Building a Retrieval-Augmented Generation (RAG) System

Brandon Des Jardins, Bonvie Fosam, Jake Horton, and George Pan

Introduction

In today's competitive job market, effectively matching candidates with suitable employment opportunities is crucial for both job seekers and employers. Traditional job search methods often rely on basic keyword matching, which can overlook the nuanced qualifications and preferences of individuals. This inefficiency not only burdens job seekers with extensive search efforts but also impedes employers from identifying the most suitable talent promptly.

Advancements in **Natural Language Processing (NLP)** and **Machine Learning (ML)** have ushered in a new era of intelligent recruitment solutions, enabling more sophisticated and accurate job matching mechanisms. By harnessing the power of semantic understanding and contextual analysis, modern job retrieval systems can transcend basic keyword correlations, delving deeper into the intricate details of resumes and job descriptions to identify optimal matches.

This project endeavors to develop an **automated job retrieval system** that leverages cutting-edge NLP techniques to match candidates' resumes with relevant job postings effectively. The system employs **embedding models** to convert textual data from resumes and job descriptions into high-dimensional vector representations, capturing the semantic essence of each document. Utilizing these embeddings, the system calculates **similarity scores** to rank and retrieve the most pertinent job opportunities for each candidate.

A critical aspect of this project is the rigorous evaluation of the system's performance across various **query types**, such as skills, location, and experience. By categorizing and analyzing how the system handles different facets of job matching, we aim to ensure a comprehensive assessment of its efficacy. The evaluation framework extends beyond the **ARES (Accuracy, Relevance, Engagement, and Satisfaction)** metrics to include traditional NLP metrics like **BLEU**, **ROUGE**, and **METEOR**, as well as measures of **system speed**. These metrics provide a multifaceted perspective on the system's ability to deliver faithful and relevant job recommendations efficiently.

Furthermore, this project emphasizes the importance of **visualizing evaluation results** to facilitate intuitive understanding and insightful analysis. Through the creation of diverse graphical representations—such as bar charts, line graphs, and scatter plots—we aim to elucidate the system's strengths and areas for improvement across different query dimensions. These visualizations not only enhance the interpretability of performance metrics but also inform future refinements and optimizations of the job retrieval mechanism.

In summary, this project addresses the pressing need for more intelligent and efficient job matching solutions in the digital age. By integrating advanced NLP techniques with robust evaluation methodologies, we strive to bridge the gap between job seekers and employers, promoting a more dynamic and responsive employment landscape. The outcomes of this endeavor hold significant implications for enhancing the user experience in job searches, optimizing recruitment processes, and ultimately contributing to the betterment of workforce development.

Methodology

In implementing the RAG based language model, a series of steps were executed to prepare and embed the data, and train and evaluate the model. First, traditional data preparation techniques were used to include setting the index, addressing missing values, making appropriate conversions to numeric values, like salary amount and longer string parameters, and normalizing the data. After cleaning and splitting the data, the data was chucked, tokenized, and embedded vector. Chunking the data is critical in avoiding loss of specificity within the data. This project uses sliding window chunking with a chunk size of 500 with an overlap of 50 considering the tradeoff between an increase in semantic relevance and large storage and processing requirements. Both the training and testing job data was embedded using batch embeddings to allow for more efficient processing and minimize time costs. Due to the limited resources in Google Colab, we utilized the **distilled version of BERT base model** from HuggingFace for embedding. From here, RAG could be implemented.

Retrieval-Augmented Generation can be divided into two components: retrieval and generation. The retrieval process allows for the most relevant job postings to be identified based on a query. The process is represented in this project by retrieve_jobs() which takes a resume embedding and number of jobs to be returned as input. One primary objective is to evaluate the similarity between the resume and job embeddings. The distance between the vectors are calculated to derive a similarity score, identifying the most relevant jobs. In this project, the top 5 most relevant job titles were identified along with its location, salary, and description. The second component of RAG, generation, serves to generate a response based on the query and retrieved documents. In this case, that is the resume and the retrieved jobs. This project uses GPT 3.5 as its model and feeds it the resume text and the retrieved jobs. Here, the model generates 5 jobs most relevant to the inputted resume. This output is used in the evaluation stage of this project.

