|  |  |
| --- | --- |
|  |  |

**ДИПЛОМСКА РАБОТА**

**Тема**

**Користење на генеративни модели за автоматско создавање на SQL прашања користејки природен јазик**

|  |  |
| --- | --- |
| Ментор | Изработил |
| проф. др.  Димитар Трајанов | Павел Стојановски Андоновиќ, индекс бр. 175022  e-mail:pavel.gjorgovski@students.finki.ukim.mk |

*Скопје, Октомври 2023*

**Содржина**

[Апстракт 1](#_Toc223681204)

[1 Вовед 2](#_Toc223681205)

[2 Структура на семинарската 2](#_Toc223681206)

[3 Тело на семинарската 2](#_Toc223681207)

[3.1 Уредување на семинарската 2](#_Toc223681208)

[3.2 Наслови 2](#_Toc223681209)

[3.3 Главниот текст 3](#_Toc223681210)

[3.4 Слики и табели 3](#_Toc223681211)

[3.5 Фусноти (забелешки) 4](#_Toc223681212)

[4 Референци 4](#_Toc223681213)

[5 Проверка на граматичката исправност 5](#_Toc223681214)

[6 Големина и обем на семинарската работа 6](#_Toc223681215)

[7 Предавање на семинарската работа 6](#_Toc223681216)

[8 Презентација 6](#_Toc223681217)

[9 Оценување 6](#_Toc223681218)

[10 Листа за проверка 6](#_Toc223681219)

# Апстракт

*Технологијата за големи јазични модели напредува со неверојатна брзина. Дел од обврски на програмерите можат да бидат поедноставени и автоматизирани со нејзиното користење. Еден од предизвиците со кои секојдневно се соочуваат податочните инженери е создавањето на добро SQL прашање. Во оваа дипломска работа целта е креирање на алатка која ќе овозможи поставување на барање на податочно ниво со користење на природен јазик, кое барање потоа автоматски ќе биде преведено во SQL прашање. Главната примена на ова решение е нетехнички лица да можат да поставуваат SQL прашања без потреба од техничкото познавање на јазикот, како и олеснување на секојдневната работа на податочните инженери.*

# Вовед

Вештачката интелигенција има направено големи напредоци во минатите години, особено во обработка на природни јазици. Генеративни модели, како ChatGPT, се имаат истакнато како моќни алати во голем број домени [1][2][3], дополнително подржани од различни методи на насочување на генеративните модели за да се добие резултат каков што корисникот очекува. Еден од секторите во кои има голем напредок е автоматско генерирање на код [4], и се очекува дека голем дел од шаблонското пишување код ќе биде многу олеснето користејќи алатки од генеративни модели.

SQL е програмски јазик

Во овој труд ќе се разгледа примената на ChatGPT во генерирање на SQL прашања со различен степен на комплексност, моментални проблеми при генерирање на SQL прашања и потенцијални решенија.

# Поврзана литература

На темата генеративни модели се пишувани голем број трудови во минатата година. Секојдневно се наоѓаат примени за ваков тип модели за создавање на нови алатки или подобрување на постоечки.

Тим од доктори по науки од универзитетите на Хонг Конг и Луксембург имаат правено истражување на тема користење на генеративни модели како ChatGPT за автоматско генерирање на код [4]. Користејки наредби кон ChatGPT од различни големини, тие ја истражуваат ефикасноста при генерирање Python код за решавање на LeetCode проблеми. Некои заклучоци од тој труд се:

* ChatGPT има потешкотија со генерализација на досега не видени проблеми и барања
* Големи наредби кон ChatGPT му попречуваат со заклучоците кои може да ги вади
* За поправање на даден код, или опис на ефектите и функцијата на постоечки код, ChatGPT перформира подобро од дотогашниот SotA модел.
* Давање на ChatGPT информации за генерички описи на проблемите го прави моделот да перформира полошо. ChatGPT има кратко внимание

Тим од биолози и медицински доктори направија истражување на ефектите на користење ChatGPT за дијагностички цели, надевајќи се дека ќе ги подобрат околините за учење, како и дијагностичките алати на доктори [2]. Како дел од едукативниот процес на докторите, следните аспекти беа оценети како позитивни за едукативниот процес користејќи го ChatGPT:

