СОФИЙСКИ УНИВЕРСИТЕТ СВЕТИ КЛИМЕНТ ОХРИДСКИ

СТОПАНСКИ ФАКУЛТЕТ

АВТОРЕФЕРАТ

Приложение на методите на машинното самообучение за решаване на бизнес задачи

Многоагентни автоматизирани системи за решаване на сложни задачи в бази данни, с помощта на големи езикови модели.

Докторант

Сергей Сергеев Вичев

Научен ръководител

Доц. д-р Ангел Ангелов Марчев, мл.

СОФИЯ, 2025

УВОД

Големите езикови модели (LLM) се превърнаха в основен катализатор за създаването на автоматизирани системи, които навлизат все по-дълбоко в различни аспекти на бизнес анализа. Паралелно с това, бързото развитие на методите за извличане и обработка на данни доведе до интеграцията на LLM с бази данни, предлагайки нови подходи за решаване на сложни проблеми.

Многоагентните архитектури, при които няколко LLM агенти си взаимодействат, се очертават като ключ към по-гъвкави и интелигентни решения. Развитието на такива системи предлага възможност за създаване на платформи, които не само извеждат информация от различни бази данни, но и аргументират, координират действия и взимат решения за следващи действия, нужни за решаването на задачата. Това поставя въпроси, свързани с методологията на изграждане, оценка и подходящо използване на подобни системи.

В контекста на съвременните организации от различни сектори (финанси, търговия, производство) автоматизацията на задълбочени бизнес анализи става все по-критична за навременни решения. Настоящата дисертация представя цялостен подход за изграждане на многоагентни LLM-базирани системи, които да осигурят ефективно търсене, анализ и аргументирано решаване на бизнес казуси. Поставя се акцент върху разработването на методи за специализиращо обучение и оценка на векторизиращи модели в Retrieval-Augmented-Generation среда, която работи с бази данни.

Основна цел на дисертацията е да бъде разработена и приложена методика за създаване на многоагентни автоматизирани системи, базирани на големи езикови модели и интегрирани с бази данни и позволяват ефективно извличане на знания, аргументирано решаване на задълбочени бизнес проблеми и оценка на критични компоненти като векторизиращи модели.

Тази цел може да бъде дезагрегирана в следните **задачи на дисертацията**:

1. Да се анализират съществуващите концепции за LLM, многоагентни системи и методи за векторно представяне, включително техните силни и слаби страни при обработка на езикови и таблични данни.

2. Да се формулират и обосноват методически принципи за разработване на автоматизирани системи, включително архитектура, функционални компоненти и механизми за взаимодействие с бази данни (RAG подход).

3. Да се създадат и тестват различни LLM агенти (единични и многоагентни), които извличат, анализират и обобщават информация от SQL бази данни с помощта на ReAct и подобни рамки.

4. Да се изследват и приложат техники за специализиращо обучение на модели за вграждане с цел подобряване на извличането на релевантна информация от специфични данни за конкретна област на приложение.

5. Да се проведат симулации и експерименти, които проверяват точността и ефективността на предложените решения, като бъдат анализирани както качествените, така и количествените резултати.

Предмет на изследване са методите, архитектурите и компонентите, свързани с разработката и оценката на многоагентни автоматизирани системи за извличане и аргументирано решаване на бизнес задачи, базирани на LLM и интегрирани с бази данни, както и процедурата за създаване на такава система.

Обекти на изследване са разнообразните бази данни, съставени от реални масиви от бизнес или оперативна информация. Анализът на съдържанието в базите данни обхваща както самите таблични структури (таблици, релации, връзки), така и съпътстващата информация (метаданни, индекси и т.н.), които заедно обуславят адекватното извличане на знания.

Фокусът е върху дълбоки невронни мрежи и трансформър архитектури, като се прилагат концепции от машинното обучение за специализиращо обучение на векторни представяния. Ограниченията включват и големината и качеството на обучителните данни, възможните пристрастия (bias) в езиковите модели и техническите ресурси, необходими за обработка на мащабни бази данни. Същевременно, фокусът върху многоагентни LLM системи предполага анализ на сложни взаимодействия, което налага тестване в контролирана среда и може да ограничи пълната приложимост в производствени условия, ако липсват подходящи ресурси и експертен екип. Въпреки че взаимодействието между човека и разработените автоматизирани системи е от принципно съществено значение, изследването не си поставя за цел да анализира детайлно човешките аспекти на комуникацията и въздействието на човека върху резултатите и те са извън обхвата. Фокусът е върху автоматизираната страна на процеса.

Дисертацията е структурирана в три основни глави и заключителна част. Първата глава разглежда теоретичните основи на LLM, многоагентните подходи и методите за семантично търсене в бази данни. Втората глава представя методиката за изграждане на автоматизираната система, описвайки детайлно архитектурата ѝ и основните ѝ компоненти (LLM агенти, RAG модул, методи за оценка). Третата глава е посветена на симулации и практически експерименти, включително специализиращо обучение на модели за вграждане,

резултати и анализ на ефективността. Накрая, в заключението са синтезирани основните приноси, изводи и препоръки за бъдещо развитие.

В настоящата разработка са включени три отделни изследователски насоки, свързани тематично, както в общата теоретичната обосновка, така и в разработената архитектура в раздел 9., където многоагентните системи и векторизиращия модел са включени като съставни подблокове. Емпиричните изследвания, поместени в Глава III, са относително обособени. За по-добра яснота на читателя е представена в Таб. 1, където е представено разпределението на текстовете относно отделните изследователски насоки по отделните аспекти - специфични теоретични, методически и приложни, а както и връзката с авторовите публикации и претенции за научни приноси, според номерацията им в приложение.

Таб. 1 Разпределение текстовете относно изследователските насоки по отделните аспекти

	Автоматизирано извличане на знания от база данни, с помощта на LLM (ReAct с памет и агенти)	Архитектура на автоматизирана система за решаване на сложни бизнес задачи с бази данни (RAGSQL)	Специализиращо обучение на векторизиращ модел (Embedding model fine-tuning)
Специфичните теоретични аспекти са разгледани в раздели:	3.3., 3.4., 4.1., 5.	4.1., 4.2., 5.	4.2, 4.3, 5.
Методическите аспекти са разгледани в раздели:	7., 10.1	9.	8., 10.2
Приложните аспекти са разгледани в разедели:	13.	-	14.
Авторски публикации по изследователската насока:	Vichev, Marchev (2025), Vichev (2024)	-	Vichev, Marchev (2024)
Претенции за научни приноси по изследователската насока:	Приноси 3, 5	Принос 1	Приноси 2, 4, 5

За целите на дисертацията са разработени следните репозиториуми:

- React SQL програмен код за решаване на 3те варианта: https://github.com/sergeyvi4ev/react-sql-agents
- RAGSQL програмен код за провеждане на експериментите: https://github.com/sergeyvi4ev/ragsql-paper
- RAGSQL-text модел и данни: https://huggingface.co/sergeyvi4ev/all-MiniLM-RAGSQL-text https://huggingface.co/datasets/sergeyvi4ev/questions_to_sql_triplets
- RAGSQL-code модел и данни: https://huggingface.co/sergeyvi4ev/all-MiniLM-RAGSQL-text https://huggingface.co/datasets/sergeyvi4ev/questions_to_evidence_triplets

Авторът изказва дълбока благодарност към семейството за тяхната постоянна подкрепа и окуражаване през целия период на изследването. Особена признателност изразявам към научния ръководител доц. д-р Ангел Марчев за безценните насоки, професионалната подкрепа и вдъхновението при разработването на настоящата дисертация. Специална признателност изказвам към рецензентите проф. д.ик.н. Антон Герунов и доц. д-р Боряна Пелова за задълбочената оценка и ценните насоки. Благодарност отправям също към ръководителя на катедрата доц. Николай Нетов, всички колеги от катедрата за оказаното съдействие и ценни препоръки, както и към останалите членове на научното жури, проф. д-р Станимир Кабаиванов, доц. д-р Александър Ефремов и доц. д-р Николай Драгомиров за проявената ангажираност и съдействие в процеса на оценяване на дисертационния труд.

Предварителни бележки по оформянето

Номериране на главите: дисертацията е структурирана в три основни глави, обозначени с римски цифри. Всяка глава е съставена така, че да отговаря на ключовите елементи на цялостно научно изследване – теоретична, методическа и емпирична част. В същото време всяка от тях може да се разглежда като самостоятелен научен текст. Поради тази причина в началото и в края на всяка глава са включени специални раздели, озаглавени съответно: „Задачи на глава…” и „Изводи от глава…”.

Оформяне на илюстративния материал: в дисертацията присъства илюстративен материал под разнообразни форми - блок-схеми, диаграми, екранни отпечатки, таблици, формули, полета с програмен код. За всеки от тези видове, се използва отделна единна система за номерация, с изключение на програмните кодове, които са без номерация. Всички авторски блок-схеми са разработени с помощта на draw.io, а диаграмите, екранните отпечатки и таблиците са разработени със стандартните инструменти за обработка на текст. Всички илюстрации са изработени без помощта на изкуствен интелект.

Използвани абревиатури, чуждици и означения:

LLM - Large Language Models (Големи езикови модели)

RAG - Retrieval Augmented Generation (генерация с обогатено извличане)

CoT - Chain of Thought (верига от мисли)

БД - база данни

SQL - Structured Query Language (структуриран език за заявки)

RAGSQL (авторско предложение за понятие) - Retrieval Augmented Generation for Structured Query Language (генерация с обогатено извличане за структуриран език за заявки).

AI - Artificial Intelligence (Изкуствен интелект)

ИИ - Изкуствен интелект

GenAI - Generative Artificial Intelligence (Генеративен Изкуствен Интелект)

RL - Reinforcement Learning (Обучение чрез подсилване)

RLHF - Reinforcement Learning from Human Feedback (Обучение чрез подсилване с обратна връзка от хора)

ANN - Artificial Neural Network (Невронна мрежа)

RNN - Recurrent Neural Network (Рекурентна невронна мрежа)

CNN - Convolutional Neural Network (Конволюционна невронна мрежа)

NLP - Natural Language Processing (Обработка на естествен език)

nDCG - Normalized Discounted Cumulative Gain (Нормализирана дисконтирана кумулативна значимост)

IR - Information Retrieval (извличане на информация)

CBOW - Continuous Bag of Words (непрекъсната торба от думи)

LSTM - Long Short-Term Memory (Невронна мрежа с дълга краткосрочна памет)

BERT - Bidirectional Encoder Representations from Transformers (двупосочното кодиращо представяне от трансформъри)

Temperature - температурен коефициент за ограничаване на халюцинации при LLM.

Loss function - функция на загуба.

Chunk - фрагмент.

Benchmark - еталон за сравнение, массиви от анотирани данни от бизнес естество и стандартизирани метрики за сравнение.

Съдържание на дисертационния труд

СЪДЪРЖАНИЕ
Предварителни бележки по оформянето
УВОД
ГЛАВА I. ТЕОРЕТИЧНИ ОСНОВИ
1. ЗАДАЧИ НА ГЛАВА I
2. СЪЩЕСТВУВАЩИ ОСНОВНИ ПОНЯТИЯ
3. ГОЛЕМИ ЕЗИКОВИ МОДЕЛИ
3.1. Основополагащи концепции
3.1.1. Обработване на естествен език (NLP)
3.1.2. Невронна мрежа с дълга краткосрочна памет (LSTM)
3.1.3. Архитектура на трансформъри и Attention механизъм
3.1.4. Модели за представяне
3.1.5. Архитектура на съвременните езикови модели
3.2. Преглед на LLM
3.2.1. Приложения на LLM
3.2.2. Развитие на LLM
3.2.3. LLM Класификация
3.2.4. Ограничения на LLM
3.2.5. Проблемът на съгласуването на ИИ
3.3. Prompt Engineering (in-context обучение)
3.3.1. Обучение с примери “zero-shot”
3.3.2. Обучение с примери “few-shot”
3.4. Способност за аргументация на LLM-и
3.4.1. Верига от мисли (Chain-of-thoughts)
3.4.2. Дърво от мисли (Tree-of-thoughts)
4. LLM ПРИЛОЖЕНИЯ В ПОЛЗА НА БИЗНЕСА
4.1. Мултимодалност
4.1.1. Рамка за аргументация и действие: ReAct
4.1.2. Генериране на код и текст-към-SQL приложения
4.1.3. Многоагентни системи
4.2. Генерация с обогатено извличане на информация (RAG)
4.2.1. Семантично търсене и RAG
4.2.2. Предизвикателства на RAG
4.2.3. Оценка на RAG системи и модели за вграждане
4.3. Специализиращо обучение на LLM-и
4.3.1. Основи на специализиращото обучение
4.3.2. Исторически контекст и еволюция
4.3.3. Методически подходи за специализиращо обучение на генеративни LLM
4.3.4. Методически подходи за специализиращо обучение на модели за представяне
4.3.5. Основни функции на загуба при модели за вграждане
4.3.6. Приложение на специализиращо обучение
5. КОНЦЕПТУАЛНА ПОСТАНОВКА НА НАСТОЯЩОТО ИЗСЛЕДВАНЕ
ГЛАВА II: МЕТОДИКА НА ИЗГРАЖДАНЕ НА АВТОМАТИЗИРАНА СИСТЕМА ЗА РЕШАВАНЕ НА ЗАДЪЛБОЧЕНИ БИЗНЕС ПРОБЛЕМИ ВЪРХУ ИНФОРМАЦИЯ ОТ БАЗА ДАННИ
6. ЗАДАЧИ НА ГЛАВА II
7. МЕТОДИ ЗА ПРОГНОЗИРАНЕ С LLM
7.1. Методи за генерация на текст-към-SQL
7.2. Методи за аргументация с LLM

7.2.1. ReAct агент без и с контекстуална памет

7.2.2. Многоагентен подход

8. МЕТОДИ ЗА ПОДОБРЕНИЕ И ОЦЕНКА НА ВГРАЖДАЩИ МОДЕЛИ ЗА RAG СИСТЕМИ С SQL БАЗИ ОТ ЗНАНИЯ.

8.1. Модели за вграждане (embedding models)

8.2. Алгоритъм за оценка на изваждане на информация

8.3. Метрики за оценка

8.4. Използвани функции за загуба при специализиращо обучение

8.5. Модели за вграждане, специфични за индустрията

9. АРХИТЕКТУРА И КЛЮЧОВИ КОМПОНЕНТИ НА АВТОМАТИЗИРАНА СИСТЕМА ЗА РЕШАВАНЕ НА СЛОЖНИ ЗАДАЧИ С БАЗИ ДАННИ (RAGSQL)

9.1. Обща схема

9.2. Блок A “Потребител”

9.3. Блок B “Потребителски интерфейс”

9.4. Блок C “LLM Агенти”

9.4.1. Описание на блока

9.4.2. Структура на блока - с един LLM агент

9.4.3. Структура на блока - многоагентна подсистема

9.5. Блок D: “Процесинг модул БД”

9.5.1. Описание на блока

9.5.2. Структура на блока

9.6. Блок E “RAG модул”

9.6.1. Описание на блока

9.6.2. Структура на блок “RAG модул”

9.7. Блок F “Модел на вграждане”

9.7.1. Описание на блока

9.7.2. Структура на блока

9.8. Блок G Трансформации върху LLM

9.8.1. Описание на блока

9.8.2. Структура на блока

10. АЛГОРИТЪМ ЗА РЕАЛИЗИРАНЕ НА СИСТЕМАТА

10.1. Програмен код за системата за извличане на знания от БД а аргументация

10.1.1. Общи модули и компоненти

10.1.2. ReAct LLM агент без памет

10.1.3. ReAct LLM агент с памет

10.1.4. Многоагентна структура

10.2. Програмен код за критичните компоненти на RAGSQL система

10.2.1. Специализиращо обучение

10.2.2. Изчисляване на метрики за оценка

11. ИЗВОДИ ОТ ГЛАВА II

ГЛАВА III: СИМУЛАЦИИ И ЕКСПЕРИМЕНТИ С АВТОМАТИЗИРАНИ СИСТЕМИ ЗА РЕШЕНИЯ ОТ БАЗИ ДАННИ

12. ЗАДАЧИ НА ГЛАВА III

13. ПРИНЦИПНО ПРИЛОЖЕНИЕ НА СИСТЕМА ЗА ИЗВЛИЧАНЕ НА ЗНАНИЯ ОТ БАЗИ ДАННИ С АРГУМЕНТАЦИЯ

13.1. Обща постановка на заданието

13.2. ReAct текст-към-SQL LLM агент без памет

13.2.1. Постановка на експеримента

13.2.2. Наблюдения от експеримента

13.3. ReAct с контекстуална памет

13.3.1. Постановка на експериментите

13.3.2. Наблюдения от експериментите

13.4. React с многоагентна система
13.4.1. Постановка на експеримента
13.4.2. Наблюдения от експеримента
13.5. Обобщение на изследването
14. ИЗСЛЕДВАНЕ НА КРИТИЧНИТЕ КОМПОНЕНТИ НА RAGSQL СИСТЕМА.
14.1. Обща постановка на заданието
14.2. Данни
14.3. Специализиращо обучение на векторизиращ модел
14.4. Анализ на резултатите
14.4.1. Първоначални резултати без специализиращо обучение
14.4.2. Резултати след специализиращото обучение
14.5. Обобщение на изследването
15. ИЗВОДИ ОТ ГЛАВА III
ЗАКЛЮЧЕНИЕ
ПРЕТЕНЦИИ ЗА НАУЧНИ ПРИНОСИ
ИНФОРМАЦИОННИ ИЗТОЧНИЦИ
ПУБЛИКАЦИИ НА АВТОРА

ГЛАВА I. ТЕОРЕТИЧНИ ОСНОВИ

СЪЩЕСТВУВАЩИ ОСНОВНИ ПОНЯТИЯ

Голям Езиков Модел (Large Language Model).

В настоящия текст се използва следната работна дефиниция: Големите езикови модели са усъвършенствани невронни мрежи, обикновено базирани на трансформърни архитектури, предварително обучени върху обширни корпуси от текст, което им позволява да разбират и генерират текст, наподобяващ човешкия, в различни задачи.

Модел за вграждане (embedding model).

В този текст се използва следното работно определение: моделът за вграждане е модел, който трансформира сложни данни, като текст или изображения, в плътни, нискоразмерни числови представяния. Тези векторни представяния улавят съществени характеристики и закономерности в данните, улеснявайки изпълнението на различни задачи като класификация, клъстеризация и препоръки.

Задание (prompt)

В този текст се използва следната работна дефиниция: Задание в NLP е текстов вход на естествен език, предоставен на ИИ модел, който служи като начална инструкция или запитване, насочващо последващия отговор или действие на модела.

Аргументация (reasoning)

Настоящият текст възприема следната работна дефиниция: Аргументация в големите езикови модели се разбира, като способността на модела да обработва и генерира последователности от междинни стъпки, които водят до смислени и логически последователни заключения, като обхваща различни типове аргументация, включително дедуктивна, индуктивна, абдуктивна и аналогична.

Специализиращо обучение (fine-tuning).

В този текст използваме следното работно определение: специализиращото обучение е техника за трансферно учене, при която предварително обучен модел (обикновено върху широка база данни или чрез самообучение) се дообучава върху по-малък, специфичен за задачата набор от данни (обикновено с надзор), за да се адаптират параметрите на модела към новата задача. В контекста на NLP и LLMs, специализиращото обучение означава започване от предварително обучените тегла на голям езиков модел и продължаване на обучението върху текстови данни, специфични за дадена област или задача, така че моделът да усъвършенства своите генеративни или аналитични способности спрямо целевата област.

ГОЛЕМИ ЕЗИКОВИ МОДЕЛИ

Големи езикови модели (LLM) - специализирани AI модели, обучени върху огромни текстови корпуси. Тези модели, реализирани във вид на Transformer архитектура, показват забележителни възможности за управление на сложни езикови задачи, извършване на основна аритметика и дори завършване на код на Python или SQL. Съществена е появата на новаторски методологии като верига от мисли (CoT), аргументация и действие (ReAct) и дърво на мисли (ToT), които са предназначени да подобрят разсъжденията и способностите за вземане на решения на LLM. Вниманието е насочено към разнообразните приложения и непрекъснатото развитие на LLMs, илюстрирайки тяхното променящо въздействие в различни области.

Обработване на естествен език (NLP)

Основен тласък в развитието дава алгоритъмът Word2Vec (Mikolov et al., 2013a, 2013b), който въвежда метод за изучаване на представяния на думи в непрекъснато векторно пространство, което позволява на думи с подобни значения да бъдат по-близо една до друга в това пространство. Тези вграждания на думи трансформират NLP, като предоставят начин за цифрово кодиране на семантично значение.

Успехът на Word2Vec демонстрира силата на вграждането на думи при улавянето на езикови структури, проправяйки пътя за по-сложни техники за дълбоко обучение.

Основно предизвикателство пред NLP е езиковата многозначност – една и съща дума или фраза може да има различни значения в зависимост от контекста. Също така човешкият език се характеризира с гъвкавост и разнообразие в изразяването на една и съща идея. Затова NLP системите трябва да се справят с задачи като разпознаване и премахване на двусмислие (word sense disambiguation), разрешаване на анафори (установяване на референцията на местоименията), както и разбиране на имплицитна информация, която не е изрично изказана. Решаването на тези проблеми традиционно изисква комбиниране на лингвистични знания и статистически методи (Cambria and White, 2014).

Архитектура на трансформъри и Attention механизъм

Следващият голям пробив в NLP идва с представянето на Transformer архитектурата (Vaswani et al., 2017) в публикация под име “Attention is all you need”. Тази архитектура революционизира NLP, като заменя рекурентните невронни мрежи (RNN) с механизми за самовнимание, което позволява паралелно обучение и по-добро разбиране на контекста.

