AI-Powered Question-Answering System Based on Web Traffic Logs

Furkan Küçük

18/08/2024

Contents

Executive summary

The program I developed filters important information from logs generated by the Apache web server and uploads the filtered information to the Pinecone vector database. It then extracts the requested information from the user query using NLP, regex, and filtering, retrieves the data from the vector database, re-filters it to enhance accuracy, and finally converts the information into natural language using the GPT-2 model to present it to the user.

How to use

To use the program, after launching it, you need to send queries to the "Ask me anything:" prompt, such as:

"Retrieve all user data from August"

"Retrieve all user data for IP: 127.0.0.1"

"Retrieve all user data from 17 August 2024, visited page: phpmyadmin/"

"Gather all data from August 17"

"Get all user information for visited page: phpmyadmin/"

Since the program does not use an overly advanced entity extraction method, spaCy's NLP processing may sometimes fail to extract key information from certain queries, leading to unexpected results.

Apache logs has stored IPv6 addresses, and because of that localhost is seen as “::1”

Detailed summary

Log parsing and uploading to database

The program parses logs generated by Apache, such as:

::1 - - [12/Aug/2024:15:22:33 +0300] "GET /dashboard/ HTTP/1.1" 200 5187 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36 OPR/111.0.0.0"

It extracts important information like IP address (::1), day/month/year (12/Aug/2024), and visited page (/dashboard/). This information is then combined into a string in the format "Day of month: {day}, Month of year: {month}, Year: {year}, IP address: {ip}, Visited page: {page}" and uploaded to the Pinecone vector database after vectorization.

Retrieving data from the database

The user’s query is processed using the NLP logic of spaCy’s "en_code_web_sm" model. Important information is extracted from the processed query with the help of NLP, and where NLP falls short, simple regex and filtering are used. The extracted information is then stored in a dictionary.

For example, if the user query is "Retrieve all user data from 17 August 2024, visited page: dashboard/", it will be written as:

{"IP_address": None, "Day_of_month": 17, "Month_of_year": Aug, "Year": 2024, "Visited_page": "dashboard/"}
The extracted information is then formatted into a string like the one used for uploading data to the Pinecone database: ("Day of month: {day}, Month of year: {month}, Year: {year}, IP address: {ip}, Visited page: {page}")
This string is used to query the Pinecone database, asking it to retrieve the top 20 relevant data entries.

Validating retrieved data

The data retrieved from the database is subjected to a simple filtering process to enhance accuracy. The information in the dictionary is checked against the data retrieved from the Pinecone database. If a match is found, the data is stored in the "filtered_logs" list; otherwise, it is not added to the list.

Sending validated data to the user:

The validated data is formatted and sent to the GPT-2 model to be converted into natural language. The natural language response generated by GPT-2 is then sent to the user.

Performance metrics

The program calculates three performance metrics: Precision, Recall, and F1 Score.

Precision: The ratio of validated data to the total data retrieved from Pinecone.

(matched_logs / total_logs)

Recall: The ratio of validated data to the total relevant data in the Pinecone database.

(matched_logs / relevant_logs)

(By default, the program retrieves 20 entries from Pinecone, but there may be more than 20 relevant entries in the database, leading to differences between Precision and Recall.)

F1 Score: The harmonic mean of Precision and Recall, providing a single metric that balances both.

(2 * (precision * recall) / (precision + recall))

Challenges

Log File Parsing

The format of Apache log files is not always consistent. Some log entries contain incomplete information, making it difficult to parse them correctly. To address this, I chose to skip entries that could not be parsed correctly and only use complete data.

Entity Extraction

Initially, I tried using the T5 model to extract important information, but neither the "t5-small" nor "t5-base" models provided satisfactory results. The models failed to correctly extract necessary information like IP addresses and dates from user queries. I then switched to using spaCy and NLP. While spaCy produced better results than T5, it still wasn’t perfect. Specific issues arose with dates, IP addresses, and visited pages. For example, spaCy sometimes identified the IP address as a random string of characters like "e:". To handle this, I developed regex and manual validation mechanisms, but these are not always sufficient, leading to misinterpreted queries.

Performance calculations

Creating accurate and relevant data sets for calculating performance metrics like Precision, Recall, and F1 Score was a challenging process. I had to add extra filtering and validation steps to measure how closely the data retrieved from Pinecone matched what the user was searching for.

Natural language generation

I experimented with various methods to generate natural language responses to send to users, encountering frequent errors. T5 failed to generate the desired text, and I faced similar issues with GPT. Through research, I better understood how to use these models and successfully implemented a working solution in the final program.

Failed methods

Entity Extraction:

I initially used Groq API to extract data by sending the user’s query to an AI. However, I later learned that this approach would not be acceptable for the project requirements, so I switched to T5.

T5 attempts:

I used "t5_small" and "t5_base" models to analyze the natural language query and extract the necessary information, but they failed to produce the desired results in almost 90% of the cases.

