A nagy nyelvi modell alapú szervezeti automatizáció lehetőségei és az autonóm ágensek kapcsolódó kihívásai

Vándor Péter* [1] és Csáki Csaba* [2]

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

Kivonat: Az elmúlt egy év során az átlag felhasználó számára is elérhetővé váltak a nagy nyelvi modelleken alapuló generatív mesterséges intelligencia megoldások. A modellek beépítése vállalati folyamatokba azonban nem magától értetődő. Manuális használathoz a felhasználónak meg kell tanulnia szabadszöveges instrukciókat („prompt”-ot) írni, míg másik lehetőség a nagyszámú segédprogramokból való választás, melyeket azonban integrálni kell, különösen akkor, ha az adott szervezet saját megoldásokat is épít. A cikkben bemutatott kutatás egy kifejezetten tesztelésre kifejlesztett mintarendszerrel különböző nagy nyelvi modelleket és ingyenes alkalmazásokat illetve adatbázis elérhetőségeket vizsgált, de elemezte saját készítésű elemek és adattáblák integrálhatóságát is vállalati feladatok szimulálásával. A legfontosabb tanulság, hogy az egyes elemek és API-k illesztése nem mindig zökkenőmentes: sok az inkonzisztencia, és a legkisebb adathiba vagy definíciós eltérés is komolyan korlátozhatja a végső kimenet használhatóságát. A hibák javításával gyorsan lehet fejlődést elérni és célzott, saját elvárásokhoz illeszkedő nyelvi alkalmazásokat írni, ami komoly lehetőségeket nyithat meg még kisebb cégek számára is.

1 Bevezetés

Az elmúlt egy évben nagyot fordult a világ a nagy nyelvi modellek hétköznapi elterjedése tekintetében: a legutóbbi MSZNYK idején már több mint két hónapja volt elérhető az OpenAI ChatGPT nevű szoftvere (OpenAI, 2022) és legalább 100 millió felhasználót jegyeztek. Ezzel az átlag felhasználó számára is elérhetővé váltak (akár ingyen vagy bővített szolgáltatások esetén nem túl magas havi díjért) a nagy nyelvi modelleken alapuló ún. generatív mesterséges intelligencia megoldások (Wasvani és mtsai, 2017).

A nagy nyelvi modellek (angol rövidítéssel LLM) lényege, mint a neve is utal rá, hogy nagyon magas, akár több száz milliárdos paraméterszámmal dolgoznak, jellemzően transzformer architektúrán enkóder vagy dekóder (esetleg mindkettő) megközelítéssel készültek és tanításukhoz szintén hatalmas (akár TB méretű) adathalmazt használtak (magyarul lásd pl. Yang és mtsai, 2023-as áttekintését). E modellek generatív jellegét az adja, hogy a fenti technikai megoldásokkal és a nyelv szemantikáját megragadó vektorreprezentációk segítségével már természetes nyelven fogalmazhatjuk meg kéréseinket és szintén természetes nyelvi formában kapunk választ, ami a felhasználó

számára a legegyszerűbben értelmezhető. Azaz a rendszer egy adott kérdésre oda illő választ generál természetes nyelven (Gillioz és mtsai, 2020; Knap és mtsai, 2023).

A nagy nyelvi modellek nemcsak a nagyközönséget, de a piaci szférát is elérték. A korábban órákat igénybe vevő feladatok akár percek alatt megvalósíthatóak nyelvi modellek támogatásával. Habár a nagy nyelvi modellekre épülő automatizáció sokszorosára növelheti bizonyos rutinszerű (szöveges) feladatok elvégzésének sebességét, mint pl. egy email megfogalmazása vagy egy adatbázis lekérdezés, a gyors elterjedés és népszerűség nem jelent automatikus beépülést a vállalati folyamatokba. A nagy nyelvi modellek bevezetése vállalati környezetbe ugyanis számos kihívást rejt magában. A saját tulajdonú adatok kezelése, az új eszközök vállalati munkafolyamatokba történő beillesztése, a modellek hálózaton keresztül történő automatikus elérése vagy a felmerülő kihívások (pl. a hallucináció elkerülése) nehézségeket okoz sok cég számára (Ayinde és mtsai, 2023).

A korai bevezetések fél-manuális megoldásokhoz vezettek, ahol a dolgozók maguk írják meg a modellel való kommunikációhoz szükséges szabadszöveges instrukciókat (angolul 'prompt'). LLM kérdések írása nem magától értetődő. A feladat nehézségét mutatja az ún. 'prompt-engineer' (lekérdezés szakértő) szerepkör elterjedése. Különösen akkor válik a feladat bonyolulttá, ha a szöveges modell-lekérdezés mellett hagyományos (jellemzően céges) adatbázisokból is össze kell szedni eredményeket, majd a kapott válaszokat integrálva diagramokat vagy prezentációt kell készíteni. A feladatok automatizációjához alkalmazások egész sora született (lásd pl. a https://theresanaifort-hat.com/ kollekciót) illetve a modellek fejlesztői egyrészt közvetlen elérést (API-kat), másrészt segédprogramokat bocsájtottak rendelkezésre (OpenAI, 2023). De ezek használata sem magától értetődő és tartogat meglepetéseket a vállalati fejlesztők számára.

Egy lehetséges megoldás például az ágens (Liu és mtsai, 2023). Az önálló döntéshozatalra és eszközhasználatra képes ágensekkel minden eddiginél gyorsabban kinyerhetőek a szükséges adatok különböző strukturált és nem strukturált formában történő tárolás esetén is. Ebben kulcsfontosságúak a központi adatbázisok, és bár a vállalati tudás egy része dokumentum formátumban létezik, ágensekkel mindkét forma elérhetővé válik.