* Подобрување на индивидуално учење – ученици имаат можност да напредуваат повеќе во материјалот кој им е послаб, како и користејќи методи персонализирани за карактерот на ученикот.
* Подобрување на дијагностичка логика – заклучоци од симптоми и начини на живот се некогаш тешки за создавање, но ChatGPT им давал перспективи кои учениците некогаш ги избезумиле.
* Помош при разбирање комплексни медицински проблеми – можноста да се разделат покомплексни проблеми во помали прости проблеми е карактеристика на ChatGPT која е документирана во повеќе различни трудови.

Проблеми и ограничувања при користење на ChatGPT за едукативниот процес на медицински лица:

* Проблеми со приватност – податоци за одредени пациенти и клиенти треба да останат приватност, но луѓето немаат доверба во таа приватност ако е понудена од страна на ВИ софтвер.
* Ризик за предрасуди или погрешни информации – иако може со параметар да се менува колку „халусинира“ ChatGPT, информациите не можат секогаш да се толкуваат како точни. Истотака, за различни теми може да содржи предрасуди предизвикани од количеството текстови на темата кои се достапни на интернетот.
* Ризик за намалување на критичко размислување и способност за самостојно решавање проблеми – иако малку потешко да се докаже од предходните 2 проблеми, се верува дека константно користење на ваков тип алатка може да предизвика намалување на ученикот да самостојно доаѓа до заклучоци, што може да е неверојатно штетно кога би станало збор за некои од покомплексните проблеми со кои ќе мора да се соочи.

Spider 1.0 Yale Semantic Parsing and Text-to-SQL Challenge е меѓународен предизвик за инженери на тема трансформирање на природен јазик во валидни SQL прашања. Разни тимови имаат пратено нивни решенија со код и научен труд, во кој се содржани нивните истражувања на темата. Сите високо рангирани тимови користат ChatGPT 4, проследени од тимови кои користат ChatGPT 3.5. Во прилог ќе бидат дефинирани резултатите на највисоко рангираниот тим на натпреварот (на датум 21/10/2023), кој е тимот *AliBaba Group* [5].

Овој труд се однесува на предизвикот за дизајнирање ефективни, ефикасни и економични решенија од Text-to-SQL засновани на LLM преку обезбедување на систематска евалуација на репер. Авторите спроведуваат сеопфатна споредба на постоечките брзи инженерски методи, вклучително претставување на прашања, избор на пример и организација на примери. Тие предлагаат ново интегрирано решение наречено DAIL-SQL, кое постигнува 86,6% точност на извршување на таблата со водачи на Spider, надминувајќи ја најсовремената технологија за 1,3%. Авторите исто така го истражуваат потенцијалот на LLM со отворен код и ги истакнуваат предностите и недостатоците на надгледуваното дотерување. Тие ја нагласуваат токенската ефикасност во брзото инженерство и споредуваат претходни студии според оваа метрика. Генерално, овој труд обезбедува подлабоко разбирање на Text-to-SQL со LLM и има за цел да инспирира понатамошни истражувања и апликации во оваа област.

Ова се позитивните и негативните заклучоци за користење на DAIL-SQL:

Позитивни:

* Постигнува висока точност на извршување (86,6% на таблата со водачи на Spider)
* Кодира знаење за структурата како SQL изјави за подобри перформанси
* Избира примери врз основа на сличности на скелетот за ефикасна употреба на токени
* Поставува нов стандард во перформансите во споредба со претходните решенија

Негативни:

* Додавањето примери може да ја намали ефикасноста на токените за LLM со отворен код
* Може да бара пристап до целосната база на податоци за оптимални перформанси

# Методи

Со цел темелно разбирање на методите на решавање кои се користени во трудов, не се користат никакви маски и ретренирања на ChatGPT освен самиот ChatGPT и LlamaIndex, кој се користи само за подршка на решението во случаи на многу големи бази на податоци.