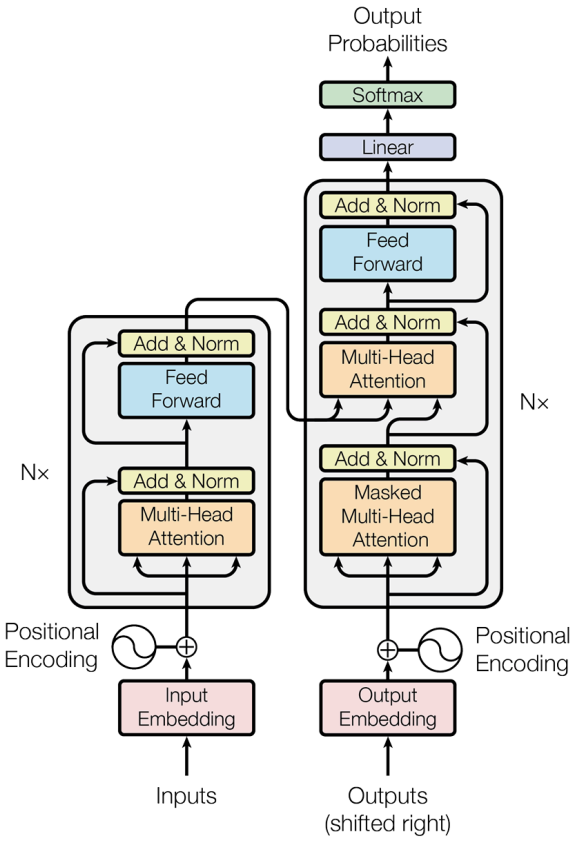
Трансформърите се състоят от кодиращи и декодиращи блокове, където:

- Кодерът обработва входни последователности и генерира контекстуализирани представяния на думи.
- Декодерът генерира изходни последователности въз основа на тези представяния.

Основната иновация на трансформърите е механизмът за самонасочване (self-attention), който позволява на модела да претегля важността на различните думи в изречението една спрямо друга, подобрявайки зависимостите на дълги разстояния.

В сравнение с RNN, които обработват текст последователно, трансформърите могат да обработват всички думи едновременно, което ги прави по-ефективни и мащабируеми. Тази иновация доведе до най-съвременна производителност в много NLP задачи, включително превод, генериране на текст и анализ на настроението.

Фиг. 1. Архитектурата на трансформър.



Източник: (Vaswani et al., 2017)

Механизмът на вниманието (attention) представлява техника, при която моделът оценява важността на различни части от входната последователност, когато предсказва всеки елемент от изхода (Bahdanau et al., 2015). В контекста на превода, например, вниманието

позволява на декодера динамично да „насочи вниманието си“ към онези думи от изходното изречение, които са най-релевантни за текущо генерираната дума.

Трансформър (Фиг. 1) е дълбока невронна архитектура, която изцяло се основава на механизми за самовнимание (self-attention) и елиминира рекурентните зависимости, присъщи на RNN (Vaswani et al., 2017). Класическият трансформър модел (напр. за машинен превод) включва кодер и декодер, изградени от многократно повтарящи се слоеве със самовнимание и позиционни кодировки (positional encoding). Всеки трансформър слой съдържа няколко паралелни „глави“ за внимание (multi-head attention), позволяващи на модела да извлече различни типове отношения (синтактични, семантични и др.) между думите, и последователност от линейни слоеве с активация (feed-forward мрежа), които обработват комбинираната информация.

Основно предимство на трансформърите е способността им да се паралелизират при обучение – всички думи в изречението могат да бъдат обработени едновременно, за разлика от RNN, които по природа работят последователно. Това прави възможно обучението на много по-големи модели върху огромни корпуси от данни, което пряко допринася за впечатляващите резултати на съвременните езикови модели (Vaswani et al., 2017).

Модели за представяне

Една от първите големи реализации на трансформър архитектура за моделите на представяне беше двупосочното кодиращо представяне от трансформъри (Bidirectional Encoder Representations from Transformers, BERT), въведено от Devlin et al. през 2018 г. За разлика от предишните модели, BERT е обучен да разбира пълния контекст на дадена дума, като взема предвид контекста на изречението преди и след нея едновременно.

BERT също въвежда концепцията за трансфер на обучение в NLP. Вместо да бъде обучаван модел от нулата за всяка нова задача, предварително обучен BERT модел може да бъде специализиращо обучен за конкретни задачи надолу по веригата, като например:

- Класификация на текста
- Разпознаване на именуван обект (NER)
- Идентификация с перифразиране

Тази парадигма за предварително обучение и специализиращо обучение значително намалява количеството етикетирани данни, необходими за NLP приложения, което прави BERT и подобни модели много ефективни.

Моделът BERT промени подхода към различни NLP задачи, включително задачи с двойки текстове като логически изводи и търсене на сходство между изречения.

Характерна черта на съвременните LLM е техният внушителен мащаб. Докато ранните модели съдържат десетки милиони параметри, днес броят на параметрите се измерва в милиарди. Например, BERT-base има около 110 милиона параметри (Devlin et al., 2019), а моделът GPT-3 достига 175 милиарда параметри (Brown et al., 2020).

Преглед на LLM

Големите езикови модели са специализирани модели за машинно обучение в рамките на дълбоко обучение и генеративен AI, обучени върху обширни текстови корпуси, понякога в петабайтов мащаб. В сферата на езиковото моделиране появата на LLM доведе до забележителна промяна. Последните развития в тази сфера се характеризират предимно с нарастващ размер на модела, отразяващ трансформативната сила на мащаба в представянето на езиковия модел. От GPT-3 на OpenAI и неговия още по-голям наследник, GPT-4 , до T5 на Google и по-новите итерации на BERT като RoBERTa (Liu et al. ,2019), LLM-ите показаха впечатляващи възможности за разбиране и генериране на човешки текст.

LLM са показали, че генерират забележително съгласувани и контекстуално богати отговори, дори когато са представени със сложни или двусмислени указания. Освен това, тези модели не само владеят езикови задачи, но са демонстрирали способност да изпълняват основни аритметични операции, да отговарят на фактически въпроси и дори да завършват код в Python или SQL.

Едновременно с това станахме свидетели на появата на модели, които дават приоритет на ефективността и компактността, без да правят компромис с производителността.

Приложения на LLM

LLM са намерили своето място в различни области, променяйки начина, по който взаимодействаме с технологиите.

В областта на образованието LLM-ите предефинират учебния опит. Забележителен пример е иновативното използване на GPT-4 от Khan Academy за разработване на Khanmigo, захранван с изкуствен интелект асистент, който служи като виртуален преподавател за учениците и помощник в класната стая за учителите, предоставяйки персонализирано, специфично за ниво образование за посрещане на различни учебни нужди (OpenAI, 2023).

Междувременно те революционизират пейзажа за създаване на съдържание, създават статии, създават поезия и дори генерират код, с инструменти като GitHub Copilot, които се оказват безценни. Например, Peng et al. (2023) провеждат контролиран експеримент, който

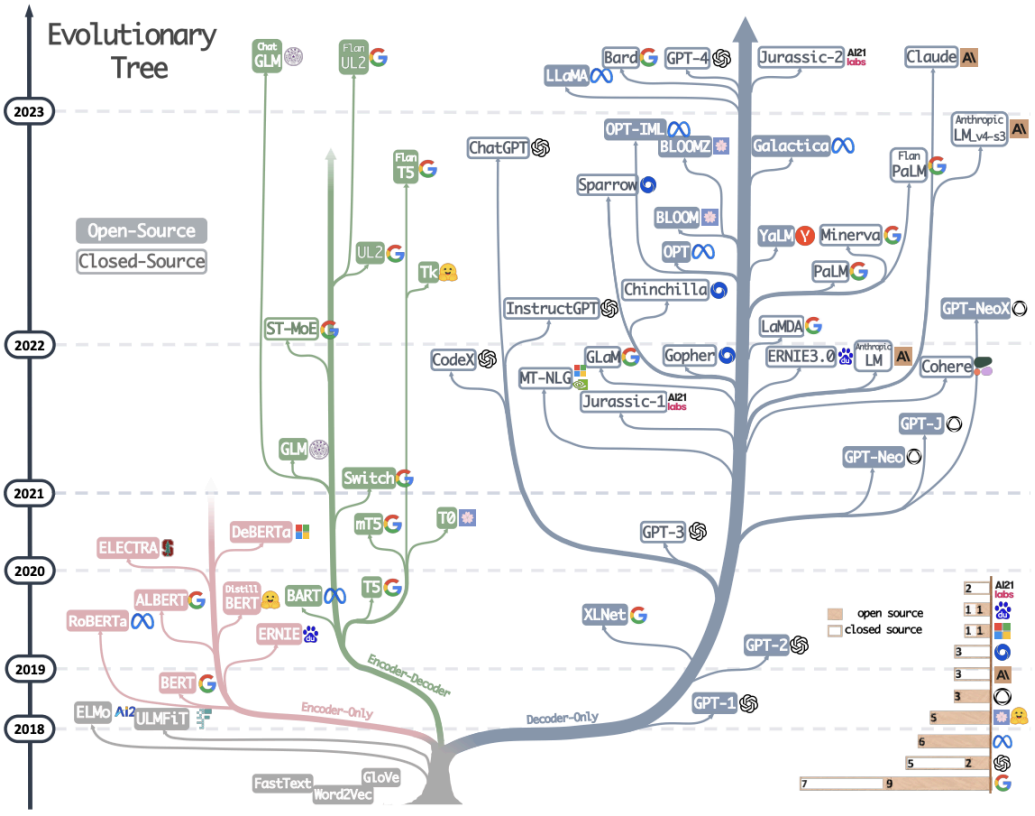
показва, че разработчиците, използващи Copilot, изпълняват задачи с 55.8% по-бързо в сравнение с тези, които не го използват.

В практически план, големите езикови модели се внедряват в различни приложения. Чатботове и виртуални асистенти като ChatGPT са изградени върху LLM и могат да водят диалог с потребители, предоставяйки помощ, информация или забавление на разбираем човешки език.

Развитие на LLM

Фиг. 2Фиг. 2 илюстрира еволюционното дърво на големите езикови модели (LLM) подредено хронологично от 2018 до 2023, като показва основните линии на развитие и връзките между различни архитектури и проекти.

Фиг. 2Фиг. 2 Дървото на еволюцията на LLM-и



Ограничения на LLM

Въпреки впечатляващите си способности, големите езикови модели имат съществени ограничения и предизвикателства. Един от основните проблеми е липсата на достоверност на знанията – LLM често генерират уверени на вид отговори, които всъщност могат да са фактически неверни – т.нар. „халюцинации“. Причината е, че моделът предсказва думи на база вероятностни зависимости, без да разбира съдържанието (Bender et al., 2021). Bender et al. (2021) образно ги наричат „стохастични папагали“, които възпроизвеждат шаблони от обучението, но без истинско знание. Това носи риск от разпространяване на дезинформация.

Свързан проблем са пристроеностите и дискриминацията в изхода на LLM. Поради това, че са обучени върху данни от интернет и други източници, те могат да наследят и усилят различни пристрастия – например стереотипи по отношение на раса, пол или етническа принадлежност (Weidinger et al., 2021). Моделът може да генерира и неподходящо или обидно съдържание, особено ако получи вход, насочващ го в такава посока. Затова, при реалното им използване често се налагат филтри и контрол върху генерираното от тях, особено в публични приложения.

Друг съществен недостатък е невъзможност за тълкуване – практически е невъзможно да се проследи как модел с толкова много параметри стига до даден отговор, което затруднява надеждното му използване в критични области. Също така, въпреки че LLM могат да имитират логическо разсъждение, те често допускат грешки при сложни логически или аритметични задачи и не притежават истинско абстрактно разбиране.

Накрая, LLM са изключително ресурсоемки – обучението им изисква огромни изчислителни мощности и електроенергия (Bender et al., 2021), а използването им в реално време също е скъпо. Това ограничава достъпността на технологията и поражда екологични притеснения.

Способност за аргументация на LLM-и

Големите езикови модели (LLM) демонстрират впечатляваща ефективност в разнообразни задачи (Brown et al., 2020) и при достатъчно голям мащаб проявяват и неочаквано появяващи се нови умения (Wei et al., 2022), сред които особено важна е способността за комплексна логическа аргументация (reasoning). Въпреки това, директното решаване на многоетапни проблеми остава предизвикателство дори за най-модерните модели (Kojima et al., 2022). Например при генериране на отговор без междинни стъпки моделите често грешат в аритметични и логически задачи. За преодоляване на тези ограничения са разработени няколко подхода, които направляват модела през серия от стъпки на

аргументация или дори през разклоняващи се пътища на мисълта. Два от най-успешните такива подхода са Chain-of-Thought (верига на мисълта) и неговата обобщена форма Tree-of-Thought (дърво на мисълта).

Верига от мисли (Chain-of-thoughts)

Chain-of-Thought (CoT) е метод, с който езиковият модел бива подтикнат да изложи хода на мисълта си стъпка по стъпка вместо да дава директен отговор. Wei et al. (2022) показват, че включването на няколко примера с решения, разписани поетапно (т.е. chain-of-thought) в подаденото към модела задание (prompt), насочва модела да генерира поредица от междинни заключения или изчисления до достигане на крайното решение. Този подход значително подобрява резултатите на LLM при аритметични, логически и символни задачи (Wei et al., 2022). Например 540-милиарден модел (PaLM), снабден с няколко CoT примера, постига нов най-добър резултат на еталон за сравнение GSM8K (математически текстови задачи), надминавайки дори специално специализирано обучен GPT-3 (Wei et al., 2022). Това подсказва, че при достатъчно големи модели способността за сложни разсъждения може да се прояви единствено чрез подходящо подканване.

Важно е, че с подобна поетапна аргументация може да се провокира и без никакви примерни решения – само с кратка инструкция към модела. Kojima et al. (2022) откриват, че добавянето на фразата „Let's think step by step" към задачата (т.нар. Zero-Shot CoT) е достатъчно, за да превърне един LLM в разсъждаващ агент в режими без примери. Този прост подход кара модела сам да генерира верига от мисли преди отговора, повишавайки напр. точността на InstructGPT при математически задачи (MultiArith) от ~18% до ~79% (Kojima et al., 2022).

LLM ПРИЛОЖЕНИЯ В ПОЛЗА НА БИЗНЕСА

Рамка за аргументация и действие: ReAct

Рамката ReAct, разработена от изследователи в Принстън и Google (Yao et al., 2022), запълва тази празнина, като предоставя структурирана методология, която позволява на LLM ефективно да управляват и изпълняват сложни работни процеси.

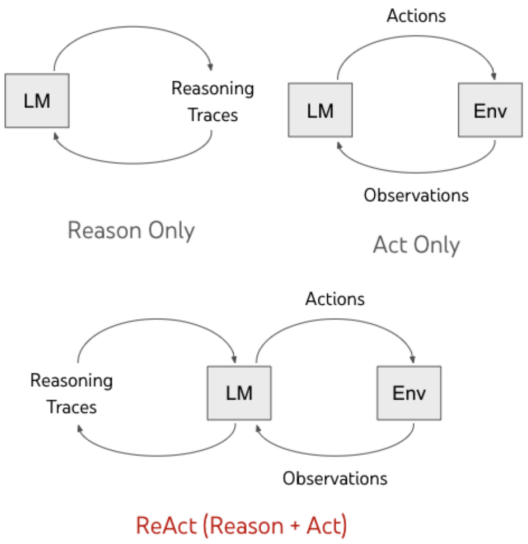
ReAct, съкратено от „Аргументация и действие" („Reason and Action)", е замислен да подобри приложението на LLM отвъд традиционното генериране на текст, за да включва динамично вземане на решения и сложно управление на задачи. Тази рамка е особено важна за задачи, които изискват LLM да участва в обширно синтезиране на информация от множество източници и процеси на вземане на решения, като тези, които се намират в

усъвършенствани системи за отговаряне на въпроси и задачи за проверка на факти, като тези, представени в еталони за сравнение HotPotQA (Yang et al., 2018) и FEVER (Thorne et al., 2018).

В основата си рамката ReAct структурира взаимодействията на LLM в цикличен модел на мисъл, действие и наблюдение (Фиг. 3Фиг. 3):

- **Мисъл (Thought)**: Тази начална фаза включва LLM концептуализиране на стратегия за подход към проблема, определяне на път на разсъждение, който ще ръководи неговите действия.
- **Действие (Action)**: Тук LLM изпълнява набор от предварително дефинирани задачи, които могат да включват заявки към бази данни, извикване на API или извършване на изчислителни функции, необходими за събиране на данни или извършване на изчисления, свързани с настоящата задача.
- **Наблюдение (Observation)**: В тази фаза LLM асимилира резултатите от своите действия, използвайки новата информация, за да коригира своето разбиране на задачата и да прецизира своя подход в следващия цикъл.

Фиг. 3Фиг. 3 Структура на рамка ReAct.



Източник: (Yao et al., 2022)

Въпреки че ReAct значително подобрява функционалността на LLM, той въвежда предизвикателства като осигуряване на устойчивост срещу неочаквани резултати или грешни входни данни. Бъдещите разработки могат да се съсредоточат върху подобряване на адаптивността на ReAct към по-широк набор от приложения, подобряване на обработката на

грешки и усъвършенстване на механизмите за обратна връзка, за да направят модела още по-устойчив и способен.

Рамката ReAct бележи критична еволюция във функционалността на LLMs, позволявайки им да предприемат сложни, многоетапни задачи, които изискват взаимодействие с външни системи и източници на данни. Чрез структуриране на взаимодействия чрез цикъл на мисъл, действие и наблюдение, ReAct не само разширява практическите приложения на LLMs, но също така подобрява техните способности за решаване на проблеми, което ги прави безценни инструменти в пейзажа на изкуствения интелект. С напредването на технологиите непрекъснатото усъвършенстване и адаптиране на рамки като ReAct ще бъде от съществено значение за оползотворяване на пълния потенциал на LLM в различни сектори, проправяйки пътя за по-сложни, автономни системи, които могат да работят ефективно в динамични и сложни среди.

Генериране на код и текст-към-SQL приложения

Съвременните големи езикови модели (LLMs) демонстрират впечатляваща способност не само да разбират естествен език, но и да генерират програмен код в различни езици (Chen et al., 2021). Тази функционалност се основава на дълбоката архитектура на трансформърите, която е обучена върху мащабни корпуси, съдържащи изходен код от множество програмни езици (Austin et al., 2021). Благодарение на това LLM могат да разпознават синтактични и семантични закономерности, както и да прогнозират последващи конструкции въз основа на подаден контекст. По този начин моделите разширяват класическите подходи за автоматизирано програмиране, позволявайки значително ускоряване на разработката и проверката на софтуер (Chen et al., 2021).

Друга ключова област, в която LLM намират широко приложение, е преобразуването на естествен език в заявки към бази данни – т.нар. text-to-SQL (Guo et al., 2019). Тук моделите се обучават да интерпретират въпроси или описания на потребителя и да ги преобразуват в коректни SQL заявки. По този начин потребител, който не владее SQL, може да формулира запитвания на естествен език и да получава желаната информация от базата данни (Guo et al., 2019). Макар ранните решения да изискват големи анотативни корпуси, съвременните LLM успяват да се адаптират към нови области на приложение и структури на данни благодарение на мащабното предварително обучение (Xu et al., 2021). Предизвикателство остава интерпретацията на двусмислия и сложни лингвистични конструкции, но бързото развитие на трансформър-базираните архитектури показва, че точността на text-to-SQL продължава да се подобрява (Chen et al., 2021). Това е особено ценно за бизнес и анализаторски среди, където времето за достъп до релевантни данни е критично.

Анализът на еталона за сравнение BIRD Benchmark (Wang et al., 2023) се утвърждава като един от основните за оценка на представянето на LLMs в задачата по генериране на SQL заявки от естествен език. Изводите от изследването показват, че въпреки напредъка, езиковите модели все още изостават значително спрямо човешката прецизност при изпълнение на текст-към-SQL задачи.

BIRD представлява мащабен еталон за сравнение, разработен за системна оценка на способността на LLMs да превръщат инструкции на естествен език в SQL заявки в контекста на големи бази данни. В изследването се идентифицират няколко предизвикателства, включително работа със „замърсени“ данни, интеграция на външни знания и оптимизация на ефективността на заявките. Впечатляващо е, че дори усъвършенстван модел като GPT-4 постига едва 54.89% точност на изпълнение, което е значително под 92.96% точност, постигната от човешки участници. Това свидетелства за съществуващото значително изоставане и необходимостта от развитие на по-надеждни подходи в тази област.

Многоагентни системи

Съвременните LLM все по-често се разглеждат и като компоненти в многоагентни системи, при които множество автономни агенти (всеки реализиран от LLM или друг алгоритъм) взаимодействат, за да постигнат обща цел (Dafoe et al., 2021). В този контекст многократно надграждащите се езикови модели играят ролята на агенти, които „преговарят“, разпределят задачи, съобразяват се със стратегии на други агенти или дори обединяват усилия в т.нар. кооперативни сценарии (Dafoe et al., 2021). Подобна парадигма е от съществено значение за приложения като координация на роботизирани екипи, дистрибутирани системи за оптимизация или системи за управление на интелигентни градове.

Интегрирането на LLM в многоагентни рамки разкрива редица предимства. Първо, езиковият модел може да анализира описания на средата и да разсъждава относно възможни стратегии за действие (Guo et al., 2024). Второ, комуникацията между отделните агенти може да се извършва чрез диалог на естествен език, улеснявайки кооперацията и споделянето на знания (Dafoe et al., 2021). Въпреки това, съществуват и предизвикателства, свързани със съгласуване (alignment) и безопасност, тъй като автономните агенти могат да вземат решения, които противоречат на човешките ценности или да генерират нежелано съдържание (Chen et al., 2021). Затова важна посока на изследване е как да се формулират механизми за съвместно обучение, при което всеки агент се адаптира спрямо останалите, запазвайки прозрачност и контролируемост (Dafoe et al., 2021).