Jelen cikkben bemutatott kutatás-fejlesztés célja egy olyan ágens létrehozása több lépcsőben, amely képes több, nagy nyelvi modellre épülő eszközt integráltan kezelni összetett lekérdezések megválaszolása és vizuális prezentálása érdekében. Ezen belül fontos, hogy képes legyen helyes és pontos lekérdezéseket írni és futtatni, hatékonyan integrálja a különböző forrásokból érkező adatokat, majd az eredményekről készítsen prezentálható diagramokat. Első körben ehhez három feladatot választottunk egy tipikus vállalati adatkeresési és vizualizációs problémát szimulálva: adatbázis lekérdezés, dokumentum feldolgozás, és diagram készítés. A feladatokat nem csak külön-külön, hanem az életszerűség érdekében integráltan is teszteltük.

A cikk bemutatja a három feladatot, ismerteti a kezelésükhöz készített ágensek jellemzőit, a felhasznált technológiákat (LLM-re épülő alkalmazásokat), majd leírja a tesztelés folyamatát mind az egyes feladatokra, mind azok integrálása esetén. A kapott eredmények bemutatása és elemzése után a cikk felhívja a figyelmet azokra a sokszor nüánsznyi csapdákra, melyekkel minden LLM-alkalmazás fejlesztőnek tisztában kell lennie, beleértve azok kezelését is.

2 Fejlesztett ágensek, alkalmazott technológiák, felhasznált adatok

Az üzleti intelligencia és az adatalapú döntéshozatal térnyerésével egyre több elemzési feladat során kell különböző (szervezeten belüli és külső) forrásból adatokat elérni, integrálni, és azok elemzése után diagramot megjeleníteni az eredményekről. Az LLM-ek megoldást jelenthetnek, hogy ilyen feladatokkal a végfelhasználóknak ne kellejen minden esetben szakértőkhöz (pl. adatelemzőkhöz, adattudósokhoz) fordulniuk, de ehhez a modelleknek rendelkezniük kell a kontextussal és a megfelelő eszközökkel az üzleti kérdés helyes és pontos megválaszolásához.

Az ilyen és ehhez hasonló egyedi és integrált vállalati feladatok automatizált megoldásához autonóm ágenseket fejlesztettünk. Az itt bemutatott ágensek LLM hívás láncolatokat használnak, melyeken belül az adott ágens képes egyéb eszközöket meghívni, amik kiegészíthetik a nyelvi modellek jelenlegi képességeit azáltal, hogy az ágens egy külső szolgáltatáshoz vagy egyedi függvényhez fordul a végső kimenet előállításához (az LLM meghívása előtt vagy után). Legegyszerűbb esetben így az ágensek képesek az LLM-ek gyermekbetegségeit orvosolni (mint pl. a matematikai jellegű vagy a betanítási adatok dátuma utánra vonatkozó kérdésekre válaszolni).

2.1 Adatbázis lekérdezés ágensek

Adatbázis lekérdezések automatizálása több megközelítéssel is lehetséges. Az ágens közvetlenül hozzáférhet az adatbázishoz és SQL nyelven írja meg a lekérdezést – nevezzük ezt 'SQL ágens'-nek. Az ágens könnyen eléri a táblákat és a séma definíciót, ezáltal egy emberi elemzőhöz hasonlóan a legfelső szintről indul és megvizsgálja az adott lekérdezéshez szükséges információkat.

Egy másik megközelítés esetén az adatokat beolvassuk egy Dataframe-be és a Python nyelv pandas könyvtárában megtalálható függvényeket használja fel az ágens a lekérdezéshez ('Dataframe ágens').

A harmadik lehetőség a ChatGPT Plus alatt elérhető Advanced Data Analysis (ADA) eszköz használata, mely nem csak adatfeldolgozásra és vizualizációra képes, hanem sokféle programozási feladat végrehajtására is. API-n keresztül nem elérhető jelenleg, csak több eszköz integrálásával közelíthető meg a funkcionalitása. A GPT-4 modell elérthető kódból, de ezen funkció nélkül. Ezért ezt manuális promptolással teszteljük a későbbi összehasonlítás érdekében.

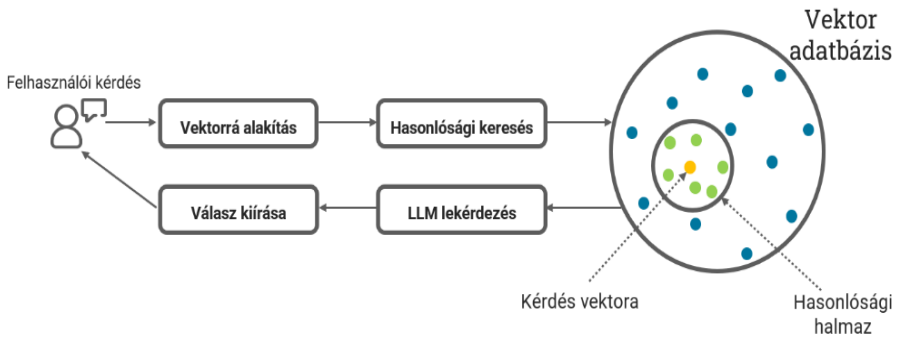
Mind a három módszerrel természetes nyelvi választ kapunk végeredményül. A chat felületen kódblokkokban, az ágenseknél pedig a verbose paraméter True-ra állításával láthatjuk a lekérdezéseket és a műveleteket is.

