Contextual Candidate Matching: An Intelligent Approach to Resume Retrieval for a given Job Description

Aanandhi Sonduri Panthangi asonduri@usc.edu
Akhilaa Sonduri Panthangi sondurip@usc.edu
Mohan Sai Ganesh Kanna mkanna@usc.edu
Mona Teja Kurakula mkurakul@usc.edu
Emma Leihe leihe@usc.edu

Abstract

Traditional job matching algorithms often use basic string matching to compare keywords in resumes with those in job descriptions. Although this approach can be useful, it tends to miss the deeper context and meaning of the skills and experiences of the applicant. This project suggests using word and sentence embeddings, which move beyond simple keyword comparisons, enabling a richer semantic analysis. By employing these advanced techniques, we can better understand the nuances and contextual relevance of applicants' abilities, enhancing the job matching process.

1 Introduction

1.1 Motivation

To effectively match resumes with job descriptions, it is essential to evaluate how well skills align. Standard Language Models typically depend on keyword similarity, risking the exclusion of qualified candidates whose language may differ from the requirements mentioned in the job description. To overcome this, we propose training a specialized neural network that generates word embeddings of skills and experiences that enable similarity score calculations. These scores will accurately assess the fit between candidates and job openings.

For example, if a job description requests "experience in developing scalable systems using modern cloud technologies", a traditional algorithm might miss a resume that states "designed and maintained backend systems using AWS and GCP". In contrast, a system powered by a neural network would recognize that "cloud-based systems" and the mention of "AWS and GCP" indicate relevant experience with modern cloud technologies, leading to a more accurate assessment of candidate qualifications. For a given input job description, the proposed model aims to select the most appropriate (top-k) resumes as output from the available pool of resumes.

1.2 Background

In the evolving job market, job matching processes often rely on traditional algorithms like Best Matching (BM-25), which primarily focus on lexical similarity between job descriptions and resumes. However, these methods often fall short in capturing the deeper semantic meaning behind the words. Dense Passage Retrieval (DPR), a state-of-the-art dual-encoder model designed for information retrieval tasks, addresses this challenge by utilizing bi-directional attention to capture complex contextual relationships between words in both the query (job description) and document (resume).

Contrastive learning further enables the model to differentiate between relevant and irrelevant passages by learning from positive and negative examples. DPR generates dense vector embeddings for both queries (job descriptions) and passages (resumes), improving retrieval effectiveness. Studies state that fine-tuning DPR on domain-specific datasets, holds the potential to significantly enhance its ability to understand nuanced semantic similarities beyond simple keyword matching, thus outperforming traditional methods for that particular domain.

In the upcoming sections, we will provide an in-depth discussion on Dense Passage Retrieval (DPR), its advancements, and the dataset used for fine-tuning. Additionally, we will explore hybrid architectures that integrate both semantic and lexical similarity, highlighting how this hybrid model performs in comparison to traditional approaches like Best Matching 25 (BM-25). Through this exploration, we aim to demonstrate how the combination of these techniques can enhance the job matching process, offering a more comprehensive and accurate solution than existing methods.

1.3 Related Work

The Dense Passage Retrieval (DPR) framework, introduced by Karpukhin et al. (2020) at Facebook, employs a two-stage process that utilizes dense vector embeddings to efficiently retrieve relevant passages. Since its introduction, DPR has undergone numerous improvements to address specific challenges in the retrieval landscape. For instance, Guu et al. (2020) expanded the original model by integrating a reranking step, significantly improving performance in open-domain question answering.

The versatility of DPR has led to its adoption in various applications, including question answering, document retrieval, and recommendation systems. Lewis et al. (2020) demonstrated its efficacy in enhancing the efficiency and accuracy of large-scale document retrieval, effectively managing diverse and complex queries. A focal point of research has been DPR's capability to scale to large datasets while maintaining high retrieval accuracy. Xiong et al. (2021) explored optimization strategies to boost efficiency, showcasing its potential for real-time applications in industrial settings.