В една система, където няколко LLM агента взаимодействат помежду си, възникват хетерогенни роли – например един агент може да се специализира в синтез на план, друг – в оценка на риска, а трети – в оформяне на комуникацията към човешките оператори. Подобна модулност дава по-голяма гъвкавост и възможност за мащабиране на системата (Guo et al., 2024). Същевременно, необходимо е да се гарантират сигурност и надеждност, тъй като грешен модел или вредно поведение на един агент могат да компрометират цялостния резултат (Dafoe et al., 2021).

Наред с това, вграждането на мултимодални LLM – способни да обработват изображения, реч или времеви серии – обещава още по-богати и сложни форми на взаимодействие между агентите (Chen et al., 2021). Например в контекста на автономното шофиране, където множество самоуправляващи се автомобили „преговарят“ за позиции по пътя, мултимодалните модели биха могли да преценяват визуални сигнали и текущите условия на трафика, за да генерират по-ефективни и безопасни маневри.

В обобщение, мулти-агентните системи, подсилени от мощни езикови модели, разкриват широка гама от приложения в автономията, роботиката, кооперативния гейминг и комплексните системи за планиране (Dafoe et al., 2021). Развитието на тези техники, в комбинация с установяването на етични и регулаторни рамки, може да ускори възприемането на AI в реалния свят, предоставяйки едновременно ползи и изисквайки постоянна бдителност по въпросите за безопасност и контрол (Chen et al., 2021).

Генерация с обогатено извличане на информация (RAG)

Генерацията с обогатено извличане на информация (Retrieval-Augmented Generation или RAG) възниква като отговор на ограниченията на чисто параметричните езикови модели при задачи, изискващи интензивно използване на знания. Ранни изследвания като REALM (Guu et al., 2020) показват, че интегрирането на диференцируема компонента за извличане с обучението на езиков модел значително подобрява достъпа до фактическа информация. Върху тази идея стъпва и Lewis et al. (2020), които формално представят RAG като рамка, съчетаваща семантично търсене с текстова генерация.

В рамката на RAG всяка заявка първо се използва за извличане на релевантни документи (обикновено чрез плътни векторни представяния или други невронни методи за информационно извличане) от външен корпус с познания, след което генеративен модел използва тези документи като контекст за създаване на отговор или изход. Тази интеграция на модул за търсене позволява на езиковия модел да обосновава своите отговори с актуална външна информация, вместо да разчита единствено на статично заучени параметри. В

резултат се наблюдава повишена ефективност при задачи, изискващи задълбочени знания, като отговаряне на въпроси с отворена област на приложение – оригиналният RAG модел постига нови най-добри резултати в редица бенчмаркове за въпрос-отговор (QA) чрез по-специфични и фактически точни отговори в сравнение с параметричен модел самостоятелно (Lewis et al., 2020).

Последващи изследвания прилагат RAG в широк спектър от проблеми, което подчертава неговата значимост. Например Guo et al. (2023) показват, че рамка с обогатено извличане подобрява текст-към-SQL семантичен парсинг: чрез семантично търсене на релевантна информация за схемата и предишни примери за заявки, тяхната RAG-базирана система по-точно превежда въпроси на естествен език в SQL заявки.

Освен халюцинацията и RAG решава и друго ограничение на LLM-ите - некачествени отговори при дълги задания или т.нар проблем „игла в купа сено“ (“needle in the haystack”) при които се наблюдава понижаване на точността, когато критичната информация е скрита сред големи обеми нерелевантен текст. (Chandrayan et al., 2024).

Появата на RAG бележи ключово развитие в NLP, като съчетава напредъка в невронното информационно извличане с мощни езикови модели. Това съчетание между семантично търсене и генерация позволява на моделите да извличат подходящи факти или примери в реално време, което от своя страна намалява халюцинациите и повишава надеждността. В резултат RAG се утвърждава като важна основа за изграждане на приложения, базирани на знания – от системи за QA с отворена област на приложение до специализирани асистенти – които изискват както плавна генерация от LLM, така и прецизност при извличането на информация (Lewis et al., 2020; Guo et al., 2023).

Предизвикателства на RAG

Въпреки своя потенциал, RAG е съпътствана от редица предизвикателства и ограничения.

Един набор от предизвикателства се съсредоточава върху халюцинациите и надеждността на предоставения контекст. Въпреки че RAG е проектирана с цел да намали халюцинациите чрез обвързване на генерацията с извлечени документи, тя все пак може да генерира неверни или измислени изходи – често в резултат от проблеми на етапа на извличане. Ако механизмът за извличане върне нерелевантни или неточни документи (т.нар. дефектни извличания), езиковият модел може да включи това подвеждащо съдържание в своя отговор. Bian et al. (2023) показват, че дори малко количество невярна информация в извлечения контекст може да „отрови“ изхода на модела: моделът проявява склонност към

свръхзависимост от предоставения текст и може да разпространи дезинформацията, илюстрирайки как „капка мастило“ (лош контекст) се разпространява в целия генериран текст.

Оценка на RAG системи и модели за вграждане

Тъй като ефективността на една RAG система зависи в значителна степен от качеството на нейната извличаща компонента, са разработени надеждни рамки за оценка с цел стандартизиране на информационното извличане в контекста на генерацията. Един от водещите примери е MTEB (Massive Text Embedding Benchmark), въведен от Muennighoff et al. (2023) като обхватен тестов набор за оценка на модели за векторно представяне на текст при разнообразни задачи, включително семантично търсене (semantic search) и извличане на информация (Information Retrieval). MTEB обхваща осем категории задачи (напр. извличане, клъстеризация, отговаряне на въпроси и др.) с 58 набора от данни на над 100 езика, което го прави една от най-обширните рамки за оценка на семантични представяния.

Едно от ключовите заключения е, че няма универсален модел за вграждане, който доминира при всички задачи по извличане – представянето варира в зависимост от областта на приложение и типа задача (Muennighoff et al., 2023). Това подсказва, че оценяването на извличането в контекста на RAG изисква широкообхватен подход: модел, който се представя отлично при уеб търсене, може да се провали при извличане от биомедицинска литература или при заявки на различни езици. Така рамки като MTEB подпомагат избора на най-подходящите модели за конкретни приложения и осигуряват надеждност на семантичната търсеща компонента на RAG.

Специализиращо обучение на LLM-и

Специализиращото обучение (fine-tuning) в машинното обучение се определя като процес, при който предварително обучен върху голям набор от данни или задача модел се дообучава (адаптира) върху нова целева задача или нов набор от данни. Счита се, че това е основна техника в по-широката парадигма на трансферното обучение, при което се пренасят знания, придобити в една задача или област на приложение (източник), за да се подпомогне обучението в друга задача или област на приложение (цел) (Pan & Yang, 2010). При специализиращото обучение се освобождават част или всички слоеве на предварително обучения модел и се дообучават с нови данни (Tan et al., 2018; Zhuang et al., 2020). По този начин се постига по-бърза конвергенция и често се наблюдават по-добри резултати в сравнение с обучението от нулата, особено когато целевата задача разполага с ограничен обем данни (Pan & Yang, 2010; Yosinski et al., 2014). Това позволява повторно използване на

вече научени характеристики, което придава особено значение на този подход в приложения, където се налага работа с по-малки или строго специализирани набори от данни.

Една от основните причини за прилагане на специализиращо обучение върху LLM е адаптацията към конкретна област – т.е. настройване на модела така, че да се представя по-добре при задачи или върху данни от специфична област, която може да не е добре обхваната в процеса на общото предварително обучение. Области като биомедицински текстове, правни документи или програмен код притежават специализирана лексика и разпределения, с които общите LLM модели може да не се справят оптимално без допълнителна адаптация. Специализиращото обучение върху корпуси или задачи от дадена област позволява на модела да интернализира знания от съответната област и да генерира по-точни и релевантни изходи в този контекст.

Потребността от адаптация към област показва ясно, че предварителното обучение само по себе си не е универсално решение – при нишови приложения на LLM, специализиращото обучение върху релевантни (дори ограничени по обем) данни може да доведе до значителни подобрения в ефективността и надеждността на модела.

ГЛАВА II: МЕТОДИКА НА ИЗГРАЖДАНЕ НА АВТОМАТИЗИРАНА СИСТЕМА ЗА РЕШАВАНЕ НА ЗАДЪЛБОЧЕНИ БИЗНЕС ПРОБЛЕМИ ВЪРХУ ИНФОРМАЦИЯ ОТ БАЗА ДАННИ

МЕТОДИ ЗА ПРОГНОЗИРАНЕ С LLM

Методи за генерация на текст-към-SQL

Превръщането на естествен език в SQL заявки или т.нар. “текст-към-SQL” е ключова стъпка в процеса на извличане на релевантни данни от бази данни. Този процес позволява на потребителите да комуникират с базата данни чрез естествен език, без да се налага да познават синтаксиса на SQL.

Подход. За реализирането на тази задача се използват големи езикови модели (LLM), които са предварително обучени върху обширни корпуси от текстови данни. Тези модели разбират контекста и намеренията зад запитванията на потребителя, като генерират точни и контекстуално обосновани SQL заявки.

Интеграция с бази данни. За да бъде системата ефективна и широко приложима, тя трябва да може да се свързва с различни типове бази данни (MySQL, PostgreSQL, SQL Server).

Това осигурява гъвкавост и многоплатформена съвместимост, което е критично за интеграцията в различни среди и приложения.

Обработка и анализ на данните. След като необходимите данни са извлечени, следващата важна стъпка е обработката и анализът на информацията. Това е възможно аналогично посредством LLM за представяне на резултатите по разбираем начин или препращането им към потребителски интерфейс.

Техники за анализ. Големите езикови модели не само подпомагат създаването на SQL заявки, но и анализират извлечените данни чрез естествен език. Те могат да:

- Описват тенденции и зависимости.
- Предлагат интерпретации на резултатите.
- Генерират препоръки въз основа на намерените зависимости.

Благодарение на тези техники, системите за обработка на естествен език и SQL заявки стават по-достъпни, по-точни и по-интуитивни за потребителите.

Методи за аргументация с LLM

В този раздел се представя предложеното решение за прилагане на аргументационните способности на Language Model (LLM) върху сложната задача. Целта е да се оцени агентното поведение на LLM чрез прилагане както на умения за генериране на код, така и на аргументационни способности в комбинация с рамката ReAct.

Експериментите се провеждат с моделите, достъпни чрез OpenAI API, като GPT-4, GPT-4o, GPT-4o-Mini, GPT-3.5 и GPT-4-turbo. Python е избран като програмен език за реализиране на решението, като се използва пакетът Langchain – отворена библиотека и свързани инструменти за улесняване на интеграцията на LLM с база данни на SQL.

Изследват се три основни подхода в рамките на три групи експерименти с цел оценка на аргументационните способности на LLM:

1. ReAct текст-към-SQL LLM агент без памет
2. ReAct с контекстуална памет
3. React с многоагентна система

Във всяка група се провеждат по 10 експеримента за всеки тип LLM и при два различни температурни коефициента (temperature) (0 и 1), където 1 е коефициент за максимално детерминистични отговори на LLM.

Системата трябва да бъде способна да формулира изводи и препоръки въз основа на получените данни. Ако получената информация не е достатъчна, системата следва да бъде способна да определи следващи стъпки, необходими за решаването на задачата. В същото време, ако необходимата информация за решаването на задачата е налице, системата следва да предостави финалният отговор.

ReAct агент без и с контекстуална памет

В подхода на **ReAct текст-към-SQL LLM агент без памет** е проектирано задание, което обхваща изискванията и ограниченията на задачата без контекстуална памет. Заданието е формулирано по начин, който насърчава LLM да извлече необходимите SQL заявки за решаване на мистерията на убийството. Чрез използването на zero-shot prompting се цели да се оцени способността на LLM да аргументира и генерира код единствено въз основа на разбирането си за описанието на задачата.

В подхода на **ReAct с контекстуална памет** се интегрират потребителски задания и се въвежда компонент за контекстуална памет, за да се предостави на LLM допълнителна информация и насоки. Намерението при включването на компонента за памет в процеса на аргументация е да се принуди LLM да поддържа контекст, което улеснява неговата аргументация и му позволява да запомня предишни отговори.

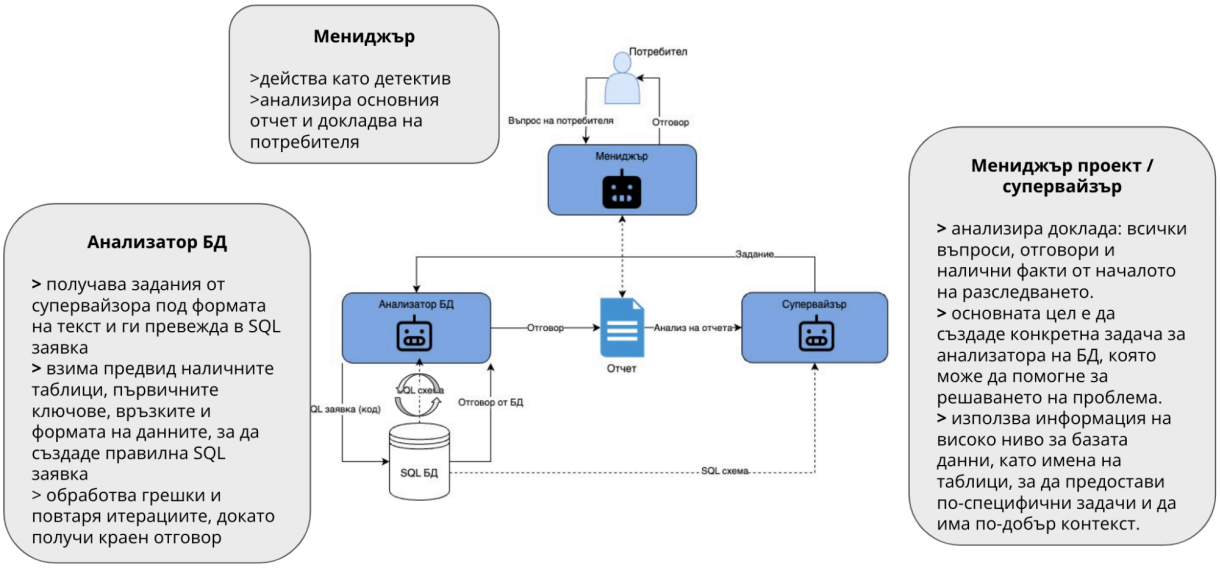
Чрез предоставяне на контекстуална памет се цели да се оцени способността на LLM да аргументира и използва контекстуална информация за генериране на коректен SQL код, достигайки до междинни отговори, които в крайна сметка да повишат шансовете за решение на задачата.

В рамките на експериментите се извършва задълбочен анализ на значими примери, илюстриращи процесите на аргументация и генериране на код от LLM. Чрез изследване на тези примери се придобиват ценни прозрения за аргументационните способности на LLM и неговата ефективност при генериране на точни SQL заявки въз основа на зададените инструкции и контекстуална информация.

Многоагентен подход

Третият подход включва React с многоагентна система

Фиг. 7 Ролите в многоагентна LLM система.



Източник: Vichev, 2024

Многоагентните системи позволяват работата на LLM да бъде ясно фокусирана върху определено задание, което следователно следва да подобрява качеството на генерираните отговори. Подобно на бизнес роли в една организация, задачите могат да бъдат разпределени към различни части от екипа (Vichev, 2024).

Фиг. 7 онагледява система, в която един и същи LLM се използва за различни цели и по различен начин и в различни части от системата. Заданията, които биват подавани към всеки LLM също са различни, по този начин всеки LLM може да бъде наричан агент.

LLM агентите могат да взаимодействат помежду си и с потребителя, в зависимост от начина на построяването на системата.

В случая системата се състои от 3 агента: мениджър, супервайзър и анализатор БД.

Ролята на мениджъра е да приема главния въпрос или задание от потребителя и да го запише в отчета или предаде на друг агент. Той има задачата да анализира отчета и да връща крайният отговор на потребителя - на предварително зададена стъпка или когато сметне, че е информацията е достатъчна, за да получи финален отговор. Може да има и функцията да спре работата на другите LLM агенти, ако прецени, че заданието е изпълнено, спрямо зададените параметри на заданието.

Супервайзорът анализира доклада: всички въпроси, отговори и налични факти от началото на разследването. Основната му цел е да създаде конкретна задача за анализатора на БД, която може да помогне за решаването на проблема или намирането на отговора.

Той използва информацията за БД на високо ниво (например: имена на таблици), за да предостави по-специфични задачи на анализатора и да има по-добър контекст.

Анализаторът БД получава задания от супервайзора под формата на текст, неговата задача е да генерира правилна SQL заявка, която да доведе до извличане на нужната информация от БД.

Той взима предвид наличните таблици, първичните ключове, връзките и формата на данните, за да създаде правилна SQL заявка, обработва грешки и повтаря итерациите, докато получи краен отговор.

МЕТОДИ ЗА ПОДОБРЕНИЕ И ОЦЕНКА НА ВГРАЖДАЩИ МОДЕЛИ ЗА RAG СИСТЕМИ С SQL БАЗИ ОТ ЗНАНИЯ.

Това изследване е насочено към фазата на извличане на данни в една RAG система в областта на текст-към-SQL приложения. Доказано е, че динамичното извличане на релевантна контекстуална информация от бази от знания (knowledge base) повишават вероятността LLM-и да генерират по-качествени и осмислени отговори. (Gao, et.al., 2024) **В този контекст, фазата на извличане играе критична роля в ефективността на цялостната RAG система, тъй като определя качеството и приложимостта на предоставената информация.**

Този раздел има за цел да разработи методология за специализирано обучение на модел с приложение за текст-към-SQL задачи, отговарящ за извличане на знания - модел за вграждане, както и да разработи методология за оценка на този вид модел с релевантни типове данни.

Модели за вграждане (embedding models)

В NLP областта съществуват два основни подхода за използване на модели за вграждане на текстове: open-source модели и модели, достъпни чрез API. Open-source моделите, като BERT, RoBERTa и DistilBERT, предоставят гъвкавост и контрол върху обучението и адаптацията към специфични задачи. Те могат да бъдат изпълнявани локално или на сървъри с GPU/TPU, което позволява тяхното специализиращо обучение и оптимизация за конкретни области на приложение.

От друга страна, API-базираните модели, като тези, предлагани от OpenAI, Cohere, и други доставчици позволяват лесен достъп до мощни NLP модели без необходимост от локално изпълнение. Този подход е особено полезен за организации, които нямат достатъчно изчислителни ресурси или експертиза в изграждането и поддръжката на модели. Въпреки

удобството, API-базираните решения могат да доведат до зависимост от външни доставчици, ограничения в заявките и съображения относно поверителността на данните. В допълнение, не всички доставчици предоставят възможност за специализиращо обучение на модели за вграждане.

Изборът между open-source и API-базирани модели зависи от нуждите на проекта – дали се търси максимална гъвкавост и локален контрол или бързо и лесно интегриране в съществуваща инфраструктура.

Някои от характеристиките на Sentence Transformer (bi-encoder) моделите са:

- Изчисляват векторно представяне (embedding) с фиксиран размер на база дадени текстове или изображения.
- Изчислението на векторното представяне е обикновено ефективно, а изчислението на косинусово подобие между векторите е изключително бързо.
- Приложим за широк спектър от задачи, като семантично текстово подобие, семантично търсене, клъстериране, класификация, намиране на парафрази и други.

Алгоритъм за оценка на изваждане на информация

Методиката за оценка е специално адаптирана за обработка на немаркирани двойки въпроси, което изключва първоначално ранжиране на предоставените данни.

Алгоритъм 1: Клас за оценяване – Методология за изчисляване на косинусово подобие и метрики за оценка

```
01: процедура COSINE_SIMILARITY_BATCH
02:     Вход: заявка и векторни представяния на кандидатите
03:     Процес:
04:     Преобразуване на векторните представяния в тензори
05:     Изчисляване на скаларно произведение и норми за изчисляване на косинусово подобие
06:     Изход: матрица на косинусовите сходства
07: край на процедурата
08: процедура GET_TOP_K_LABELS
09:     Вход: матрица на косинусово подобие
10:     Процес:
11:     Сортиране и извличане на топ-k индекси и етикети
12:     Изход: списък с топ-k етикети за всяка заявка
13: край на процедурата
14: процедура CALCULATE_METRICS
15:     Вход: top_k_labels, context_labels, subset_k, metrics
16:     Процес:
17:     Изчисляване на попадения (hits) и нормализирана кумулативна печалба (nDCG) за топ-k етикетите
18:     Изход: стойности за hits и nDCG
19: край на процедурата
```

Източник: Vichev & Marchev, 2024

Това и означава, че:

- За всеки въпрос съществува само една релевантна информация, а за останалата информация се приема допускането, че не са релевантни.
- В началото на експеримента предварително зададено „правилно“ съвпадение (етикет) между въпросите и извлечената информация не е известно за модела, а само за изследователя.
- Всички извлечени данни се обработват наравно – няма предварително определена йерархия на значимост. Ранкирането се извършва след получаване на векторните представяния от модела.
- Оценката на извличането се извършва динамично – след като информацията бъде извлечена, тя се оценява спрямо етикета.

Оценяване на релевантността

В рамките на избраните *max*K индекси, алгоритъмът изчислява честотата на попаденията (Hit Rate) и Нормализирано дисконтирано кумулативно подобрение (nDCG) за всеки въпрос и стойност на K. Този процес включва проверка дали индексът на въпроса съвпада с индекса на съответния запис. По този начин за всеки въпрос се изчисляват HitRate@K и nDCG@K.