T5 fine-tuning:

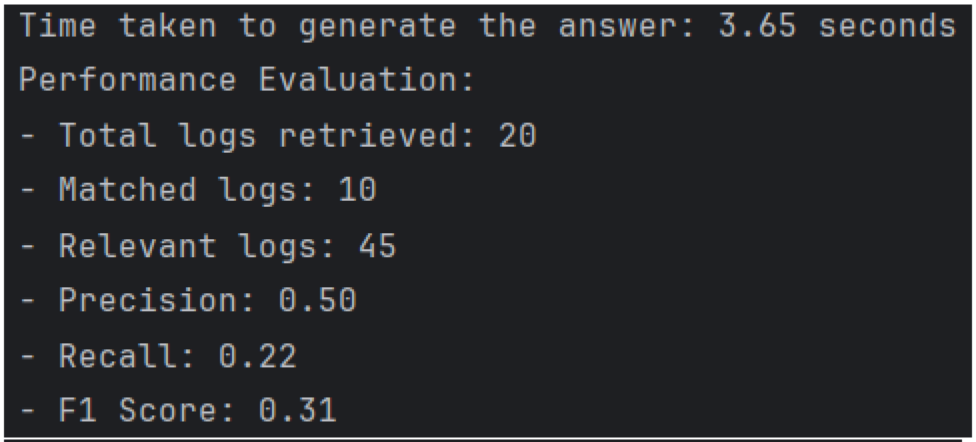
To resolve issues with T5, I created a dataset with 500 rows of queries and extracted information. I tried fine-tuning the T5 model using this dataset. Due to time constraints and the large size of the dataset relative to my computer’s performance, I adjusted TrainingArguments (learning rate, batch size, epochs) for faster results. I believe the main reason for not achieving the desired outcome from fine-tuning was that I rushed the process. With sufficient time and a more extensive dataset, I think fine-tuning would have produced better results.

After six hours of rushed fine-tuning, I still didn’t get the desired results from the T5 model and switched to spaCy NLP.

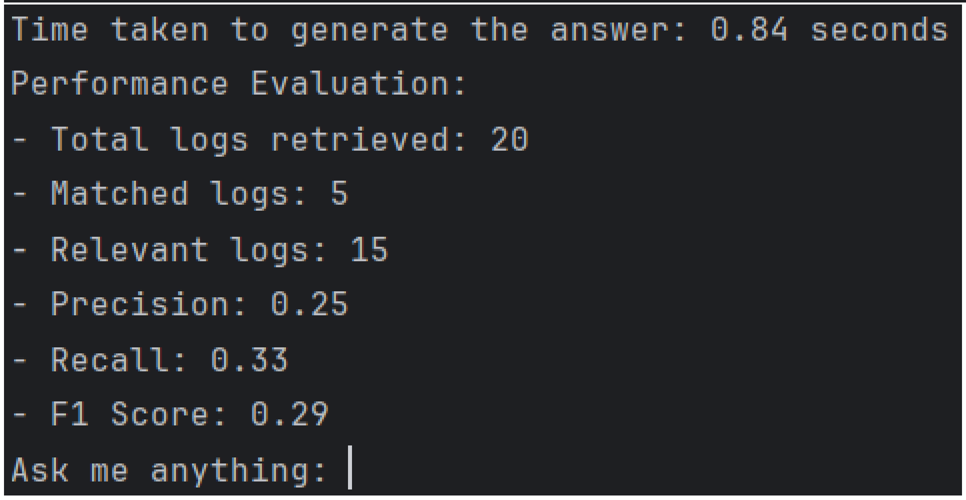
Natural language generation:

Initially, I used the Groq API to send the retrieved data to an AI for generating a response, but had to abandon this approach for the same reasons. I then switched to GPT-2. When I didn’t get the desired results with GPT-2, I tried sending all the data as coherent sentences in a single string to GPT-2. This approach finally gave me the results I wanted.

Performance



```
Time taken to generate the answer: 3.65 seconds
Performance Evaluation:
- Total logs retrieved: 20
- Matched logs: 10
- Relevant logs: 45
- Precision: 0.50
- Recall: 0.22
- F1 Score: 0.31
```



```
Time taken to generate the answer: 0.84 seconds
Performance Evaluation:
- Total logs retrieved: 20
- Matched logs: 5
- Relevant logs: 15
- Precision: 0.25
- Recall: 0.33
- F1 Score: 0.29
Ask me anything:
```

The program uses a straightforward method to measure performance. It measures the time elapsed from when the user asks a question until the question is answered and uses this time to gauge performance.

Future Works

Fine-tuning:

Since I'm not getting the desired results from the spaCy NLP method I'm currently using, I plan to retry the T5 fine-tuning process, which I previously had to rush due to time constraints. This time, I'll use a larger dataset and allow the tuning process to take hours or even days. With sufficient time and a comprehensive dataset, I believe fine-tuning will yield much better results, potentially allowing me to switch entirely to a T5-based question interpretation method and eliminate the regex and manual validation methods I currently use as fail-safes.

Improving program accuracy:

The program currently retrieves 20 entries by default from the Pinecone database. Instead of doing this, I could analyze the user's query to retrieve all relevant entries from the database, thereby improving the program's expected accuracy. Additionally, vector normalization could be applied to the 384-dimensional vectors to reduce their dimensionality, increasing the speed and accuracy of retrieving meaningful data and mitigating the "**Curse of Dimensionality**" to some extent.

Performance:

To improve the speed and efficiency of the program's NLP and GPT methods, I could make some optimizations. For example, using a fine-tuned LLM model specifically for entity extraction would likely produce faster results than a general-purpose LLM model. The same applies to the GPT model used for generating responses, a model specifically designed or fine-tuned for the program's needs would increase response speed.

Libraries

time: For time delays and timing functions.
re: For text processing and entity extraction using regex.
datetime: For handling and converting date formats.
spaCy: For natural language processing and entity extraction.
Pinecone: For managing the vector database.
Sentence_transformers: For text vectorization.
Transformers: For using the GPT-2 model.
(Failed methods) Datasets
(Failed methods) csv
(Failed methods) T5

References

Apache: https://httpd.apache.org/docs/2.4/
spaCy: https://spacy.io/api/doc
Pinecone: https://docs.pinecone.io/reference/api/introduction
I resolved many integration issues by referring to various StackOverflow topics (The errors were generally due to Python and/or package version incompatibilities).
I also used ChatGPT to create more efficient regex patterns.