Comparative evaluations indicate that DPR consistently outperforms traditional retrieval methods in accuracy and relevance. Recent studies by Chen et al. (2022), further validate the superiority of DPR across various benchmark datasets and tasks. Despite its effectiveness, DPR may encounter challenges with complex queries, such as extracting specific details from lengthy texts or addressing multi-step reasoning tasks.

2 Dataset Description

The proposed dataset is a resume corpus[1] (Jiechieu and Tsopze, 2021) consisting of approximately 30,000 resumes in .txt format, each annotated with corresponding occupation labels (class labels) in .lab format. These labels are multi-class, meaning each resume can be associated with multiple role titles based on the candidate's work experience.

2.1 Resume Corpus (resumes_corpus.zip)

This file comprises individual resumes in text format (.txt), with each resume's associated occupation labels provided in a corresponding .lab file. This facilitates easy access to the full-text resumes and their multi-label occupational annotations.

[1] https://github.com/florex/resume_corpus

2.2 Normalized Classes (normalized_classes.txt)

This is a text file (.txt) that standardizes diverse occupation titles by mapping them to a set of 10 primary categories (class labels), enabling consistent categorization and interpretation. The 10 normalized occupation classes are:

1. Security_Analyst
2. Systems_Administrator
3. Project_Manager
4. Database_Administrator
5. Software_Developer
6. Front_End_Developer
7. Web_Developer
8. Java_Developer
9. Network_Administrator
10. Python_Developer

Role titles can be company specific and normalized classes help us to funnel down all these redundant titles into fixed number of defined classes. For example, a React Developer, will be mapped to the Front_End_Developer role in normalized occupation classes. This structured dataset and labels from normalized_classes.txt serves as a valuable resource for training, fine-tuning and evaluation.

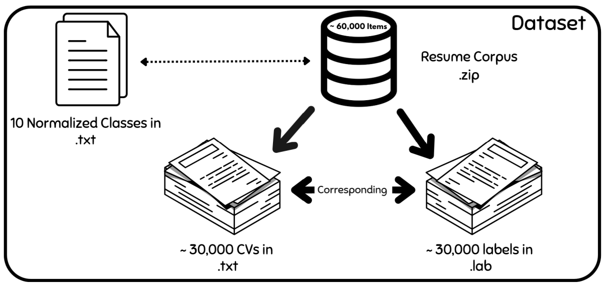

Figure 1: Dataset Breakdown.

3 Methodology

3.1 Data Preparation

The first step in our workflow is to prepare the train, validation, and test datasets in the required format. Since the core query in our problem is the job description and the current dataset lacks this information, we leverage Groq to generate them.

To ensure balance across the 10 class labels, we select 100 resumes per class, leading to a total of 1,000 resumes. Using a carefully designed, high-quality prompt, each resume is fed into Groq to generate a corresponding job description. Each record in the resulting dataset consists of:

- resume (Resume text)
- categories (Class labels)
- job_description (Generated text)

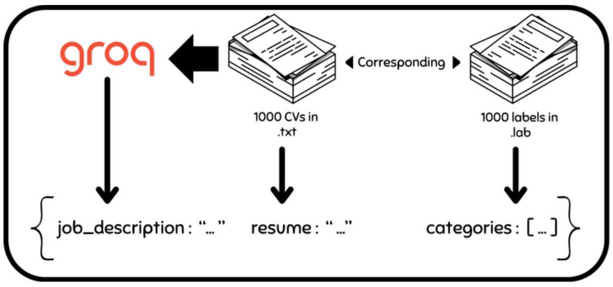

Figure 2: Train Data for fine-tuning.

This dataset serves as the training data for fine-tuning the DPR base model (DPR-BASE), which was originally trained on open-domain question answering tasks (NQ - Natural Questions). By fine-tuning it to the specialized task of job matching, we adapt the model to better understand and retrieve relevant resumes based on job descriptions.