2.2 Dokumentum feldolgozás ágensek

A ma elérhető nagy nyelvi modelleket nagyméretű, az általános tudást lefedni próbáló korpuszon tanították be. Kiválóan válaszolnak a korpuszban szerepelő témákkal kapcsolatos kérdésekre, azonban az üzletmenethez kritikus, nem nyilvános vállalati dokumentációkban fellelhető információkhoz nem férnek hozzá. Erre kínál megoldást a Retriaval Augmented Generation (RAG) keretrendszer, amelyben az LLM egy külső

forrásból (jellemzően dokumentumokból) kinyeri a kontextushoz releváns információkat a lefuttatás során. Az általános tudás specifikus tudással egészül ki.

A dokumentum beolvasása után felbontjuk azt az LLM által kezelhető méretű és a jelentéstartalmat megőrző, egyenlő hosszúságú részekre, szükség szerint átfedéseket definiálva és fix (meta)adatokat hozzáadva a konzisztencia megőrzése miatt. A szövegrészekre bontást a modellek tokenlimitje teszi szükségessé, míg az átfedések biztosítják a szemantikai kontextus megőrzését a szövegrészek között. Esetünkben 1000 karakter egy egység 200-as átfedéssel. A szövegrészekből számszerű vektorreprezentációkat (ún. FAISS vektorokat – lásd később) hozunk létre az OpenAI beágyazás modelljének megfelelően, majd a keletkezett vektorokat egy (saját) vektoradatbázisban tároljuk a későbbi kereshetőség érdekében. A 9 fájl szövegéből 836 részlet készült. Amikor beérkezik egy kérdés az ágens felé, akkor ez is vektorizálásra kerül. A vektor adatbázisból a kérdés vektor és a tárolt vektorok közelségén, azaz szemantikai hasonlóságon alapuló kereséssel visszakapjuk az n legrelevánsabb szövegrészletet a kérdéshez. Utolsó lépésként az így kapott szövegrészeket a felhasználói kérdéshez hozzácsatoljuk és meghívjuk a nyelvi modellt a (lokális kontextussal) kiegészített bemenettel (lásd 1. ábra).



1. ábra: A dokumentum feldolgozás ágens működési vázlata (saját munka)

2.3 Diagramgenerálás ágens

A Dataframe ágens egyik előnye, hogy az alatt hívhatók vizualizációs szubrutinok is a pandas könyvtár felhasználásán túl. A diagramok elkészítéséhez két megközelítést alkalmaztunk. A matplolib könyvtár segítségével az ágens lefutási eredményében közvetlenül megrajzoltattuk a diagramot. Ez első adatvizualizációként megfelel, de nem a professzionális környezetben elvárt minőség. Ezért a második megközelítés egy Echarts JSON megíratása volt. Az Apache Echarts vállalati környezethez tervezett és dashboardok készítéséhez felhasznált diagramtípusokról nyújt részletesen szerkeszthető mintákat. Az Echarts diagramok elkészítéséhez viszont módosítanunk kell a kérdéseken: Echarts JSON struktúra megírására kérjük meg az ágenst, amit egy HTML fájlba illesztve állítjuk elő az eredményt.

2.4 Adatforrások

A teszt esetben felhasznált adatok az infojegyzet.hu Orszagok táblájára (https://infojegyzet.hu/webszerkesztes/mysql/orszagok/) épül, melyet kiegészítettünk saját készítésű Varosok és Emberek táblával (mely a felhasználói céges adatok integrálását hivatott szimulálni). Az Orszagok táblában megtalálható 194 ország fővárosa, földrajzi régiója, államformája, autójele, pénzneme, pénzjele, telefonhívószáma. A számszerű adatok a terület km^2-ben, a népesség 1000 főben megadva, a főváros népessége szintén 1000 főben, és az ország GDP-je USA dollárban Az Emberek tábla 15 kitalált személy adatait tartalmazza. A tábla létrehozásával a JOIN és GROUP BY típusú lekérdezések tesztelése volt a cél. A varos_id a város azonosítója, ahol a személy jelenleg él. A további adatok az életkor, az állampolgárság, a nem és az éves fizetés az ország pénznemében megadva. A Varosok tábla a személyekhez kapcsolódó városokat tartalmazza. Az orszag_kod az ország 3 betűből álló ISO szabvány szerinti kódja. A további adatok az országrész neve és a város népessége.

Kilenc további PDF dokumentum a (fenti, kiegészített) adatbázist lekérdező kérdésekhez felhasznált városok magyar nyelvű Wikipédia oldalait tartalmazza, amely lehetővé teszi többek között a földrajzi és éghajlati jellemzők, kulturális és turisztikai nevezetességek, történelmi és gazdasági tények beemelését a kérdésekbe. A dokumentumok szöveget, képeket és táblázatokat is tartalmaznak.

2.5 Felhasznált technológiák és paraméterezésük

A LangChain egy open source Python és Javascript nyelvekhez elérhető könyvtár, amelyet LLM-alapú alkalmazások fejlesztéséhez terveztek. A LangChain segítségével definiálható ágensek képesek eldönteni, hogy a kérdés megválaszolásához szükséges-e az eszköz meghívása vagy az egyszerű LLM kimenet is elégséges a feladathoz.

A dokumentumok vektorizálását és kereshetőségét a Facebook AI Similarity Search (FAISS) Python könyvtárral oldottuk meg. Ha van egy vektorhalmazunk, ez a FAISS segítségével felindexelhető és a lekérdezés vektort (query vector) használva a leghasonlóbb vektorok nagy sebességgel kikereshetőek euklideszi távolságot vagy koszinusz hasonlóságot használva.

A Retrieval Augmented Generation (RAG) az LLM válaszok minőségét és megbízhatóságát javítja külső adatforrások csatolásával. A módszer biztosítja az általunk választott, megbízható forrást, továbbá a felhasználók hozzáférést nyernek a modell forrásaihoz, így ellenőrizhetővé válnak a kapott LLM válaszok.

