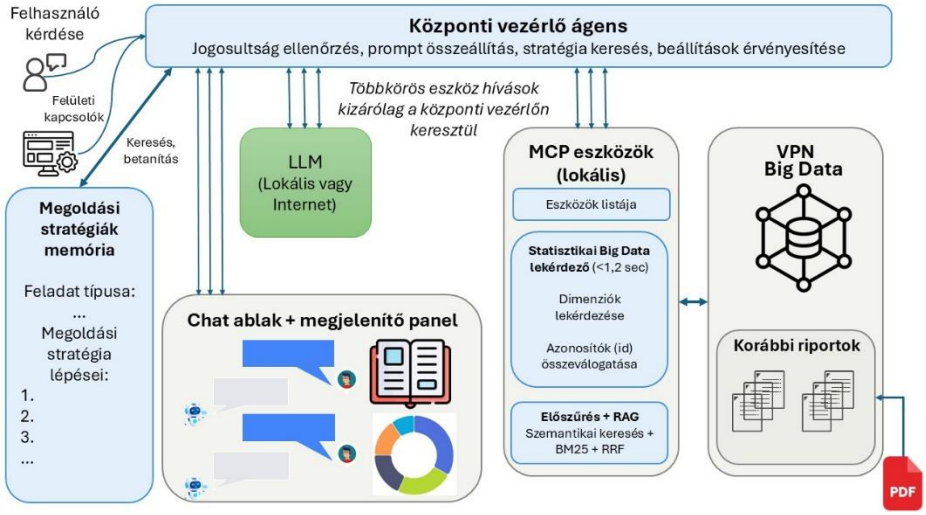
A vezetői kommunikációs kihívás megoldására a javasolt megoldás egy intuitív chatbot felület és egy dedikált ágens architektúra. A saját ágens-alapú chatbot architektúra tervezésekor két fő irányelvet vezettünk be és alkalmaztunk a hallucinációk csökkentése céljából: a *hibrid vezérlés* és a *folytatólagos prompttanulás* elvét. Ezeket az elveket a módszertani részben bemutatott technikai megoldásokra épített sajátságos architektúrában valósítottuk meg. A következőkben a tervezési fázisban megfogalmazott, fejlesztést irányító elveket, majd az ennek hatására kidolgozott rendszer architektúrát mutatjuk be, külön kiemelve a működés folyamatát. A fejlesztés első fázisában optimalizációs teszteket futtattunk a tényleges alkalmazás előtt.

A hibrid vezérlés a determinisztikusan, algoritmussal megoldható feladatrészeket továbbra is explicit kódból kezeli, míg a nyelvi modell hatáskörét leszűkíti annak általános nyelvi képességeit kihasználó feladatkörökre. Utóbbihoz tartozik a felhasználói kérdés jelentéstartalmának dekódolása és a megfelelő lekérdezési eszköz (függvény) kiválasztása és felparaméterezése, valamint a válasz megfogalmazása és összefoglaló készítése. Már a folyamat elején a felhasználó kezébe adjuk a vezérlést az elérési felületen állítható kapcsolókkal, amelyek determinisztikusan, algoritmus segítségével megváltoztatják a lekérdezések paraméterezését (pl. számítási mód kiválasztása) vagy egy más lefutási ágba terelik az ágenst (pl. riportokban keresés). Ez

a hibrid megközelítés teret ad az algoritmikus működésnek az LLM előnyeinek kihasználása mellett. A hibrid vezérlés fő kihívása az egyensúly megtalálása a jól definiált algoritmusok és a széleskörű feladatvégzésre alkalmas nyelvi modellek között, ahol az optimális működés elérése az adott feladat igényeitől függ.

A folytatólagos prompttanulás elve azt jelenti, hogy az egyes kérdéstípusoknál előforduló nem determinisztikus működést a felhasználó képes korrigálni általános megoldási stratégiák leírásával. Ezek beépülnek az ágens belső tudásbázisába, és minden új beérkező kérdésnél egy RAG (Retrieval Augmented Generation) keresést hajt végre az ágens a tudásbázis szövegében, hogy csak az adott feladattípushoz tartozó releváns kontextust töltse be a modell rendelkezésére álló információk közé.