To fine-tune the model and evaluate its performance, we require validation data to adjust the hyper-parameters and test data to measure the model's efficiency. The primary goal is to provide a job description as input and retrieve the top-k most relevant resumes from a given resume pool. For this task, we consider a pool size of 20 resumes per query (job description), where 5 resumes are positive samples (relevant to the job description), and 15 resumes are negative samples (irrelevant resumes). To ensure a fair evaluation, we construct a total of 100 queries, with 10 queries for each class, for both validation and test datasets ensuring no overlap. The validation and test datasets are created using the following process.

For each record or query in validation and test dataset:

1. Select a specific class label.
2. Iterate through the resume corpus and identify 5 resumes whose corresponding `.lab` files contain the selected class label. These resumes serve as the positive samples.
3. Similarly, select 15 resumes whose `.lab` files do not contain the target class label. These resumes act as negative samples.
4. Aggregate the text from the positive samples and pass it to Groq using a high-quality prompt, similar to the process used for generating the training data. Groq generates a job description relevant to the selected class label using the text from 5 positive resume samples.

Each record in the validation and test datasets consists of:

- label (The class label serving as the ground truth).
- pos (List containing text from 5 positive sample resumes)
- neg (List containing text from 15 negative sample resumes)
- description (Job Description text generated using 5 positive resume samples)

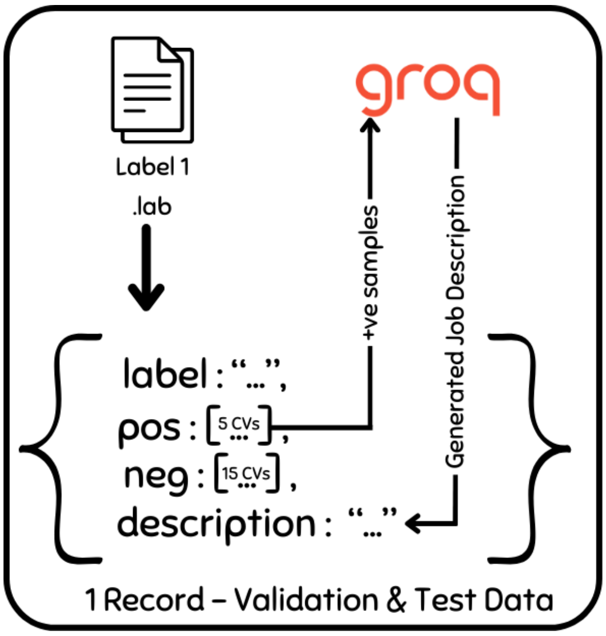

Figure 3: Validation and Test Data sample.

Overall, each query or record in validation and test datasets have a Class label, Job Description, and a pool of 20 resumes, including 5 positive samples and 15 negative samples.

For evaluation, the job description serves as the query, and the goal is to retrieve the top-k resumes, where $k = 5$ in our case. The expectation is that the model accurately identifies all 5 positive resume samples as the most relevant resumes within the pool of shuffled 20 candidate resumes (5 positive and 15 negative). Given the multi-class nature of the dataset, this setup mirrors real-world scenarios where resumes may correspond to multiple class labels. The model's performance is evaluated based on how accurately it ranks the 5 positive samples in the top-k retrieved results using several metrics.