Az Apache Echarts egy ingyenes web alapú adatvizualizációs könyvtár, amely interaktív és JSON séma alapján részletesen szerkeszthető diagramokat kínál. Az Echarts JSON egyben tartalmazza a komponenseket, stílusokat, adatokat és interakciókat, ezáltal könnyen validálható formátum.

Az Advanced Data Analysis (ADA) funkció a GPT-4 modell alatt a ChatGPT Plus és Enterprise verziókban elérhető olyan összetett megoldás, amely az adatbeolvasást, elemzést és vizualizációt kombinálja a chat felület fizető felhasználói számára.

Az ágensek esetében az OpenAI és ChatOpenAI paramétereket próbáltuk ki a GPT-3.5-Turbo-0613 és a GPT-4-0613 modellekkel, amik kifejezetten a függvényhívás támogatására lettek finomhangolva. A függvény leírása alapján felismeri, mikor érdemes

egy függvényt meghívni és milyen inputot kell átadni a sikeres futtatáshoz. A LangChain típusok közül az OpenAI Functions-t teszteltük, amely célja, hogy megbízhatóbban kezelje a függvényhívásokat a finomhangolt modellekkel. Minden esetben „0" temperature értéket alkalmaztunk a jobb reprodukálhatóság céljából. Az SQL ágens az SQLDatabaseToolkit eszközcsomagot alkalmazza, a Dataframe ágens a create_pandas_dataframe_agent() függvénnyel definiálható, egy vagy több df megadásával.

2.6 Az elkészült ágensek működése és tesztelésük folyamata

Először a két adatbázis lekérdező megoldást (a Dataframe ágenst és az SQL ágenst) teszteltük a fenti adatbázison. Az Orszagok adatbázishoz nyilvánosan elérhető kérdéssorból 12 egyre nehezedő kérdést választottunk ki (lásd 1. táblázat első oszlopa). Következő lépésként az egyéb dokumentumok beolvasását és a több forráson alapuló kérdéseket próbáltuk ki. Végül a diagramábrázolással kibővítve az adatvizualizációs lehetőségek hatékonyságát vizsgáltuk mindkét adatforrás (az adatbázis és a dokumentumok) együttes felhasználásával. Az adatbázis lekérdezésnél minimális előkészítés után is jó eredményt vártunk a két ágenstől a legújabb GPT modellek kódolási képességeit ismerve.

Mindkét ágens bemenetét kontextussal bővítettük a jobb teljesítmény eléréséhez. Általánosan definiálható segítség az oszlopok leírása, amely nélkül az adatok mértékegységei, számítási módszerei vagy a felvehető értéktartományuk hiányoznak az ágensek tudásából. A táblák közötti kapcsolatot is megfogalmaztuk, hiszen ez kritikus azok összekapcsolásához és közös értelmezéséhez. Szintén általánosan megadható néhány példa lekérdezés, ez az ún. few-shot learning megközelítés. Végül szükség esetén specifikus esetekre megfogalmazhatóak kontextusfüggő folyamatleírások és tanácsok.

3 Kapott eredmények

3.1 Adatbázis lekérdezések

Az Orszagok adatbázishoz elérhető kérdéssor egyre nehezedő kérdéseinek tesztelése sok érdekességet fedett fel a két módszerben. Az ágenseknek kezdetben csak a megadott adatbázis és a kérdés állt rendelkezésre. Problémába ütközéskor ezt kibővítettük az oszlopok tartalmának rövid leírásával, majd példa lekérdezésekkel. Az eredmények alapján elmondható, hogy a Dataframe ágens jobban teljesített az SQL ágensnél, de érdemes részletesen is megvizsgálni a hibákat (1. táblázat).

Két kérdés helyes megválaszolása után az SQL ágens helytelen választ adott a „*Melyik ország autójele TT?*" kérdésre, mivel az első kérdés mintájára az „orszag" oszlop alapján szűrt. Az egyes oszlopok leírásának megadásával már megtalálta a helyes választ. A Dataframe ágens más problémába futott: nem találta a TT autójelet, mivel egy szóköz szerepelt a string végén. Az extra karakter az SQL ágens lekérdezésének eredményében is megjelenik, ennek ellenére adattisztítás nélkül is megtalálta. Ugyanakkor adattisztítás után a Dataframe ágens teljesített jobban: a kontextus megadása nélkül is

teljesítette a feladatot. Az SQL ágensnek már a 3. kérdéstől, míg a Dataframe ágensnek a 7. kérdéstől volt szüksége az oszlopleírásokra.

1. Táblázat: Az ágensek és a GPT-4 Plus felület válaszai a tesztkérdésekre

Kérdések	Dataframe ágens	SQL ágens	GPT-4 Chat (ADA) prompt
Mi MADAGASZKÁR fővárosa?	Helyes	Helyes	Helyes
Melyik ország fővárosa OUAGADOUGOU?	Helyes	Helyes	Helyes
Melyik ország autójele a TT?	Adattisztítás után helyes	Oszlopok rövid leírása után helyes, adattisztítás nem szükséges	Jelezve, hogy a lekérdezés visszaad egy országot helyes: .str.strip()
Melyik ország pénzének jele az SGD?	Helyes	Oszlopok rövid leírása + 3 példa után helyes	Helyes
Melyik ország nemzetközi telefon-hívószáma a 61?	Helyes	Oszlopok rövid leírása + 3 példa után helyes	Helyes
Mekkora területű Monaco?	Kérdés kibővítése után helyes	Oszlopok rövid leírása + 3 példa után helyes	Helyes
Hányan laknak Máltán?	Oszlopok rövid leírása után helyes	ChatOpenAI paraméterrel, eddigi segítségekkel helyes	Helyes
Hány lakosa van a Földnek?	1000-rel szorzás lemarad, ha nem szerepel a kérdésben	ChatOpenAI paraméterrel, eddigi segítségekkel és a kérdés kibővítésével helyes	Helyes, a formázást leszámítva (ezer milliárd helyett)
Mennyi az országok területe összesen?	Oszlopok rövid leírása után helyes	ChatOpenAI paraméterrel, eddigi segítségekkel helyes	Helyes
Mennyi az országok átlagos népessége?	Oszlopok rövid leírása után helyes	ChatOpenAI paraméterrel, eddigi segítségekkel közel helyes (kerekítési hiba)	Helyes
Mennyi a Föld népsűrűsége?	1000-rel szorzás lemarad, ha nem szerepel a kérdésben	Csak pontos számítási definícióval helyes: „Összes ország népsűrűség = (összes népesség) / (összes terület)”	Helyes
Hány 1.000.000 km^2-nél nagyobb területű ország van?	Oszlopok rövid leírása után helyes	ChatOpenAI paraméterrel, eddigi segítségekkel helyes	Helyes