A két elvet leképező saját architektúránk többszöri eszközhívásokra, akciókra, majd ezek kiértékelésére, reflektálásra épül. A kiértékelés során az ágens hozza meg a döntést a rendszer prompt, a docstring tartalma, a felhasználói prompt, az eddigi csevegést eltároló memória és az akciók segítségével begyűjtött adatok segítségével, hogy készen áll-e a végleges kimeneti válasz megfogalmazására.



1. ábra: A Big Data ágens architektúrája (saját munka)

Az ágens-alapú chatbot architektúra hat fő komponensből tevődik össze (lásd 1. ábra): (1) a központi vezérlő ágens, (2) az LLM (lokális vagy API használaton keresztül), (3) az MCP eszközök, (4) a Big Data adatbázis, (5) a megoldási stratégiákat tartalmazó tudásbázis és (6) a felhasználói felület, ami áll egy chat ablakból és egy megjelenítő panelből. A központi vezérlő ágens irányítja a megoldási munkafolyamat menetét, mely függ a beérkező kérdéstől és a felületen aktív beállításoktól. Először ellenőrzi a felhasználói jogosultságokat, majd, ha ezek megfelelőnek bizonyultak, érvényesíti a felhasználó által alkalmazott beállításokat és a beérkezett kérdést feldolgozva lekérdezi a tudásbázisban tárolt stratégiákat. Ezt követően a kiválasztott LLM számára listázza az eszközöket és megjeleníti a stratégiákat. Az LLM ekkor vagy

rögtön válaszol (ritka), vagy egy MCP eszközt választva felparaméterez egy függvényt, és visszakéri annak az eredményét. Az eredmény alapján döntést hoz, majd további lekérdezéseket futtat, ha nem elegendő az információ, vagy visszatér a végső válasszal a felhasználó számára, ha már elegendő információt gyűjtött össze. Érdemes érvelő modellt alkalmazni, mely képes levezetni, hogy a kapott eredmény megfelelő-e a kérdés megválaszolására.

Fontos megjegyezni, hogy nem az LLM hívja direktben az MCP eszközöket, hanem miután megkapta az eszközlistát, és kiválasztotta, felparaméterezte a megfelelő függvényt, akkor visszatér a függvényhívással, amit a vezérlő ágens hív meg lokálisan, és ezen a ponton algoritmikusan közbe tud avatkozni új paraméterek átadásával, meglévők megváltoztatásával vagy a teljes vezérlés eltérítésével.

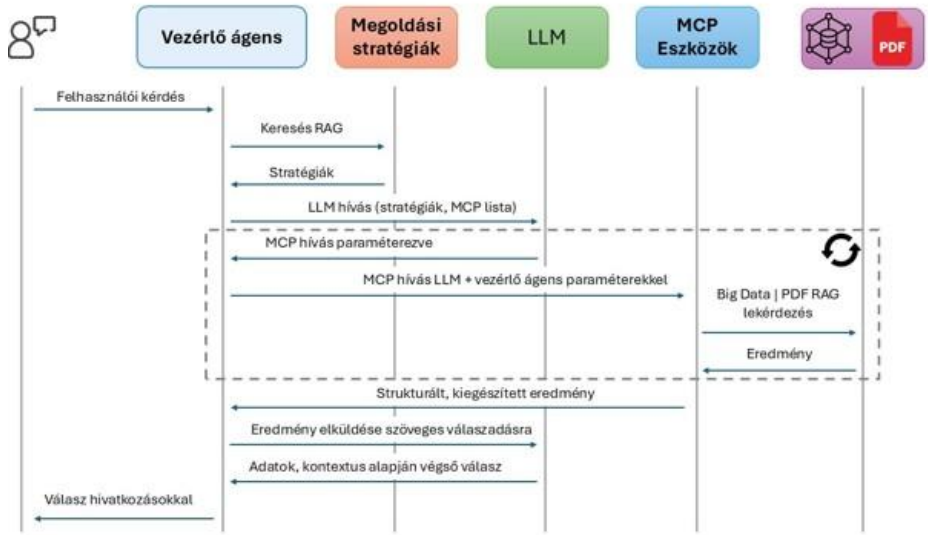
A Big Data lekérdező ágensben a rendszer prompt tartalma a legerősebb vezérlő tényező az LLM számára, hiszen ez minden interakció során érvényes. Az utána következő felhasználói kérésekben, utasításokban hiába próbáljuk felülírni a rendszer promptban megfogalmazott iránymutatásokat, az LLM tesztjeink alapján ragaszkodik azok betartásához. Hangsúlyos figyelmeztetésként fogalmaztuk meg az LLM számára, hogy (1) nem térhet el a tárgytól a felhasználó kifejezett kérésére sem; (2) mindenképpen valamilyen eszközt kell használnia a válaszadáshoz, amihez megadható az eszközök listája; (3) ha hiányos az információ a lekérdezés paraméterezéséhez, akkor mindenképpen kérdezzen vissza a konkrét hiányzó paraméterek megnevezésével; és (4) amennyiben nincs olyan eszköze vagy nem tudja felparaméterezni a felhasználó kérdésének megfelelően, akkor ezt jelezze a válaszában és ne találjon ki számokat. Már az első teszek bizonyították, hogy a jól megfogalmazott rendszerutasítás jelentősen csökkentette a pontatlan válaszok arányát és a visszakérdezés sok esetben több lépéses, komplex feladatmegoldást eredményezett. A csevegési előzmények elküldése tovább erősíti a természetes nyelvi kommunikációt és ebből következően az eredmények pontosságát, mivel általában az egyik kérdésből következik a másik (pl. az előző eredmény további szempontok szerinti bontása) és az ember adottnak tekinti a korábbi kérdés-válaszokban definiált paramétereket.