Целиот проблем на превод на природен јазик во SQL прашање е поделен на 7 сегменти:

1. Поставување на прашање или барање на природен јазик
2. Читање на примероци од табелите од базата на податоци
3. Создавање на описи на табелите врз база на примероците од табелите користејќи ChatGPT.
4. Зачувување на описите за понатамошно користење за прашања, како и запишување на истите како индекси во VectorStore користејќи LlamaIndex
5. Барање на табели најсоодветни за одговарање на прашањето или барањето користејќи ги индексите од LlamaIndex
6. Користење на избраните табели и ChatGPT за создавање SQL прашање кое ги задоволува потребите
7. Користење на ChatGPT за опис на процедурата на стигање до даденото SQL прашање.

## ChatGPT

ChatGPT е јазичен модел развиен од страна на OpenAI, базиран на GPT (Generative Pre-trained Transformer) архитектурата. Ова е како функционира ChatGPT:

1. **Претренирање:** ChatGPT се претренира со големо количество текстуални податоци од интернет. Во текот на претренирањето, моделот учи статистички обрасци, граматика и контекст на човечкиот јазик. Оваа фаза помага му на моделот да изгради фундамент на разбирање на јазикот.
2. **Доработка:** По претренирањето, ChatGPT се доработува со специфични податочни множества и задачи за да стане корисен во реални апликации. Во овој процес, моделот се обучува на прилагодени податоци, вклучувајќи ги прашањата и одговорите, за да стане способен да дава одговори со контекст.
3. **Генерирање на јазик:** Кога корисникот внесува текст, како на пример прашање, ChatGPT генерира одговор базиран на статистичките обрасци и знаењето кое го научил во претренирањето и доработката. Моделот користи своето разбирање на јазик и контекст за да создаде кохерентен и контекстуално релевантен текст.
4. **Зачувување на контекст:** ChatGPT го зачувува контекстот за време на разговорот. Тој ги помни претходните прашања на корисникот, како и сопствените одговори, за да обезбеди контекстуално релевантни одговори. Ова му овозможува да води значајни и интерактивни разговори со корисниците.
5. **Ограничувања:** Иако е впечатливо способен, ChatGPT има свои ограничувања. Понекогаш може да генерира неточни или безсмислени одговори. Тој е осетлив на фразирањето на влезот и може да генерира содржина со предрасуди или несоодветен карактер ако податоците за обука содржат такви предрасуди. OpenAI има воведено мерки за безбедност за да ги ограничи овие проблеми и да спречи генерирањето на штетен содржина. За неточни информации, постои влезен параметар кој дефинира колку ChatGPT ќе измислува податоци кои не ги знае со сигурност.
6. **Историјата на Развој на GPT**

Првата итерација на моделот GPT (Generative Pre-trained Transformer) беше објавена во 2018 година, и тоа ја постави основата за развојот на ChatGPT. Првобитно, моделот бил трениран на огромна колекција на текстови за да генерира текст кој прилично реално симулира човечки разговор. Одличноста во GPT архитектурата е во нејзината способност да учи од огромни количества податоци и да произведува текст во разни стилови и теми.

Следна итерација беше GPT-2, објавена во 2019 година, која не беше достапна за јавноста поради можна злоупотреби во областа на лажни вести и забуна. Но, откако беа земени потребните мерки за обезбедување на безбедност, овој модел стана доспупен и донесе значителни подобрувања во генерирањето на текст.

Предпоследна итерација на GPT беше GPT-3, кој беше објавен во 2020 година и кој стана основа за голем број апликации, вклучувајќи го и ChatGPT. Оваа архитектура се надгради и дополни со нови техники и податоци, што резултираше во GPT-3.5, која беше објавена во 2021 година и ја подобри способноста на системот за разговор и генерирање на текст.

Најновата итерација на GPT е GPT-4, кој беше објавен во 2023 годинам кај кој се тргнати голем број на ограничувањата присутни во GPT-3.5, но кој бара месечна претплата за да се користи.

Во овој труд се користи исклучиво GPT-3.5.