A 4. kérdéshez az SQL ágens által elsőre megírt lekérdezés magát a pénzjelet adta vissza. A pénzjel oszlop leírásának részletezése nem hozott javulást, ezért few-shot-learning megoldást alkalmaztunk. A 3 példa alapján helyes lekérdezés és válasz érkezett.

Komolyabb problémát jelentett a ragozott alak megjelenése a kérdésben. Az SQL ágens minden esetben a ragozott formát („Máltán”) írta az SQL lekérdezésbe és a few-

shot-learning megoldás sem segített. Mivel üres volt a lekérdezés eredménye, hallucinálva kitalált egy számot: 43 700. További teszteléssel kiderült, hogy nem konzisztens a válasz és a „*Ha üres a lekérdezés eredménye, akkor ne találj ki nem valós eredményt.*" mondattal bővítés sem oldotta meg a hallucinációt, ezért további kontextus leírások helyett a paraméterezés megváltoztatását próbáltuk ki. Végül a ChatOpenAI paraméter használata az OpenAI helyett vezetett helyes eredményre, így már jól kezelte a ragozást. A ZERO_SHOT_REACT paramétert is kipróbáltuk, de a kérdésben specifikálás nélkül csak az OPENAI_FUNCTIONS paraméterrel írt olyan lekérdezést, ami az ezres szorzást is elvégzi. Utóbbi viszont csak magát az SQL lekérdezést adja vissza, nem futtatja a kódot.

A „*Mennyi az országok területe összesen?*" kérdésnél mindkét ágens helyesen elvégezte a műveletet és az oszlopleírásban megadott km^2 mértékegységet is felhasználta.

A „*Mennyi a Föld népsűrűsége?*" kérdésnél az 1000 fővel felszorzásra mindkét ágenst újra figyelmeztetni kellett az oszlopleíráson túl, amely vonatkozó részletét figyelmen kívül hagyták. Az SQL ágens továbbá a „*népsűrűség = népesség / terület*" definícióra elrontotta a számítást, mert először osztott és utána összegzett: „*SUM(nepesseg/terulet)*". Csak az „*Összes ország népsűrűség = (összes népesség) / (összes terület)*" pontosítással jutott helyes eredményre.

A GPT-4 ADA esetén a chat felületen az első prompt-ban megadtuk az oszlopok rövid leírását, ez a segítség minden kérdésnél rendelkezésre állt. Továbbá a „*Te egy kiváló adatbázis szakértő vagy sok év tapasztalattal. Feladatod, hogy lekérdezéseket készíts SQL nyelven (PostgreSQL), amelyek visszaadják a következő kérdésekre az eredményt.*" feladatkör leírással zártuk a prompt-ot. Kisebb hibák fordultak elő, pl. az adatfájl újbóli feltöltésére volt szükség az utolsó kérdések előtt, valamint formázási hibák jelentek meg. A GPT-4 ADA az adattisztítási problémát képes volt korrigálni: a 4. kérdést elsőre elrontotta, de jelezve, hogy a lekérdezésnek van eredménye, a string végén található szóközt helyesen levágta.

Az információszerzés következő szintje amikor több táblát kell összekapcsolni. Ezért olyan kérdéseket tettünk fel, amelyek megválaszolásához legalább 2 táblát, de inkább mindhármat össze kellett kapcsolni (JOIN). A Varosok-Emberek kapcsolatot id, az Orszagok-Varosok kapcsolatot 3 karakteres országkód biztosítja. 6 különböző kérdést tettünk fel, pl. „*Hány magyar lakik külföldön?*", amelyek közül a Dataframe ágens 2, az SQL ágens 5, a GPT-4 felület pedig 6 kérdésre válaszolt helyesen. Az SQL ágens az első 5 kérdést az oszlopok leírása alapján megválaszolta, a 6. kérdésnél azonban csak a teljes megoldási folyamat leírása segített. Ezzel szemben a Dataframe ágens már a 2. kérdésnél folyamatleírásra szorult. A 3. kérdés megválaszolásához még részletesebb segítséggel lehetett eljutni (országkód kikeresésének beszúrása a promptba). Az utolsó, legbonyolultabb kérdésnél végtelen ciklusba futott az ágens és többféle segítség megadásával sem lehetett helyes eredményt elérni. A GPT-4 chat felületen a 2. kérdés megválaszolásánál először hiba lépett fel, de automatikusan tudta önmagát korrigálni és végül helyes eredményre jutott.