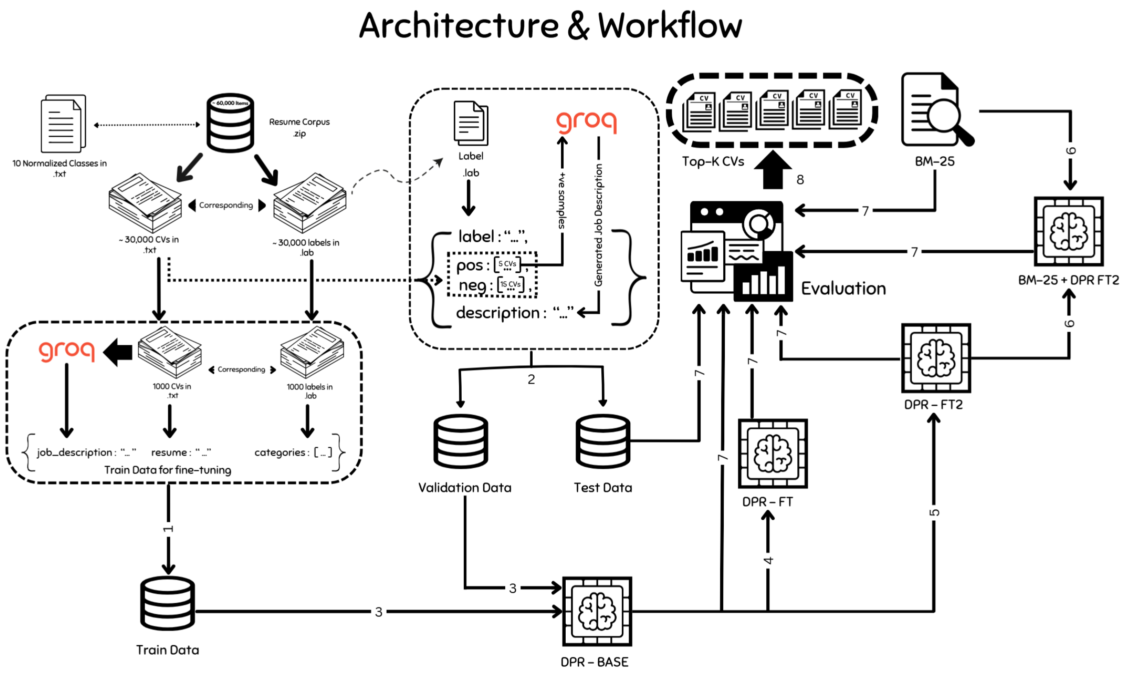

Figure 4: Proposed Architecture and Workflow.

3.2 Workflow

By the end of the data preparation phase, there are 1000 records in the training data, 100 records in the validation data, and 100 records in the test data in their respective formats. This results in considering a total of 5000 non-overlapping resume samples from our resume corpus, as each validation and test record consists of 20 resumes each. However, Training Data contain only one resume per record. The training set is used to fine-tune the DPR base model (DPR-BASE), a system designed for passage retrieval. DPR encodes both a "query" (job description) and a "passage" (resume) into a high-dimensional vector space using BERT-based encoders, where the dot product or cosine similarity between these vectors determines the relevance of a resume to a job description.

During the fine-tuning phase, Backpropagation is used along with the Adam optimizer for adaptive learning rate. Initial model considers a batch size of 8 to compute gradients and update the weights (due to computational constraints). The validation data can be utilized to monitor the training progress, ensuring a consistent decrease in model loss while avoiding overfitting on the training data. The best checkpointed models, selected based on accuracy and Mean Average Precision (MAP), Mean Normalized Discounted Cumulative Gain (MNDCG) are saved as DPR-FT. These models contain embeddings and are fine-tuned specifically for job description–resume retrieval tasks.

Next version of the model is fine-tuned with a batch size of 16. Increasing the batch size offers several benefits in this context. A larger batch size allows the model to compute gradients based on a broader set of samples, reducing gradient variance and leading to more stable updates. This stability can improve the model's generalization performance, making it more robust during inference. This model is saved as DPR-FT2, selected based on the best checkpoint determined by the previously mentioned metrics.

The final version of the proposed architecture is an ensemble model that combines BM-25 and DPR-FT2, effectively leveraging the strengths of both sparse and dense retrieval techniques. BM-25 is a sparse retrieval model that focuses on exact term matching, making it highly effective at capturing lexical similarities between job descriptions and resumes. Its precision in identifying explicit keyword matches makes it particularly valuable when the input queries contain important keywords.