A Big Data lekérdezés (a központi, legtöbbet használt függvény) esetén általában 20+ dimenzió és akár 100+ értékmező is megadható, melyek példánkban tipikusan idő és területi behatárolások, és csoportosítások, melyekre az aggregálás jellemzően több milliárd rekordon fut le. Az LLM számára részletesen és minél pontosabban el kell magyarázni az egyes dimenziók lehetséges értékeit és használati módjukat. A kisebb számosságú attribútumok esetében javasolt, hogy az MCP eszköz leírása tartalmazza az összes választható értéket és ezt külön jelezzük is az LLM felé, hogy csak ezen értékek közül választhat. A nagyobb (pl. 1000+) elemszám esetén a token-alapú árazás és a túl hosszú kontextus miatt érdemes csak röviden leírni az értékkészletet, néhány példát adni a sokszor előforduló esetekre, valamint egy külön MCP segédfüggvényt biztosítani a dimenzió értékeinek keresésére vonatkozóan. Ha az LLM magabiztosan rosszul paraméterez, akkor az MCP lekérdező függvény mindjárt a legelején kiszűri a nem megfelelő értékeket, a válaszban jelzi a hibát, és az LLM figyelmébe ajánlja a dimenzió kereshetőségét biztosító segédfüggvényt. Tesztjeink szerint több esetben is ez vezetett el egyes feladatok megoldásához.

Előfordulnak olyan dimenziók, ahol egy összetett attribútum struktúrával rendelkező entitást kell beazonosítani és erre általában az entitás neve önmagában nem elegendő

(pl. személynevek azonossága). A pontos eredmény elérése érdekében ekkor egy olyan eljárást kell az LLM számára definiálni, amely az entitás jellemzőivel paraméterezhető és a visszatérési értékében rekordonként egy egyértelmű egyedi (adatbázis) azonosítót tartalmaz. Ha az egyedi id-k összegyűjtésre kerültek, akkor már a fő Big Data lekérdező precízen felparaméterezhető és az eredmény is a felhasználó kérésének megfelelően pontos lesz.

Amennyiben két fogalom között egyértelmű megfeleltetés végezhető és csak az egyikkel paraméterezhető az MCP, akkor az LLM számára néhány példával ez az összefüggés megvilágítható és mellette a transzformációt elvégző program is megadható (pl. nemzetiség – országnév).



2. ábra: A Big Data ágens kommunikációs diagramja

Két fő lefutási ága van a kérdéseknek (lásd 2. ábra, elágazás): ezek az adatbázisra és a PPTX formátumú riportokra vonatkozó kérdések mentén különülnek el. Míg az előbbi esetben csak egy-két beállítás befolyásolhatja a folyamat lefutását, addig az utóbbi esetben mindig előszűrést alkalmaz az ágens a megfelelő riport kiválasztására a riportra vonatkozó fő dimenziók (metaadatok) és összefoglaló szerint. Az előszűrés után zajlik le az adott szöveges dokumentumban a részletes keresés: a RAG rendszer szemantikai és a BM25 keresést RRF újra rangsorolással egyesítő komplex keresési folyamata.

