Chat2Data: An Interactive Data Analysis System with RAG, Vector Databases and LLMs

Xinyang Zhao
Tsinghua University
Beijing, China
xy-zhao20@mails.tsinghua.edu.cn

Xuanhe Zhou
Tsinghua University
Beijing, China
zhouxuan19@mails.tsinghua.edu.cn

Guoliang Li
Tsinghua University
Beijing, China
liguoliang@tsinghua.edu.cn

ABSTRACT

Traditional data analysis methods require users to write programming codes or issue SQL queries to analyze the data, which are inconvenient for ordinary users. Large language models (LLMs) can alleviate these limitations by enabling users to interact with the data with natural language (NL), e.g., result retrieval and summarization for unstructured data and transforming the NL text to SQL queries or codes for structured data. However, existing LLMs have three limitations: hallucination (due to lacking domain knowledge for vertical domains), high cost for LLM reasoning, and low accuracy for complicated tasks. To address these problems, we propose a prototype, Chat2Data, to interactively analyze the data with natural language. Chat2Data adopts a three-layer method, where the first layer uses Retrieval-Augmented Generation (RAG) to embed domain knowledge in order to address the hallucination problem, the second layer utilizes vector databases to reduce the number of interactions with LLMs so as to improve the performance, and the third layer designs a pipeline agent to decompose a complex task to multiple subtasks and use multiple round reasoning to generate the results in order to improve the accuracy of LLMs. We demonstrate Chat2Data with two real scenarios, unstructured data retrieval and summarization, and natural language-based structured data analysis. The online demo is available at *http://vdemo.dbmind.cn.*

PVLDB Reference Format:
Xinyang Zhao, Xuanhe Zhou, and Guoliang Li. Chat2Data: An Interactive Data Analysis System with RAG, Vector Databases and LLMs. PVLDB, 17(10): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, technical report, and other artifacts have been made available at http://vdemo.dbmind.cn/.

1 INTRODUCTION

Existing data analysis methods have big barriers for ordinary users who cannot write programming code and SQL queries. Recently, machine learning techniques have been widely applied in data management systems to assist users in lowering barriers of using databases and improving efficiency, such as learned query optimizes [4], learned indexes [1], learned database configuration tuning [7]. However, traditional ML-based optimizing methods suffer from drawbacks such as high dependency on training data, poor generalization capability for data schema and distribution changes, and low interpretability for multi-round dialogue. These issues make it more difficult to adapt to different scenarios.

Fortunately, the emerging of large language models (LLMs) brings new opportunities to solve these problems. LLMs are statistical language models trained on vast amounts of textual data, which could be used to handle NLP tasks, such as understanding, summarization and generation tasks. In data management fields, LLMs could be used in data cleaning [2], database optimization [8], SQL equivalence determination [5], and root cause detection [6].

Although LLMs have brought breakthroughs in many fields, leveraging the capabilities of LLMs to make data management more intelligent, simpler, and low code presents the following three challenges: hallucination (due to without real-time data and domain knowledge for vertical domains), high cost for LLM reasoning, and low accuracy for complicated tasks.

Challenge 1. Lack of Domain-Specific knowledge. Firstly, LLMs may generate incorrect information or hallucinations for vertical domain data. Additionally, LLMs are insensitive to timeliness, models trained on old data could not promptly reflect the latest updates in database management systems. For example, the newest version of PostgreSQL for ChatGPT 3.5 is PostgreSQL 13.4, while the latest one is PostgreSQL 16.2 which is released in Feb 2024. Therefore, to effectively apply LLMs to database management, domain-specific knowledge needs to be integrated into the model in order to avoid hallucination and outdated information.

Challenge 2. High Cost of Interacting with LLMs. Secondly, LLMs consume huge resources to reason the results, which involve computation-intensive model inference or expensive API usage. For example, GPT 4 takes minutes to do reasoning. It is important to reduce the cost by avoiding to frequently interact with LLMs.

Challenge 3. Low Accuracy of Complicated Tasks. Thirdly, existing LLMs may not get high-quality results for complicate tasks, e.g., transforming NL to SQL queries (for data analytics) and pandas APIs (for visualization). LLMs need to comprehend the user intention and generate the complex pipelines to process the query. Therefore, converting intricate user requirements into instructions that the model can understand is a challenge. Moreover, one-round interaction may not get reasonable reasoning results and it requires to support multi-round interaction and reasoning.

