Retrieval-Augmented Generation (RAG) System for Personalized Skill Recommendations and Career Advice

Table of Contents

1. Introduction

The Retrieval-Augmented Generation (RAG) system is designed to provide personalized skill recommendations and career advice to users based on their queries about specific job roles. The system leverages both retrieval-based techniques and generative models to craft tailored responses.

2. System Architecture

The RAG system comprises three main components:

Data Preparation:

Objective: Utilize the provided jobs data to create document embeddings using Sentence Transformers.

The first step involves loading and preprocessing job data. The data typically includes fields such as job title, job description, job requirements, and career level. The job description and job title are combined into a single text field to create a comprehensive representation of each job posting. This combined text is then used for embedding creation.

Embedding Creation

Using Sentence Transformers, the combined job details are encoded into dense vectors. Sentence Transformers are powerful models that create embeddings capturing the semantic meaning of the text, making it suitable for similarity searches. The embeddings are created in chunks to handle large datasets efficiently.

Retrieval Component

The retrieval component leverages FAISS (Facebook AI Similarity Search) for efficient vector similarity search. FAISS is a library for fast nearest neighbor search in large datasets, providing the ability to quickly find similar vectors. The job embeddings are indexed using FAISS, and user queries are transformed into vectors to perform similarity searches, retrieving the most relevant job postings.

Generation Component: Generate personalized responses using a generative model.

Generative Component

The generative component utilizes a pre-trained language model (such as LlamaForCausalLM) to craft personalized responses. Once the relevant job postings are retrieved, the generative model uses these postings to generate coherent and contextually relevant career advice and skill recommendations. The model ensures that the responses are tailored to the user's query and provide valuable insights.

3. Implementation Details

1. Data Preparation

- Data Loading: Job data is loaded from a CSV file, which includes fields like job title, job description, job requirements, and career level.

- Preprocessing: The job description and requirements are combined into a single text field to create a comprehensive representation of each job posting. The 'requirements' column is then dropped to streamline the dataset.

2. Embedding Creation

- Model Initialization**:** A Sentence Transformer model is initialized to create embeddings for the combined job details.
- Chunk Processing**:** The data is processed in chunks to handle large datasets efficiently. Each chunk of combined text is encoded into dense vectors.
- Concatenation**:** The embeddings from all chunks are concatenated into a single matrix, representing the entire dataset in a dense vector format.

3. Retrieval Component

- FAISS Indexing: A FAISS index is initialized with the appropriate dimensions to store the job embeddings. The embeddings are added to the FAISS index for efficient similarity search.
- Query Processing: User queries are encoded into vectors using the same Sentence Transformer model. The query vector is then used to search the FAISS index, retrieving the most relevant job postings based on vector similarity.

4. Generation Component:

- Model Loading: A pre-trained generative model (e.g., LlamaForCausalLM) is loaded, and the model is moved to GPU if available to ensure efficient processing.

- Optimize Model Loading and Usage by using torch's half-precision (FP16) to reduce memory usage.

- Response Generation: The generative model uses the retrieved job postings to generate a personalized response. The response includes tailored skill recommendations and career advice based on the user's query and the retrieved job details.

Evaluation Metrics

Relevance Metrics:

- Precision at k (P@k): If applicable, based on advice retrieval.
- Mean Reciprocal Rank (MRR): If applicable, based on advice retrieval.
- User Satisfaction Score: A human evaluation metric where users rate the relevance of the advice.

Coherence Metrics:

- Human Evaluation: Users rate the coherence and usefulness of the advice.
- BLEU Score: Evaluates the quality of the text generated by the model compared to reference responses.
- ROUGE Score: Measures the overlap of n-grams between the generated response and reference responses.

In order to proof evaluation concept we use only Cosine Similarity Matrix to simplify the evaluation with no need to a ground truth, for further evaluation comparisons can be executed in order to get customer satisfaction.

PS: as I faced a challenge while generating the model which exploit all Ram resources in colab notebook I tried to limit the maximum of each job response may that affect on last response accuracy; we can enhance it by saving the model and use a new session by the saved model to get better results without making limitations to the search so we will get more accurate results

Conclusion

The RAG system combines the strengths of retrieval-based techniques and generative models to provide personalized skill recommendations and career advice. By leveraging Sentence Transformers for embedding creation, FAISS for efficient similarity search, and a generative model for crafting responses, the system delivers relevant and coherent recommendations tailored to user queries. This documentation outlines the architecture, implementation details, and usage instructions, providing a comprehensive guide for setting up and using the RAG system.