1. **Апликации на ChatGPT**

ChatGPT има бројни апликации во разни области, како што се:

1. **Системи за разговор:** ChatGPT се користи за да создава виртуелни асистенти и чатботови кои можат да обезбедат корисни информации и решенија на прашања на корисниците.
2. **Едукација:** Во образованието, ChatGPT може да се искористи за индивидуални наставни програми, обуки и објаснување на концепти во различни предмети.
3. **Здравство:** ChatGPT може да обезбеди информации и совети за здравствена нега, како и да помогне во понатамошни истражувања во областа на медицината.
4. **Креирање на содржина:** ChatGPT може да биде корисен за автоматско генерирање на текстови за веб-страници, блогови и други медиуми.
5. **Консултирање и поддршка:** Овој систем може да се користи за виртуелни консултации и техничка поддршка за корисници.
6. **Заклучок за ChatGPT**

ChatGPT е резултат на структурен напредок во областа на природни јазици и вештачка интелигенција. Неговите корени се во историјата на истражувањето во машинско учење и генеративни модели на текст. Со големите можностите за примена, ChatGPT се очекува да има значителен влијание во разни области и ќе се продолжи со истражување и напредок во иднина.

## LlamaIndex

LlamaIndex, претходно познат како GPT Index, е иновативна рамка за податоци специјално дизајнирана за поддршка на развој на апликации базирани на LLM.

Тој нуди напредна рамка која им овозможува на програмерите да интегрираат различни извори на податоци со големи јазични модели. Ова вклучува различни формати на датотеки, како што се PDF и PowerPoint презентации, како и апликации како Notion и Slack, па дури и бази на податоци како Postgres и MongoDB. Рамката носи низа конектори кои помагаат при внесување податоци, олеснувајќи ја интеракцијата со LLM.

Покрај тоа, LlamaIndex има неверојатно ефикасен интерфејс за пребарување на податоци во датотеки. Оваа функција им овозможува на програмерите да внесат било какво прашање за LLM и, за возврат, да добијат излез кој е и богат со контекст и зголемен со знаење. Оваа функција на LlamaIndex, заедно со мапирањето на json датотеки во индекс, е причината за неговото користење во овој труд.

Во суштина функционира како интерфејс кој управува со интеракциите со LLM преку создавање на индекс од понудените влезни податоци. Овој индекс потоа се користи за да одговори на какви било прашања поврзани со дадените податоци. LlamaIndex има разновидност да креира различни видови на индекси, како што се индекси на вектор, дрво, листа или клучни зборови, во зависност од специфични барања кои се доделени.

LlamaIndex нуди широк спектар на алатки кои ги олеснуваат процесите на внесување податоци, пребарување, структурирање и интеграција со различни рамки за апликации.

## Поставување на прашање или барање на природен јазик

Првиот чекор кој мора да се направи при извршувањето на процесот е спецификација на барање, како и дефинирање на која база на податоци ќе се користи.

Барањето треба да содржи кој тип на информации се бара, како и ограничувања. Не треба да содржи информации за табелите, препорака кои табели да се користат, или кои клучеви да се користат при спојување на две или повеќе табели.

Прашањата треба да се на англиски јазик, и не бараат техничко знаење за работа со бази на податоци

Пример прашање:

What are the names of all female students who play football?

Ова барање личи како прашање што би било побарано од личноста која работи со базите на податоци, и би очекувале да ни одговори со листа од имињата на сите женски студенти кои играат фудбал.

Толкување на резултатите на ова прашање ќе се наоѓа во следниот сегмент, „Резултати“, но овој пример служи за да се демонстрира како треба да личат прашањата поставени.

Во реален случај, ќе треба да се воспостави конекција со база на податоци користејќи информации за најавување, доделување ограничувања на пристап итн. За овој труд, се користат само базите од Spider. Пред секое прашање, мора во конфигурациската датотека да се специфицира за која база на податоци треба да се генерира SQL прашање. Минатото прашање е поставено за „game\_1“ базата на податоци.

Прашањата во трудов се влечени од официјалните Spider прашања за нивниот натпревар.

Други пример прашања:

* How many singers do we have?
* What are the names of properties that are either houses or apartments with more than 1 room?
* Find the number of owners who do not own any dogs at this moment.