4 A belső tudásbázis építése: folytatólagos tanulás

A döntéstámogató rendszerekben alkalmazott LLM-alapú megoldások esetében gyakran előfordulnak ismétlődő hibák, mivel nincsen elég kontextusa a modellnek az adott üzleti helyzetről (Cheung, 2024). A folytatólagosan tanítható belső tudásbázis,

amit egyfajta tartós memóriának is tekinthetünk, lehetővé teszi, hogy a rendszer az elkövetett hibákból általános következtetéseket vonjon le a feladattípusok megoldásával kapcsolatban és a jövőben ezeket hasznosítsa a hasonló kérdések megoldása során. A kontextuson belüli tanulás könnyen áttekinthetővé és ellenőrizhetővé teszi az ágens memóriáját, ezáltal egy transzparens és követhető tanulási folyamatot biztosítva.

E megközelítés elválasztja egymástól az LLM általános nyelvi tudását és az adott szervezetnél előforduló, domain-specifikus műveleti szabályokat. Annak eldöntése, hogy mely feladattípusokhoz szükséges megoldási stratégiát hozzárendelni, nem triviális feladat, amelyet elsősorban a rendszer felhasználói tudnak helyesen mérlegelni. A nyelvi modellre bízni ezt a döntést csak további bizonytalanságot vezetne be, valamint, ha a felhasználó tisztában van a helyes megoldással, rengeteg felesleges próbálkozástól szabadítjuk meg az ágenst. A felhasználó által biztosított helyes megoldás úgy kerül eltárolásra, hogy nem maga a válasz szövege, hanem a feladat típusa és a hozzá tartozó megoldási stratégia legyen visszakereshető és alkalmazható.

Az egyszerű memóriához képest itt általános megoldási helyzeteket definiálunk, melyek a feladatok széles körét lefedik és a szükséges megoldási stratégia is ennek megfelelően kerül megfogalmazásra. Tehát egyfajta típusfeladatok megoldási útmutatóját nyújtjuk át a rendszernek, hogy determinisztikussá tegyük az eredményeket. Tesztjeink alapján ugyanis volt több olyan feladat, amikor a paraméterezést elrontotta a rendszer, majd rájött a hibára és onnan kezdve véletlenszerű eloszlásban három stratégiát alkalmazott: (1) meghívta újra, immár helyes paraméterezéssel a Big Data lekérdezőt; (2) belenyugodott az eredménybe és közölte a felhasználóval, hogy ez így sikerült; (3) teljesen rossz paraméterezéssel hívta újra a lekérdezést.

A többlépésben lekérdezhető komplex problémák megoldása kihívást jelent a mai legfejlettebb LLM számára is. Ilyen esetekben a lefutás nem determinisztikus, több úton is elindulhat a nyelvi modell. Ennek a bizonytalanságnak a kezelésére alkalmas a belső tudásbázis, szerepét a következőkben egy konkrét példával illusztráljuk.

A turisztikai adathalmazban az első ’n’ külföldi küldőországok lekérdezésekor a kezdeti eredmény első helyen mindig Magyarországot tartalmazza. Ez egy döntési pontot jelent, mivel így csak n-1 külföldi ország szerepel az eredményben. Az LLM az esetek egy részében n-1 országot sorol fel a felhasználónak, másik részében új lekérdezést indít n+m elemre, ahol m jellemzően 5 és így már vissza tudja adni a kért n elemet. Az első megoldás hibás, míg a második token- és időpazarlással jár. A feladattípushoz megoldási stratégia definiálásával mindig egy lépésben n+1 elemet kér le a modell, ezáltal 100%-ban pontos lekérdezések valósulnak meg ennél a kérdésnél.

Az elemzők rendszeresen készítenek bizonyos időszakokra és területekre vonatkozó, grafikonokat is tartalmazó jelentéseket, melyek azonos (esetünkben turisztikai) mutatókkal dolgoznak. Ezek a riportok szemantikailag nagyon hasonlók, mert ugyanazon kulcsfontosságú mutatókkal készülnek, ezért a keresési eredményeket nehéz rangsorolni. Mivel több száz nagy hasonlóságú riport áll rendelkezésre, ezért kellett egy szűrési lépést (szabályt) bevezetni a metaadatok és a generált összefoglaló alapján, és az összes dokumentumban keresés helyett riportonként keresni a megfelelő adatokat.