Evaluation

Augmented Retrieval Evaluation for Semantics (ARES) aims to assess the ability of these models to extract information from text data and generate cohesive output, specifically in reference to semantic relevance and factual accuracy. In this project, ARES was used in assessing context relevance, answer faithfulness, and answer relevance. First, context relevance measures how well the retrieved job aligns with the context of the query. Here, it was determined by capturing the overlap between predetermined common skills that relate to the resumes and the resume or job text. The predetermined common skills, represented as strings, include technical verbiage that are expected to be in resume and job test like ‘python’, ‘machine learning’, ‘project management’, and more. It is important to note that the resulting context relevance is dependent upon this preset common skills library. A resume and job match in regard to skills if they contain the same overlapped skills. The match is quantified using a ratio between the number of shared skills over the job’s number of skills. A similar procedure to quantify relevance was repeated resume and job text location and experience. The overall context relevance score was calculated as a sum of the skills, experience, and location matches each multiplied by a respective weight. Based on importance, skill had a weight of 0.5, experience had a weight of 0.3, and location had a weight of 0.2. In this project the average context relevance was 0.52.

Next, answer faithfulness is a measure of how factual the retrieved jobs are. In this project the job title, description, location, and minimum salary of the retrieved jobs and the jobs in the training data set are compared. If they are the same, an increment starting from zero is increased by one. The faithfulness score is determined by a ratio between the final increment value and the total number of retrieved jobs. The average answer faithfulness score in this project is 1.00. Answer relevance measures how well the retrieved job addresses the queried resume. In this project, answer relevance is calculated by applying equal weights of 0.5 to the similarity score and context relevance scores. Using these two metrics allows us to appropriately compare the inputted resume with the retrieved job and assess the relevance. The average answer relevance was 0.74.

Figure 1 shows how the system performs across the different query types (skills, location, and experience). We see that answer faithfulness consistently performs the best with a value of 1.00 across all query types. This is probably due to the fact that our job database stayed the same throughout the process, and we are not generating job descriptions from scratch. Instead, every job our system retrieves comes directly from the job database. Context relevance consistently performs the worst as it varies slightly around 0.50. This can be interpreted as 50% of the time the retrieved job is relevant to the context of the resume. This metric is less than optimal however, its strength is heavily reliant on the common skills library established prior to evaluation. Here, the common skills library had just 10 skills which greatly limits the context relevance from reaching its full potential. In a real world application, this common skills library would include a much larger set of skills, which would likely contribute to a higher context relevance score. Answer relevance is consistently around 0.74, meaning that around 74% of the time, the retrieved job is relevant to the queried resume. This output is heavily reliant on the output of context relevance. The poorer context relevance subsequently decreases the answer relevance.

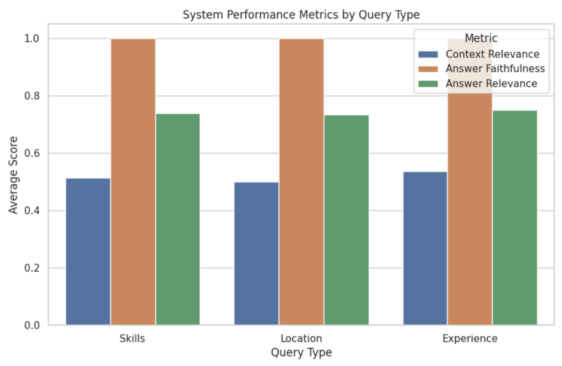

Figure 1: System Performance Metrics by Query Type

The evaluation using ARES was followed by an evaluation using BLEU, RoGUE, and METEOR. Bilingual Evaluation Understudy (BLEU) evaluates how the LLM generated output matches a set of ground truth texts. In this case, the LLM generated jobs and the retrieved jobs. This is important to capture because it communicates how natural-sounding, relevant, and semantically accurate the generated output is. Broadly, BLEU compares the n-grams between the generated output and reference texts. This project utilizes the existing library, nltk.translate.bleu_score, to execute the BLEU evaluation. The average bleu score across the different query times is 0.018.

Recall-Oriented Guided Understudy for Gisting Evaluation (RoGUE) measures the faithfulness and relevance of the generated output compared to the retrieved data. There are three variants of RoGUE included in this project which are RoGUE 1, RoGUE 2, and RoGUE Long (L). For all three variants the scores were calculated using the existing library, rouge_score. RoGUE 1 computes this comparison using a unigram overlap in which each word is individually considered.

So, the contents of each job are compared on a word-by-word basis. The resulting mean score for RoGUE 1 was 0.30 across the query types. RoGUE 2 computes the same comparison using a bigram overlap which captures more context between the generated and retrieved jobs. The resulting mean score for RoGUE 2 was 0.04. Lastly, RoGUE L is tailored for longer texts where it is important that the output is coherent and relevant. This is appropriate in this project because a considerable amount of information about each job is expected in the output. The mean score for RoGUE L was 0.13 across the query types.

Lastly, a Metric for Evaluation of Translation with Explicit Ordering (METEOR) was calculated. This is a sophisticated step up from BLEU as it captures the meaning overlap between the generated text and the retrieved text. The existing library, nltk.translate.meteor_score, was used to find an average METEOR score of 0.21 across all query types. Figure 2 depicts the breakdown of the different evaluations on each query type.

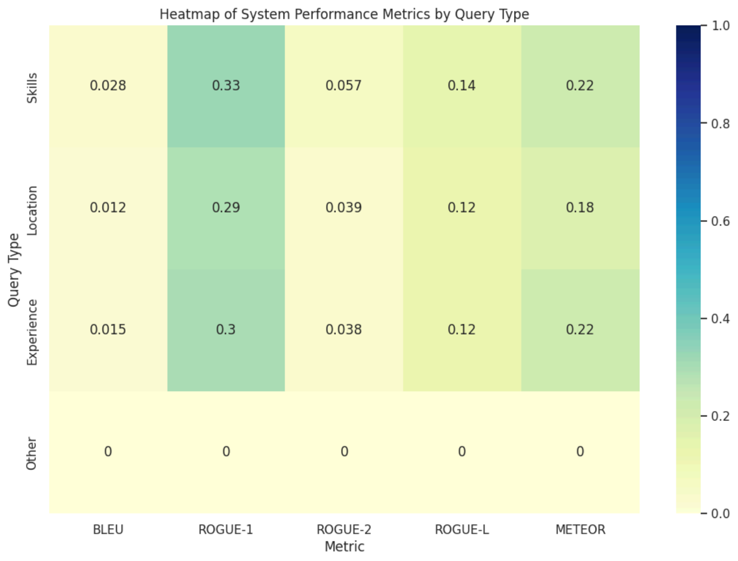

Figure 2: System Performance Metrics by Query Type

These scores are less than optimal, however, this can be justified because it would be unreasonable to expect the contents of the retrieved jobs to be exact matches to the ground truth jobs. Due to the long-text nature of this output, there is more potential for variances which leads to decreases in these metrics. This consideration must be taken into account when interpreting these results.

Model Performance Analysis

As seen in table 1, on average, skill queries were marginally the fastest query type and location queries were the slowest. These times are quite reasonable when it comes to making an assessment of the efficiency of data retrieval. A strong evaluation can be made by establishing benchmark times to compare these times to.

	Average Time	Max Time	Min Time
Skill Queries	0.0076 seconds	0.121 seconds	0.0062 seconds

Location Queries	**0.0091 seconds**	**0.0139 seconds**	**0.0074 seconds**
Experience Queries	**0.0078 seconds**	**0.0109 seconds**	**0.0064 seconds**

Table 1: Query Speed by Query Type

Recommendation and Next Steps

In completing this project, a few growth opportunities that can be addressed in subsequent iterations were identified. First, this model can be adjusted to improve the performance of the model and subsequently the accuracy and relevance of retrieved data and generated output. For example, the chunking size can be better optimized to achieve a reasonable balance between access to granular pieces of information and obtaining an overall contextual understanding of the text. Identifying the right chunk size will help maximize evaluation metrics like relevancy, recall, precision, and computation time. Another adjustment that can be made is to change the LLM used in forming the vector embeddings. For example, given more computational resources, switching the model from DistilBERT to Base BERT could allow for more sophisticated modeling and lead to improved model performance.

To attempt to improve the evaluation metrics, specifically context relevance and answer relevance the common skills library can be extended to include more words. This would increase opportunities for matches between the resumes and jobs. Additionally, it would be helpful to re-evaluate the weights used in calculating the context relevance and answer relevance scores to ensure each measurement contributes an appropriate amount to the final score. Over the next iterations, the goal would be to have a model sophisticated enough to handle a more complex data set, and robust enough for wide scale use.