Project Title:

Elastic Open search capability expansion

Santu Banerjee (netID: santub2@illinois.edu)

Project Keywords: #document-search #BM-25-ranking #information-retrieval #NLP #LLM #AWS #document-result-Cache #chatbot #gemini

Project Description:

Problem:

At Enterprise level, distributed system services logged each interactions and transactions. These logs are very important to monitor service health, incident management and debugging purpose. Elastic Open Search get used to show that logging info but there is a time lag between the actual incident time and the time when that incident showed up in Open search. And for each time it does a complete scan in logs to produce the result. Performance of Semantic search is very bad and does not return results in most cases. It performs better in exact keyword search scenarios .

High level Approach:

Run indexing on logs as part of preprocessing. Rank documents using initial query and save the results in cache. If similar query get used then use NLP and LLM to determine similarity and then get the result from memory instead of running BM25 based search engine. If new query is not similar with previous queries then return result from BM25 based search engine and update cache to be used for subsequent queries.

Evaluation:

Effectiveness will be measured by response time and results of queries.

Implementation:

This project implements an interactive chatbot that searches and analyzes AWS service logs using BM25 search algorithm and natural language processing. The chatbot provides an intuitive interface for querying log data and returns relevant log entries based on user queries.

Features:

- Interactive chat interface using Streamlit
- BM25-based log search functionality
- Caching mechanism for faster repeated queries
- Natural language query processing
- Real-time log analysis and parsing
- Support for various AWS service log formats

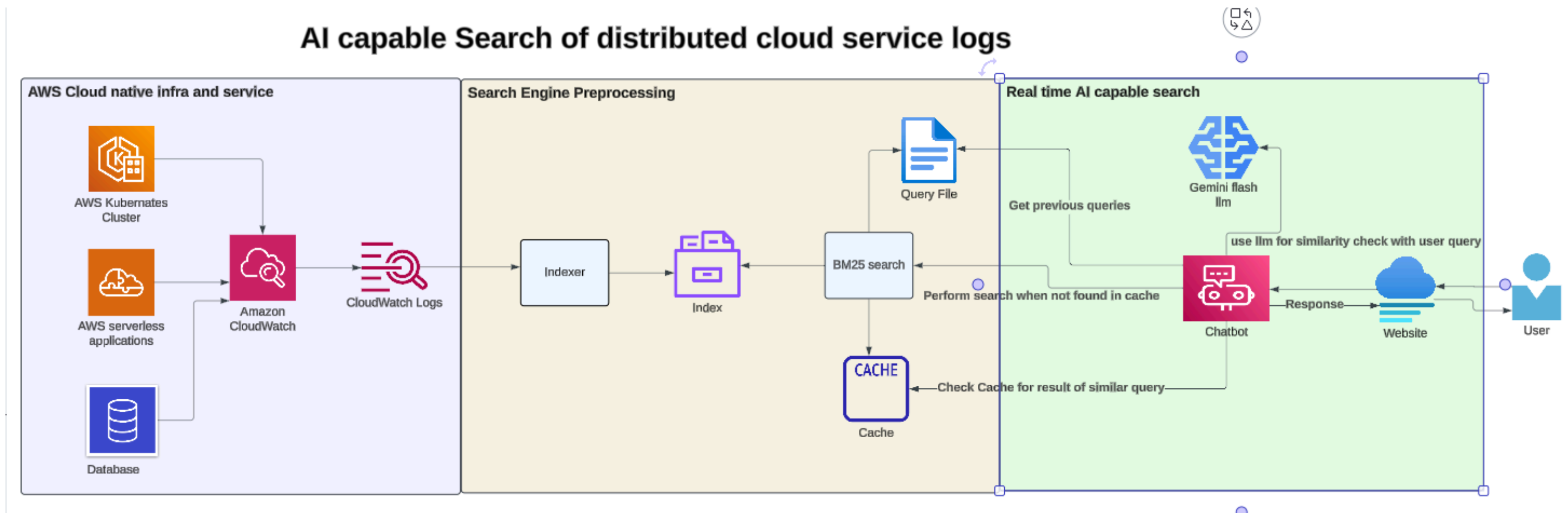
Prerequisites:

- Python 3.10 or higher
- Streamlit
- Google Generative AI API access
- Required Python packages (see requirements.txt)

Project Structure:

```
project/
├──── code/
|     ├──── chatbot.py
|     └──── generate_data.py
├──── data/
|     └──── logs/
|           └──── aws-service-logs/
├──── processed_corpus/
├──── indexes/
├──── requirements.txt
└──── README.md
```

Project high level architecture:



AI capable Search of distributed cloud service logs
AWS Cloud native infra and service
AWS Kubernetes Cluster
AWS serverless applications
Database
Amazon CloudWatch
CloudWatch Logs
Search Engine Preprocessing
Indexer
Index
BM25 search
Query File
CACHE
Cache
Real time AI capable search
Gemini flash llm
Get previous queries
use llm for similarity check with user query
Perform search when not found in cache
Check Cache for result of similar query
Chatbot
Response
Website
User

Log Files:

Cloud native distributed service generate logs for each API calls, transactions etc. Due to privacy concerns, synthetic cloud watch log file has been created for this project.