A vállalati elemző riportok esetében a korábban már kifejlesztett saját RAG (Vándor és Csáki, 2025) algoritmusunkat kívántuk használni, de felmerült az a probléma, hogy

az azonos jellegű riportok szövegezésben alig különböznek, csak a számok változnak, mert például egy másik időszakra vonatkoznak. Ebben az esetben a releváns szövegrészeket megtaláló szemantikai kereső nagyon sok, értelmezésüket tekintve a felhasználói kérdésre tökéletesen válaszoló szövegrészt talál eltérő értékekkel. A felhasználó sok esetben nem definiálja az időszakot, hanem egyszerűen a legfrissebb adatokra kíváncsi. Ennek következtében nem alkalmazható egy lépésben a RAG algoritmus, hanem szükségessé vált bevezetni egy riport előválasztó lépést. A dokumentumok feldolgozása során LLM-el általános, a riport típusára vonatkozó összefoglalót generáltatunk az időszakok megemlítése nélkül, valamint az egyes riportokra vonatkozó metaadatok (terület, időszak, típus) is kitöltésre kerülnek. A felhasználó kérdése alapján az LLM listázza a releváns riport típusokat és az azon belül rendelkezésre álló riportok legfontosabb adatait, majd kiválasztja a megfelelő dokumentum azonosítókat és ezekkel hívja meg a RAG alapú rendszert. A RAG MCP eszköz egy saját LLM hívás segítségével megfogalmazza a választ, amihez algoritmikusan hozzákapcsoljuk a három legrelevánsabb szövegrészlet (chunk) hivatkozását, melyeket a központi vezérlő ágens azonnal továbbít a felhasználó felé és nem futtat keresztül a többi feladathoz használt LLM-en az utolsó lépésben. A felhasználó pedig megtekintheti a riport hivatkozott oldalát.

5 Grafikonrajzolás és adatelemzés

Adatbázis lekérdezések eredményeinek döntési felhasználása esetén igen fontos funkció az adatok szemléletes megjelenítése és elemzése. Ezért az ágens rendszer felhasználói lekérdezésekre adott válaszokat nem csupán szöveges formában, hanem interaktív grafikonokon keresztül is megjelenítheti, ezzel elősegítve a döntéshozók gyors és intuitív információ-feldolgozását. A grafikus megjelenítés célja, hogy a nagyméretű, komplex adathalmazokból kiemelje a releváns trendeket, eltéréseket és összefüggéseket, ezáltal támogatva a valós idejű vezetői döntéshozatalt.

A feladatmegoldáshoz hasonlóan a grafikonrajzolás során is rögzítésre kerülnek a legfontosabb paraméterek: csak Echarts oszlop- és kördiagramok készítése engedélyezett az előre definiált hivatalos színpalettával. Az eszköz azonban rugalmas, mivel lehetővé teszi az egyszerű diagramokon túl a többdimenziós összehasonlításokat is (pl. időszakok vagy kategóriák direkt összehasonlítása).

A kötelező paraméterek biztosítják az alapvető funkcionalitást, míg az opcionális beállítások a testreszabhatóságot. A diagramok címének, feliratainak generálása, formátumának beállítása, elhelyezése már az ágens feladata. Az értékek megjeleníthetők az oszlopokon belül vagy kívül, a kördiagramoknál pedig formázási lehetőségek állnak rendelkezésre, amelyek megjeleníthetik a neveket, értékeket és százalékos arányokat egyaránt. A tooltip és jelmagyarázat elhelyezését is ágens kezeli.

Az eszköz a kimenetét speciális formátumban adja vissza: egy JSON alapú konfigurációs objektum tartalmazza a teljes diagram specifikációt, mely közvetlenül értelmezhető a frontend komponensek által és a diagram azonnal megjelenik a felhasználói felületen.

6 Lekérdezési eredmények

A feladatok komplexitását vizsgálva négy szintet állapítunk meg. A feladatok között egyszerűnek tekintjük az egy lépésben lekérdezhető és megválaszolható kérdéseket, amelyeknél az első eszközhívás után képes válaszolni az ágens, az eszközválasztást nem számítva plusz lépésnek. A bonyolultság következő foka, amikor valamelyik MCP segédfüggvényt kell igénybe venni először a helyes paraméterek beállításához és a végső lekérdezéshez. Majd a három vagy több, de nem nagy számosságú lépésben megoldható feladatok következnek, amelyek esetén a végső eredményhez nem elég egy jól paraméterezett Big Data lekérdezés, hanem több aggregációs lekérdezés eredményét kell kombinálni megfelelően. Egy alternatív példa a riport generálása mintaszövegbe beillesztéssel, ahol a megfelelő számok kinyerése és beillesztése többkörös lefutást eredményez. Komplex feladatok esetén (mint például az aggregált célérték keresése egy nagy számosságú tartományban) pedig előfordult log2(N) lépést igénylő folyamat is. Megjegyezzük, hogy ez utóbbi esetben a paraméter értéktartomány többszöri felezésével érhető el a megoldás. Az első és az utolsó szint ritkán fordul elő, a legtöbb feladat két-három lépést igényel.