1. Изчисление на попаденията (Hits): Дефинира се вектор $hits$, където всеки елемент $hits_i$, се изчислява като:

$$H_i = I(L_i \in T)$$

,където:

- *I*(condition) е 1, ако условието е изпълнено, и 0 в противен случай.
- L е масив от контекстни етикети

2. Изчисляване на *nDCG*:

$$\mathrm{NDCG}_i = \begin{cases} \frac{1}{\log_2(P[L_i]+1)} & L_i \in P \\ 0 \end{cases} \text{ for } i = 1, \ldots, n$$

където P е речник с позиции на етикетите за топ-k етикетите в T_i, като всеки етикет се съпоставя с неговия ранг, започващ от 1.

Категоризация на данни

Генерираният изход за всеки въпрос, наречен сурови данни, преминава през процес на категоризация въз основа на предварително зададени критерии като модел,

обучителни/тестови подмножества и специфики на базата данни. Тази стъпка гарантира, че последващите анализи са структурирани и контекстуализирани спрямо вътрешните характеристики на данните. (Vichev & Marchev, 2024)

Метрики за оценка

В настоящото изследване за оценка на експериментите се използва метриката nDCG@K, която предлага редица предимства в контекста на задачата. Тя осигурява количествена и лесна за тълкуване оценка на качеството на извлечените документи, като взема предвид както релевантността, така и позицията на всеки елемент в класирането (Jarvelin & Kekalainen, 2002). Това я прави особено подходяща за системи, при които е от значение не само кои фрагменти текст се връщат, но и в какъв ред. Допълнително предимство на nDCG@K е възможността за директна съпоставка с външни стандартизирани еталони за сравнение като MTEB, където nDCG@10 се приема за основна метрика (Muennighoff et al., 2023), което улеснява валидирането на резултатите и позиционирането им в по-широк изследователски контекст. В допълнение към това, Thakur et al. (2021) в друг еталон за сравнение на модели за извличане на информация (BEIR) установяват, че nDCG@K предлага надеждни резултати както за бинарни, така и за градуирани оценки на релевантност, което допълнително потвърждава приложимостта ѝ.

Наред с това, nDCG@K е особено ефективна в условия, при които са налични човешки анотации (Jarvelin & Kekalainen, 2002). В нашия случай се разполага с над 5000 такива записа и няколко вида модели и видове експерименти, което позволява прилагането на тази метрика без нужда от допълнителна автоматизирана оценка. Това я прави значително по-ефективна и мащабируема при провеждането на множество експерименти с различни модели, като същевременно се гарантира висока прецизност и устойчивост на резултатите.

Съществуват и други подходи за оценка на RAG системи, като предложените в рамките на “ragas”, където се използват LLM модели за оценка на извлечените и генерираните отговори. Тези метрики са особено полезни при липса на човешки анотации, тъй като позволяват автоматизирана оценка без да са необходими анотирани данни. Въпреки това, когато са налични надеждни анотации, както е в настоящото изследване, използването на метрики като nDCG@K води до по-ефективна, възпроизводима и прозрачна оценка.

Използват се метрикита *nDCG@K*, където *K* ⎕ има стойности от 3, 5 или 10 извлечени елемента, за оценка на ефективността на избраните модели за векторно представяне. Това сравнение на метрики не само подчертава производителността на всеки модел, но също така е

широко прието и съпоставимо с утвърдените еталони за сравнение за извличане на информация в рамките на RAG. (Vichev & Marchev, 2024)

Алгоритъм 2: Оценка и изчисляване на метрики в контекста на текст-към-SQL

```
01: процедура ОЦЕНКА
02:     Вход: DataFrame със заявки и метаданни
03:     Процес:
04:     Итерация и търсене в колекцията за всеки запис
05:     Проверка на съвпаденията чрез сравняване на идентификатори
06:     Изход: списъци с резултати, идентификатори на резултати и съвпадения
07: край на процедурата
08: процедура ИЗЧИСЛЯВАНЕ_НА_МЕТРИКИ
09:     Вход: данни, k-стойности, метрики
10:     Процес:
11:     Изчисляване на средните стойности на метриките, групирани по етикет
12:     Изход: DataFrame с изчислените средни стойности
13: край на процедурата
```

Източник: Vichev & Marchev, 2024

Тази методология има за цел надеждно да тества ефективността на моделите за задачи за извличане на данни, като включва извличане на резултати и изчисляване на гореспоменатите метрики.

Използвани функции за загуба при специализиращо обучение

За да се подобри ефективността на векторните представяния в специфични за задачата контексти, като извличане на информация за text-to-SQL, е необходимо допълнително специализиращо обучение на предварително обучени модели. Изборът на подходяща функция на загуба е критичен фактор в този процес, тъй като той определя начина, по който моделът се научава да различава релевантни от нерелевантни примери. В този подраздел се разглеждат основни функции на загуба, приложими за специализиращо обучение на модели за вграждане, като основа за последващото сравнение и анализ на резултатите в т. 8.5.

MultipleNegativesRankingLoss е много подходяща функция на загуба, ако има само положителни двойки, например двойки от сходни текстове като двойки парафрази, двойки дублиращи се въпроси или двойки (заявка, отговор). (Henderson et al., 2017)

Тази функция на загуба очаква като вход партида (batch), състояща се от двойки изречения (a_1, p_1), (a_2, p_2), ..., (a_n, p_n), където се приема, че (a_i, p_i) са положителна двойка, а (a_i, p_j) за $i! = j$ е отрицателна двойка.

За всяко a_i се използват всички останали p_j като отрицателни примери, т.е. за a_i има 1 положителен пример (p_j) и $n - 1$ отрицателни примера (p_j). След това се минимизира негативната лог-вероятност за “softmax” нормализираните оценки.

Производителността обикновено се подобрява с увеличаване на размера на партидите.

Може също така да се предоставят един или няколко трудни отрицателни примера за всяка положителна двойка, като данните се структурират по следния начин: (a_1, p_1, n_1), (a_2, p_2, n_2). Тук n_1 е труден отрицателен пример за (a_1, p_1). В този случай функцията на загуба ще използва за двойката (a_i, p_i) всички p_j за $j! = i$ и всички n_j като отрицателни примери.

Математическото описание:

$$\mathscr{L}(a_i, p_i) = -\log\left(\frac{\exp(\text{sim}(a_i, p_i))}{\sum_{k=1}^{K}\exp(\text{sim}(a_i, c_k))}\right)$$

където:

- a_i е векторът на анкъра (заявка);
- p_j е векторът на съответния позитивен пример (отговор);
- K е общият брой на кандидатите (обикновено $K = n + m$, където n е броят на примерите в батча, а m — брой на допълнителните трудни негативи);
- $sim(a_i, c_{k)}$ е функция за подобие между векторите a_i и c_k

Модели за вграждане, специфични за индустрията

Въпреки че широко достъпните модели за вграждане демонстрират висока ефективност при разнообразни задачи, специфичните предизвикателства в сферата на text-to-SQL, като разбирането и съпоставянето на заявки на естествен език с точни SQL кодови примери или доказателства, изискват по-целенасочен подход в извличането на информация. Синтактичните и семантичните особености на SQL заявките налагат адаптации, които да съответстват тясно на схемата на базата данни и намерението зад заявката, като това ги отличава от стандартните задачи по извличане на информация. (Vichev & Marchev, 2024)

Следователно, адаптацията към конкретна област на приложение на тези модели в контекста на извличане на информация за SQL текстове е от съществено значение за оптимизиране на ефективността на търсенето по подобие в задачите по text-to-SQL.

Преодоляването на тези предизвикателства може да допринесе за развитието на изследванията в областта на text-to-SQL, което ще направи данните по-достъпни за неекспертни потребители и ще подобри процесите на вземане на решения в конкретните сектори. (Vichev & Marchev, 2024)

Затова в тази работа се изследват и сравняват различни модели за вграждане и извличане на информация в контекста на text-to-SQL. Реализираме специализирана оценъчна рамка и адаптираме BIRD Bench набора от данни, за да го направим подходящ за задачи по извличане на информация. Освен оценката на съществуващите модели, извършваме специализиращо обучение на предварително обучен Sentence Transformer модел и представяме резултати от сравнителния анализ.

Чрез оценка на ефективността на извличането на информация в контекста на text-to-SQL, целим да изградим по-стабилна рамка, която не само подобрява точността на генериране на SQL код, но и разширява практическата приложимост на интерфейсите на естествен език за работа със сложни бази данни. Това изследване може да проправи пътя към по-интуитивни и ефективни инструменти за заявяване на данни, като в крайна сметка демократизира достъпа до данни и аналитиката.

Тези резултати са в съответствие с актуални изследвания, които подчертават потенциала на специализиращо обучените модели за подобряване на NLP задачите чрез стратегии за адаптация към специфични области на приложение.

АРХИТЕКТУРА И КЛЮЧОВИ КОМПОНЕНТИ НА АВТОМАТИЗИРАНА СИСТЕМА ЗА РЕШАВАНЕ НА СЛОЖНИ ЗАДАЧИ С БАЗИ ДАННИ (RAGSQL)

Обща схема

Този раздел разглежда архитектурата на RAGSQL системата, която е предназначена за автоматизирано решаване на сложни задачи с използване на SQL бази данни, многоагентна LLM структура и RAG.

RAGSQL представлява подход, който комбинира Retrieval-Augmented Generation (RAG) и текст-към-SQL техники за подобряване на взаимодействието с базите данни чрез използване на големи езикови модели и динамично извличане на релевантни примери, които да допълват LLM заданието. Основната идея зад RAGSQL е да се използват методите на извличане на релевантна информация от текстови източници и автоматизирано генериране на SQL заявки,

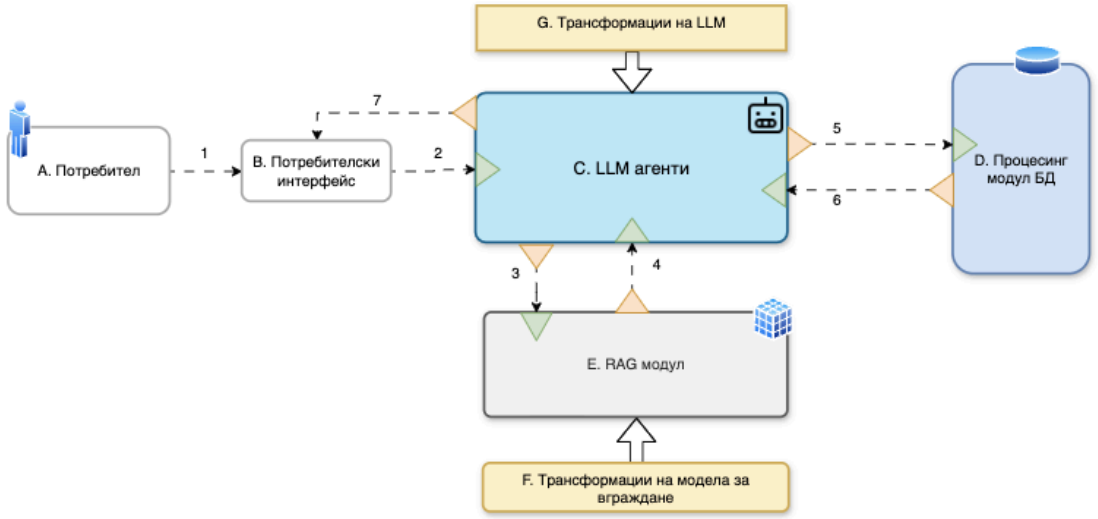
за да се отговори на сложни аналитични въпроси. Вместо ръчно създаване на заявки или директно използване на текст-към-SQL модели, RAGSQL разчита на хибриден подход, който включва както търсене на свързани данни от външни източници, така и аргументация върху тези данни и генериране на SQL заявки с помощта на LLM-и.

Основният принцип на RAGSQL е първо да извлече релевантните данни чрез алгоритми за търсене и извличане от RAG система, като по този начин допринася към заданието, с което LLM трябва да работи. След това езиковият модел използва този контекст, за да формулира по-точни SQL заявки, които могат да бъдат директно изпълнени върху целевата база данни. Това намалява излишната обработка, като предоставя само релевантна информация на модела и подобрява точността на генерираните отговори.

Приложенията на RAGSQL включват широк набор от сценарии: автоматизирано генериране на сложни отчетни заявки, подобрено извличане на знания от многопластови данни, както и улесняване на взаимодействието с бази данни за потребители без специализирани технически познания. Възможността за комбиниране на аргументация, извличане и динамично създаване на SQL команди прави RAGSQL мощен инструмент за справяне с високоспециализирани задачи, изискващи бърз и точен достъп до информация.

За разлика от MAG-SQL (Xie et al., 2024) и други подобни изследвания, където целта на агентите е да генерират правилна SQL заявка чрез итеративно подобряване и корекции на частични Sub-SQL заявки, в рамките на тази дисертация се решава значително по-сложна задача. Вместо да се фокусира само върху създаването на SQL заявка, разработената система използва SQL заявките като междинна стъпка за постигане на основна крайна цел, а работата на агентите се фокусира върху анализ на извлечените данни, планиране на следващи стъпки и вземане на решения до окончателното решаване на поставената задача. Процесът включва динамична оценка на резултатите от заявките, аргументация върху наличните факти и създаване на нови планове за действие в случай на непълна или противоречива информация. По този начин приносът на тази работа се състои в разработването на многоагентна система, използваща рамката ReAct, която интегрира text-to-SQL генерация, анализ на резултати и стратегическо планиране в единен цикъл за решаване на сложни бизнес задачи. В същото време, резултатите от изследвания като MAG-SQL могат да бъдат използвани за допълване и подобряване на разработената система, като повишат надеждността и качеството на генерираните SQL заявки в процеса на разследване и вземане на решения.

Фиг. 8 Обща архитектура на RAGSQL система с LLM агент/и.



Източник: автор

Легенда:

1. **Въвеждане на въпрос/задача от потребител.**
2. **Подаване на заданието към LLM агенти.**
3. **Заявка от LLM агенти за извличане от RAG База от Знания.**
4. **Извличане на релевантната информация от БЗ.**
5. **Подаване на SQL заявка към SQL БД.**
6. **Подаване на извлечената информация от БД към LLM агенти.**
7. **Окончателен отговор към потребителя.**

В архитектурата на RAGSQL могат да съществуват следните информационни потоци между основни структурни блокове, които не са задължително разположени в последователен ред:

1. **Въвеждане на заданието.** Потребителят задава заявката си чрез удобен интерфейс, който може да бъде под формата на чат, диалогова система или специализиран уеб формуляр. Интерфейсът трябва да бъде интуитивен и да позволява ясно формулиране на въпроса или задачата.
2. **Подаване на заданието към LLM агенти.** След като потребителят въведе заданието, то автоматично се прехвърля към подсистемата за обработка от LLM агенти. Тази подсистема анализира текста на заявката, като идентифицира основните концепции, ключови думи и формулировки, необходими за следващите етапи.
3. **Заявка от LLM агенти за извличане от RAG База от Знания.** LLM агентите изпращат заявка към RAG подсистемата, която извършва първоначално търсене в предварително индексираната документация. Целта е да се открие релевантна информация, която ще

бъде използвана като контекст при генерирането на SQL заявки или окончателни отговори. След изпълнението на SQL заявката и връщането на резултатите, LLM може да генерира допълнителна заявка към RAG, ако възникне необходимост от допълнителен контекст или пояснения, свързани с получените данни. Това се случва, когато:

- Първоначалната информация е частична или не напълно разрешава задачата.
- Изисква се допълнителна документация, описания на таблици или примерни случаи за коректно интерпретиране на данните.

Резултатът от тази допълнителна заявка може да бъде нова текстова информация, която:

- Подпомага LLM при генерирането на коректна SQL заявка.
- Обогатява крайния отговор, предоставен на потребителя.

Този процес гарантира, че LLM работи с максимално точен и пълен контекст, като комбинира структурирани (SQL) и неструктурирани (RAG) данни за създаване на качествени и аргументирани отговори.

4. **Извличане на релевантната информация от БЗ.**

   RAG подсистемата идентифицира текстови документи и релевантни фрагменти, които предоставят контекст за зададената задача. Тези фрагменти могат да включват техническа документация, ръководства за базата данни, описания на таблици или примери за подобни SQL заявки. Извлечената информация се връща към блок “LLM агенти”, който я обработва и използва при съставянето на следващите заявки.

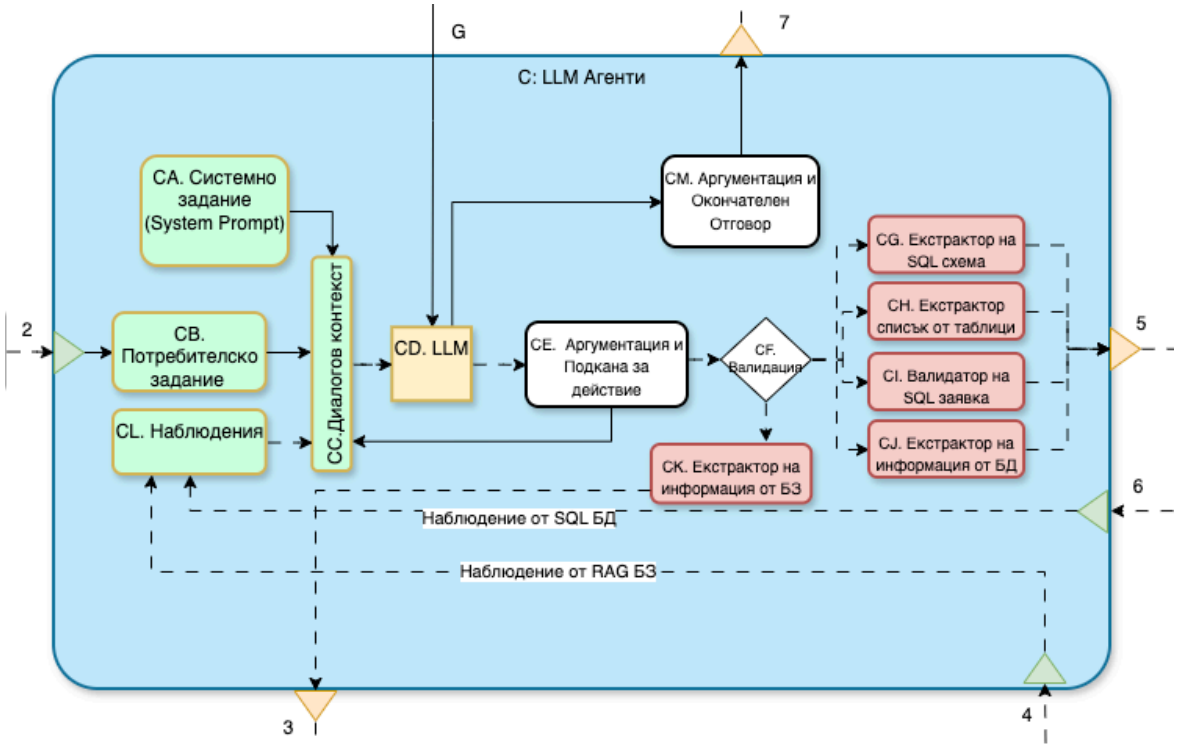
5. **Подаване на SQL заявка към БД.** Информацията от предходните стъпки се комбинира, за да се създаде валидна SQL заявка за извличане на информация на SQL схеми, таблици или бизнес данни. LLM модулът, решавайки текст-към-SQL задача, генерира заявката въз основа на извлечените и интерпретирани данни. Тази заявка може директно да се изпълни върху базата данни, за да се получат крайните резултати.
6. **Извличане на данни от SQL БД**. След подаването на SQL заявката към базата данни, системата изпълнява заявката и обработва резултатите. Изпълнението на заявката връща структурирани данни, които могат да бъдат под формата на таблици, агрегирани стойности, технически параметри на базата или други релевантни формати. Тези данни се извличат от базата данни и след това се предоставят за по-нататъшна обработка на LLM агентите.
7. **Окончателен отговор към потребителя.** След извличането на необходимите данни и допълнителните пояснения, системата формулира ясен и точен краен отговор. Този

отговор обединява резултатите от SQL заявката, допълнителната информация от RAG подсистемата (ако има такава) и контекста на първоначалното задание. Всяка част от събраната информация се структурира и представя по разбираем за потребителя начин. Крайният отговор може да бъде поднесен като отчет, таблица с данни, текстово обяснение или комбинация от тези формати. Основната цел е потребителят да получи пълна, коректна и добре обоснована информация, която да му помогне при решаването на поставения аналитичен въпрос или задача.

Блок C “LLM Агенти”

LLM агентите са автономни единици, които управляват взаимодействието между базите данни и езиковите модели. Те са програмирани да извършват специфични задачи като анализ на текст, откриване на модели и предоставяне на предложения за решения на задачи.