## Читање на примероци од табелите од базата на податоци

Читањето на примероци од табелите треба да се случи само првиот пат при поставување на барање од дадена база на податоци, и освен тоа само кога „корисникот“ ќе побара рачно да се ажурираат податоците.

За целта на овој труд е користена колекцијата на бази на податоци понудена од Spider, која користи SQLite. Во овој труд нема да се разгледуваат различните потенцијални опции за користење на различни табели со различни методи на оптимизација (Redshift оптимизација по време наспроти BigQuery оптимизација по податоци изминати и сегментација по клуч на партиција). Иако остатокот од кодот не е зависен од методот на влечење на податоците, финалниот изглед на истите треба да е составен од името на базата како име на фолдерот кој содржи .csv датотеки крстени по имињата на табелите во базата.

Пример:  
-college\_1  
--CLASS.csv  
--COURSE.csv  
--PROFESSOR.csv

Секоја од датотеките треба да содржи мал број редови од табелите, како и имињата на колоните. Бројот на редови кои се читани е поставен како 50, со можност да се смени. Препорачани вредности се 20-200, каде 20 би се користело во случај на табели со преголем број колони, додека 200 може да е користено кога станува збор за табели со 2-3 колони. Има 2 причини за ставање на горна граница за примероци:

* Штедење меморија во случај на читање на голем број бази на податоци
* Овозможување на ChatGPT да ги прочита податоците од примерокот покрај горната граница на вредности која ја содржи

Овие примероци од табелите во следниот чекор ќе се користат за опис на табелите, така што мора заедно со инструкциите кои ќе бидат доделени на ChatGPT да не зафаќа повеќе од 16000 симболи.

Во случај да веќе постои фолдер со примероци за дадената база на податоци, овој чекор се скока. Тоа е со цел да не се читаат податоци од базата повторно при секое прашање од корисникот.

## Создавање на описи на табелите врз база на примероците од табелите користејќи ChatGPT.

После зачувување на резултатите од примероците се користи ChatGPT за да се генерираат описи на табелите и да се зачуваат во одделна дестинација. Форматот на излезот ќе е речиси идентичен со форматот на зачувување на примероците, со тоа што ќе се чуваат описите во .json датотеки

Пример:  
-college\_1  
--CLASS.json  
--COURSE.json  
--PROFESSOR.json

За создавање опис се користи ChatGPT 3.5, со вредност на temperature параметарот наместен на 0. Тоа е со цел да избегнеме халусинација на моделот, која може да доведе до вметнување колони и описи за колони кои не постојат во оригиналната табела.

**За добивање на описи, се користи следната системска наредба:**

You are tasked with explaining the columns and their meaning based on the column names as well as the row data in the given table. Give the results in a dictionary format, where for every key (column name) there is a value (explanation of column), followed by the data type of the values in SQL. If a column has numeric values aside from 1 and 0, do not explain it as boolean.  
   
 \*\*\* IF AN ID IS PRESENT, ALWAYS EXPLAIN WHAT THE ID IS FOR, USING YOUR BEST GUESS. \*\*\*  
  
 -> Example:  
 {  
 "CustomerID": "The unique identifier for each customer. DataType: int",  
 "Purchases": "The number of purchases that customer has made. DataType: int",  
 "ProductName": "The name of the purchased product. DataType: varchar",  
 "table\_explanation": "The table contains information on purchases made by customers. Each row represents a product purchased by a specific customer."  
 }  
 -> End example  
  
 Add to that dictionary a key named 'table\_explanation', where the value is the explanation of the entire table.  
 THE RESULTS MUST BE IN A JSON FORMAT. ALWAYS USE A DOUBLE QUOTES AND SEPARATE THE KEY-VALUE PAIRS WITH COMMAS  
 The table\_explanation needs to be inside the dictionary.  
 Transform the dictionary to a json format file.  
 Do not invent column names that are not already present in the table.

За да се разбере причината за оваа наредба, ќе се разгледаат сите компоненти од кои се состои.

You are tasked with explaining the columns and their meaning based on the column names as well as the row data in the given table.

Првично мора да се постави обврската на моделот. Сите последователни информации се со цел да се подобри извршувањето на оваа основна обврска. Му се посочува дека треба да ги користи и имињата на колоните и содржината во редиците за да создаде заклучок за што е содржано во колоната.