3.2 Kombinált eset: Adatbázis és dokumentum

Nem minden esetben elegendő az adatbázis vizsgálata a felmerülő kérdések helyes megválaszolására. Az eddig bemutatott adatbázis lekérdezések kiegészíthetőek vektor

adatbázis keresésekkel, hogy vállalati dokumentumokkal augmentáljuk az ágensek tudását. Az oszlopok rövid leírását minden esetben megkapták az ágensek a 3 tábláról (lásd fent: Országok, Varosok, Emberek).

A kombinált esetekben a Dataframe ágens szerepelt jobban. Az SQL ágens segítség után sem volt képes végrehajtani a feladatot. Ha megadtuk a keresési sorrendet (először a dokumentumrészletek között keress), akkor is hosszú angol nyelvű választ kaptunk, ami szerint az adatbázisban nem található a kérdésre vonatkozó információ. A LangChain SQL ágens minden esetben az adatbázist priorizálta a többi elérhető forrással szemben. A Dataframe ágens részletes tesztelése során 3 típusú nehézséget tapasztaltunk: pdf beolvasási hibák, a forráshasználat sorrendje és a hasonlósági keresés hibái.

A táblázatok beolvasása problémát jelentett a PyPDF2 PdfReader használatával, csak az oszlopneveket olvasta be helyesen a gyakorlatban. Éppen ezért az uniós tagországok listáját a kérdés után az „Információk a válaszhoz:" blokkban megadtuk a szükséges esetekben. Egy helyen nyomdai hibába ütköztünk: „*A város hegygerincre épült.*" mondattal kezdődő feladatra a hasonlósági keresés nem találta meg a releváns részt a dokumentumokban. Ennek oka, hogy a beolvasás után a szövegrészletben a „*hegygerinc re*" formában szerepelt a szó, ami pontos egyezéssel sem kereshető ki.

Az első kombinált kérdésnél: „*Ki lakik Aquincum városában?*" az ágens azonnal az adatbázis városai közt kereste Aquincum várossal az egyezést, annak ellenére, hogy a hasonlósági keresés által visszaadott 4 szövegrészből 2 tartalmazta a szükséges információt. Ezen szövegrészleteket a kérdés utáni „*Információk a válaszhoz:*" blokkban adtuk meg és a sorrendre felhívva a figyelmet már sikeresen válaszolt. Ellenpéldaként említhető egy komplexebb feladat, a „*Valaki a Humboldt egyetemen tanít jó fizetéssel. Ki lehet az?*". Ez esetben az ágens részletes indoklással, helyes sorrendben hajtotta végre a műveleteket. A pénznemet is megfelelően azonosította az oszlopleírás alapján.

A hasonlósági keresés a „*Tisza és a Maros folyók találkozása*" esetben a vártnál rosszabbul teljesített. Nem állt fenn beolvasási hiba, ennek ellenére nem találta meg a szükséges szövegrészletet. Végül az ágens számára pontos egyezésen alapuló kereséssel biztosítottuk az információt. Az eredmények alapján a hasonlósági keresés önmagában nem mindig elegendő, a régebbi megoldással kiegészítés javítja az eredményességet. Érdemes megjegyezni, hogy a hasonlósági keresés általánosan alkalmazható, míg a fix keresés minden esetre speciális.

2. Táblázat: Dataframe ágens és GPT-4 Plus válaszai a kombinált kérdésekre

Kérdések	Dataframe ágens	GPT-4 Chat (ADA) prompt
Ki lakik Aquincum városban?	Helyes, másodszorra sorrend információval	Helyes, másodszorra sorrend információval
Ki lakik a Tisza és a Maros folyók találkozásánál?	Pontos egyezéssel helyes	Pontos egyezéssel helyes
Valaki a Humboldt Egyetemen tanít jó fizetéssel. Ki lehet az?	Helyes	Helyes
A város hegygerincre épült. Legmagasabb pontja 70 méterrel található a tengerszint felett. Mennyi az átlagfizetése az itt lakóknak?	Második segítség: csak releváns szövegrésszel helyes	Pontos egyezéssel helyes (Everton kulcsszó)

A kombinált eseteknél (2. táblázat) a fair tesztelés érdekében a GPT-4 ADA felületen megadtuk a kódban lefuttatott hasonlósági keresés kimenetét (a 4 leghasonlóbb szövegrészlet) a kérdések mellett. Elsőre a GPT-4 is az adatbázisban kereste az információt. Szükség esetén a fix keresést is lefuttattuk és hozzáadtuk a keresési kimenetekhez. Az utolsó kérdésnél a Dataframe ágens csak akkor végzett helyes számítást, ha a hasonlósági keresés 4 eredményét kivettük az információk közül és csak a fix keresés eredményével dolgozott. A GPT-4 ADA jól kezelte a hosszabb információblokkot és elsőre megtalálta Liverpool városát.

3.3 Komplex problémák megoldása: az esetek integrálása

Ebben a fejezetben olyan feladatok megoldását kérjük az ágensektől és a GPT-4 felülettől, amelyek magukba foglalják mind az adatok lekérdezését, a vektorizált dokumentumok értelmezését és a grafikonok létrehozását mindhárom forrás felhasználásával. Négy kérdés tettünk fel, melyek közül 3 alkalommal az adatbázisból és a dokumentumokból is ki kellett olvasnia az ágensnek a szükséges információkat. Egy esetben csak az adatbázis alapján készíttettünk vizualizációt, ami könnyebben kezelhető, azonban gyakoribb felhasználás lehet valós vállalati környezetben.

A „*Kérlek készíts kördiagramot Nagy-Britannia városainak népességéről.*” feladathoz generált matplotlib diagramon helyesen szerepelnek a százalékok, a kimenet szöveges részében viszont hibásak – összegük nem is adja ki a 100%-ot. Az Echarts elkészítéséhez átfogalmaztuk a feladatot: „*Kérlek készítsd el az ECHARTS JSON leírását egy kördiagramnak Nagy-Britannia városainak népességéről.*”. Mindkét diagramon jól leolvashatóak az adatok és illeszkedő magyar címmel látta el őket az ágens.