To address these challenges, we propose Chat2Data, an LLM-enhanced data analysis platform. Chat2Data engages in natural language (NL) and multi-round dialogues with users to reduce the burden of interacting with the underlying data and achieve low-code (even zero-code) ability. Chat2Data extracts domain data, splits them into paragraphs based on their semantics, generates the

Proceedings of the VLDB Endowment, Vol. 17, No. 10 ISSN 2150-8097.
doi:XX.XX/XXX.XX

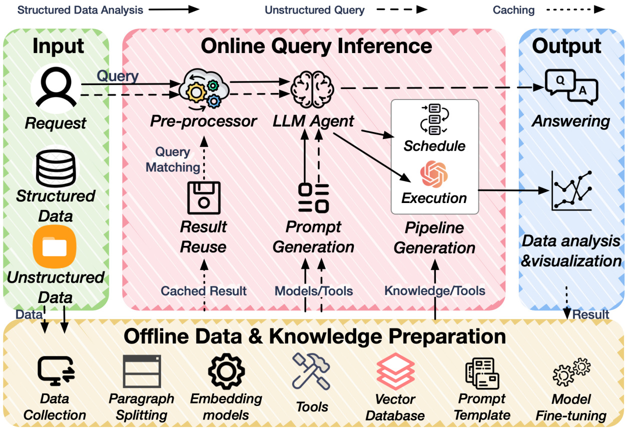


Figure 1: Chat2Data Workflow Overview.

embedding of each paragraph, and maintains these embeddings in vector databases. Then given an NL query, Chat2Data searches the local knowledge based on the query embedding and local knowledge embedding using vector databases, and uses the search results as prompts to input to LLMs. For unstructured data, Chat2Data uses LLMs to summarize the results. For structured data, Chat2Data uses LLMs to translate natural language queries into SQL queries to analyze the data and call the pandas APIs to visualize the results. Chat2Data also uses vector databases to cache the frequent queries, and if the cache hits, Chat2Data directly uses the cached results to answer the queries, thus avoiding to interact with LLMs.

2 CHAT2DATA SYSTEM DESIGN

To make data analysis more intelligent and low code, we built an interactive data analysis system called Chat2Data, which utilizes the Retrieval Augmentation Generation (RAG) for augmenting domain knowledge, vector databases for reducing the latency, LLM techniques for reasoning and inference, and LLM agent for generating multi-round pipeline to process complicated tasks. The overview of Chat2Data is shown in Figure 1. Chat2Data contains three main components, including *knowledge management, prompt generation via RAG, pipeline generation via LLM agent.* Specifically, the knowledge management module provides domain-specific knowledge for augmenting the LLMs. Given vertical domain data, it extracts the knowledge, generates the embeddings for the knowledge, and stores the embeddings and the extracted knowledge in vector databases. It also extracts tools and their corresponding explanations and puts them and their embeddings into vector databases, which are used for the online inference process. The prompt generation module extracts the user's intent, retrieves related knowledge from the vector database based on user intent, generates prompts with domain knowledge and tools, and inputs them into LLM. The pipeline generation module transfers the complex query into multiple operations and generates a pipeline. The LLM utilizes the input query and additional knowledge to generate results.

2.1 Offline Data and Knowledge Preparation

The offline vertical domain data and knowledge preparation provides the knowledge and APIs required to initialize our systems, which will be used to generate prompts to augment the query. The offline data and knowledge preparation first collects relevant knowledge and APIs. Then for the knowledge (and text explanations for APIs), Chat2Data splits the knowledge into text chunks based on their semantics. Then for each text chunk, Chat2Data selects an embedding model, generates an embedding, and inserts the embedding into vector databases. Later, for online query processing, Chat2Data generates embedding for the query, uses the vector databases to search relevant knowledge and APIs in order to generate effective prompts and input the prompt to the LLMs in order to improve the inference quality.

(1) **Knowledge Data Collection**: This component collects variable data sources, supporting both structured data and unstructured data. For structured data, it collects structured data from relational database or tabular data uploaded by users. It provides NL-based data analysis tasks on structured data. For unstructured data, it accepts the unstructured data uploaded by users, and supports NL-based data retrieval and summarization.

