EHRAgent Reproduction and Extension

PETER IAT KEI HO, University of Texas, Austin, USA

Even though EHRAgent based on gpt4 was proven to have better result than other solution by the work this report is based on, There are still many challenges in reproduction and optimization.

CCS Concepts: • **Applied computing → Health care information systems**; • **Software and its engineering → Software prototyping**; • **Information systems** → *Structured Query Language*; • **Computing methodologies** → *Learning from critiques*; Learning to rank.

Additional Key Words and Phrases: EHR, Agent, LLM, Azure, OpenAI, Python, Debug, SQL, MIMIC

ACM Reference Format:
Peter Iat Kei Ho. 2024. EHRAgent Reproduction and Extension. 1, 1 (December 2024), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

According to https://www.thebalancemoney.com/causes-of-rising-healthcare-costs-4064878, cost of healthcare has significantly increased over past decades and years, which can be a risk for health and well being of Americans impacting every age group. I believe there are many factors with many different solutions. One of them is leveraging technology to help medical professionals to focus their time and energy on patients and/or research but not simple data retrieval or analysis.

LLM (Large Language Model) has gained signficant popularity over the past years. It can capture semantics of text, memorize past conversations, and generate code. This report evaluates past work that utilizes MIMIC[1][2][3] dataset by reproducing and considering alternatives in data retreival and analysis.

2 Related Work

This a reproduction and alternation of part of the work - ["EHRAgent: Code Empowers Large Language Models for Complex Tabular Reasoning on Electronic Health Records"]https://arxiv.org/abs/2401.07128. EHRAgent is an LLM agent empowered with a code interface, to autonomously generate and execute code for complex clinical tasks within electronic health records (EHRs). The original project page is available at [this link](https://wshi83.github.io/EHR-Agent-page/). [4]

Text to SQL generation is another related project that the former utilizes its evelation logic with slight modification. Information of the project can be found at https://github.com/wangpinggl/TREQS/blob/master/README.md [5]

3 Methodology

3.1 Agent Execution Steps

(0) Predefined knowledge is intialized as memory, including question, knowledge, and associated solution.

Author's Contact Information: Peter Iat Kei Ho, peter-ho@live.com, University of Texas, Austin, Austin, Texas, USA.