Mind a strukturált adatbázis lekérdezések, mind a nem strukturált riportokban keresés tesztelésre került több nyelvi modellel. A 28 egyre nehezedő lekérdező feladatban az egyszerű, egyetlen szűrést igénylő kérésektől (pl. „Hány turista járt Baján 2025-ben?”) az időszakösszehasonlításon és százalékszámításon át egészen a többlépéses grafikonrajzolásig (pl. „Készíts kördiagramot 2025-ben a legtöbb turistát fogadó településekről. Az első 7 település mellett jeleníts meg egy egyéb kategóriát.”) terjedt. A visszautaló kérdésekkel egy csevegésen belül zajlott a tesztelés, valós kérdés-válasz folyamot szimulálva.

A kiválasztott modellek között egyaránt szerepel lokális, nyílt (gpt-oss, Qwen, Llama) modell és API-n elérhető, zárt (GPT, Gemini) modell is. A távolról elérhető modellek leginkább a teljesítmény mérése és az összehasonlítás szempontjából kerültek be, a gyakorlati megoldás esetében csak lokális szerveren (L40 kártyákkal) futtatott modellek jöhetnek szóba. Az adatbázisban csak anonimizált hash található, személyes adatok nem kerültek ki a tesztelés során.

1. táblázat: Modellek teljesítménye 14 adatbázis lekérdezés feladaton

Modell	Eredmény	Minőség
GPT-5	100,0%	visszakérdez (+), grafikon szerkezete furcsa (-)
GPT-5 mini	96,4%	1 paraméter kihagyása
GPT-4o	82,1%	összehasonlítás (-), 1 paraméter kihagyása, komplex megoldási stratégia (-)
gpt-oss-120b	32,1%	paraméterek pontosan definiált neveit sem adja meg jól
gpt-oss-20b	32,1%	paraméterek pontosan definiált neveit sem adja meg jól
Qwen3-235B	71,4%	összehasonlítás (-), grafikonrajzolás (-), megoldási stratégia (-)
Llama-4-Maverick-17Bx128E	42,9%	grafikon készítés kérés nélkül, rossz kimeneti mező választás, végtelen ciklus

A gyakorlati tapasztalatok azt mutatták, hogy a GPT-5 és mini modelleket alkalmazva magas pontosság érhető el. Azonban nem várt probléma, hogy a modellek egyes paraméterek behelyettesítését ritkán végezték el helyesen, pl. a szálloda típus kiválasztása is ezek közé tartozik a „Sorold fel az 5*-os szállodákat…” típusú kérdéseknél.

A modellek teljesítményét vizsgálva kizárólag a GPT-5 tudta megválaszolni az összes kérdést, mindezt úgy, hogy visszakérdezett a bizonytalan megfogalmazásoknál (pl. szoba vagy férőhely kapacitás). A kisebb GPT-5 mini esetében már megfigyelhető paramétertévesztés. A 4o nem érvelő modell, ezért az összehasonlítási feladatban hibázott és az utolsó, „egyéb” kategória létrehozására vonatkozó megoldási startégiát sem tudta sikeresen alkalmazni. A lokális modellek közül a gpt-oss modellek mérettől függetlenül súlyos hibákat vétettek, legtöbbször a paraméterek pontosan definiált neveit sem tudták helyesen behelyettesíteni a függvénybe. Az OpenAI nyílt modelljei nem rendelkeznek a GPT-5-höz hasonlítható minőségű eszközhívás betanítással, ez látszik az eredményeken. A Qwen3 összességében a legjobb nyílt modell lett, de számos problémával rendelkezik, mint a hibás grafikonrajzolás és a megoldási stratégiák nem megfelelő kihasználása. A Llama 4 modell váratlan hibákat vétett, két alkalommal grafikont készített kérés nélkül, egy kérdésnél változtatás nélkül, végtelen ciklusban újra lekérte az adatokat, és hallucinációt produkált az utolsó kérdésnél. A teljes kérdéssorozatban ez az egy tipikus hallucinációs hiba, az összes többi hibatípus az utasítások, stratégiák nem megfelelő értelmezéséből ered.