(2) **Paragraph Splitting**: To address the hallucination problem, we utilize the vertical domain data to augment the answers of user queries. Thus it is important to retrieve the most important and relevant domain data. However, the vertical domain data may be scattered and it is vital to split the domain data into text chunks based on their semantics. A naive paragraph-based split method may not achieve high-quality partitions. To address this problem, we propose a learning-based model to partition the domain data. We split the domain data into sliding window, generate an embedding for each slide window, and select the window with high independent semantics as partitions, i.e., the window embedding is standalone and different from other windows.

(3) **Embedding Model Selection**: The embedding model is important to the retrieval quality. We first select the widely-used embedding models and thousands of datasets with different data distributions, and train a classification model to predict the best embedding model for each data distribution. Then given a new domain data, we use the model to select the most appropriate model.

(4) **Tools**: This component collects the relevant APIs (e.g., database APIs, local pandas APIs) for data analysis and visualization tasks. Note that for each APIs, we need to add an explanation in order to make LLMs understand the APIs.

(5) **Vector Database**: To facilitate the domain knowledge retrieval, we use the vector databases to accelerate the efficiency. Given a query embedding, the vector database can efficiently find the most similar data embeddings based on embedding similarity functions. Moreover, the vector databases also need to support both predicate filter and vector search to improve the recall.

(6) **Prompt Template Preparation**: This component prepares effective templates, where each template contains instructions (e.g., translating NL to SQL queries), some examples (e.g., NL2SQL pairs), domain knowledge explanation, API explanation, etc. The prepared templates will be used for online query processing.

(7) **Model Fine-tuning.** To better grasp domain knowledge, the LLM can be fine-tuned to improve the result quality. In this case, we prepare some training examples with ground truth. Moreover, we may also need to provide human feedback to improve the fine-tuning quality. For example, for NL2APIs, which transforms NL

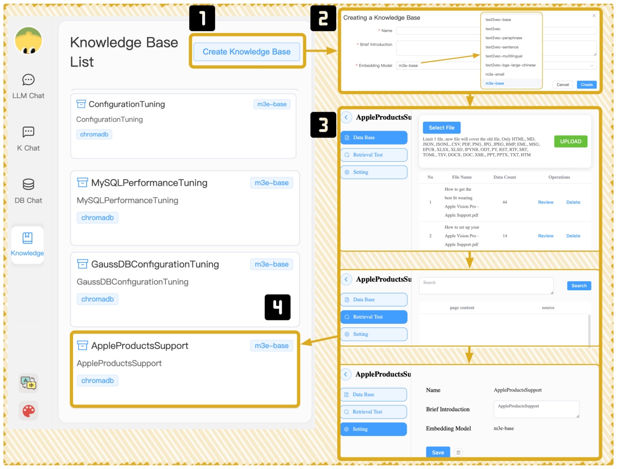
queries to pandas APIs, we generate many pairs of NL and APIs, and we use these pairs to fine-tune the models, e.g., LLama2 [3].

2.2 Online Query Inference

Query Pre-processor Module: Given an online query, Chat2Data first utilizes the embedding model to transform queries into corresponding vectors. Furthermore, Chat2Data analyzes the query intent and decides whether to use a single-round processing or a multiple-round processing based on the task difficulty. If it can be answered in a single round, Chat2Data then generates the prompts by searching the domain knowledge and APIs using the vector databases and inputting these prompts to LLMs. If it has to be answered by multiple rounds, Chat2Data uses LLM agents to generate a multiple-round pipeline. Moreover, to reduce the overhead of frequently interacting with LLMs, we also use a cache layer to improve the performance. Chat2Data first checks whether the query is cached in the cache layer using the vector databases. If so, Chat2Data directly uses the cached results to answer the query; otherwise interacts with LLMs and caches the frequent queries.

(1) **Prompt Generation via RAG:** Given a user query, Chat2Data pre-loads the necessary models or tools from the *knowledge management* module. For tasks like unstructured data query, two types of dialogue modes are provided: (*i*) LLM dialogue mode, using the fine-tuned LLM model and (*ii*) Knowledge base mode, using a RAG-enhanced LLM model. Based on the user's selection, Chat2Data loads the corresponding model from the knowledge management module. For the Knowledge base mode, users first set parameters like LLM temperature, number of knowledge matches, knowledge match score threshold, and history of dialogue discourse. Chat2Data utilizes vector databases to search relevant knowledge and then generates prompts based on these knowledge. For tasks like structured data analytics, Chat2Data searches relevant knowledge. Considering the case of translating NL to SQL, traditional LLMs may not capture the local data schema and local data values. Chat2Data will search relevant schemas and values as prompts to input to LLMs. Moreover, Chat2Data also detects the query intent based on LLMs. According to the query intent, Chat2Data also searches relevant APIs, e.g., pandas API for data analytics, and asks LLMs to call relevant APIs to process the query request.