ACM XXXX-XXXX/2024/12-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

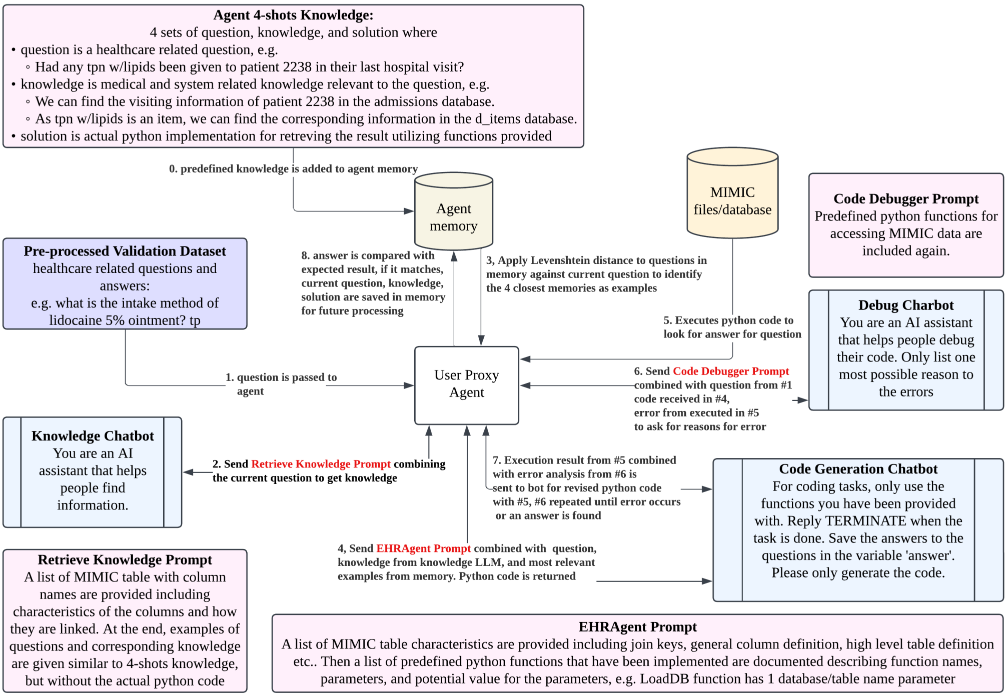
Agent 4-shots Knowledge:
4 sets of question, knowledge, and solution where
• question is a healthcare related question, e.g.
◦ Had any tpn w/lipids been given to patient 2238 in their last hospital visit?
• knowledge is medical and system related knowledge relevant to the question, e.g.
◦ We can find the visiting information of patient 2238 in the admissions database.
◦ As tpn w/lipids is an item, we can find the corresponding information in the d_items database.
• solution is actual python implementation for retreving the result utilizing functions provided
0. predefined knowledge is added to agent memory
Agent memory
MIMIC files/database
Code Debugger Prompt
Predefined python functions for accessing MIMIC data are included again.
Pre-processed Validation Dataset
healthcare related questions and answers:
e.g. what is the intake method of lidocaine 5% ointment? tp
8. answer is compared with expected result, if it matches, current question, knowledge, solution are saved in memory for future processing
3, Apply Levenshtein distance to questions in memory against current question to identify the 4 closest memories as examples
5. Executes python code to look for answer for question
Debug Charbot
You are an AI assistant that helps people debug their code. Only list one most possible reason to the errors
1. question is passed to agent
User Proxy Agent
6. Send Code Debugger Prompt combined with question from #1 code received in #4, error from executed in #5 to ask for reasons for error
Knowledge Chatbot
You are an AI assistant that helps people find information.
2. Send Retrieve Knowledge Prompt combining the current question to get knowledge
7. Execution result from #5 combined with error analysis from #6 is sent to bot for revised python code with #5, #6 repeated until error occurs or an answer is found
Code Generation Chatbot
For coding tasks, only use the functions you have been provided with. Reply TERMINATE when the task is done. Save the answers to the questions in the variable 'answer'. Please only generate the code.
Retrieve Knowledge Prompt
A list of MIMIC table with column names are provided including characteristics of the columns and how they are linked. At the end, examples of questions and corresponding knowledge are given similar to 4-shots knowledge, but without the actual python code
4, Send EHRAgent Prompt combined with question, knowledge from knowledge LLM, and most relevant examples from memory. Python code is returned
EHRAgent Prompt
A list of MIMIC table characteristics are provided including join keys, general column definition, high level table definition etc.. Then a list of predefined python functions that have been implemented are documented describing function names, parameters, and potential value for the parameters, e.g. LoadDB function has 1 database/table name parameter

Fig. 1. EHRAgent execution flow

(1) A question from validation data set is passed into UserProxyAgent
(2) During initialization process of main code generation chat, UserProxyAgent sends question to Knowledge chatbot to retrieve knowledge regarding the question applying chain of thought. Few shot learning is also applied as the prompt to the knowledge chatbot includes examples of questions and their corresponding knowledge.
(3) Retrieve 4 most similar examples by comparing current question with questions in memory according to levenshtein distance, applying few shot learning.
(4) Invoke code generation chatbot with table definition, function definition, knowledge retrieved from knowledge chatbot, examples from memory.
(5) Execute code retrieved from bot and identify answer or errors. IF errors were encountered, capture errors and provide potential hints in resolving them.
(6) When error was thrown in the last step, functions documentation, code executed and execution error are sent to debugger chatbot to identify potential fixes.
(7) Original error message and suggestions returned by debugger chatbot are packaged as response back to code generation cahtbot for it to review and regenerate a better response. As chat with code generation chatbot continues until error occured related to exceeding max token size or termination message was identified.
(8) The response is then compared with predefined answer, if the answer was correct, the question, knowledge and solution are all stored in memory for future reference.

3.2 Issues Regarding Original Work

1. `Cost`: The table cost was referenced in multiple places of the original implementation including examples, validaton questons and answers, but in MIMIC-III v1.4, cost is not a file available for download, nor in the demo dataset. Any question related to cost can't be answered, and solution based on the example referring to cost would result in a table not found error, which can't be recovered.
2. `Python Package dependency`: In requirements.txt, autogen version 1.0.16 was referred, but 0.3.2 is the latest when this report was written. 0.3.3 was released afterwards, but it's still quite far from 1.0.16.
3. `Sqlite3 calendar calculation`: After setting up sqlite3 by importing MIMIC-III v1.4 csv files, sqlite3 database datetime function doesn't use current date even though current_time was passed in. It used 2000-01-01, so with input of -1 day, 1999-12-31 was returned.
4. `LoadDB requirement`: Since LoadDB reads the whole table into memory, for larger table, out of memory exception was thrown. Due to re-reading the same csv or csv.gz files for every LoadDB command for various questions, disk IO can cause unnecessary delays when reading against the same data.
5. `Validation set issue`: When trying to identify reasons for incorrect responses, SQL embedded in the validation data set was used to identify and verify given answers. There are answers that are incorrect based on data retrieved from MIMIC-III v1.4.
6. `Execution challenge`: Since some code generated by chatbot can be quite inefficient, there are times that it takes more than hours to run, and sometimes it can be hard to tell how long it will take for the process to terminate. So with more than 500 validation questions, it takes more than days to finish evaluating 1 implementation.