2. táblázat: Modellek teljesítménye 6 riportokban keresés feladaton

Modell	Eredmény	Minőségi hibák	Idő (sec)
GPT-5	100%	felesleges információk	52
GPT-5 mini	100%	felesleges információk	66
GPT-4o	67%	2 hiányzó adat; táblázat értelmezése	19
gpt-oss-120b	100%	legjobb megfogalmazás és minőség	44
gpt-oss-20b	67%	2 hiányzó adat; táblázat értelmezése	17
Qwen3-80B	50%	3 hiányzó adat; 1 db fájl kizárása; hivatkozás blokk rossz	28
Qwen3-235B	67%	2 hiányzó adat; táblázat értelmezése	42
Llama-4-Maverick-17Bx128E	0%	összes PDF kizárása	17
Gemini-2.5-flash	83%	1 hiányzó adat	28
Gemini-2.5-flash-lite	67%	2 hiányzó adat; válasz nem koherens	14

A riportokban keresés a RAG módszerrel kapott eredményeket mutatja meg, 6 adat lekérdezésével tesztelve. A GPT-5 és GPT-5 mini modellek bár hibátlanul vizsgáztak, a vártnál több tokent használtak el és felesleges információkat is közöltek a válaszokban. A 4o két esetben nem találta meg az adatokat, beleértve egy táblázat közepén található adatpontot. A gpt-oss-120b 100%-os eredményt produkált, ráadásul

a legjobb megfogalmazással és minőséggel, felesleges információk nélkül. A 20 milliárdos, kisebb oss modell a 4o-val egyenértékű válaszokat nyújtott, a táblázat értelmezése ennek a modellnek is kihívást jelent. A nagyobb Qwen3 is 2 hibával, míg a kisebb már 3 hibával szerepelt – utóbbi az egyik adatpontot tartalmazó fájlt kizárta és másik riportban keresett. A Llama 4 Maverick nem megfelelő dokumentumokban végezte az információkeresést.

Az 1 perc körüli feldolgozási idők a GPT-5 modelleknél a válasz előtti érvelési folyamatnak tudhatók be, éppen ezért szignifikánsan gyorsabb a 4o modell 20 másodperc alatti ideje. A nyílt, lokálisan futtatható modelleknél jelentősen függ a sebesség a paraméterszámtól: 100 milliárd felett több, mint 40 másodperc, míg 20 milliárd aktív paraméter alatt kevesebb, mint 20 másodperc a mért válaszidő. A kifejezetten sebességre optimalizált Gemini-2.5-flash modellek a válaszadás felgyorsítása céljából kerültek tesztelésre. A 2.5-flash-lite modell a legjobb feldolgozási időt érte el (14 sec), de a válasz minősége nem elegendő a gyakorlati alkalmazáshoz: a 2 hiányzó adaton túl a válasz koherenciája sem érte el a kívánt szintet.

7 Tanulságok összefoglalása

A valós üzleti körülmények között pontos és hallucinációmentes válaszokat szolgáltató, természetes nyelven lekérdezhető ágens-alapú chatbot rendszer a szervezeti tudásbázis újszerű, de megbízható kiaknázását teszi lehetővé. A válaszok nemcsak reprodukálták a manuálisan lekérdezhető eredményeket, de validálhatóak is a függvénybe beillesztett paraméterek segítségével. Emellett a legtöbb rossz lekérdezés felismerhető a hibás felparaméterezésből, így erre a későbbiekben akár egy korrekciós módszer is építhető.

Az adatbázis lekérdezés feladatokban csak a GPT-5 szerzett 100%-os eredményt. A nyílt modelleknek kihívást jelentett a feladatsor, a gpt-oss modellek esetén a rendszer prompt rövidítése vagy angol utasítások megadása orvosolhatja a gyengébb eredményeket. A nagyobb paraméterszámú modellek általában jobb teljesítményt nyújtottak. A riportokban történő keresési feladatoknál a GPT-5 modellek hibátlan eredményeket értek el, míg a gpt-oss-120b modell a legjobb válaszminőséget nyújtotta felesleges információk nélkül. A feldolgozási idők elemzése kimutatta, hogy a modellméret és a válaszadási sebesség között szoros, negatív irányú összefüggés található: a nagyobb modellektől szignifikánsan lassabb válasz várható.