(2) **Pipeline Generation via LLM:** With the knowledge-enhanced LLM model, user inputs are transformed into sequences of APIs. This extracted pipeline could be matched with corresponding data and rules stored in the knowledge base, allowing for rewriting and optimizing the transformed pipeline. In this case, Chat2Data executes the processes provided by the request parser module to handle and fulfill user requests. Our system is capable of: (i) utilizing knowledge or tools stored in the *knowledge base* module to further determine the required data sources and optimize the execution pipeline; (ii) executing the current pipeline and evaluating the quality of the generated results with user feedback; (iii) for unsatisfied outcomes, leveraging models from the knowledge base module to re-generate new execution pipelines. Chat2Data also designs an effective scoring method to evaluate the results based on previous queries and LLMs feedback. If the scoring function gives a low value, Chat2Data will regenerate new pipelines to refine the answers; otherwise Chat2Data will return the current answers.



Embedding Model m3e-base

Limit 1 file, new file will cover the old file, Only HTML, MD, JSON, JSONL, CSV, PDF, PNG, JPG, JPEG, BMP, EML, MSG, EPUB, XLSX, XLSD, IPYNB, ODT, PY, RST, RTF, SRT, TOML, TSV, DOCX, DOC, XML, PPT, PPTX, TXT, HTM

No	File Name	Data Count	Operations	
1	How to get the best fit wearing Apple Vision Pro - Apple Support.pdf	44	Review	Delete
2	How to set up your Apple Vision Pro - Apple Support.pdf	14	Review	Delete

Name AppleProductsSupport

Brief Introduction AppleProductsSupport

Embedding Model m3e-base

Figure 2: Vertical Domain Data and Knowledge Management

(3) **Effective Caching via Vector Databases:** Chat2Data caches frequently asked questions to the cache layer. Chat2Data generates the embeddings of user queries, and caches the embedding and corresponding answers with high frequency in the cache. Given a query, Chat2Data first generates an embedding of the query, finds highly similar queries in the cache, and returns the cached answers to the query; if Chat2Data cannot find similar queries, Chat2Data will input the query to LLMs. To improve the cache hit and knowledge search hit, Chat2Data also designs a multi-way search method to improve the recall, which not only uses embeddings but also uses keyword search and BM25 to improve the matching possibility.

3 DEMONSTRATION SCENARIOS

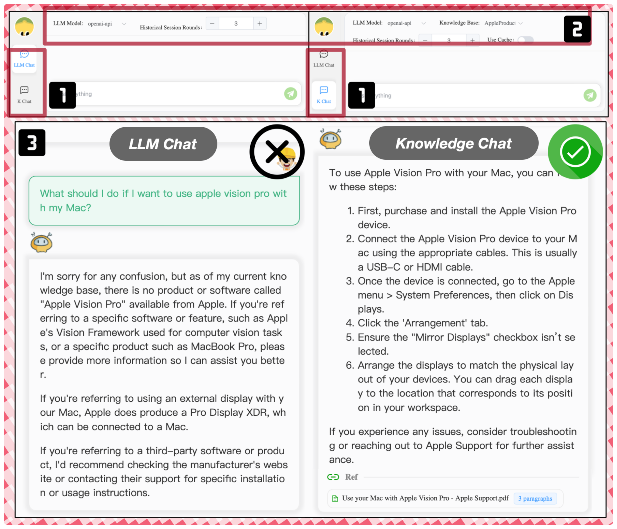
In this section, we present the implementation and scenarios of Chat2Data. For vector databases, we use milvus. For LLMs, we use ChatGPT API and also fine-tune LLama2. For pipeline generation agent, we extend LangChain framework to support unstructured data retrieval and structured data analysis. For vertical domain data, Chat2Data allows users to upload extra data (e.g., database or files), generate knowledge bases, and customize knowledge-enhanced LLM with just a few clicks.

Our demonstration aims to demonstrate the capability of Chat2Data to surpass traditional data analysis systems. Next, we will describe three scenarios in Chat2Data's demo. The first scenario demonstrates the knowledge management module. Then we showcase the user's interaction with the interface for unstructured data retrieval and summarization. The third scenario presents the interface for users to execute structured data analysis tasks.

3.1 Knowledge Management

In this scenario, we utilize the product knowledge from Apple[1]. First, we click the "Create Knowledge Base" button (❶) to create a new knowledge base named "Apple Products Support", fill in the relevant information about the knowledge base (❷), and select the embedding model to use (m3e-base). After clicking the create button, we could use buttons located on the left (❸) to manage the knowledge base. Chat2Data allows users to upload various types

[1] https://www.apple.com/



LLM Model: openai-api Historical Session Rounds: 3

LLM Model: openai-api Knowledge Base: AppleProduct Historical Session Rounds: 3 Use Cache:

What should I do if I want to use apple vision pro with my Mac?

I'm sorry for any confusion, but as of my current knowledge base, there is no product or software called "Apple Vision Pro" available from Apple. If you're referring to a specific software or feature, such as Apple's Vision Framework used for computer vision tasks, or a specific product such as MacBook Pro, please provide more information so I can assist you better.

If you're referring to using an external display with your Mac, Apple does produce a Pro Display XDR, which can be connected to a Mac.

If you're referring to a third-party software or product, I'd recommend checking the manufacturer's website or contacting their support for specific installation or usage instructions.

To use Apple Vision Pro with your Mac, you can follow these steps:

1. First, purchase and install the Apple Vision Pro device.
2. Connect the Apple Vision Pro device to your Mac using the appropriate cables. This is usually a USB-C or HDMI cable.
3. Once the device is connected, go to the Apple menu > System Preferences, then click on Displays.
4. Click the 'Arrangement' tab.
5. Ensure the "Mirror Displays" checkbox isn't selected.
6. Arrange the displays to match the physical layout of your devices. You can drag each display to the location that corresponds to its position in your workspace.

If you experience any issues, consider troubleshooting or reaching out to Apple Support for further assistance.

Use your Mac with Apple Vision Pro - Apple Support.pdf 3 paragraphs

Figure 3: Unstructured Data Retrieval and Summarization.

of files (e.g., JSON, excel, PDF). We upload several PDF files which contain knowledge for Apple products. After offline knowledge preparation processing, these uploaded files will be transferred into vector data, and saved in the vector database. The created knowledge base will be displayed in the "Knowledge Base List" (❹). In addition, Chat2Data provides a "retrieval testing" button to check whether the knowledge retrieval is successful, and a "Setting" button to modify or delete the current knowledge base.

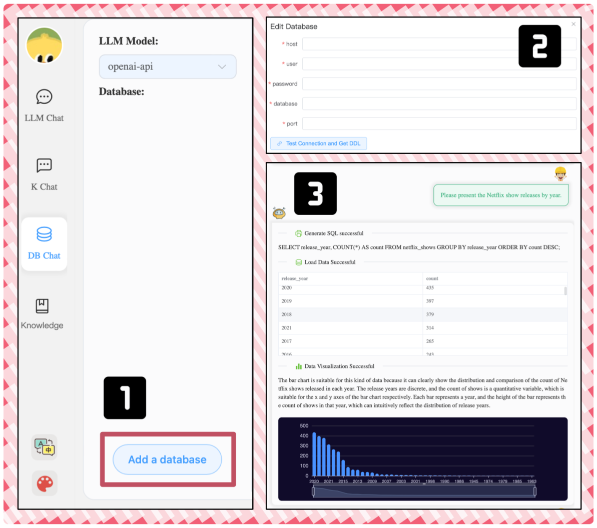
3.2 Unstructured Data Retrieval and Summarization

This scenario involves performing unstructured data retrieval and summarization tasks. By clicking the "LLM Chat" and "K Chat" buttons on the left sidebar(❶), users could switch the two modes of fine-tuned LLM and knowledge-augmented LLM. The configuration settings for these two modes are shown in Figure 3 (❷).

We choose the "Apple Products Support" knowledge base to illustrate Chat2Data's unstructured data retrieval and summarization ability. In LLM Chat mode, asking questions such as "how to use Apple Vision Pro in Mac", and Chat2Data provides better answers. As shown in Figure 3 (❸), comparing the answers generated by these two modes, we find that in LLM Chat mode, LLMs are insensitive to timeliness. The data used to train the model does not contain relevant information about this product. Therefore, LLM Chat mode cannot provide effective answers to this question. In contrast, in K-Chat mode, our system can provide effective answers to this question by retrieving it in the additional knowledge base and summarising them to provide better answers.

3.3 Structured Data Analysis

This scenario involves demonstrating the structured data analysis processing. We use the "Netflix Movies and TV Shows" dataset[2] to illustrate the process of structured data analysis. This dataset consists of listings of movies and TV shows available on Netflix. It contains the details such as the title, country, cast, directors,



LLM Model: openai-api

Database:

* host

* user

* password

* database

* port

Please present the Netflix show releases by year.

SELECT release_year, COUNT(*) AS count FROM netflix_shows GROUP BY release_year ORDER BY count DESC;

release_year	count
2020	435
2019	397
2018	379
2021	314
2017	265
2016	243

The bar chart is suitable for this kind of data because it can clearly show the distribution and comparison of the count of Netflix shows released in each year. The release years are discrete, and the count of shows is a quantitative variable, which is suitable for the x and y axes of the bar chart respectively. Each bar represents a year, and the height of the bar represents the count of shows in that year, which can intuitively reflect the distribution of release years.

Figure 4: Structured Data Analysis.

ratings, etc. We click the "Add a database" button to establish a connection to the database(❶). Some connection information, such as database username and host, should be provided(❸). Once successfully connected to the database, this interface will display the database's Data Definition Language (DDL) information. We can input the requirements with natural language into the dialogue box. For example, we want to analyze the release year of TV shows. After clicking the query execution button below, Chat2Data will generate prompts by extracting the database schema and values, and get the SQL query described by natural language. After executing the query, Chat2Data will display the query results and visualization results. As shown in Figure 4 (❸), Chat2Data shows the generated SQL, query result, visualization result and a brief visualization description.

ACKNOWLEDGMENTS

This paper was supported by National Key R&D Program of China (2023YFB4503600), NSF of China (61925205, 62232009, 62102215), Zhongguancun Lab, Huawei, TAL education, and Beijing National Research Center for Information Science and Technology (BNRist). Guoliang Li is the corresponding author.

REFERENCES

[1] Zhaoyan Sun, Xuanhe Zhou, and Guoliang Li. 2023. Learned Index: A Comprehensive Experimental Evaluation. *Proc. VLDB Endow.* 16, 8 (2023), 1992–2004.
[2] Nan Tang, Ju Fan, Fangyi Li, and et al. 2021. RPT: Relational Pre-trained Transformer Is Almost All You Need towards Democratizing Data Preparation. *Proc. VLDB Endow.* 14, 8 (2021), 1254–1261.
[3] Hugo Touvron, Louis Martin, Kevin Stone, and et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). https://doi.org/10.48550/ARXIV.2307.09288 arXiv:2307.09288
[4] Xiang Yu, Chengliang Chai, Guoliang Li, and Jiabin Liu. 2022. Cost-based or Learning-based? A Hybrid Query Optimizer for Query Plan Selection. *Proc. VLDB Endow.* 15, 13 (2022), 3924–3936. https://doi.org/10.14778/3565838.3565846
[5] Fuheng Zhao, Lawrence Lim, Ishtiyaque Ahmad, Divyakant Agrawal, and Amr El Abbadi. 2023. LLM-SQL-Solver: Can LLMs Determine SQL Equivalence? *arXiv preprint arXiv:2312.10321* (2023).
[6] Xuanhe Zhou, Guoliang Li, and et al. 2023. D-bot: Database diagnosis system using large language models. *arXiv preprint arXiv:2312.01454* (2023).
[7] Xuanhe Zhou, Guoliang Li, Jianhua Feng, and et al. 2023. Grep: A Graph Learning Based Database Partitioning System. *SIGMOD* (2023).
[8] Xuanhe Zhou, Zhaoyan Sun, and Guoliang Li. 2024. DB-GPT: Large Language Model Meets Database. *Data Science and Engineering* (2024), 1–10.

[2] https://www.kaggle.com/datasets/shivamb/netflix-shows