3.3 Adjustments

1. `Cost`: Since examples related to cost won't be relevant and validation questions related to cost will not work, 24 questions and answers involving costs were removed. Examples related to costs were also adjusted to better illustrate other implementation.
2. `Python Package dependency`: Autogen 0.3.2 was used instead of 1.0.16, flaml[automl] was added due to other dependency. And mariadb 1.1.11 was added to support accessing MariaDB.
3. `Sqlite3 calendar calculation`: Calendar calculation was changed to MariaDB DATE_ADD which evaluates duration based on current datetime instead of 2000.
4. `LoadDB requirement`: LoadDB and several other functions are changed to defer execution similar to Apache Spark, so only result after filtering and selected columns is returned and read from data source.
5. `Execution challenge`: Due to the time sensitive nature of this report, when the evaluation of a question takes more 45 minutes, the process is killed with question id marked as not complete and restarted.
6. `Database backend`: To help improve data operations, instead of loading from files every time, all operations invokes MariaDB backend with database indexes setup for larger tables.
7. `New functions`: Given database operations aren't as straightforward to be mapped with results returning multiple values or list when only one value is needed, new functions are added including GetValue getting a single, GetValues getting multiple values, GetCount counting the number of rows of a data set.

4 Result

Table 1. Execution Result

	Number of Question	Success Rate	Completion Rate
EHRAgent from original work[4]	581	53.10	91.72
EHRAgent w/o cost	556	26.08	69.96
EHRAgent with MariaDB w/o cost	556	26.08	69.96

Since the cost file is not available and some of the question's ground-truth answer is incorrect after verifying by the provided SQL against MIMIC data, I believe it's likely that the dataset of the original work[4] is different from the MIMIC data set used in this report. With the result being comparable between EHRAgent and EHRAgent with MariaDB, the code generation has not improved even though some new functions are provided. But the amount of time it took to finish processing the full set of questions has improved quite a bit. Due to the process being randomly crashed and processing of some questions were killed inconsistently , duration of processing was not accurately captured. But the amount of time it took to finish without MiraDB was signiificantly larger than the processing with MariaDB.

5 Conclusion

EHRAgent[4] promised significant improvement comparing to related work. This report attempts to reproduce it, evaluate problems during the process, and make adjustments to the solution. But even though the dataset mentioned being MIMIC-III, the dataset doesn't seem to match the MIMIC-III dataset available on PhysioNet - https://physionet.org/content/mimiciii/1.4/ which caused a big difference between the original result and reproduced results.

Even though improved solution with MariaDB took signficantly less time, correction rate and completion rate were the same. I believe incorrect ground truth can also caused this correction rate to be lower than expected in both evaluation. Potential next steps can be:

- having a set of verification questions and ground truth answers matching the test data.
- adjusting initial memory so they are more diverse.
- persisting memory captured by answering questions correctly so that when the process hangs and needs to be destroyed, memory can be recovered.
- providing more robust filtering examples, as some generated code has applied multiple filtering which can be incorrect in some cases.

References

[1] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P.C. Ivanov, R. Mark, J.E. Mietus, G.B. Moody, C.K. Peng, and H.E. Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. (2000), e215–e220. https://doi.org/10.13026/C2XW26

[2] Alistair Johnson, Tom Pollard, and Roger Mark. 2016. MIMIC-III Clinical Database. version 1.4 (2016). https://doi.org/10.13026/C2XW26

[3] Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data* 3 (2016). Issue 1. https://doi.org/10.1038/sdata.2016.35

[4] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce Ho, Carl Yang, and May D Wang. 2024. Ehragent: Code empowers large language models for complex tabular reasoning on electronic health records. *EMNLP* (2024).

[5] Ping Wang, Tian Shi, and Chandan K Reddy. 2020. Text-to-SQL Generation for Question Answering on Electronic Medical Records. In *Proceedings of The Web Conference 2020*. 350–361.