A feladattípusokhoz megfogalmazott megoldási stratégiák beépítésével az ágens a nem egyértelműen elvégezhető kérések kezelésére is megtanítható. A visszakérdezés fontos pontja a bizonytalanság redukálásának. Az LLM-ek rugalmassága és a determinisztikus folyamatok biztonsága teszi lehetővé a nagyfokú autonómiát.

Az itt bemutatott megoldás új eszközt kínál a Big Data közvetlen vezetői hozzáférhetőségére a szervezeten belül és elősegíti a pontos, adatvezérelt döntéshozatalt. Bár egy adott, konkrét esethez készült ágens architektúra került bemutatásra, a megoldás más adatbázishoz is hozzákapcsolható, és annak sajátságaira átszabható a lekérdező eszközök kiválasztásával és illesztésével.

Bibliográfia

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., ... & Zhao, S. (2025). gpt-oss-120b & gpt-oss-20b model card. arXiv preprint arXiv:2508.10925.

OpenAI. (2025, August 13). GPT-5 system card. OpenAI. https://cdn.openai.com/gpt-5-system-card.pdf

Altman, S. (2025, January 6). *Reflections*. Sam Altman. https://blog.samaltman.com/reflections

Anthropic. (2024). *Introducing the Model Context Protocol*. Anthropic.com. https://www.anthropic.com/news/model-context-protocol

Cheung, M. (2024). A Reality check of the benefits of LLM in business. *arXiv preprint arXiv:2406.10249*.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., ... & Mehta, S. V. (2025). Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261.

Dong, Z., Guo, Y., Han, C., Ma, W., Xiong, J., Wen, S., & Xiang, Y. (2024). AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways. *ACM Computing Surveys, 57*, 1 - 36.

GitHub. System prompts and models of AI tools. (2025). GitHub - x1xhlol/system-prompts-and-models-of-ai-tools: FULL Augment Code, Claude Code, Cluely, CodeBuddy, Comet, Cursor, Devin AI, Junie, Kiro, Leap.new, Lovable, Manus Agent Tools, NotionAI, Orchids.app, Perplexity, Poke, Qoder, Replit, Same.dev, Trae, Traycer AI, VSCode Agent, Warp.dev, Windsurf, Xcode, Z.ai Code, dia & v0. (And other Open Sourced) System Prompts, Internal Tools & AI Models. https://github.com/x1xhlol/system-prompts-and-models-of-ai-tools

Händler, T. (2023). A Taxonomy for Autonomous LLM-Powered Multi-Agent Architectures. *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management.*

Introducing ChatGPT agent: bridging research and action. (2025). Openai.com. https://openai.com/index/introducing-chatgpt-agent/

Introduction - Model Context Protocol. (2025). Modelcontextprotocol.io; Model Context Protocol. https://modelcontextprotocol.io/docs/getting-started/intro

Mei, L., Yao, J., Ge, Y., Wang, Y., Bi, B., Cai, Y., Liu, J., Li, M., Li, Z.-Z., Zhang, D., Zhou, C., Mao, J., Xia, T., Guo, J., & Liu, S. (2025). *A Survey of Context Engineering for Large Language Models*. ArXiv.org. https://arxiv.org/abs/2507.13334

Meta. (2025). The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. Meta.com. https://ai.meta.com/blog/llama-4-multimodal-intelligence/

Sapkota, R., Roumeliotis, K. I., & Karkee, M. (2025). AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468*.

Vándor, P. & Csáki, Cs. (2025). RAG módszerek implementálásának kihívásai egy célorientált chatbot rendszer kontextusában. In: Berend, Gábor; Gosztolya, Gábor; Vincze, Veronika (Eds.) Magyar Számítógépes Nyelvészeti Konferencia online edition, Hungary: Szegedi Tudományegyetem TTIK, Informatikai Intézet (2025) 278 p., pp. 97-110., 14 p.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., ... & Qiu, Z. (2025). Qwen3 technical report. arXiv preprint arXiv:2505.09388.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., & Cao, Y. (2022, October). React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Zhao, A., Huang, D., Xu, Q., Lin, M., Liu, Y., & Huang, G. (2023). ExpeL: LLM Agents Are Experiential Learners. *AAAI Conference on Artificial Intelligence*.