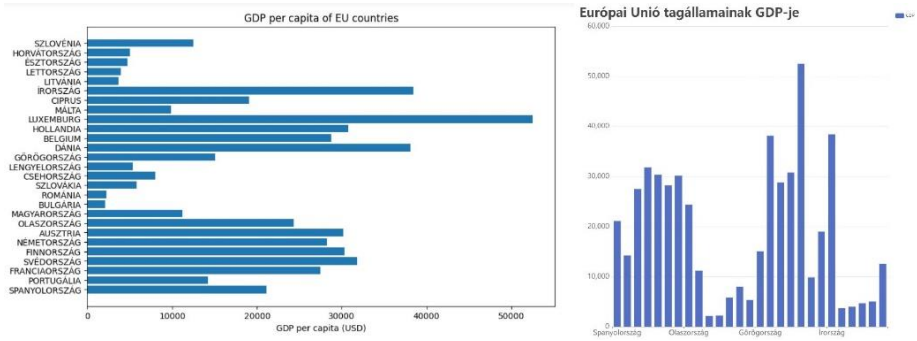
A „*Kérlek készíts oszlopdiagramot az Európai Unió országainak GDP-jéről.*” feladathoz már az EU országok kikeresése is szükséges volt a dokumentumokból, majd az országok szűrése ez alapján. Az ágens ezt egy új, df1_EU Dataframe létrehozásával oldotta meg. A fektetett oszlopdiagramról az összes országnév jól leolvasható és helyes adatokat ábrázol. A cím ez esetben angol a magyar kérdés ellenére, az x tengelyen viszont az USD valutát is feltüntette. Az Echarts diagramon a pontos értékek is megjelennek, ha az egeret egy adott oszlopra visszük. Hátrány, hogy csak így tudjuk leolvasni a pontos országot az országnevek elforgatásának hiánya miatt. A magyar nyelvű cím, beosztások, színek és interaktív funkciók, mint a pontos értékeket közlő tooltip jobb összhatást eredményeznek az előbbi diagramnál. További szerkesztés szükséges, ha dashboardon szeretnénk felhasználni.

A „*Kérlek készíts vonaldiagramot az Európai Unió országainak népsűrűségéről csökkenő sorrendben.*” feladatban mindkét módszer oszlopdiagramot hozott létre a kért vonaldiagram helyett. Az adatok csökkenő sorrendbe rendezése nem okozott problémát. Mind a matplotlib, mind az Echarts diagramon elmaradt az 1000-rel szorzás a számítási mód megadása nélkül.

Az utolsó feladat a „*Kérlek ábrázold a fizetéseket oszlopdiagramon a Humboldt egyetem városában lakók esetében.*” volt. Mindkét módszerrel helyes adatokat ábrázoló oszlopdiagramot kaptunk vissza.

A komplex feladatoknál jelentkeztek igazán az SQL ágens limitációi – általánosabb kódolási képesség nélkül nem tudta elkészíteni a diagramokat és a JSON formátumot

sem, ehhez újabb eszközökkel kellene felruházni. Egy eszközös ágensek esetén egyértelmű a Dataframe ágens szélesebb körű alkalmazhatósága és hatékonysága.



2. ábra: Európai Unió tagállamainak GDP-je: Matplotlib vs. Echarts (saját munka)

3.4 Összehasonlítás

Az adatbázis lekérdezés feladatban a kontextus megadása nyújtja a legnagyobb segítséget az ágenseknek. A legegyszerűbb kérdéseket leszámítva a legjobb modellnek is szüksége van további információkra az adatokról. A few-shot-learning módszer is hatékonynak bizonyult a pontosság növelésében. Amennyiben egy valós rendszerben számos korábbi lekérdezés áll rendelkezésre, érdemes ezeket felhasználni az ágens futtatásakor. Egy számítási eszköz bevezetésének hasznosságára utal, hogy az SQL ágens egy esetben a nem létező „Multiply" eszközt próbálta meghívni. A LangChain könyvtár használatakor a paraméterezés is jelentősen befolyásolhatja az eredményt.

Bár mind az OpenAI, mind a ChatOpenAI paraméter használatakor megadható a pontos nyelvi modell, amit az ágens használ, mégis eltéresek tapasztalhatók e kettő között. Az egyszerű LLM interface-el ellentétben, ami nyers szöveget kezel bemenetként és kimenetként, a chat modell interface üzeneteket alkalmaz, mint az AIMessage, HumanMessage és SystemMessage.

A dokumentum-feldolgozás során tapasztalt PDF beolvasási hibák elkerülésére érdemes más beolvasási módszereket is kipróbálni a PyPDF2 helyett, mint pl. a táblázatok beolvasását támogató Camelot vagy PyMuPDF.További fejlesztésként érdemes lehet matematikai eszközökkel kibővíteni az ágenst a szorzási és kerekítési hibák elkerülése céljából. A diagramkészítés során akkor tapasztalhatunk súlyos hibát, amikor az adatok lekérdezése helytelen. Ez kiemeli az adatbázis lekérdezési folyamat és ehhez kapcsolódó információk kritikusságát, hiszen az maga a diagramábrázolás folyamat első szükséges lépése egyben.

Az SQL ágens a vártnál gyengébben teljesített az egy táblás adatbázis lekérdezések során, viszont több táblát igénylő feladatokon jól teljesített. A Dataframe ágens az SQL ágenshez képest jobb eredményt ért el egy táblás lekérdezéseken, az adattáblák összekapcsolásához azonban részletes segítséget igényelt és egy esetben végtelen ciklusba futottunk. A GPT-4 ADA megoldás nyújtotta a legjobb válaszokat, viszont ez a legnehezebben beilleszthető létező folyamatokba.