Give the results in a dictionary format, where for every key (column name) there is a value (explanation of column), followed by the data type of the values in SQL. If a column has numeric values aside from 1 and 0, do not explain it as boolean.

Следната наредба го дефинира изгледот на излезот. Резултатот сакаме да е во форма на речник за да може резултатот понатаму да го зачуваме како json датотека. Се прецизира клуч-вредност парот како име на колона проследено од описот на колоната, и податочниот тип кој е присутен. Ако не се специфицира податочниот тип, настануваат потешкотии при генерирање на SQL прашања при сврзување на табели со различно крстени колони. При иницијалното тестирање се случуваше да integer вредности ги запишува како boolean, дури и кога постоеа случаи на вредности различни од 0 и 1, па крајот на овој сегмент на наредбата служи да се избегне таков случај.

\*\*\* IF AN ID IS PRESENT, ALWAYS EXPLAIN WHAT THE ID IS FOR, USING YOUR BEST GUESS. \*\*\*

Кога во името на табелата и вредностите на табелата нема доволно очигледна информација за што е содржано во табелата, ChatGPT некогаш враќа генерички опис кој личи вака:

"C\_ID": "The unique identifier for every row. DataType: int",

Вакви случаи се проблематични кога настанува спојување на различни табели при генерирање на SQL прашање, па затоа се дава наредба на моделот која форсира дефинирање на значењето на колоната. При успешен обид, дефиниција на ID колона личи вака:

"C\_ID": "The unique identifier for each customer. DataType: int"

Примерот кој го даваме на програмата е:

-> Example:  
 {  
 "CustomerID": "The unique identifier for each customer. DataType: int",  
 "Purchases": "The number of purchases that customer has made. DataType: int",  
 "ProductName": "The name of the purchased product. DataType: varchar",  
 "table\_explanation": "The table contains information on purchases made by customers. Each row represents a product purchased by a specific customer."  
 }  
 -> End example

Дефинираме почеток и крај на примерот затоа што моделот покажа подобри перформанси кога беа убаво поделени сегментите на наредбата. Овој пример користи реални описи на една од табелите понудени од Spider. table\_explanation клучот е додаден во примерот, но објаснувањето за него се наоѓа во следниот сегмент на наредбата. Важно е да го додадеме во примерот, затоа што ако не е додаден можат да настанат проблеми каде што ChatGPT ќе го внесе после крајот на речникот, што ќе ја скрши структурата и ќе го направи текстот да не може правилно да се запише како json датотека.

Add to that dictionary a key named 'table\_explanation', where the value is the explanation of the entire table.

Овде се дефинира вредноста на table\_explanation клучот во речникот, кој содржи краток опис на табелата, каков што може да заклучи ChatGPT од податоците дадени. Овие описи понатаму се важни за заклучување кои табели се корисни за одговарање на секое прашање.

THE RESULTS MUST BE IN A JSON FORMAT. ALWAYS USE A DOUBLE QUOTES AND SEPARATE THE KEY-VALUE PAIRS WITH COMMAS  
The table\_explanation needs to be inside the dictionary.  
Transform the dictionary to a json format file.

Понекогаш ChatGPT ги двои table\_explainer-от и описите на колоните, па изглезот личи вака:

{  
 "CustomerID": "The unique identifier for each customer. DataType: int",  
 "Purchases": "The number of purchases that customer has made. DataType: int",  
 "ProductName": "The name of the purchased product. DataType: varchar"   
 },  
"table\_explanation": "The table contains information on purchases made by customers. Each row represents a product purchased by a specific customer."

Овој формат на изгледот не е компатибилен со форматот на json датотека, па ќе има проблеми при зачувување. За среќа, предходната наредба ги тргна сите случаи на ваков формат на излезот при зачувување.

Do not invent column names that are not already present in the table.

Финалната наредба кон системот, која додефинира дека не треба да се напишат колони кои не постојат. temperature вредноста е наместена на 0, но ова е финален слој на заштита против зачувување непостоечки табели.