Фиг. 9 Структура на блок C “LLM агенти” (с един агент)

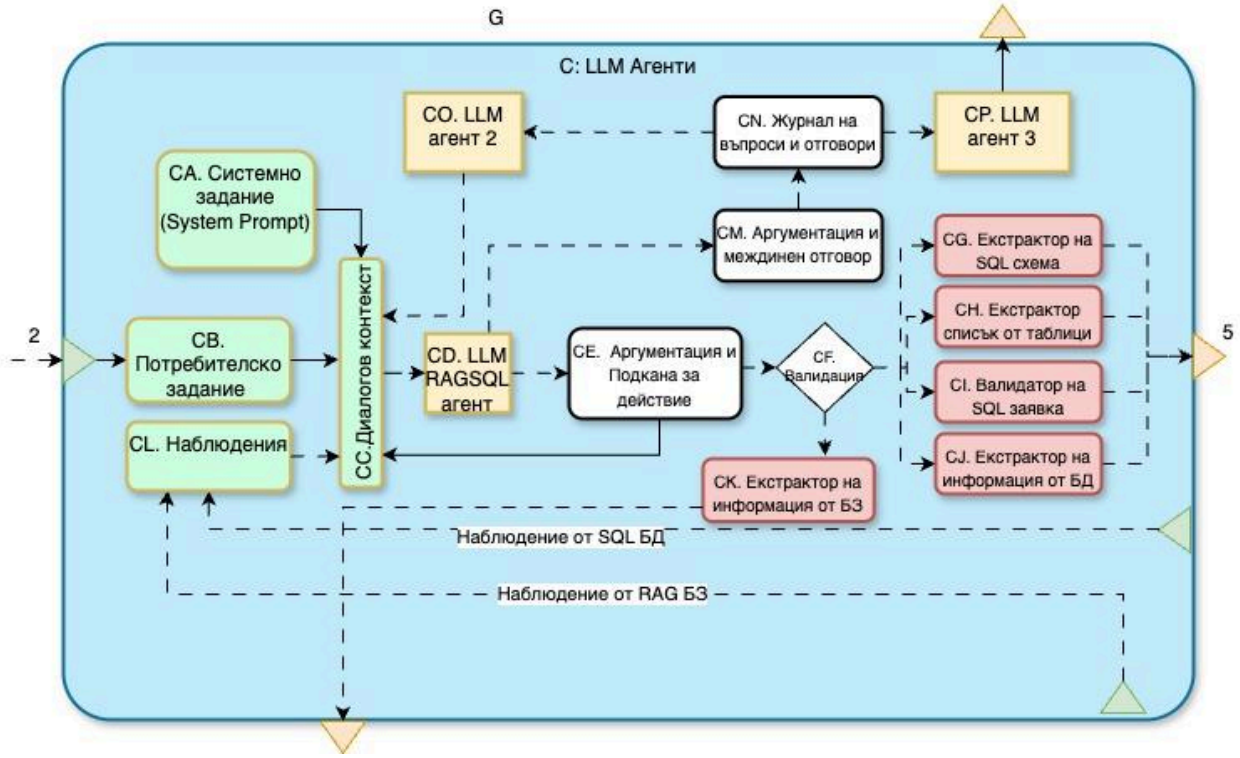


Източник: автор

RAGSQL агентът работи чрез координиран процес, включващ LLM, валидация и специализирани инструменти (програмни блокове) за извличане на информация от SQL база данни и RAG база от знания. Основните елементи на блока са показани на Фиг. 9

Архитектурата на блока може да се разгледа от гледна точка на входните данни и процес на работа.

Фиг. 10. Структура на блок С “LLM агенти” (многоагентна)



Източник: автор

Фиг. 10. по-горе представя комуникацията в многоагентна структура, в която съществуват два допълнителни LLM агента и допълнителен блок:

CN. Журнал на въпроси и отговори

Това по същество е дневник на всички въпроси и отговори на RAGSQL агента (CD) без “вътрешния” му диалог.

CO. LLM агент 2

Този LLM агент има за цел да генерира следващата най-релевантна задача за LLM RAGSQL агент (CD), прочитайки всичко до момента от журнала CN.

CP. LLM агент 3

Този LLM агент има за цел да докладва на потребителя, анализирайки работата на другите агенти до момента (от записите в CN). Допълнително агентът може да “вземе решение” за спиране на работата на системата, ако прецени, че събраната информация е задоволителна и основната задача на потребителя е решена.

Многоагентната архитектура, илюстрирана на фигурата, въвежда допълнителни нива на автономност и координация чрез три специализирани компонента, всеки с ясно дефинирана роля в управлението и оценката на аргументационния процес. Този подход позволява динамично генериране на задания (чрез CO) и критичен мета-анализ на системната дейност (чрез CO), основан на хронологията от въпроси и отговори, съхранявана в журнала CN. Системата демонстрира не само оперативна гъвкавост, но и способност за саморефлексия, като интегрира механизъм за самостоятелно прекратяване на изпълнението при постигане на целите.

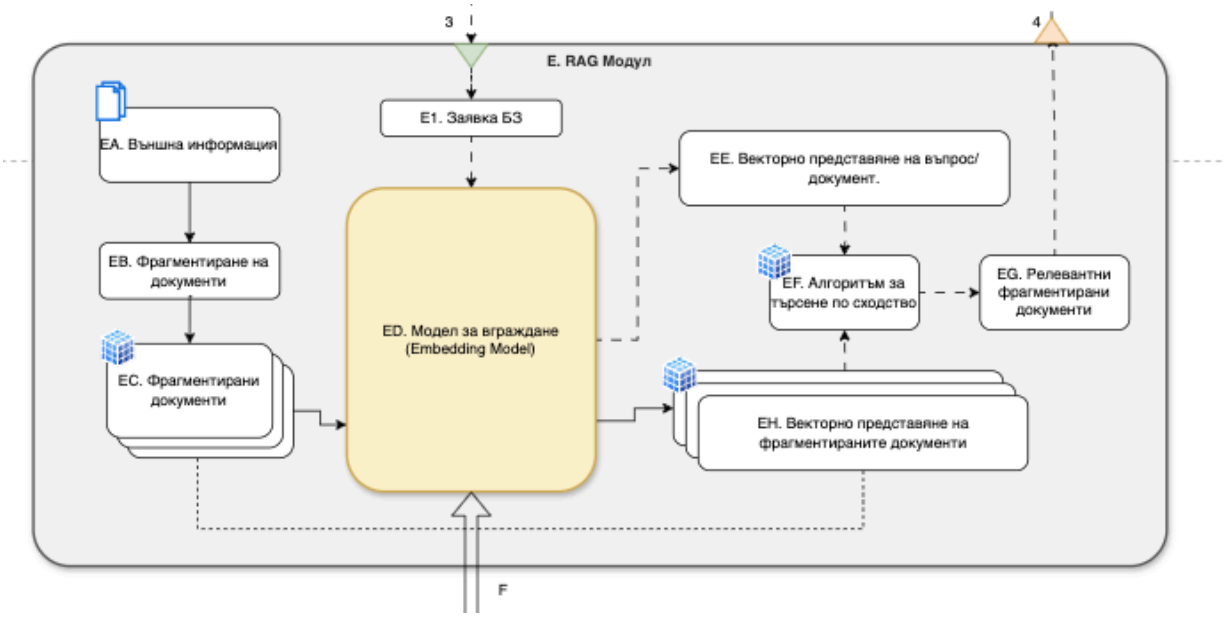
Блок E “RAG модул”

RAG се състои от две основни фази – подготовка на външна информация и извличане/генериране на отговори. В първата фаза външните данни се обработват и индексират, за да бъдат леснодостъпни по време на следващите етапи. Във втората фаза, наречена извод (inference), моделът извлича релевантна информация от предварително обработената база данни и я обогатява с контекста, зададен от потребителя. Обединените данни се използват за генериране на отговори, които са високо точни и съобразени с въпроса. По този начин RAG успява динамично да предостави оптимизирани решения чрез ефективно използване на външни знания и съвременни методи за обработка на естествен език.

Тази подсистема обединява процесите на извличане на информация и генериране на текст, за да подобри качеството на генерираните от LLM отговори. Този блок е съществен за решаването на сложни задачи, особено изискващи актуална и специализирана информация, която не е била заложена в тренираните данни на съответния LLM.

RAG подсистемата се състои от няколко основни елемента взаимодействащи помежду си (Фиг. 12).

Фиг. 12 Структура на блок “RAG модул”



Източник: автор

EA. Външна информация

Тази част от RAG модула обхваща всички външни източници на текстови или структурирани данни, които могат да послужат като контекст при решаване на дадена задача. Външната информация включва документация, технически ръководства, описания на схеми, логове, както и други текстови файлове, които предварително са индексирани и достъпни за системата. Основната функция на този компонент е да предостави база от знания, която да подпомогне извличането на релевантна информация, необходима за решаването на текущата задача.

EB. Фрагментиране на документи

Процесът по фрагментиране на документи може се извършва съгласно предварително дефинирани правила или алгоритми, които вземат предвид структурата на текста, дължината на параграфите или логическите раздели в документа, например: заглавия на текст, параграфи или редове в таблици. Критериите включват осигуряване на смислова завършеност, подходяща дължина спрямо ограниченията на използвания модел за вграждане и контекстова независимост на всеки фрагмент. В някои случаи се прилага и специфична бизнес логика за групиране на свързани части от съдържанието.

В контекста на text-to-SQL приложенията логиката на фрагментацията следва да бъде такава, че всеки фрагмент да представялва отделна SQL заявка. В такъв случай всяка SQL

заявка може да бъде разглеждана като един логически фрагмент, тъй като представлява самостоятелна смислова единица.

EC. Фрагменти на документ

Това са индивидуалните текстови сегменти, които се получават след разделянето на документа. Всеки фрагмент съдържа сравнително самостоятелно смислово завършено съдържание, но може да изисква допълнителен контекст от съседни фрагменти, за да бъде напълно разбираем. Тези фрагменти служат като основни единици, върху които RAG подсистемата работи при извличането на информация, създаването на векторни представяния (embeddings) и определянето на тяхната релевантност спрямо поставения въпрос.

ED. Модел за вграждане

Моделът за вграждане е основен компонент на RAG подсистемата, който преобразува текстовите фрагменти и потребителските заявки в многомерни векторни представяния. Тези векторни представяния запазват семантичната структура на текста, което позволява ефективно търсене и сравнение между различни текстови елементи. Моделът за вграждане може да бъде предварително обучен върху големи корпуси от текстове и оптимизиран за специфични задачи, като търсене на информация, извличане на контекстуални зависимости и подобряване на точността на RAG процеса.

E1. Заявка БЗ (База от Знания)

Заявката към БЗ е форматирано текстово запитване, което RAG системата използва, за да извлече релевантна информация от базата от знания. Тази заявка може да бъде директно генерирана от потребителското задание или модифицирана от LLM в зависимост от диалоговия контекст. Форматът на заявката е оптимизиран за алгоритмите за търсене, като може да включва ключови думи, семантични зависимости и допълнителни филтри за по-прецизно извличане на информация. Тя се използва за създаване на векторното представяне (EE), което след това се използва в EF (Алгоритъм за търсене на подобие), за да идентифицира най-близките по смисъл фрагменти от документи (EG).

EE. Векторно представяне на въпрос/документ

Този компонент преобразува потребителското задание или въпрос в числово векторно представяне, наречено embedding. Векторните представяния на въпроса улавят основните концепции, намерения и смисъл на текста. Те са входни данни за алгоритъма за търсене, който определя най-релевантните фрагменти от документа.

EF. Алгоритъм за търсене по подобие

Алгоритъмът за търсене е ключова част от RAG модула, отговаряща за намирането на релевантна информация. Той сравнява векторното представяне на въпроса с векторните представяния на фрагментите от документи, за да идентифицира онези части от текстовата база, които са най-близки във векторното просранство до векторното представяне на запитването.

В зависимост от използвания подход, алгоритъмът може да прилага метрики за подобие като косинусово подобие (cosine similarity) или евклидово разстояние (euclidean distance), за да определи най-релевантните фрагменти. Тези методи позволяват бързо и ефективно измерване на семантичната близост между заявката и документите, като гарантират висока точност при извличането на информация.

Косинусовото подобие между вектор на въпрос q и вектор на един документен фрагмент d се изчислява като:

$$\text{similarity}(q, d) = \cos(\theta) = \frac{q \cdot d}{\|q\|\|d\|},$$

където:

$q \cdot d$ е скаларното произведение на векторите,

$||q||$ и $||d||$ са дължините (нормите) на векторите.

Колкото по-близо до 1 е резултатът, толкова по-голямо е семантичното съответствие между въпроса и фрагмента.

Формулата на Евклидово разстояние:

$$\text{distance}(q, d) = \sqrt{\sum_{j=1}^{n} (q_j - d_j)^2}$$

където:

- $n \in N$ е размерността на векторното пространство;
- q_j и d_j са съответно j-тите компоненти на векторите q и d;

Колкото по-малка е стойността на евклидовото разстояние, толкова по-близки са векторите в семантичното пространство.

Алгоритъмът използва и параметър K, който определя броя на фрагменти, които трябва да бъдат върнати като резултат. Вместо да избира само един най-релевантен фрагмент,

алгоритъмът извлича K на брой фрагмента с най-голямо косинусово подобие или най-малко евклидово разстояние спрямо запитването.

В случай, когато метриката на подобие е косинуво подобие:

$$D_{\text{best}} = \arg \max_{D \subseteq \{d_1, \ldots, d_n\},\ |D| = k} \sum_{d_i \in D} \text{similarity}(q, d_i),$$

а когато е Евклидово разстояние:

$$D_{\text{best}} = \arg \min_{D \subseteq \{d_1, \ldots, d_n\},\ |D| = k} \sum_{d_i \in D} \text{distance}(q, d_i),$$

където: D_{best} е множество от k фрагмента.

EG. Релевантни фрагментирани документи

Релевантните фрагменти на документ са тези текстови сегменти, които алгоритъмът за търсене определя като най-близки по семантично значение за текущото задание (E1). Те се добавят към CL (Наблюдения), така че LLM да може да ги използва при генерирането на аргументация, заявки към база данни или крайния отговор на системата.

EH. Векторно представяне на фрагментирани документи.

Векторните представяния на документи представляват числови вектори, създадени за всеки фрагмент от документ. Те се използват за улесняване на процеса на търсене и сравнение в RAG модула. Тези представяния съхраняват семантичната структура и смисъла на текста, което позволява на системата ефективно да определи релевантността на всеки фрагмент спрямо зададеното задание.

1. АЛГОРИТЪМ ЗА РЕАЛИЗИРАНЕ НА СИСТЕМАТА

Програмните кодове са публично достъпни на следните репозиториуми:

- React SQL програмен код за решаване на 3те варианта:

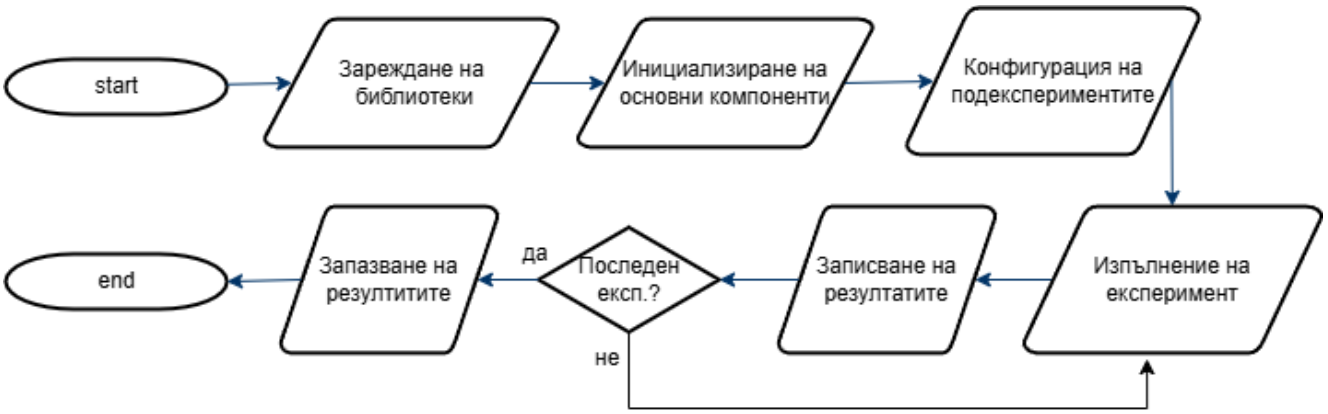
  https://github.com/sergeyvi4ev/react-sql-agents

- RAGSQL програмен код за провеждане на експериментите:

  https://github.com/sergeyvi4ev/ragsql-paper

Общата структура на алгоритъма на програмния код е представена на Фиг. 14

Фиг. 14 Общата структура на алгоритъма



Източник: Автор

ГЛАВА III: СИМУЛАЦИИ И ЕКСПЕРИМЕНТИ С АВТОМАТИЗИРАНИ СИСТЕМИ ЗА РЕШЕНИЯ ОТ БАЗИ ДАННИ

В настоящата глава са включени две независими изследвания, които имат следните задачи:

- Провеждане на изследване за принципно приложение на система за извличане на знания от бази данни с аргументация. Основната идея е принципна проверка за възможността такава система да работи, използвайки предложена в методическата част на тази дисертация архитектура на системата. Изследването да включи 3 различни подхода:
    - ReAct текст-към-SQL LLM агент без памет
    - ReAct с контекстуална памет
    - React с многоагентна система

Във второто изследване да се изследват аспектите, които са най-критично важни за работа на подобен тип система, а именно:

- Специализиращо обучение на модел за вграждане за текст-към-SQL приложения;
- Оценка на модели за вграждане при системи за извличане на знания предназначени за текст-към-SQL приложения.

Използваните данни за изследванията в раздел 13 е база данни от SQL Murder Mystery, а в раздел 14 се използват напълно различни данни, като се използват анотациите от еталон за сравнение BIRD, съдържащ 80 бази данни от 37 различни бизнес области на приложение.

ПРИНЦИПНО ПРИЛОЖЕНИЕ НА СИСТЕМА ЗА ИЗВЛИЧАНЕ НА ЗНАНИЯ ОТ БАЗИ ДАННИ С АРГУМЕНТАЦИЯ

Обща постановка на заданието

В този раздел се цели да се оцени способността за решаване на задачи от LLM в контекста на образователна задача. Конкретната задача, която се оценява, е стандартно учебно упражнение, обикновено задавано на студенти, които посещават курсове по SQL.

Задачата, известна като "Мистериозно Убийство на SQL" (SQL Murder Mystery), включва решаването на измислено убийство чрез използване на SQL заявки и предоставена база данни, съдържаща съответни записи, свързани с престъплението (Canale и Farinetti, 2021),. Целта на това упражнение е да се идентифицира убиецът чрез заявки и анализ на информацията, съхранена в базата данни.

Базата данни по своята структура не е по-различна от класически бизнес насочени бази данни. Броят на записите в базата данни е с общо ~55 хил., а броят колони - 37 колони, което може по размер е съпоставимо с база данни на малко и средно предприятие.

Използваме този конкретен казус, защото вече е добре измислен и многократно изпробван. В хода на играта участниците постепенно разкриват следващите стъпки от задачата, което прави преживяването интригуващо и естествено.

Начинът на поставяне на задачата е такъв, че изисква от участника да открие сам същността ѝ в хода на действието. Първоначално не е ясно дори какво точно е престъплението – играчът трябва сам да стигне до това заключение, след което да идентифицира убиеца. След разкриването на извършителя, постепенно се установява и кой стои зад него като подбудител. Това я превръща в многослойна задача, при която всяко ново разкритие води до следващо ниво на разбиране и анализ. Подходът ангажира логическото мислене и насърчава активно търсене на връзки между уликите.

Всичко това го прави много подходящо за експериментиране с LLM агенти, тъй като задачата изисква последователно разсъждение, планиране, работа с контекст и способност за интерпретиране на частична информация.

Чрез оценката на изпълнението на LLM агент или агенти на тази задача се цели да се получат представи за способностите за аргументация на тези модели, когато се прилагат за извличане на информация от бази данни с помощта на техники текст-към-SQL.

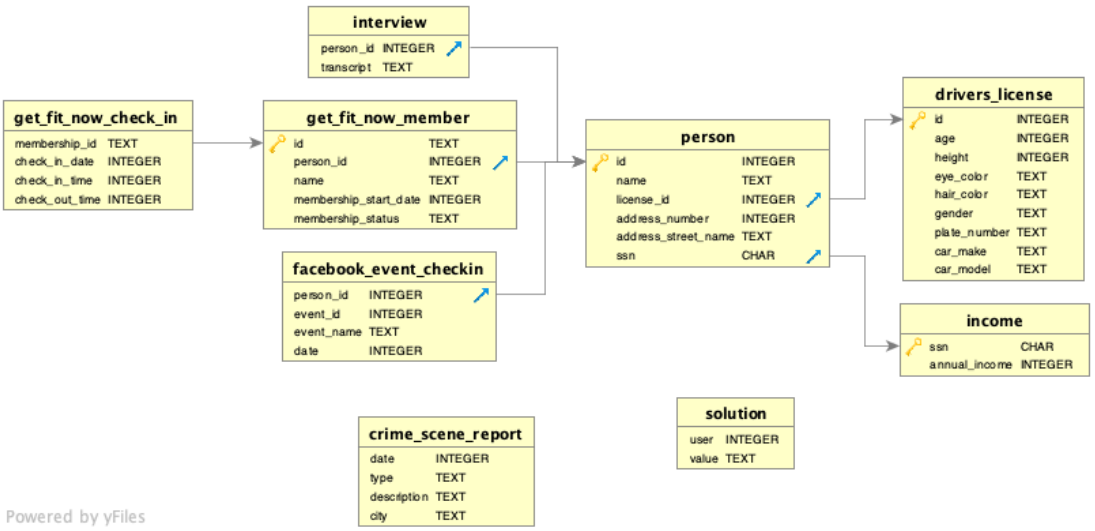
За целите на това изследване са избрани три различни подхода за експерименти:

1. ReAct текст-към-SQL LLM агент без памет
2. ReAct текст-към-SQL LLM агент с памет
3. Многоагентна LLM система с ReAct текст-към-SQL LLM агент

Използваните модели за експериментите са LLM предоставени от OpenAI с различна способност.

SQL схемата на базата данни е представена по-долу (Фиг. 15)

Фиг. 15 SQL схема на базата данни.



Източник: Canale и Farinetti, 2021

Очаква се тези таблица да съдържат данни, които могат да бъдат заявени и анализирани чрез SQL за разкриване на улики и решаване на мистериозното престъпление.

Задачата представлява че в движение научаваш за нови задачи. Още в първата стъпка си поставяш нова задача. Подходяща за проверка на способностите за аргументация.