A kombinált feladatokon az SQL ágens hibás válaszokat adott, a Dataframe ágens és a GPT-4 ADA pedig közel azonos eredményeket ért el. A keresési sorrendet segítséggel helyesen kezelték. A GPT-4 ADA jobban teljesített az ágensnél több megadott keresési szövegrész esetén. A komplex feladatok tesztelése mutatta meg a Dataframe ágens általános Python programozási képességeit a diagramok létrehozása által. Sokkal többet tud a Dataframe kezelésnél, mindössze erre a feladatra lett kiélezve az eszköz.

Pragmatikus szemszögből érdemes megemlíteni, hogy a GPT-4 ADA megoldás ChatGPT Plus előfizetést, tehát havi 20 dollárt követel. Az API-alapú ágensek bemeneti és kimeneti tokenek szerint vannak árazva. A gpt-3.5-turbo-0613 0,0015 dollár 1000 bementi tokenre és 0,002 dollár 1000 kimeneti tokenre. A gpt-4-0613 0,03 és 0,06 dollár azonos kategóriákban. Az OpenAI november 6-án bejelentette az új GPT-4-Turbo és a GPT-3.5-Turbo használati árak bemenet esetén harmadára, kimenet estén felére csökkentését és az open source modellek által létrehozott versenyhelyzet miatt várhatóan a jövőben is folytatódik ez a tendencia (OpenAI, 2023). Az API-alapú modellekre épülő ágensek teljesítménye messze felülmúlja az open source modell ágensek eredményeit és a GPT-4 közülük is kiemelkedik – ami hasonló az irodalomban (pl. Liu és mtsai, 2023) található eredményekhez.

4 Pragmatikus tanulságok

Az integrált megoldás megvalósításához a Dataframe ágenst ajánljuk. Az adattáblák előkészítésével illetve az összekapcsolások elvégzésével az ágensnek történő átadás előtt javítható a rendszer hatékonysága a komplex, sok táblás adatbázisokon. Egyes feladatok megoldásához a kontextusfüggő segítségek is elengedhetetlenek, tehát az LLM-ek általános tudása még nem elegendő a biztos eredményekhez. Éppen ezért érdemes minél több segítséget adni a rendszernek és az ágenst testre szabni, felkészíteni egy kiválasztott feladathalmazra. A kontextusfüggő segítségeket érdemes több esethez újrahasznosíthatóan megírni, amennyiben lehetséges.

Mivel az ágens a vállalati rendszerbe integrálható, ezért helytelen válasz esetén ellenőrzéseket is érdemes megvalósítani, amelyek visszaküldik az ágensnek a kérdést és a hibát. A lekérdezésekre adott válaszoknak mindig az eszköz meghívásán kell alapulniuk. Ebből következik, hogy az ágens által megírt kód ellenőrzésével kiszűrhető, mikor alapul a kódlefutás eredményén és mikor tényleges lefutás nélküli, hallucinált eredményen a válasz. Amennyiben az SQL ágens jobban teljesített volna, ezen ágens alkalmazásakor a táblákat előkészítő lépés nem lenne kritikus. A magas hibaarány, a kombinált és komplex feladatokra alkalmatlanság, továbbá a (csak itt előforduló) angol nyelvű válaszok miatt nem ajánljuk gyakorlati használatra.

A GPT-4 ADA felület azonos szinten vagy jobban teljesített a Dataframe ágensnél a tesztelt feladatokban. Azonban a testre szabás, valamint a kódlefutás és válasz ellenőrzés nem alkalmazható e felület használatával. Ez a fajta bővíthetőség az API-n keresztül elérhető megoldások fő előnye.

Bibliográfia

Ayinde, L., Wibowo, M. P., Ravuri, B., & Emdad, F. B.: ChatGPT as an important tool in organizational management: A review of the literature. Business Information Review 40(3), 137-149 (2023).

Gillioz, A., Casas, J., Mugellini, E., and Abou Khaled, O.: Overview of the Transformer-based Models for NLP Tasks. In 15th Conference on Computer Science and Information Systems FedCSIS, pp. 179-183. IEEE (2020).

Knap, Á., Dömsödi L B., Mogyorósi P., Szigeti P., Tóth A., Virág A. és Kmetty Z.: Magyar nyelvű időjárásjelentések nyelvi modell alapú automatizált generálása. XIX. Magyar Számítógépes Nyelvészeti Konferencia Szeged, 2023. január 26–27. pp. 275–287. Szegedi Tudományegyetem (2023).

Liu X, Yu H, Zhang H, Xu Y, Lei X, Lai H, Gu Y, Ding H, Men K, Yang K: Agentbench: Evaluating LLMs as agents. arXiv preprint arXiv:230803688, https://doi.org/10.48550/arXiv.2308.03688 (2023).

OpenAI Blog: Introducing ChatGPT. https://openai.com/blog/chatgpt (2022).

OpenAI Blog: New models and developer products announced at DevDay, https://openai.com/blog/new-models-and-developer-products-announced-at-devday (2023).

Yang, Z. Gy., Dodé, R., Ferenczi, G., Héja E., Jelencsik-Mátyus, K., Kőrös, Á., Laki, L. J., Ligeti-Nagy, N., Vadász, N., Váradi, T.: Jönnek a nagyok! BERT-Large, GPT-2 és GPT-3 nyelvmodellek magyar nyelvre. In XIX. Magyar Számítógépes Nyelvészeti Konferencia Szeged, 2023. január 26–27. pp. 247–262. Szegedi Tudományegyetem (2023).