Search Engine Preprocessing:

As part of preprocessing, index files get created by indexer and loaded in Index and Preprocessed corpus folders.

Real time AI capable search:

Langchain and Gemini is used to create chatbot. When user perform a search, then chatbot does a look up on previous queries and use natural language processing via LLM for similarity search. If similarity is found then get the result from cache. Otherwise perform BM25 search to perform ranking and produce result. It also update cache with the new query, so that similarity can be used in subsequent searches.

User Interface:

- Interactive chat interface
- Real-time response streaming
- Clear history option
- Error handling and user feedback

Error Handling:

The application includes comprehensive error handling for:

- API authentication issues
- Invalid queries
- Connection problems

- Data processing errors

Dependencies:

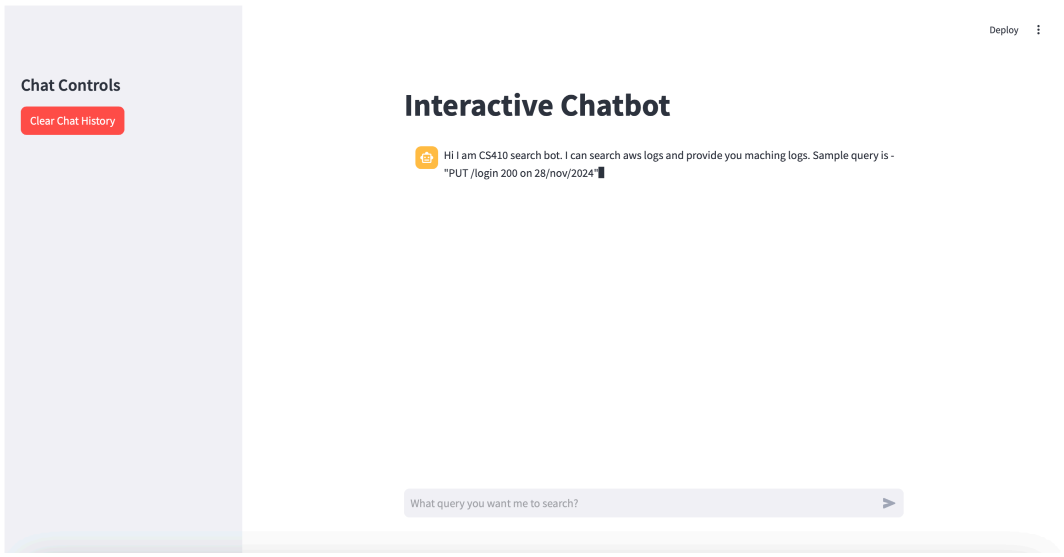
- streamlit
- langchain-google-genai
- google-generativeai
- python-dotenv
- google-auth
- langchain-google-genai
- pyserini==0.38.0
- openjdk=21
- faiss-cpu
- maven
- lightbgm
- PyTorch

Environment creation:

- Create conda environment using environment.yml or requirement.txt file.
- Activate conda environment.
- Generate an API key using https://ai.google.dev/gemini-api/docs/api-key and copy the api key.
- Export API key for the environment - export GOOGLE_API_KEY = ‘your api key here’.

Usage:

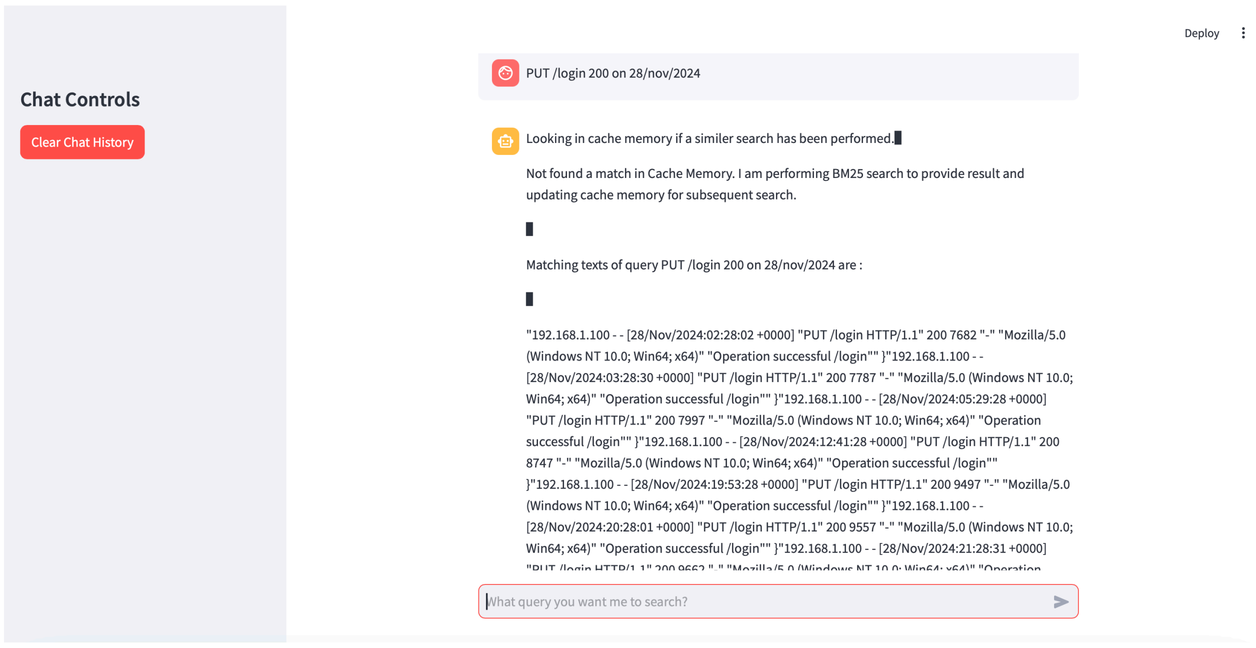
- Start the application - streamlit run code/chatbot.py
- Access the web interface at http://localhost:8501/



Interactive Chatbot

Hi I am CS410 search bot. I can search aws logs and provide you maching logs. Sample query is - "PUT /login 200 on 28/nov/2024"

- Enter queries, for example - “PUT /login 200 on 28/nov/2024”
- Chatbot explains each step it is performing and provide relevant documents following BM25 page ranking retrieval method.
-



PUT /login 200 on 28/nov/2024

Looking in cache memory if a similer search has been performed.

Not found a match in Cache Memory. I am performing BM25 search to provide result and updating cache memory for subsequent search.

Matching texts of query PUT /login 200 on 28/nov/2024 are :

```
"192.168.1.100 - - [28/Nov/2024:02:28:02 +0000] "PUT /login HTTP/1.1" 200 7682 "-" "Mozilla/5.0
(Windows NT 10.0; Win64; x64)" "Operation successful /login"" }"192.168.1.100 - -
[28/Nov/2024:03:28:30 +0000] "PUT /login HTTP/1.1" 200 7787 "-" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64)" "Operation successful /login"" }"192.168.1.100 - - [28/Nov/2024:05:29:28 +0000]
"PUT /login HTTP/1.1" 200 7997 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)" "Operation
successful /login"" }"192.168.1.100 - - [28/Nov/2024:12:41:28 +0000] "PUT /login HTTP/1.1" 200
8747 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)" "Operation successful /login""
}"192.168.1.100 - - [28/Nov/2024:19:53:28 +0000] "PUT /login HTTP/1.1" 200 9497 "-" "Mozilla/5.0
(Windows NT 10.0; Win64; x64)" "Operation successful /login"" }"192.168.1.100 - -
[28/Nov/2024:20:28:01 +0000] "PUT /login HTTP/1.1" 200 9557 "-" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64)" "Operation successful /login"" }"192.168.1.100 - - [28/Nov/2024:21:28:31 +0000]
"PUT /login HTTP/1.1" 200 9662 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)" "Operation
```

Performance Optimization:

- Implemented caching for frequent queries
- Optimized BM25 search parameters
- Efficient log parsing and processing

Future Enhancements:

- Format chatbot response in more user friendly manner.
- Implemented more natural language based queries by summarizing query and finding important words based on the topic.
- Enhance chatbot to provide analytics capability.

Known Issues:

- Gemini-1.5-flash model has quota limit. Due to that reason it show ‘ 429 resource has been exhausted’ error when quota limit exceeds.