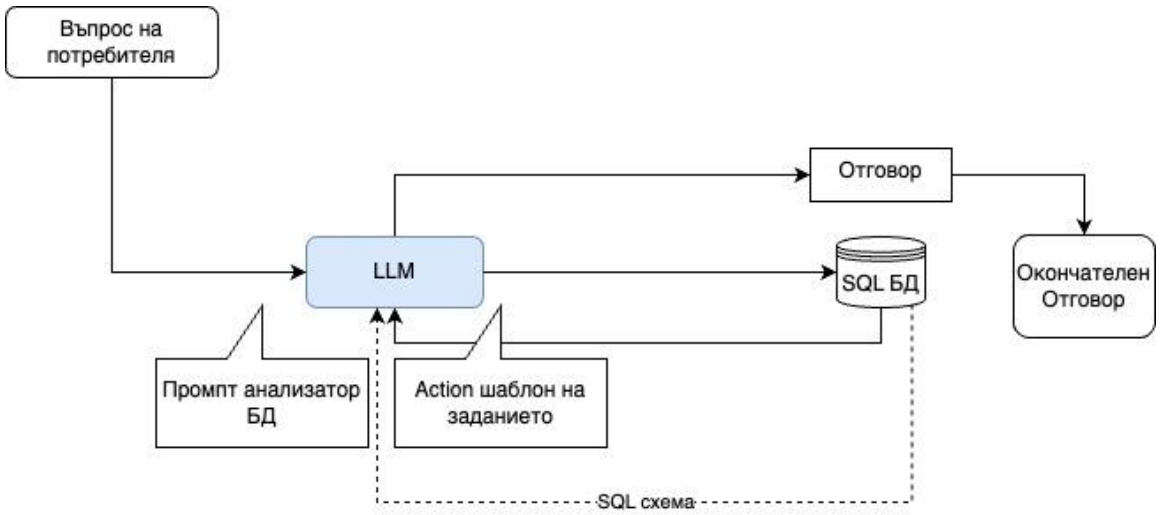
1.1. ReAct текст-към-SQL LLM агент без памет

1.1.1. Постановка на експеримента

В първата група експерименти бе приложен подходът с ReAct текст-към-SQL LLM агент без памет. Схемата (Фиг. 16) представлява архитектурата на информационния поток на една итерация, която се състои от следните компоненти:

1. **Въпрос на потребителя**.

Фиг. 16 Процесът на работа на ReAct .



Източник: автор

2. **LLM SQL агент**: На този етап LLM SQL агент генерира мисловна верига и създава SQL код на базата на тях.

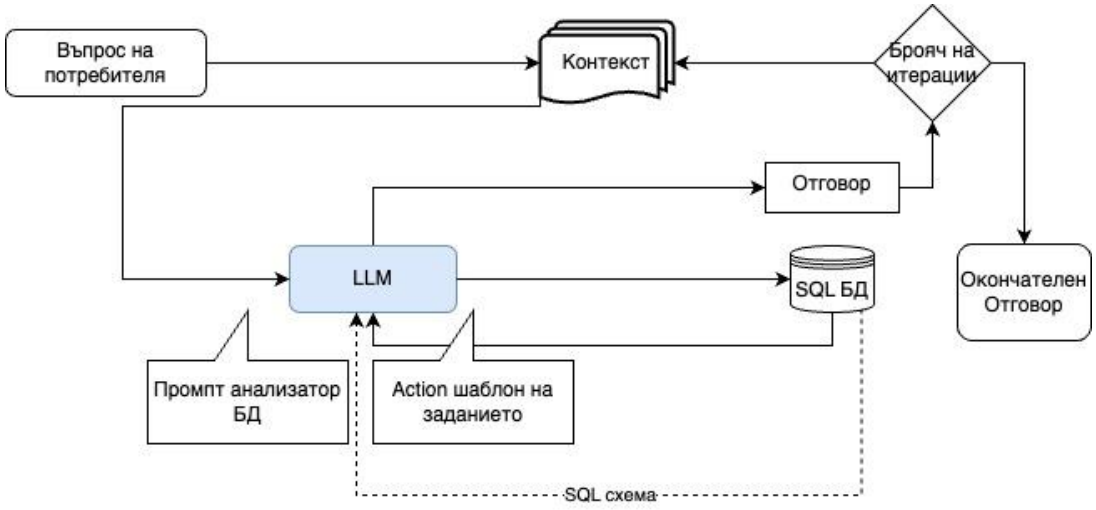
3. **SQL БД**: SQL база данни, от която се извлича информацията.

4. **Отговор**: това е финалният отговор на LLM агента.

5. **ReAct шаблон на заданието**: шаблонът на заданието се използва за проектиране на задания ("prompt engineering") и се подава заедно с въпроса на потребителя (1).

7. **Action шаблон на заданието**: Това задание е използвано за генериране на текст-към-SQL и извличане на резултата от базата данни. Функцията, която се използва, предава информация за диалекта на SQL в зависимост от типа на базата данни и информацията за таблиците, присъстващи в базата данни, което помага за генерирането на коректния синтаксис на SQL заявките.

Фиг. 19 Процесът на работа на ReAct агент с памет.



Източник: автор

Благодарение на добавения контекст в този експеримент LLM агентът успява да стигне до ключов липсващ факт от базата данни, като конкретно проверява регистрационния номер на автомобила. Това води до окончателен и обоснован отговор, който представлява правилното решение на задачата.

Въпреки че LLM агентът демонстрира висока степен на автономност в извличането на релевантни факти и формирането на аргументирани заключения, при увеличението на контекста се наблюдава влошаване на генерираните SQL заявки.

1.2. React с многоагентна система

Настоящият експеримент е насочен към разработването на изцяло автономна система, която цели поетапно решаване на задачата. Целта е да се създаде интелигентна многоагентна система, способна да извършва сложни анализи и аргументация, използвайки само релевантен контекст при генериране на SQL.

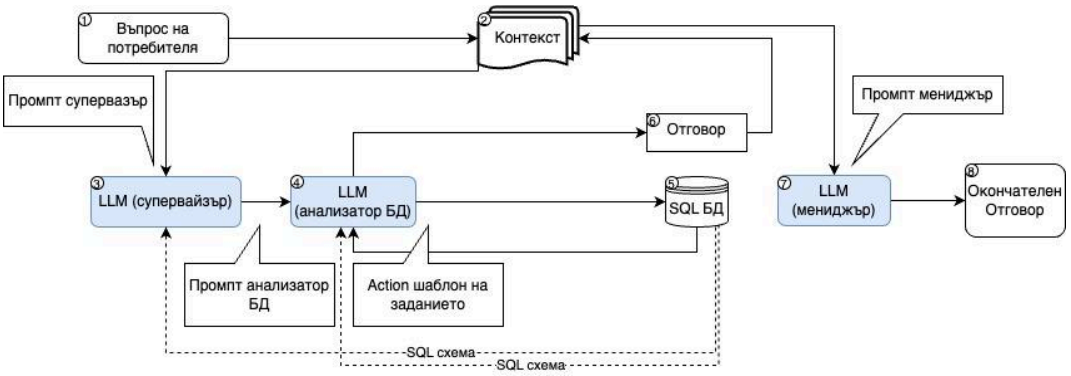
Този експеримент надгражда група експерименти 2, в който въпреки високата степен на автоматизация, прекалено големият контекст влияеше върху качеството на генерираните SQL заявки. В новата фаза на изследването се цели елиминиране на тези зависимости, като се разработи по-усъвършенстван механизъм за дългосрочна памет и адаптивно вземане на решения.

За тази цел се въвежда многоагентна система, като се добавят 2 допълнителни LLM агента, които:

- да дават предложения за следваща най-разумна стъпка и
- да анализират дали цялата информация до момента, е достатъчна да се постигне крайният резултат.

По този начин, създавайки многоагентна автоматизирана система.

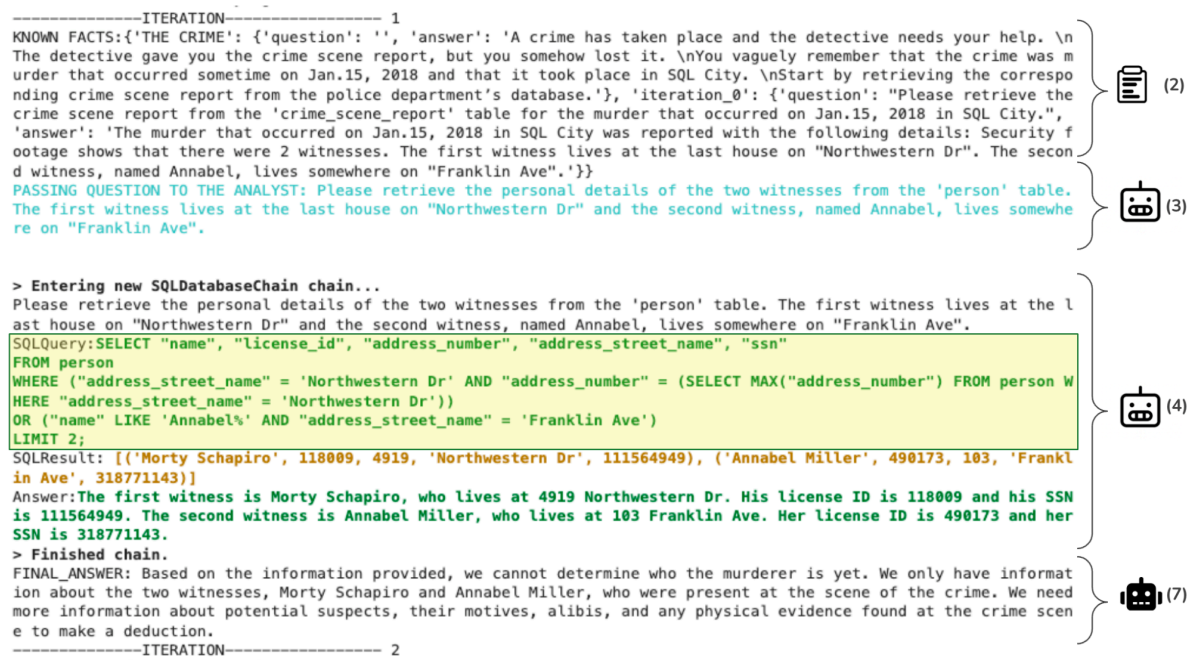
Фиг. 27 Потокът на информация (експеримент 3)



Източник: Vichev, 2024

Фиг. 27 по-горе визуализира информационният поток между LLM агентите в построената система за експеримент 3. Всеки елемент е нумериран от 1 до 8.

Фиг. 28 Диалог между LLM агентите (пример 1)

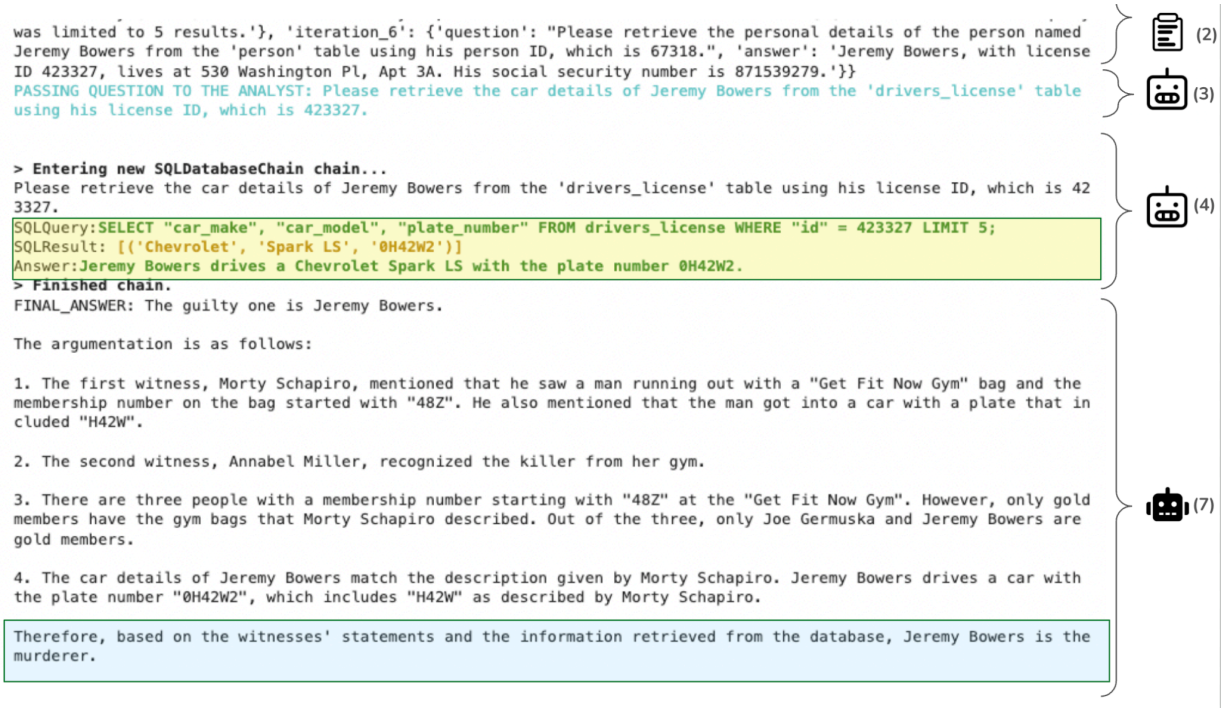


Източник: Vichev, 2024

Диалогът от началото до края на една цяла итерация между LLM агентите (а именно втората итерация) е показан в примера по-горе (Фиг. 28). Докладът и агентите са номерирани с препратка към Фиг. 27.

В следващата итерация (Фиг. 30) екипът успява да зададе такъв въпрос, който да провери предположението, да генерира SQL заявка, която да извади съответната информация и да стигне до заключението.

Фиг. 30 Диалог между LLM агентите (пример 3)



Източник: Vichev, 2024

1.3. Обобщение на изследването

Това изследване се фокусира върху способността на системите, използващи генеративен изкуствен интелект, да решават сложни задачи на база информация получена от бази данни с използване на метода ReAct. Резултатите са забележителни, особено при добавяне към системата функция на памет и при добавяне на допълнителни агенти.

Проведени са тридесет различни експерименти от 3 различни подхода:

- ReAct текст-към-SQL LLM агент без памет
- ReAct с контекстуална памет
- Многоагентна LLM система с ReAct текст-към-SQL LLM агент

В сценариите, при които LLM а трябва да изпълни задачата без памет, се установява, че LLM агентът не успява да намери окончателно решение в повечето случаи. Основните проблеми в тези случаи включват преждевременно прекъсване на итерациите или генериране на грешни SQL заявки.

Резултатите се различават значително, когато към LLM агента добавяме включен контекст на паметта. В този случай LLM успява да намери решения за 6 от 10 задачи, което демонстрира значително подобрение в сравнение с предишната група експерименти.

Най-добрият резултат, получаваме при многоагентна структура, в която системата от агенти се справят в 7 от 10-те експеримента.

Въпреки подобренията, определени грешки все още са често срещани, включително случаи на некоректни SQL заявки, неправилни отговори и ситуации, в които моделът поема в неправилна посока при обработката. Тези резултати подчертават предизвикателствата при автоматизирането на текст-към-SQL задачи с помощта на LLMs, но също така разкриват техния потенциал, особено когато задачите се разделят на поетапни операции и се допълват с контекстна памет. Обобщение на резултатите е представено в Таб. 2.

Таб. 2 Обобщение на резултатите от изследването.

Група експерименти	**Резултат**
1.ReAct без памет	0/10
2.ReAct с памет	6/10
3.Многоагентна структура	7/10

Оценяване:

Експериментите се оценяват на база достигане на финален отговор:

- True Positive: крайният отговор е даден и е верен: 1 т.
- True Negative / False Positive: крайният отговор липсва или е грешен: 0 т.

В заключение, направените изводи подчертават значението на поетапния подход при решаване на проблеми и влиянието на контекста на заданието върху представянето на LLM. Бъдещата работа включва изследване на алтернативни техники за аргументация, специализиращо обучение на отворени LLM модели и допълнително усъвършенстване на подходите за създаване на задания.

В контекста на оценяването - при успех системата предоставя краен, конкретен и верен отговор на зададения въпрос. При неуспех операторът получава частични сведения, които могат да подпомогнат решението със следващ промпт. Този аспект демонстрира, че дори в случаи на неуспех системата запазва помощна роля, вместо да бъде напълно нефункционална. Това от друга страна предполага, че ролята на човека в процеса може да е от съществено значение за оптимизиране на взаимодействието и повишаване на ефективността на системата.

LLM агентите представляват мощна AI технология, способна да подпомага човешки задачи. Техните аргументационни способности са очевидни, въпреки че резултатите все още не могат да бъдат напълно надеждни във всички възможни сценарии. (Vichev, 2024)

LLM агентите все още могат да правят неверни твърдения и да предоставят грешни съвети. Този проблем става особено значим, когато моделите изпълняват сложни аргументационни вериги, особено в комбинация със задачи за генериране на код, което увеличава риска от подвеждане на потребителите. Доверието на потребителите към подобни системи, особено за критично важни задачи, са от съществено значение. Една потенциална бъдеща насока за развитие е специализиращото обучение на LLMs за задачи, ориентирани към конкретни области на приложение, с цел повишаване на тяхната точност и надеждност.

При верижни задачи, където началните изводи определят посоката на аргументацията, проверката на фактите от човек е от съществено значение, особено в ранните етапи.

По отношение на бъдещата работа съществуват няколко възможни насоки за изследване. Една потенциална посока е проучването на алтернативни съвременни техники за аргументация, като Tree of Thought (ToT) или self-checking, с цел допълнително подобряване на аргументационните способности на LLM при тази задача.

Допълнително, по-нататъшното усъвършенстване и изследване на техники за създаване на задания (prompt engineering) може да подобри аргументацията и способността за генериране на код на LLM. Непрекъснатата оптимизация и адаптация на заданията могат да доведат до значително повишаване на общата ефективност на решението.

ИЗСЛЕДВАНЕ НА КРИТИЧНИТЕ КОМПОНЕНТИ НА RAGSQL СИСТЕМА.

Обща постановка на заданието

Докато RAG предлага обещаващ подход чрез динамично извличане на релевантна информация въз основа на потребителското запитване, като по този начин намалява натоварването върху самото задание, този метод въвежда нови усложнения. Например, ефективността на RAG зависи в голяма степен от релевантността на извлечената информация, което от своя страна е обусловено от качеството на използваните алгоритми за търсене по подобие. Въпреки че за задачата на извличане на информация (Information Retrieval) съществуват различни еталони за сравнение (benchmarks) от различни области, конкретното приложение на информационното извличане върху текст-към-SQL данни остава слабо изследвано. Наблюдава се съществена празнота в методите за оценяване, които са специално насочени към анализ на въздействието на търсенето по подобие при извличането на релевантна информация при генерирането на SQL заявки.

Фиг. 31 Текст-към-SQL данни - прилики и разлики

Question:	Who is the CBSA officer of the post point in the area with the highest number of employees?
Evidence:	CBSA officer refers to CBSA_name; the highest number of employees refers to MAX(employees)
SQL query	SELECT T1.CBSA_name FROM CBSA AS T1 INNER JOIN zip_data AS T2 ON T1.CBSA = T2.CBSA WHERE T2.employees = (SELECT MAX(employees) FROM zip_data)

Легенда: червен цвят отбелязва токени за личност или име на компания; син - сравняване или агрегиране; жълт - географски термини.

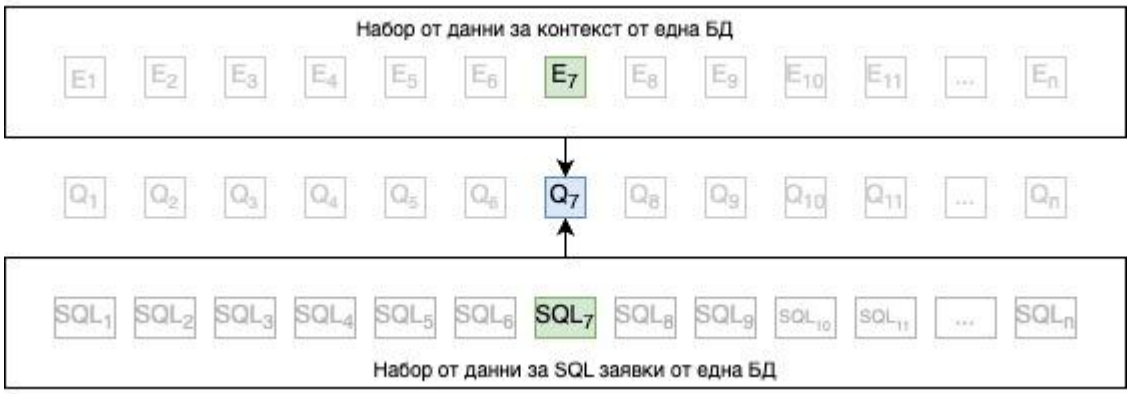
Източник: Vichev & Marchev, 2024

Фиг. 31 представлява примерен запис от масива с данни, използван за експериментите. С цветови код са отбелязани сходства между думите в един зададен въпрос от потребителя и синтаксиса на една SQL заявка, която позволява да се отговори на този въпрос, както и често срещани практики при кръщаване на колони в SQL бази данни.

В експерименталната постановка основната задача на системата е намирането на релевантният контекст (E_i) или релевантната SQL заявка (SQL_i) за съответен въпрос (Q_i) от всички възможни записи за контекст (E) или SQL заявки (SQL) за съответната база данни. За всеки въпрос съществува само един правилен контекст или една правилна SQL заявка, която се приема за тази, предоставена от BIRD набора от данни. По този начин, наборът от данни съдържа n “правилни” двойки въпрос-контекст (Q_i-E_i) или въпрос-SQL заявка (Q_i-SQL_i) и n-1 на брой некоректни контекстуални данни или SQL заявки за всеки даден въпрос (Фиг. 32).

Наборът от данни за контекст и SQL заявки, от който се търси релевантният запис се формира отделно за всяка база данни.

Фиг. 32 Типове данни в експерименталната постановка



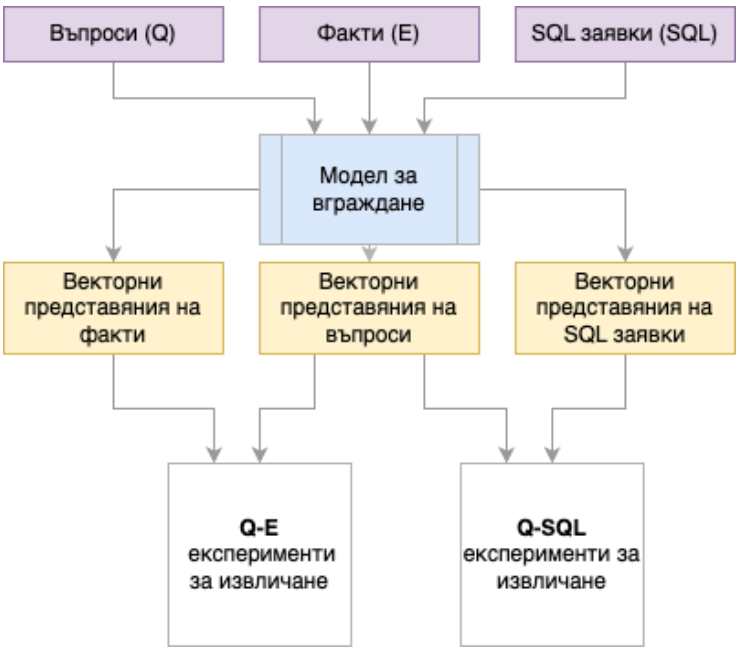
Източник: Автор

Провеждат се две групи експерименти (Фиг. 33):

A. Въпрос-Контекст (Q-E): Този експеримент включва извличане на най-релевантните записи за контекст от наличния набор въз основа на векторните представяния (embeddings) на въпросите.

B. Въпрос-SQL заявка (Q-SQL): По аналогия, този експеримент включва съпоставяне на въпроси с най-релевантните SQL заявки въз основа на техните векторни представяния.

Фиг. 33 Методика на оценка на модели за вграждане



Източник: Vichev & Marchev, 2024

1.4. Данни

За целта на експериментите, се адаптира наборът от данни от широко приетия междуобластен еталон за сравнение - BIRD, който е предназначен за мащабна оценка на текст-към-SQL системи и обхваща над 37 професионални области (като търговия, здравеопазване, технологии, образование, спорт и др), 12751 уникални двойки въпрос-SQL заявка и 95 бази данни.

1.5. Специализиращо обучение на векторизиращ модел

Използва се предварително обучен базов модел - “all-MiniLM-L6-v2”, който разполага с компактна архитектура с 22.7 милиона параметри, оптимизирани за генериране на векторни представяния. Този модел се подлага на специализиращо обучение за целта на експеримента, за да адаптира способностите си по-точно към специфичните изисквания на генерирането на SQL заявки от заявки на естествен език. (Vichev & Marchev, 2024)

Параметри на специализиращото обучение:

Данни: специализиращото обучение се провежда върху подмножество от набора от данни на еталона за сравнение - BIRD, който включва 5 269 положителни двойки. Тези двойки

се състоят от съответстващи въпроси на естествен език и съответстващия им контекст или SQL заявки, като по този начин се гарантира, че моделът усвоява директната връзка между текстовете на въпросите, контекста и SQL командите.

Функцията на загуба (loss function): Използва се Multiple Negatives Ranking Loss с косинусово подобие (cosine similarity) като метрика за разстояние. Този избор е насочен към подобряване на способността на модела да разграничава тясно свързани текстове (въпроси и контекст, въпроси и SQL заявки), като моделът се фокусира върху приближаване на положителните примери в пространството на векторните представяния и отдалечаване на несвързаните (отрицателни) примери.

Използването на Multiple Negatives Ranking Loss е особено подходящо в задачи, при които имаме само положителни двойки – например парафрази, двойки въпрос-отговор, (заявка-отговор) или език източник, език цел (Henderson et al., 2017). Освен това, тази функция за загуба е ефективна в контексти с много големи множествени пространства на кандидатите, където пълно изчисляване на вероятностите е непрактично и е необходимо приближение чрез негативни примери (Henderson et al., 2017). Също така, тя е приложима когато целта е не класификация, а директно ранжиране на правилните отговори по-високо от неправилните, което по-добре отговаря на реалните задачи за търсене и извличане на информация (Henderson et al., 2017). Поради този ред причини, тази функция на загуба е предпочитана и подходяща за специализиращото обучение.

Използване на хардуер: Обучението се извършва на T4 GPU с 15 GB RAM, което позволява ефективна обработка на изчисленията на модела и зареждането на данни.

Режим на обучение: Моделът преминава през 10 епохи на обучение с 43 стъпки за загряване, за да се гарантира постепенното регулиране на скоростта на учене, което оптимизира процеса на обучение за по-добра сходимост.

Оценка на моделите

Моделите се оценяват в два различни сценария, за да се анализират тяхната гъвкавост и устойчивост:

- **Оценка в рамките на същата базата данни (In-DB)**: Този сценарий включва тестване на моделите върху същите бази данни, които са използвани по време на специализиращото обучение. Той проверява представянето на модела в познати контексти, използвайки отделено подмножество от обучително-валидационните записи.
- **Оценка извън базата данни (Out-of-DB)**: В този сценарий моделите се тестват върху изцяло нови бази данни, които не са били част от обучителните данни. Този подход е от

съществено значение за оценката на способността на модела да се обобщава върху различни схеми на данни и SQL заявки.

Тези оценъчни настройки допринасят за разбирането както на ефективността на модела в условия, сходни с тези на обучителната среда („In-DB“), така и на неговата способност да се адаптира към нови, непознати среди („Out-of-DB“).

Анализ на резултатите

1.5.1. Първоначални резултати без специализиращо обучение

Чрез използваната методология за оценка се докладват първоначалните резултати за избрани модели в група експерименти A (въпрос-контекст) и експерименти B (въпрос-SQL заявка). За моделите с отворен код (open-source) се включва размерът на моделите (в милиони параметри) и размерността на векторното представяне на всеки модел (Таб. 4).

И в двата експеримента със SQL заявки и въпроси най-висок резултат се постига от в моделите на OpenAI. Освен това, наблюдава се по-висок диапазон на резултатите в оценяването на двете групи експерименти в сравнение с MTEB еталон за сравнение, което показва разликата в сложността на задачите.

Таб. 4 Първоначални резултати без специализиращото обучение

Името на модела	Size	Emb. Dim.	MTEB bench.	Q-E	Q-SQL
all-mpnet-base-v2	110	768	43.81	68.13	64.46
all-MiniLM-L6-v2	23	384	41.95	69.63	74.66
OpenAI-3-small	-	1536	51.08	81.9	**79.51**
OpenAI-3-large	-	3072	**55.44**	**88.05**	79.46
OpenAI-ada-002	-	1536	49.25	82.54	80.62
Cohere-eng-v3	-	1024	55	80.09	68.18

Забележка: Метриката е nDCG@10. Q-E: Група експерименти Въпрос-Контекст. Q-SQL: Група експерименти Въпрос-SQL заявка.

Източник: Vichev & Marchev, 2024

Резултати след специализиращото обучение

В това изследване се оценява моделът с отворен източник (open-source) all-MiniLM-L6-v2, след изпълнено специализираното обучение, който е най-малкият модел сред избраните за експеримента. Показано е, че моделът генерира значими и контекстуално богати

векторни представяния, съществени за динамичното извличане на информация. Методологията за оценка беше стандартизирана за всички модели, за да се гарантира сравнимост на резултатите.

Като се има предвид, че наборът от данни включва единствено положителни двойки, за специализиращото обучение е използвана функцията на загуба Multiple Negatives Ranking Loss. Тази функция е оптимална за набори от данни без присъщи негативни примери, тъй като принуждава модела да различава положителен пример от множество негативни примери в рамките на същия пакет (batch).

Ефективността на специализирания модел върху заданията на експериментите е демонстрирана в сравнителната таблица с резултати по-долу (Таб. 5). Получените модели след специализираното обучение са RAGSQL-text и RAGSQL-code.

Като се има предвид, че наборът от данни включва единствено положителни двойки, за специализиращото обучение е използвана функцията на загуба Multiple Negatives Ranking Loss. Тази функция е оптимална за набори от данни без присъщи негативни примери, тъй като принуждава модела да различава положителен пример от множество негативни примери в рамките на същия пакет (batch).

Ефективността на специализирания модел върху заданията на експериментите е демонстрирана в сравнителната таблица с резултати по-долу (Таб. 5). Получените модели след специализираното обучение са RAGSQL-text и RAGSQL-code.

Таб. 5 Резултати след специализиращо обучение

Експеримент	**Въпрос-Контекст (Q-E)**		**Въпрос-SQL заявка (Q-SQL)**	
Модел	**In-DB**	**Out-of-DB**	**In-DB**	**Out-of-DB**
all-mpnet-base-v2	65.11	63.59	67.69	70.93
all-MiniLM-L6-v2	75.55	70.23	68.86	71.56
OpenAI-3-large	80.27	75.08	87.85	88.1
OpenAI-3-small	80.53	75.01	81.56	81.34
OpenAI-ada-002	81.62	76.53	82	83.17
Cohere-eng-v3	80.66	76.36	67.49	70.53
RAGSQL-text	**91.80**	**84.99**	84.5	83.07
RAGSQL-code	85.21	77.91	**93.02**	**89.62**

Забележка: Метрика: nDCG@10. Оценките в **удебелен шрифт** показват най-доброто представяне във всяка категория. In-DB: обозначава набора за валидация при обучението със същите БД. Out-of-DB: представлява набора за валидация при обучението с нови БД.

Източник: Vichev & Marchev, 2024

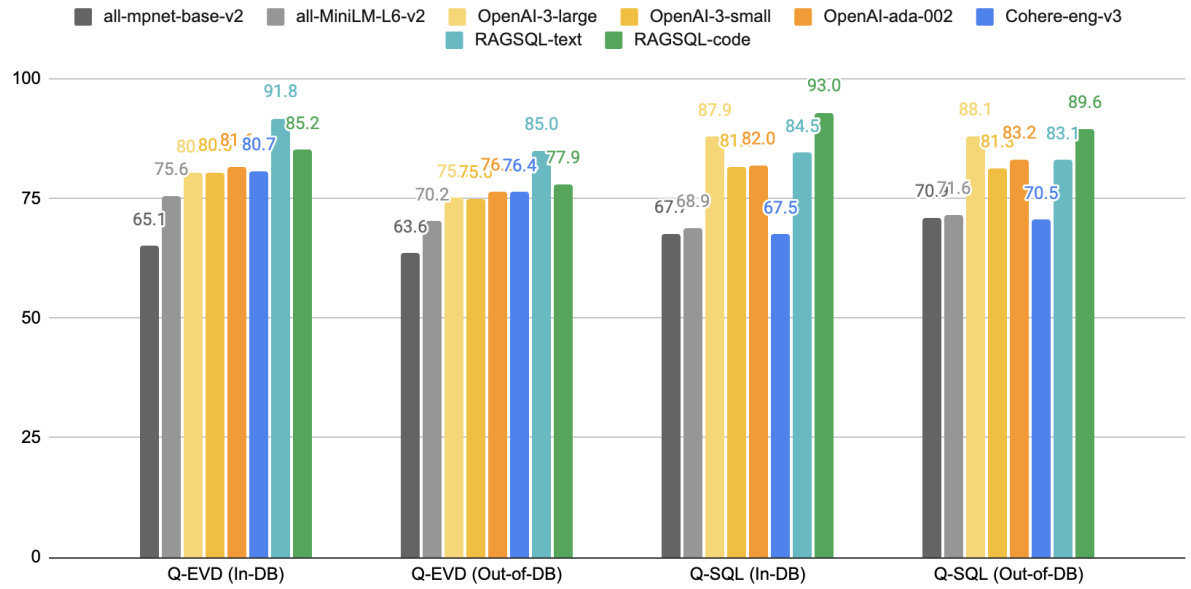
RAGSQL-text моделът е специализиращо обучен с данни от група “въпрос-контекст”. Предназначението на такъв тип модел е да извлича релевантна информация, като например:

- спецификите на наименованията за съответната базата данни;
- релевантни термини и описание на таблици
- примери и насоки за правилния синтаксис
- специфични категории на данни.

Съответно, моделът RAGSQL-code е специализиращо обучен с данни от група въпрос-SQL заявка. Предназначението на такъв тип модел е да извлича релевантни примери на SQL заявки, които биха помогнали на LLM-ите да обогатят примерите в заданието, и като крайна цел да увеличат вероятността да се състави коректна SQL заявка.

При двата получени модела, се надгражда способността им да генерират такива вектори за представяне, че те могат по-ефективно да идентифицират релевантни зависимости между въпросите и контекстната информация или SQL код. Това надграждане позволява на моделите по-прецизно да извличат релевантни данни. (Vichev & Marchev, 2024)

Фиг. 34 Резултати след специализиращо обучение (nDCG@10)



Източник: Автор

Диаграмата по-горе (Фиг. 34) визуализира информацията от табличен вид и представя четирите подгрупи експерименти.

Специализирано обучените модели RAGSQL-text и RAGSQL-code превъзхождат всички съвременни модели, избрани за сравнение, по всички оценявани метрики в съответната група, което показва висока точност при извличането на най-релевантната информация. Освен това, моделите демонстрират стабилно представяне, както с информация от същите бази данни, от които са взети данни за трениране (In-DB), така и с информация от БД, които са абсолютно нови за получените модели (Out-of-DB). Това говори за приложимостта на получените модели извън създадената експериментална среда.

1.6. Обобщение на изследването

Това изследване разглежда ролята на RAG и моделите за вграждане за подобряване на текст-към-SQL приложения, в частност в контекста на LLM-и, които генерират SQL заявки от входни данни на естествен език. Чрез прилагане на специално разработената методология за оценка и адаптиране на данни от широко признатия еталон за сравнение - BIRD, това изследване сравнява съществуващите модели за вграждане и представя специализирано обучен модел за вграждане с отворен достъп, който превъзхожда едни от най-съвременните модели избрани за изследването, както при експериментите за извличане на данни тип “въпрос-контекст”, така и при “въпрос-SQL заявка”.

Адаптацията на BIRD и представената в тази статия методология за оценка могат да служат като надежден метод за оценяване на качеството на други модели за вграждане, фокусирани върху текст-към-SQL задачи. Представеният методологичен подход и акцентът върху оценяването както в in-database, така и в out-of-database среда демонстрират цялостен метод за тестване и валидиране на модели, осигурявайки надеждност и приложимост на резултатите в различни бази данни.

Получените модели за вграждане RAGSQL-text и RAGSQL-code, в резултат на специализирано обучение, показват добра обобщаваща способност в множество професионални сфери. Получените резултати показват, че значително по-малки модели с ефективно специализиращо обучение могат да доведат до по-висока производителност за конкретната задача. След като доказват превъзходството си, получените модели не се ограничават до използвания набор от данни и използваните БД, а могат да бъдат приложени в различни бизнес сценарии от реалния свят, изискващи текст-към-SQL функционалности. (Vichev & Marchev, 2024)

Проведеното изследване показва, че специализирано обучените модели за вграждане могат ефективно да се справят с данни от в областта на текст-към-SQL приложения в различни сфери, водейки до значителни подобрения. Тези констатации са в съответствие с последните

изследвания, които подчертават потенциала на специализирано обучените модели за подобряване на NLP задачите чрез устойчиви стратегии за адаптация към област на приложение.

Заедно с резултатите са споделени данните, използвани за обучение, както и получените модели.

- RAGSQL-text модел и даннни:

https://huggingface.co/sergeyvi4ev/all-MiniLM-RAGSQL-text

https://huggingface.co/datasets/sergeyvi4ev/questions_to_sql_triplets

- RAGSQL-code модел и данни:

https://huggingface.co/sergeyvi4ev/all-MiniLM-RAGSQL-text

https://huggingface.co/datasets/sergeyvi4ev/questions_to_evidence_triplets

ЗАКЛЮЧЕНИЕ

В изследването са разгледани и обобщени най-съществените концепции за LLM, методите за векторизиране и идеите за многоагентни системи, формирайки теоретичния фундамент на дисертационния труд.

Разгледани и синтезирани бяха базовите концепции, свързани с машинно обучение, дълбоки невронни мрежи и големи езикови модели (LLM), като особено внимание бе отделено на трансформър архитектурите. Анализът показа, че LLM могат да се разглеждат като основен градивен елемент за създаване на автоматизирани системи, които работят върху неструктуриран текст и данни от бази данни, благодарение на способността им да улавят дълбоки семантични зависимости.

Допълнително, бяха дискутирани множество подходи за векторно представяне (embeddings), включително методи за двупосочна трансформър обработка (BERT) и авторегресионни генеративни модели (GPT). Въз основа на тези анализи беше констатирано, че многоагентните архитектури позволяват разпределена координация между отделни LLM агенти, повишавайки ефективността при решаването на по-комплексни проблеми.

Проучването откри, че ключов ограничителен фактор е рискът от „халюцинации“ и пристрастия в изходите на LLM, което налага допълнителни механизми за валидиране и етичен контрол. Изследваните теоретични постановки подчертаха важността на Retrieval Augmented Generation (RAG) като подход, позволяващ да се съчетаят големите езикови модели с реални бази от знания, което намалява вероятността от неверни отговори. Тези изводи изградиха стабилна теоретична платформа за по-нататъшно разработване на методики, свързани със специализиращо обучение и оценка на производителността на системите.

Формулирана е методологична рамка, която показва конкретните стъпки за проектиране и реализация на многоагентна система с LLM и бази данни, обосновавайки архитектурните избори и техническите решения.

На базата на разгледаните теоретични основи бе разработена детайлна методология, която стъпва на няколко ключови стълба: дефиниране на архитектурата, избор на подходящи LLM агенти и механизми за контролирано взаимодействие с базите данни. Обосновано бе, че системата следва да включва поне три основни модула – LLM агент(и), RAG модул за извличане на релевантно знание и процесинг модул за SQL заявки, за да се осигури надеждно и ефективно свързване на естествения език с табличната информация в бази данни.

Във втората глава на дисертацията е предложена блокова диаграма, илюстрираща процеса на обработка на потребителска заявка, при който LLM агентът извършва анализ на

входа, RAG компонентът осъществява извличане на данни от базата, след което се генерира отговор, базиран на реални факти. Специално внимание бе обърнато на механизма ReAct (Reason+Act), при който агентът динамично редува между разсъждения (chain-of-thought) и конкретни операции (например SQL заявки), осигурявайки прозрачност на стъпките при решаването на задачи.

Методологията включва и препоръки за използване на многоагентна организация, където няколко LLM агенти си поделят отговорности (напр. един фокусиран върху лингвистична анализа, а друг – върху векторно търсене). Въведени са критерии за оценка на отделните компоненти, обхващащи както точност на извличане, така и бързодействие, което позволява мултикритериален анализ на крайната система.

Създадени и проучени бяха различни LLM агенти, показвайки как архитектурните и функционалните разлики влияят върху крайните резултати.

В съответствие с методологията, бяха създадени няколко варианта на LLM агенти – едни, които използват ReAct без памет, други с контекстуална памет и трети, организирани в многоагентна структура. Основната разлика между тези вариации е в начина, по който агентът управлява аргументацията (reasoning) и как съхранява предишна информация (memory), за да дава по-точни отговори при последователни заявки.

Във втора глава е описан “ReAct LLM агент без памет”, който показва добра производителност при директни SQL заявки, но се затруднява при поредица от въпроси, изискващи натрупване на контекст. За да се преодолее това ограничение, бе проектиран “ReAct LLM агент с контекстуална памет”, който запаметява част от миналите стъпки, подобрявайки ефективността при сложни бизнес казуси, разпростиращи се в няколко итерации. Третата реализация – “Многоагентна структура” – въведе три независими LLM агента, които си взаимодействат през споделен слой, като единият генерира SQL заявки, а другият следи за логическа последователност и валидност на резултатите.

Резултатите от симулациите в Глава III показаха, че многоагентните подходи предоставят по-висока степен на адаптивност и надеждност при задачи, свързани с обединяване на данни, аргументация и корекция на грешки. Изводът е, че гъвкавото комбиниране на различни агентни роли значително повишава качеството и скоростта на получените решения, особено когато сложността на заявките е висока.

В изследването бе показано как специализиращото обучение може значително да подобри качеството на извличане на информация и, оттам, финалните решения на автоматизираната система.

В контекста на големите езикови модели и retrieval-ориентираните системи, ключова част от изследването бе насочена към техниките за специализиращо обучение на модели за вграждане. В Глава II и III бяха представени принципи за избор на подходящи базови модели, методи за инициализация и процедури за специализиращо обучение върху специфични данни за дадена област на приложение.

Симулациите показаха, че след допълнителни епохи на обучение, фокусирани върху конкретни бизнес термини, се повишава точността при извличане на релевантни записи и се намаляват пропуските. Особен интерес предизвикаха резултатите след специализиращо обучение на модели за вграждане, при което nDCG метриката се подобри осезаемо, а времето за извличане остана в практични граници. Наличието на по-точни векторни представяния улесни и работата на LLM агентите, които получаваха по-релевантен контекст за аргументация и решаване на задачи, свързани със сложни SQL заявки.

Систематичният подход при специализиращо обучение включваше разделяне на данните в обучителни и валидиращи множества, анализиране на точността спрямо baseline модел и коригиране на хиперпараметрите. Този процес демонстрира потенциала на специализираните векторни модели да работят в хармония с LLM, осигурявайки по-добра цялостна производителност на RAG-системите.

Използвани са конкретни експериментални доказателства, показващи силните страни и ограниченията на различните конфигурации на системата.

В третата глава на дисертацията бяха описани и осъществени няколко експериментални постановки, които измерваха както точността на извличане от базата данни, така и качеството на крайния отговор (аргументация и валидност). За да се избегне субективност, в експериментите се прилагаха обективни метрики като nDCG за оценка на търсенето по векторно подобие, и различни способи за оценка на логическата съгласуваност при SQL заявките.

Анализирани бяха три групи експерименти, свързани с ReAct без памет, ReAct с памет и многоагентни структури, където всяка група включваше поне петнадесет примерни бизнес казуса с нарастваща сложност. Резултатите показаха, че агентът без памет е ефективен при еднократни и конкретни заявки, но се затруднява при контекстно зависими въпроси, което съответства на теоретичната обосновка.

Разширеният агент с памет и особено многоагентният сценарий регистрираха по-висока степен на точност при комплексни поредици от въпроси, където се налага натрупване на данни и координиране на действия. Симулациите със специализиращо обучени модели за вграждане доказаха, че подходящата векторизация на специфични данни за конкретна област на

приложение води до по-точна селекция на контекст, което допринася за по-пълни и релевантни отговори. Изводът от емпиричната оценка е, че предложената методика успешно адресира предизвикателствата по извличане на данни и аргументация, особено при по-мащабни бази данни.

Въз основа на така изпълнените задачи може да се заключи, че основната цел на дисертацията – разработването на методология за създаване на многоагентни автоматизирани системи, които използват големи езикови модели за анализ и решаване на задълбочени бизнес проблеми, базирани на информация от бази данни – е постигната. Представените теоретични проучвания, детайлната методология, разработените прототипи на LLM агенти, експерименталните резултати и формулираните препоръки показват, че интегрирането на LLM с RAG подходи и многоагентни архитектури води до подобрена ефективност и надеждност при извличането и аргументираното решаване на задачи. Така дисертацията осигурява цялостна рамка за изграждане и оценка на подобни системи, добавяйки съществен научен и приложен принос в сферата на изкуствения интелект и бизнес анализа.

ПРЕТЕНЦИИ ЗА НАУЧНИ ПРИНОСИ

Методически приноси

1. Разработена е архитектура на система за пълно и автоматично решаване на задачата за бизнес анализ, която включва многостъпково планиране на задачите, многослойната координация между отделни LLM агенти и анализ на получените справки (RAGSQL). Системата работи в режим text-to-SQL-to-insights, т.е. без намеса от потребителя в целия процес след първоначален въпрос до краен отговор и без необходимост потребителят да използва програмен SQL код, за да получи полезни резултати. Представената система е модулна и позволява смяна на отделните блокове с други техни варианти, като същевременно те работят автономно, което спомага за устойчивостта при грешки.

2. Предложен е нов подход за специализиращо обучение на модели със семантично вграждане, водещ до подобрени резултати. В подхода се използва по иновативен начин за извличане на контекст база от анотирани знания от известен еталон за сравняване - Bird - предназначен за оценяване на генеративни модели, се използват по иновативен начин за извличане на контекст. Открива се възможност за лесен трансфер на обучение, при смяната на анотираните знания от друга област. Използваният модел за вграждане може да бъде в пъти по-малък по размерност и въпреки това да получава съществено по-добри резултати.

3. Разработена система, използващи рамка ReAct за решаване на сложни бизнес задачи, използващи SQL бази данни, включваща паралелна работа на няколко LLM агенти с различни допълващи се роли. Приложимостта на системата е изследвана и съпоставена по резултати с други два подхода от рамката ReAct.

Приложни приноси

4. Създадена е автоматизирана експериментална постановка, позволяваща оценка на модели за семантично вграждане за SQL задачи. Експерименталната рамка включва два специализирано обучени и общодостъпни модели за вграждане и два адаптирани и общодостъпни набора от анотирани знания за специализирано обучение на модели. Така подготвените набори от знания обхващат 80 реалистични бази данни от разнообразни бизнес области.

5. Разработени са софтуерни решения и съответните документирани програмни кодове за две приложения - многоагентна структура, с протокол за комуникация и собствена имплементация на памет и оригинална имплементация на алгоритъма за оценка на RAGSQL системата.

ИНФОРМАЦИОННИ ИЗТОЧНИЦИ

1. Mitchell, T. M. (1997). Machine learning. New York, NY: McGraw-Hill.
2. Bridle, J. S. (1989). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Advances in Neural Information Processing Systems (Vol. 2, pp. 211–217).
3. Sheppard, J. G. (1970). Introduction to learning machines (NASA Technical Note D-5790). Washington, DC: National Aeronautics and Space Administration.
4. Schmidhuber, J. (2014). Deep Learning in Neural Networks: An Overview. arXiv preprint arXiv:1404.7828.
5. Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 65(6), 386-408.
6. Bengio, Y. (2013). Deep learning of representations: Looking forward. arXiv preprint arXiv:1305.0445. Retrieved from https://arxiv.org/abs/1305.0445
7. Zhang, Z., Cui, P., & Zhu, W. (2019). Deep learning: A selective overview. arXiv preprint arXiv:1904.05526. Retrieved from https://arxiv.org/abs/1904.05526
8. Gubrud, M. (1997). Nanotechnology and International Security. Presented at the Fifth Foresight Conference on Molecular Nanotechnology.
9. Higham, D. J., & Grindrod, P. (2024). Diffusion Models for Generative Artificial Intelligence: An Introduction. arXiv:2312.14977.
10. Sun, Y., Jang, E., Ma, F., & Wang, T. (2024). Generative AI in the Wild: Prospects, Challenges, and Strategies. arXiv:2404.04101.
11. Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2023). Generative AI. arXiv:2309.07930.

12. Sengar, S. S., Hasan, A. B., Kumar, S., & Carroll, F. (2024). Generative Artificial Intelligence: A Systematic Review and Applications. arXiv:2405.11029.
13. McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (1955). *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*.
14. McCarthy, J. (2007). From Here to Human-Level AI. Artificial Intelligence, 171(18), 1174–1182.
15. Shabbir, J., & Anwer, T. (2018). Artificial Intelligence and its Role in the Near Future. arXiv preprint arXiv:1804.01396.
16. Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2018). Recent Advances in Recurrent Neural Networks. arXiv preprint arXiv:1801.01078. https://arxiv.org/abs/1801.01078
17. Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv preprint arXiv:1506.00019. https://arxiv.org/abs/1506.00019
18. Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). Dive into deep learning. Cambridge University Press.
19. Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Yin, B., & Hu, X. (2023). Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond. arXiv. https://doi.org/10.48550/arXiv.2304.13712
20. Jana, A., Paudel, B., Sarker, M. K., Ebrahimi, M., Hitzler, P., & Amariucai, G. T. (2022). Neural fuzzy extractors: A secure way to use artificial neural networks for biometric user authentication. Proceedings on Privacy Enhancing Technologies, 2022(4), 86–104. https://doi.org/10.56553/popets-2022-0100
21. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. arXiv preprint arXiv:cs/9605103.
22. Sutton, R. S., & Barto, A. G. (1998). Reinforcement Learning: An Introduction. MIT Press.
23. Kaufmann, T., Weng, P., Bengs, V., & Hüllermeier, E. (2023). A Survey of Reinforcement Learning from Human Feedback. arXiv preprint arXiv:2312.14925. https://arxiv.org/abs/2312.14925
24. Zhu, B., Jordan, M., & Jiao, J. (2023). Principled Reinforcement Learning with Human Feedback from Pairwise or K-wise Comparisons. arXiv preprint arXiv:2301.11270. https://arxiv.org/abs/2301.11270
25. Lambert, N., Gilbert, T. K., & Zick, T. (2023). The History and Risks of Reinforcement Learning and Human Feedback. arXiv preprint arXiv:2310.13595. https://arxiv.org/abs/2310.13595
26. Leblanc, B. (2024). On the relationship between interpretability and explainability in machine learning. arXiv preprint, arXiv:2311.11491.
27. Microsoft. (n.d.). Introduction to Windows Machine Learning. Microsoft Learn. Retrieved March 16, 2025, from https://learn.microsoft.com/en-us/windows/ai/windows-ml/overview
28. Song, C., Ristenpart, T., & Shmatikov, V. (2017). Machine learning models that remember too much. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17) (pp. 587–601). ACM. https://doi.org/10.1145/3133956.3134077
29. Wang, R., Erus, G., Chaudhari, P., & Davatzikos, C. (2023). Adapting machine learning diagnostic models to new populations using a small amount of data: Results from clinical neuroscience. arXiv preprint, arXiv:2308.03175.
30. Liddy, E. (2001). Natural Language Processing. In A. Kent & H. Lancour (Eds.), Encyclopedia of Library and Information Science. Boca Raton, FL: CRC Press
31. Chowdhury, G. G. (2003). Natural language processing. Annual Review of Information Science and Technology, 37(1), 51–89
32. Dale, R., Moisl, H., & Somers, H. (2000). Handbook of Natural Language Processing. New York: Marcel Dekker
33. Shao, M., Basit, A., Karri, R., & Shafique, M. (2024). Survey of Different Large Language Model Architectures: Trends, Benchmarks, and Challenges. arXiv:2412.03220
34. Ji, W., Su, L., Xu, J., Zhang, L., & Chang, E. Y. (2025). An Overview of Large Language Models for Statisticians. arXiv:2502.17814

35. Kalyan, K. S. (2023). A Survey of GPT-3 Family Large Language Models Including ChatGPT and GPT-4. arXiv preprint arXiv:2310.12321. https://arxiv.org/abs/2310.12321
36. Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). Large Language Models: A Survey. arXiv preprint arXiv:2402.06196. https://arxiv.org/abs/2402.06196
37. Chen, D., Pellizzoni, P., & Borgwardt, K. (2023). Fisher Information Embedding for Node and Graph Learning. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), Proceedings of the 40th International Conference on Machine Learning (Vol. 202, pp. 4839–4855). PMLR.
38. Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised Deep Embedding for Clustering Analysis. In M. F. Balcan & K. Q. Weinberger (Eds.), Proceedings of The 33rd International Conference on Machine Learning (Vol. 48, pp. 478–487). PMLR.
39. Vatsal, S., & Dubey, H. (2024). A survey of prompt engineering methods in large language models for different NLP tasks. arXiv preprint arXiv:2407.12994. https://arxiv.org/abs/2407.12994
40. Jégou, H., Douze, M., & Johnson, J. (2024). The Faiss library. arXiv preprint arXiv:2401.08281.
41. Schubert, E., Lang, A., & Feher, G. (2021). Accelerating Spherical k-Means. In Similarity Search and Applications (pp. 3-15). Springer International Publishing.
42. Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4), 824-836.
43. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527–1554. DOI: 10.1162/neco.2006.18.7.1527
44. Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., & Smith, N. (2020). Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. arXiv:2002.06305 [cs.CL].
45. Bergmann, D. (2024, March 15). What is fine-tuning? IBM AI Topics. Retrieved from https://www.ibm.com/think/topics/fine-tuning
46. Darrin, M., Formont, P., Ben Ayed, I., Cheung, J. C. K., & Piantanida, P. (2024). When is an Embedding Model More Promising than Another? arXiv:2406.07640
47. Tao, C., Shen, T., Gao, S., Zhang, J., Li, Z., Tao, Z., Ma, S. (2024). LLMs are Also Effective Embedding Models: An In-depth Overview. arXiv:2412.12591
48. OpenAI. (2023). Powering virtual education for the classroom. Посетено на 20.01.2024: https://openai.com/index/khan-academy/
49. Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot*. arXiv preprint arXiv:2302.06590.
50. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv preprint arXiv:1907.11692.
51. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.
52. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research, 21*(140), 1–67.
53. Mikolov, Tomas; Chen, Kai; Corrado, Greg; Dean, Jeffrey (2013a). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781
54. Canale, L., and Farinetti, L. (2022). SQL Murder Mystery: a serious game to learn querying databases. In 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 129-138). IEEE.

55. Bahdanau, D., Cho, K. and Bengio, Y. (2015) 'Neural Machine Translation by Jointly Learning to Align and Translate'. Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).

56. Cambria, E. and White, B. (2014) 'Jumping NLP Curves: A Review of Natural Language Processing Research'. IEEE Computational Intelligence Magazine, 9(2), pp. 48-57.

57. UKPLab. (n.d.). *sentence-transformers* [Computer software]. GitHub. https://github.com/UKPLab/sentence-transformers

58. Hochreiter, S. and Schmidhuber, J. (1997) 'Long Short-Term Memory'. Neural Computation, 9(8), pp. 1735-1780.

59. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013) 'Distributed Representations of Words and Phrases and their Compositionality'. Advances in Neural Information Processing Systems, 26 (NIPS 2013), pp. 3111-3119.

60. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gómez, A.N., Kaiser, Ł. and Polukhin, I. (2017) 'Attention Is All You Need'. Advances in Neural Information Processing Systems, 30 (NIPS 2017), pp. 5998-6008.

61. Young, T., Hazarika, D., Poria, S. and Cambria, E. (2018) 'Recent Trends in Deep Learning Based Natural Language Processing'. IEEE Computational Intelligence Magazine, 13(3), pp. 55-75.

62. Pennington, J., Socher, R., Manning, C., 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

63. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., (2018). Improving Language Understanding by Generative Pre-Training. OpenAI. [https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf]

64. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y. (2022) ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629

65. Austin, J., Odena, A., Barham, P., Cisse, M., & Brown, T.B. (2021) 'Program Synthesis with Large Language Models'. Advances in Neural Information Processing Systems, 34, pp. 11941–11953.

66. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P., Kaplan, J. et al. (2021) 'Evaluating Large Language Models Trained on Code'. arXiv preprint arXiv:2107.03374

67. Dafoe, A., Bachrach, Y., Hadfield, G., Horvitz, E., Moskva, H., Hughes, N. et al. (2021) 'Cooperative AI: Machines must learn to find common ground'. Nature, 593, pp. 33–36.

68. Guo, J., Pasupat, P., Liu, Z., & Liang, P. (2019) 'Towards complex text-to-SQL in cross-domain database with intermediate representation. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 4524–4535.

69. Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N., Wiest, O., Zhang, X. (2024) Large Language Model based Multi-Agents: A Survey of Progress and Challenges. arXiv:2402.01680

70. Xu, S., Semnani, S., Campagna, G., Lam. M. (2021) AutoQA: From Databases To QA Semantic Parsers With Only Synthetic Training Data. arXiv:2010.04806

71. Amodei, D., et al. (2016) 'Concrete problems in AI safety'. arXiv preprint arXiv:1606.06565.

72. Bender, E. M., et al. (2021) 'On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?'. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 610–623.

73. Bommasani, R., et al. (2021) 'On the opportunities and risks of foundation models'. arXiv preprint arXiv:2108.07258.

74. Bostrom, N. (2014) Superintelligence: Paths, Dangers, Strategies. Oxford University Press. ISBN 978-0-19-967811-2

75. Russell, S., Dewey, D. & Tegmark, M. (2015) 'Research priorities for robust and beneficial artificial intelligence'. AI Magazine, 36(4), pp. 105–114.

76. Weidinger, L., et al. (2021) 'Ethical and social risks of harm from language models'. arXiv preprint arXiv:2112.04359.
77. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2021). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. arXiv:2107.13586.
78. Radford, A. et al. (2019). Language models are unsupervised multi task learners. OpenAI Technical Report.
79. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching networks for one shot learning. Advances in Neural Information Processing Systems, 29, 3630–3638.
80. Kojima, T. et al. (2022). Large Language Models are Zero-Shot Reasoners. Advances in Neural Information Processing Systems, 35 (NeurIPS 2022).
81. Wang, X. et al. (2023). Self-Consistency Improves Chain-of-Thought Reasoning in Language Models. Proceedings of the International Conference on Learning Representations (ICLR 2023).
82. Wei, J. et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. Advances in Neural Information Processing Systems, 35 (NeurIPS 2022).
83. Yao, S. et al. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. Advances in Neural Information Processing Systems, 36 (NeurIPS 2023).
84. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … & Amodei, D. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems (NeurIPS 2020) (Vol. 33, pp. 1877–1901).
85. Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In Advances in Neural Information Processing Systems (NeurIPS 2020) (Vol. 33, pp. 12449–12460).
86. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT 2019 (pp. 4171–4186). Association for Computational Linguistics.
87. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014) (pp. 580–587). IEEE.
88. He, K., Girshick, R., & Dollár, P. (2019). Rethinking ImageNet pre-training. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2019) (pp. 4918–4927). IEEE.
89. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504–507. https://doi.org/10.1126/science.1127647
90. Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018) (pp. 328–339). Association for Computational Linguistics.
91. Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191
92. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems 27 (NIPS 2014) (pp. 3320–3328). MIT Press.
93. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., … & He, Q. (2021). A comprehensive survey on transfer learning. Proceedings of the IEEE, 109(1), 43–76.
94. Aggarwal, C. C., & Zhai, C. (2012). *A survey of text clustering algorithms*. In C. C. Aggarwal & C. Zhai (Eds.), Mining Text Data (pp. 77–128). Springer.
95. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2024). Retrieval-Augmented Generation for Large Language Models: A Survey (Version 5) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2312.10997
96. Angelov, D. (2020). *Top2Vec: Distributed Representations of Topics*. arXiv:2008.09470.
97. Gao, T., Yao, X., & Chen, D. (2021). SimCSE: Simple contrastive learning of sentence embeddings. *Proceedings of EMNLP*, 6894–6910.

98. Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *Proceedings of CVPR*, 1735–1742.

99. Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems, 20*(4), 422–446.

100. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... & Zaremba, W. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems, 35*, 27730–27744.

101. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *Proceedings of EMNLP-IJCNLP*, 3982–3992.

102. Saravanakumar, K. K., Ballesteros, M., Chandrasekaran, M. K., & McKeown, K. (2021). Event-driven news stream clustering using entity-aware contextual embeddings. *Proceedings of EACL (Vol. 1)*, 2330–2340.

103. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *Proceedings of CVPR*, 815–823.

104. Gunel, B., Du, J., Conneau, A., & Stoyanov, V. (2020). *Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning*. arXiv:2011.01403

105. Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Geng, R., Huo, N., Zhou, X., Ma, C., Li, G., Chang, K. C. C., Huang, F., Cheng, R., & Li, Y. (2023). *Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs*. arXiv preprint arXiv:2305.03111

106. Adlakha, V., BehnamGhader, P., Lu, X. H., Meade, N., & Reddy, S. (2024). *Evaluating correctness and faithfulness of instruction-following models for question answering*. Transactions of the Association for Computational Linguistics, 12, 681–699. (Published online 2023)

107. Bian, N., Liu, P., Han, X., Lin, H., Lu, Y., He, B., & Sun, L. (2023). *A drop of ink makes a million think: The spread of false information in large language models*. arXiv preprint arXiv:2305.04812.

108. Chen, J., Lin, H., Han, X., & Sun, L. (2023). *Benchmarking large language models in retrieval-augmented generation*. arXiv preprint arXiv:2309.01431.

109. Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020). *REALM: Retrieval-augmented language model pre-training*. In Proceedings of the 37th International Conference on Machine Learning (ICML).

110. Guo, C., Tian, Z., Tang, J., Li, S., & Wang, T. (2025). *Multi-pattern retrieval-augmented framework for text-to-SQL with Poincaré-skeleton retrieval and meta-instruction reasoning*. Information Processing & Management, 62(3), 103978. (Early access 2023)

111. Lewis, P., Pérez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., & Riedel, S. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. In Advances in Neural Information Processing Systems 33 (NeurIPS 2020) (pp. 9459–9474).

112. Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2023). *MTEB: Massive Text Embedding Benchmark*. In Proceedings of the 17th Conference of the European Chapter of the ACL (EACL) (pp. 2014–2037).

113. Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., & Manning, C. D. (2018). Hotpot: A Dataset for Diverse, Explainable Multi-hop Question Answering. arXiv:1809.09600.

114. Thorne, J., Vlachos, A., Christodoulopoulos, C., & Mittal, A. (2018). FEVER: a large-scale dataset for Fact Extraction and VERification. arXiv:1803.05355.

115. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv preprint arXiv:2104.08663v4.

116. Henderson, M., Al-Rfou, R., Strope, B., Sung, Y.-H., Lukács, L., Guo, R., Kumar, S., Miklos, B., & Kurzweil, R. (2017). Efficient natural language response suggestion for Smart Reply. arXiv preprint arXiv:1705.00652.

ПУБЛИКАЦИИ НА АВТОРА

201. Sergey Vichev, Angel Marchev, Reasoning capabilities of large language models on information extracted from databases using Text-to-SQL, PROCEEDINGS OF THE 49TH INTERNATIONAL CONFERENCE “APPLICATIONS OF MATHEMATICS IN ENGINEERING AND ECONOMICS”, 2025, doi:https://doi.org/10.1063/5.0247481, Ref, IF (- 2025), SCOPUS, SJR (- 2025), представен доклад на конференция AMEE: Юни 2023, представен последен вариант на текста за публикуване: Юли 2023

202. Sergey Vichev, A Multi-Agent Framework for Large Language Model-Driven SQL Analysis, KSI Transactions on KNOWLEDGE SOCIETY, vol:XV, issue: Nr. 1, 2024, ISSN (print):1313-4787, представен доклад на PyData семинар: Декември 2024, представен последен вариант на текста за публикуване на: Февруари 2024

203. Vichev, Sergey, Marchev, Angel, RAGSQL: Context Retrieval Evaluation on Augmenting Text-to-SQL Prompts, International IEEE Conference proceedings, IS, 2024, doi:10.1109/IS61756.2024.10705186, Ref, SCOPUS, SJR (- 2024) представен доклад на конференция IEEE IS: Август 2024, представен последен вариант на текста за публикуване: Юли 2024

204. Vichev, Sergey, Application of K-means Clustering Algorithm with RFM for Customer Segmentation in Logistics Sector., Vanguard Scientific Instruments in Management, issue:19, 2023, ISSN (print):ISSN: 1314-0582, Ref, др.(CEEOL), представен доклад на конференция VSIM: Септември 2022, представен последен вариант на текста за публикуване: Декември 2022