Минатата колекција наредби му е дадена на системот како системска команда. Како корисничка команда, пак, се доделува следното:

Using the instructions given in the system prompt, explain the following table  
The column list is: {columns}  
The table you need to explain is:  
{sample}

Во оваа наредба, {columns} е заменето со листа од колоните кои ги содржи табелата, додека во sample е даден примерокот од табелата со форматиран излез, така што имињата на колоните се клучеви и вредностите на колоната се дадени како листи вредности.

Оваа функција се користи на секој примерок кој беше прочитан од базата, и резулатите се запишуваат во излезниот фолдер како {име\_на\_табела}.

Во случај да веќе постои фолдер со описи на табелите на дадената база на податоци, овој чекор се скока. Тоа е со цел да не се повикува ChatGPT повторно при секое прашање од корисникот.

**Постоечки недостатоци:**

Нажалост, дел од информацијата на примероците се губи со описите на колоните. На пример, понекогаш ChatGPT нема да препознае категориска варијабла ако таа е претставена како integer (има 4 можни вредности: 0,1,2,3 , но ChatGPT ја толкува како нумеричка непрекината вредност.

Покрај тоа, можноста да се зачуваат описи за табелите овозможува работа со повеќе табели истовремено, и многу помага при ограничувањата на влезни карактери во ChatGPT, особено при работа со големи бази.

## Зачувување на описите за понатамошно користење за прашања, како и запишување на истите како индекси во VectorStore користејќи LlamaIndex

Во минатиот чекор беа напишани описите на табелите кои понатаму ќе се користат за генерирање SQL прашања. Но, за да можат да се изберат само релевантните табели при создавање прашање, истите мораме да ги зачуваме во VectorStore користејќи LlamaIndex. Тоа ги трансформира целосните описи на табелите во вектори кои се зачувани по ембединзи, кои содржат клучни информации за описите кои можат понатаму да се пристапат со соодветни барања.

Во случај да веќе постои индекс со описи на табелите на дадената база на податоци, овој чекор се скока. Тоа е со цел да не се создава нов VectorIndex повторно при секое прашање од корисникот.

Од овој момент, прв пат се користи барањето поставено од корисникот за да се добие одговорот. Следната наредба се праќа на LlamaIndex:

What tables do I need to make a query that answers the following user's question:   
  
###  
  
# Question:  
{query\_str}  
  
###   
  
# Tables:  
{context\_str}  
  
###  
  
Only provide the table names and suggested columns in the answer.  
The format of the result should look like the following example:  
  
-> Example:  
<table\_name>: type, index, address  
<table\_name>: name, last\_name, experience  
-> End of example.  
  
Do not start the answer with Answer:  
Also, keep all tables in separate lines  
Keep the table names with the same lowercase and uppercase values as it originally was.  
  
\*\* DO NOT USE TABLES THAT ARE NOT RELEVANT TO THE QUESTION EVEN IF THEY CONTAIN SIMILAR COLUMNS TO THE TARGET \*\*  
  
###  
  
# Answer:

Во {query\_str} се заменува прашањето поставено од корисникот и во {context\_str} се внесени имињата на табелите кои припаѓаат на базата на податоци.

What tables do I need to make a query that answers the following user's question:

Ова е основната наредба која посочува што треба да се добие на излез. Во овој случај, излезот треба да ги содржи табелите кои треба да се користат за SQL прашање

# Референси

1. Menekşe, M. (2023). Envisioning the future of learning and teaching engineering in the artificial intelligence era: opportunities and challenges. Journal of Engineering Education, 112(3), 578-582. <https://doi.org/10.1002/jee.20539>
2. Malik Sallam, Nesreen A. Salim, Muna Barakat and Ala’a B. Al-Tammemi (2023) - ChatGPT applications in medical, dental, pharmacy, and public health education: A descriptive studyhighlighting the advantagesand limitations
3. Michael Dowling, Brian Lucey, ChatGPT for (Finance) research: The Bananarama Conjecture, Finance Research Letters, Volume 53, 2023
4. Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, Tegawendé F. Bissyandé (2023) - Is ChatGPT the Ultimate Programming Assistant -- How far is it?
5. Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, Jingren Zhou (2023) - Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation