LAZY-MAN'S DATABASE MANAGEMENT SYSTEM

許喬淇
生物機電工程學系碩一
R12631001

謝欣妤
生物機電工程學系碩一
R12631009

鄭朝鴻
生物機電工程學系碩一
R12631013

連震宇
生物機電工程學系碩一
R12631012

ABSTRACT

The Lazy-Man's Database Management System simplifies querying unstructured and semi-structured data by allowing users to input documents and natural language, which are then processed using Large Language Models (LLMs) to generate structured tables and SQL queries. Inspired by the Evaporate system, this tool consists of a user interface built with Streamlit, the Evaporate model for data extraction, Google Gemini for text-to-SQL tasks, and a management system for database operations. Evaluated using spider databases with 100 questions of varying difficulty, the system achieved an overall accuracy of 0.77, with higher error rates for more complex queries, demonstrating its effectiveness and areas for improvement.

1. INTRODUCTION

1.1 Motivation

In real-life situations, most of the information we handle is unstructured or semi structured data, which is hard to define the complex relationship within it. For example, images and web pages require humans to reconstruct to be suitable for relational databases. Moreover, Structured Query Language (SQL) requires users to have a deep understanding of the database schema and SQL syntax. This poses significant barriers for non-technical users who may struggle to write complex SQL queries, leading to inefficiencies and limited access to valuable data insights. Moreover, even for experienced SQL users, constructing intricate queries can be time-consuming and error-prone, especially when dealing with large and complex database schemas. Text-to-SQL emerged as a solution to address the challenges associated with querying databases using traditional SQL statements. However, copying output SQL statements from Text-to-SQL to DBMS interfaces is quite trivial work. In this case, we aim to develop a system to cope with unstructured data, extract necessary information, and bridge Text-to-SQL to DBMS. The system would be capable of letting users input unstructured data or spoken language and returning prompt and precise responses to the users.

1.2 Background Information

The conversion of heterogeneous data presents a longstanding conundrum in the realm of data management [3, 7, 12, inter alia.]. To address this formidable challenge, existing methodologies often rely on presumptions about the data domain [10, 14, 15], the syntactic frameworks encapsulating valuable insights [2, 3, 4, 5, 6, 9, 12, 13], or the formats which data is stored, like XML files [8]. For example, Cafarella et al. [6] concentrated their efforts on representing facts as triples in earlier iterations, underpinned by hypernymic "is-a" relationships between entities. In recent years, the surge of deep learning methodologies has significantly influenced various system architectures [13]. Nonetheless, there were still several problems and limits in the neural system. Firstly, they necessitate domain and document format in specific training. Secondly, the neural system excels at reasoning sentences rather than long article files. Lastly, their efficacy relies heavily on sophisticated linguistic tools to impose structure into unorganized textual data. For the management of language model data, Chen et al. [16] introduce a system designed for querying heterogeneous data lakes through the utilization of in-context learning. In the data programming part, Simran Arora et al. [1] introduce and assess Evaporate, a system that employs LLMs to produce organized perspectives of semi-structured data lakes.

1.3 Main Approach

Our goal with the LAZY-MAN'S DATABASE MANAGEMENT SYSTEM is to create a user-friendly interface that allows easy access to and retrieval of information from databases. Inspired by the paper "Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes," we implemented the system described by the authors, Evaporate, which is available on GitHub. Users can input unprocessed semi-structured or unstructured data, and Evaporate will generate reasonable attributes from the documents. By leveraging existing LLMs, we produce our candidate functions to extract possible target attributes, filter these functions, and output the results in a structured table. For the user interface, we developed our web application using Streamlit, which is built on Python. We employ Google's Gemini Pro LLM to help read database content and generate SQL queries. The operational flow of the LAZY-MAN'S DATABASE MANAGEMENT SYSTEM is as follows: Users upload documents they want to process into our system. The output from Evaporate is then passed to Gemini as a knowledge base. Users can input natural language queries to request the desired results. Our system executes the SQL queries generated by Gemini in MySQL, displaying the query results and SQL syntax in the user interface. If users are not satisfied with the results or the generated SQL syntax, we provide an option for syntax modification to ensure the most accurate results are delivered.

2. PRELIMINARIES

We first point out the problems encountered and expected system outcomes.

2.1 Problem Setting

Organizations often seek valuable insights from diverse data sources, such as the web, corporate data lakes, and electronic health records, using Information Extraction (IE). IE is the process of converting unstructured data into structured data. In the current era of information proliferation, the exponential growth of data has

made IE increasingly crucial. However, in their raw form, these unstructured data sources provide limited support for analytical queries and require labor-intensive and time-consuming processing. Consequently, many domains have shifted to automatic or semi-automatic IE methods.

Some researchers using general LLM extract easily usable structured data from unstructured documents (e.g. HTML webpages, PDFs, text) [1, 11]. However, iterative prompt tuning and testing are required to enhance structured data extraction performance, which may force the user to understand the nature of the unstructured data. In addition, these systems can only generate the data structure but lack the ability to create SQL schemas. As a result, users may encounter challenges in crafting prompts that produce optimal results, and those without background knowledge in database languages would confront obstacles when creating database schemas using the generated structured data.

In addition to the challenges of extracting structured data, the general unfamiliarity with SQL language among non-engineering users presents another significant hurdle. In many scenarios, individuals need to retrieve specific information from databases; however, those who lack knowledge of SQL language must invest considerable time in understanding the database's attributes and learning how to construct and execute SQL queries. Consequently, although databases are designed for rapid information retrieval, their usability is significantly diminished for users without SQL proficiency, thereby reducing their overall utility.

In light of these challenges, we aim to develop an automated, end-to-end system that minimizes user effort. The system will integrate LLMs, the Evaporate model for unstructured-to-structured data transformation, and our self-designed management system. Further details will be illustrated in section 3.

2.2 Review Large Language Models

In this section, we will cover definitions and background knowledge of LLMs.

LLMs, such as OpenAI ChatGPT, Microsoft Copilot, and Google Gemini, are multimodal artificial intelligence systems that were built based on Transformer architecture to process variable-length input sequences of text or images.

These models were trained using a technique called unsupervised learning on vast amounts of data, including text, emails, chats, and documents. During training, the model learns to predict the next word in a sentence based on the context of the words that came before it. This process allows the model to develop a deep understanding of language structure, grammar, and semantics.

LLMs have been the focus of extensive research across various domains. In data mining, LLMs offer significant potential in pattern recognition and predictive analytics; in database management, LLMs can understand the semantic meaning of natural language and then generate or optimize queries according to user instructions (Figure 1).

Figure 1: Application of LLM on SQL.

2.3 Expected Outcomes

The expected outcome of Lazy-Man's Database Management System would be an automated, end-to-end, and user-friendly system. The system would be able to structurize unstructured and semi-structured data by generating structured views, moreover, it utilizes LLM to parse user input commands and the structure and content of the database. By understanding the content of the database, the SQL queries generated by the model can achieve higher accuracy, avoiding queries that do not meet the user's requirements. We expect the results of this research to be capable of structuring various data formats and demonstrate their accuracy and robustness in understanding user semantics and converting them into SQL queries.

3. LAZY-MAN'S DATABASE MANAGEMENT SYSTEM PROTOTYPE

3.1 Overview

The proposed Lazy-Man's Database Management System consists of four parts: user interface, Evaporate, Google Gemini, and Management System. In the following sections, we delve into each of these components in detail, exploring their functionalities, design principles, and how they interact to form a cohesive and efficient database management system.

3.2 User-interface Design

Our project, titled 'Lazy-Man's Database Management System,' embodies the design philosophy that we aim to provide users with a rapid information searching function without requiring excessive thought or background knowledge. As a result, our user interface is straightforward and easy to understand, as shown in Figure 2.

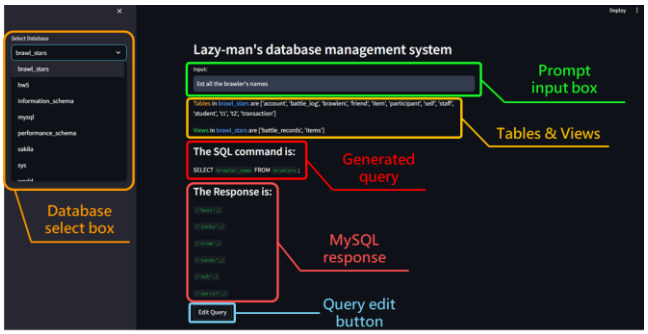

Figure 2: User interface.

The user interface was built using Streamlit, an open-source Python framework specifically designed for data scientists and AI/ML engineers. It allows engineers to create interactive data applications on web pages. In the web page we designed, users can select the desired database using the **Database select box**. Subsequently, information about **tables and views** will be displayed, allowing users to understand which data is accessible. At this point, users can enter their queries in the **Prompt input box**, detailing the information they wish to obtain. The application will then return the generated SQL query as well as the output results from MySQL.

Additionally, we also designed an **Edit query** button. When the generated query does not fully align with the user's instruction, the user can utilize this button to fine-tune the query to obtain the desired information (Figure 3).

Figure 3: Edit Query button

3.3 Evaporate

Evaporate, a system powered by LLMs that generates structured views of heterogeneous, semi-structured data lakes. It automates structured information extraction from diverse documents without manual labeling or domain-specific training. Evaporate offers a user-friendly interface for inputting documents, automatically producing structured views. Evaporate implementations, including direct and code extraction strategies, highlighting a cost-quality trade-off. Additionally, Evaporate-Code+ with weak supervision for quality enhancement at low costs. Figure 4 illustrates the Evaporate process, from input format, extracting attributes, generating and selecting appropriate functions, to finally outputting a structured table.

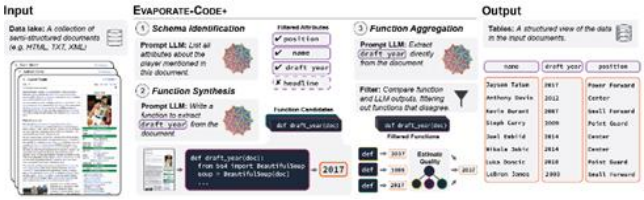

Figure 4: Evaporate operational flowchart [1].

Evaporate-Direct (Direct Extraction): This component of Evaporate involves the LLM directly extracting values from documents. The prompt guides the LLM to both identify the schema and extract values accordingly [1].

Evaporate-Code (Code Synthesis): In this aspect of Evaporate, the LLM synthesizes code used to process documents at scale [1]. This process involves two sub-tasks: (1) schema synthesis: The LLM identifies the schema and (2) function synthesis: The LLM generates code functions for processing the documents.

Evaporate-Code+ (Code Synthesis + Aggregation): This extended version of Evaporate enhances quality while maintaining low costs through the following steps: (1) schema identification, as in Evaporate-Code, (2) function synthesis, as in Evaporate-Code, except that multiple candidate functions are generated per attribute, and (3) function aggregation: Introducing a weak supervision (WS)-based algorithm to aggregate different predictions for attribute values across documents, thereby improving quality while controlling costs [1].

Figure 5 shows how Evaporate integrated the prompt. The prompt template, which includes placeholders for in-context examples and the inference example (i.e., data lake documents), is applied to each document in the data lake [1].

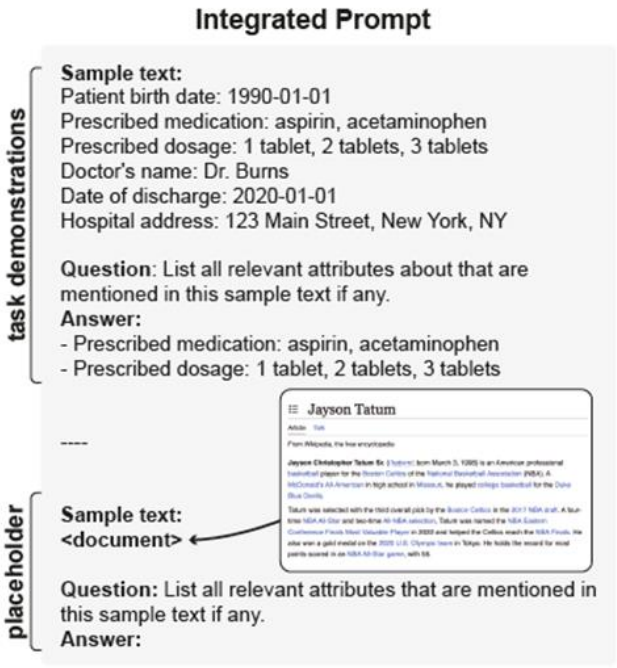

Figure 5: Prompt for Evaporate-Direct structure [1].

Figure 6 shows an Evaporate function synthesis prompt. The in-context examples show pairs of text snippets and functions to extract an attribute of interest [1].

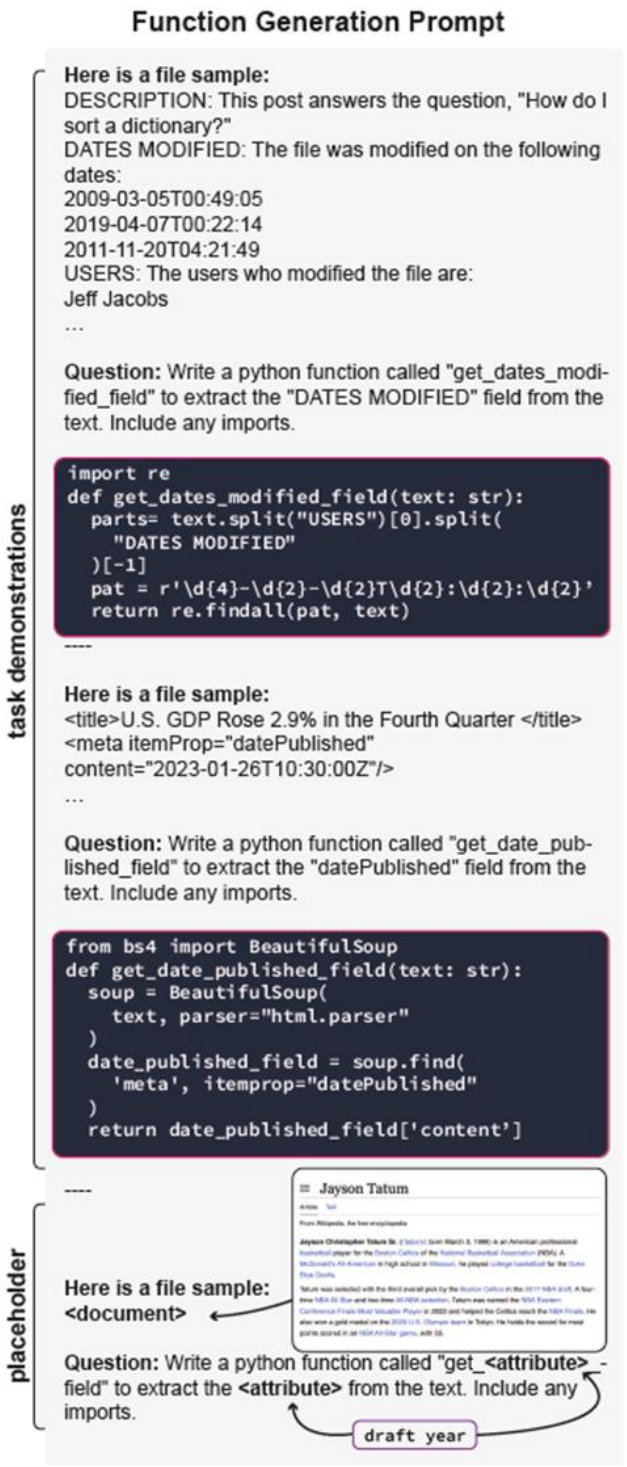

Figure 6: A representative prompt for function synthesis, containing two data lake agnostic in-context examples [1].

3.4 Google Gemini

In this study, we utilized Google Gemini 1.5 Pro to handle text-to-SQL tasks. The model is able to process up to 1 million tokens with a context window. Here we delve into Google Gemini, an LLM designed for various tasks, including text-to-SQL. We would explore its structure and how it compares to other LLMs in assisting text-to-SQL.

3.4.1 Structure

The structure of Google Gemini leverages a combination of Transformer architecture and the **Mixture-of-Experts (MoE)** technique. MoE divides the model into multiple smaller expert networks to specialize in different sub-tasks. This technique enhances efficiency and potentially improves performance on specific tasks like text-to-SQL.

3.4.2 Comparison with other LLMs

Compared to primarily text-focused LLMs like ChatGPT or Copilot, Gemini's ability to handle diverse data types could prove beneficial in text-to-SQL tasks. It is able to utilize additional information such as the exported SQL files from databases to enhance its understanding of user intent and formulate precise SQL queries.

3.5 Management System

The management system is the key to connecting the aforementioned applications (Figure 7). It is capable of performing various operations on the database, including retrieving desired data, understanding the database tables, and making any necessary modifications to the database. By pre-reading the database, it can determine which tables are needed based on the user's input requests and read only the necessary table contents. For structured data desired by the user, the management system inputs the unstructured or semi-structured data into Evaporate to obtain structured data. Subsequently, the management system inputs the user's requests, the extracted data from the database, and the generated structured data into Google Gemini. Then, through SQL queries obtained via the LLM, it performs operations on the database. Finally, the information desired by the user is displayed on the user interface.

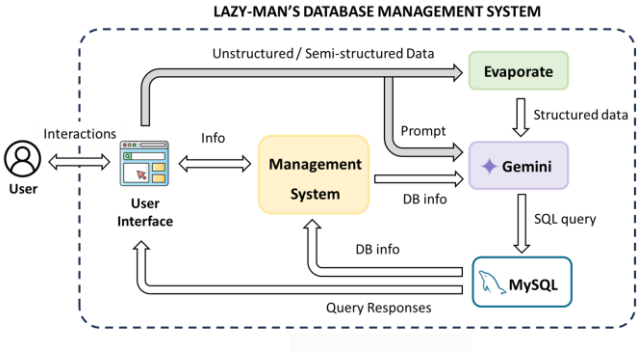

Figure 7: System Operation Diagram.

4. EVALUATION

4.1 Experiment Data

Based on our experiment, we utilized spider databases (Figure 8) to validate the accuracy of Lazy-man's database management system. In this study, we selected four random examples from the spider databases as our test subjects.

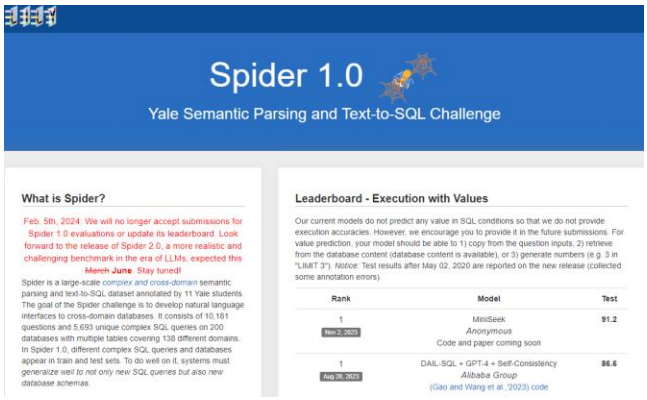

Figure 8: The Spider website.

4.2 Experiment Setting

In the experiment process, we tested our lazy-man management system by using a total of 100 questions and assigning 25 questions to each database. We classified the questions into different levels, including Easy, Medium, and Hard. In this study, we referred to the grading method on the Spider website and simplified it to three difficulty levels. There are examples of the questions.

- Easy: What is the number of cars with more than 4 cylinders?

  Output:
  SELECT COUNT(*)
  FROM car_data
  WHERE cylinders > 4

- Medium: For each stadium, how many concerts are there?

  Output:
  SELECT T2.name, COUNT(*)
  FROM concert AS T1 JOIN stadium AS T2
  ON T1.stadium_id = T2.stadium_id
  GROUP BY T1.stadium_id

- Hard: Which countries in Europe have at least 3 car manufacturers?

  Output:
  SELECT T1.country_name
  FROM countries AS T1 JOIN continents
  AS T2 ON T1.continent = T2.cont_id
  JOIN car_markers AS T3 ON
  T1.country_id = T3.country
  WHERE T2.continent = 'Europe'
  GROUP BY T1.country_name
  HAVING COUNT(*) >= 3

After classification, we have 40 Easy, 39 Medium, and 21 Hard questions. In Figure 9, it shows the distribution of questions within each database.

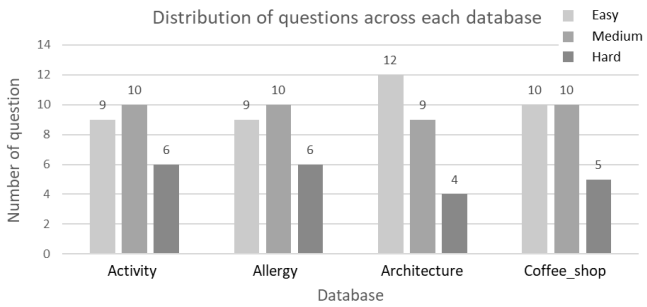

Figure 9: The distribution of questions from different levels within each database.

Next, these 100 questions were utilized in our lazy-man database management system, and compared the output answers with the original spider database answers.

4.3 Experiment Results

In this section, we discuss the results of validation. The mean accuracy is 0.77. According to different levels, the accuracy for Easy, Medium, and Hard questions are 0.85, 0.77, and 0.62 respectively. In Figure 10, it compared the number of correct and incorrect answers across different levels. Although the number of questions assigned to the Hard is less, the proportion of incorrect answers is higher than in the other. This performance shows that the higher error rates for more complex queries highlight both the system's effectiveness and the potential areas for enhancement.

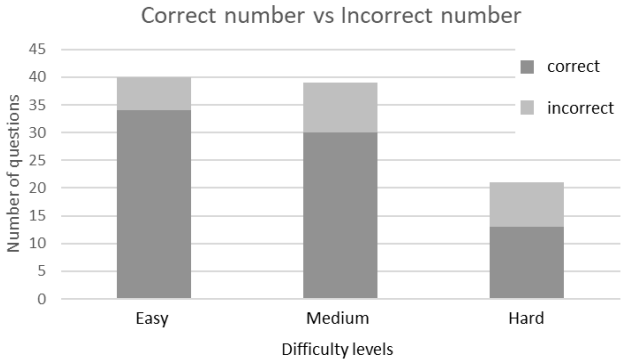

Figure 10: The correct and incorrect numbers in different levels.

5. DISCUSSION

5.1 Limitation

Our LAZY-MAN'S DATABASE MANAGEMENT SYSTEM still has considerable room for improvement. These areas include implementing the full functionality of Evaporate, increasing the amount of data that can be processed at once, improving the response speed of the LLM, and verifying its accuracy. During testing, we occasionally encounter unsupported syntax and errors. Additionally, we need to address how to provide feedback when users encounter issues. These are the current limitations of our research and problems that need improvement.

5.2 Future Works

Based on feedback from our peers, our future work will focus on optimizing the system's performance and stability. We will also fully integrate Evaporate's functionality into our system. Additionally, we will consider whether our system can continue to operate quickly and stably when handling larger databases. Currently, the accuracy of our system's responses needs improvement. We will experiment with different models and fine-

tune them to better align with user data. To enhance the user experience, we will also provide solutions for any errors that occur during system operation.

6. CONCLUSION

We have successfully integrated MySQL with an LLM to establish a Lazy-Man's Database Management System. Currently, this system can handle SQL queries of easy to medium complexity. This system effectively assists users who are not familiar with SQL queries by allowing them to interact with the database using natural language. Additionally, it integrates Evaporate to structure unstructured data that users wish to apply to the database. This system is a significant boon for those seeking a more effortless database management experience.

7. REFERENCE

[1] Arora, S., Yang, B., Eyuboglu, S., Narayan, A., Hojel, A., Trummer, I., & Ré, C. (2023). Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433*.

[2] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. A survey on open information extraction. In Proceedings of the 27th International Conference on Computational Linguistics, 2018.

[3] Eugene Agichtein Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, 2000.

[4] Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Mausam, and Soumen Chakrabarti. Openie6: Iterative grid labeling and coordination analysis for open information extraction. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.

[5] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. Open language learning for information extraction. 2012.

[6] Michael J. Cafarella, Christopher Re, Dan Suciu, Oren Etzioni, and Michele Banko. Structured querying of web text. In Conference on Innovative Data Systems Research (CIDR), 2007.

[7] Michael J Cafarella, Dan Suciu, and Oren Etzioni. Navigating extracted data with schema discovery. In WebDB, pages 1–6, 2007.

[8] Minos Garofalakis, Aristides Gionis, Rajeev Rastogi, Sridhar Seshadri, and Kyuseok Shim. Xtract: A system for extracting document type descriptors from xml documents. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 165–176, 2000.

[9] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. In *AAAI*, 2004.

[10] P. DeRose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. *VLDB*, 2007.

[11] Peng, R., Liu, K., Yang, P., Yuan, Z., & Li, S. (2023). Embedding-based retrieval with llm for effective agriculture information extracting from unstructured data. *arXiv preprint arXiv:2308.03107*.

[12] S. Brin. Extracting patterns and relations from the worldwide web. In *WebDB*, 1998.

[13] Shaowen Zhou, Bowen Yu, Aixin Sun, Cheng Long, Jingyang Li, Haiyang Yu, Jian Sun, and Yongbin Li. A survey on neural open information extraction: Current status and future directions. *IJCAI22*, 2022.

[14] T.S. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu. Avatar information extraction system. *IEEE Data Eng*. Bull, 2006.

[15] Xiang Deng, Prashant Shiralkar, Colin Lockard, Binxuan Huang, and Huan Sun. Dom-lm: Learning generalizable representations for html documents. 2022.

[16] Zui Chen, Zihui Gu, Lei Cao, Ju Fan, Sam Madden, and Nan Tang. Symphony: Towards natural language query answering over multi-modal data lakes. *CIDR*, 2023.