On the other hand, DPR-FT2 is a dense retrieval model fine-tuned specifically for job description–resume matching. Unlike BM-25, DPR-FT2 maps the job descriptions and resumes into a shared high-dimensional vector space using dual encoders based on BERT. The cosine similarity between these embeddings captures deeper semantic relationships, enabling the model to retrieve resumes even when there are no exact keyword overlaps. However, dense retrieval models like DPR-FT2 can sometimes miss exact matches, especially when key lexical terms are not encoded properly in the embeddings.

By combining BM-25 and DPR-FT2 into a hybrid ensemble model, we harness the complementary strengths of both approaches. BM-25 ensures that important lexical information is preserved and utilized, while DPR-FT2 captures semantic meaning and contextual nuances. The hybrid architecture bridges the gaps that arise when either model is used independently. Specifically, it mitigates the limitations of sparse retrieval, which may fail to capture semantic meaning, and the weaknesses of dense retrieval, which can overlook exact matches.

The ensemble model combines the results from both BM-25 and DPR-FT2, often using a weighted fusion or ranking strategy to prioritize resumes that are consistently relevant across both retrieval methods. As a result, the hybrid system is expected to deliver improved performance, outperforming the standalone BM-25 and DPR-FT2 models. This advancement is particularly beneficial in real-world job description–resume retrieval tasks, where both lexical precision and semantic understanding are crucial for accurate and robust results.

As per Karpukhin et al. (2020), accuracy increases if we fine tune DPR on more number of training samples. Additionally, larger batch sizes are more efficient on modern hardware, such as GPUs, as they enable better parallelism. As a result, training can be faster while maintaining or improving accuracy and other evaluation metrics. However, it is important to balance this with available computational memory and ensure convergence is not adversely impacted.

3.3 Fine-Tuning DPR

The DPR model relies on constructing a dataset of positive and negative pairs. Each job description acts as a query q_i, and resume is treated as passage p_i. Positive pairs consist of job descriptions and their corresponding resumes, while negative pairs use resumes from other classes in the same batch. The idea is to bring the positive pairs together.

In each training batch, encoded job descriptions and resumes are structured as matrices:

$$Q = \begin{bmatrix} \leftarrow q_1 \rightarrow \\ \leftarrow q_2 \rightarrow \\ \vdots \\ \leftarrow q_N \rightarrow \end{bmatrix}_{N\times d} \qquad P = \begin{bmatrix} \leftarrow p_1 \rightarrow \\ \leftarrow p_2 \rightarrow \\ \vdots \\ \leftarrow p_N \rightarrow \end{bmatrix}_{N\times d}$$

where N is the batch size, and d is the BERT encoding dimension (768). The dot product between the question and passage matrices produces a similarity score matrix S:

$$S = \begin{bmatrix} s(q_1,p_1) & s(q_1,p_2) & \dots & s(q_1,p_N) \\ s(q_2,p_1) & s(q_2,p_2) & \dots & s(q_2,p_N) \\ \vdots & \vdots & \ddots & \vdots \\ s(q_N,p_1) & s(q_N,p_2) & \dots & s(q_N,p_N) \end{bmatrix}_{N\times N}$$

Each element $s(q_i, p_j)$ represents the similarity between a job description and a resume. The diagonal of S represents the similarity scores for positive pairs, while the off-diagonals serve as negative pairs. By applying the softmax function row-wise on S, the model computes probabilities for each job description, assigning higher values to relevant resumes. Loss is calculated using the negative log-likelihood of the positive pairs' probability scores.

4 Loss Function & Optimization

$$L = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\text{diag}\left(\text{softmax}_{\text{row}}(S)\right)\right) \quad (1)$$

Equation (1) represents the loss function L used for training, where N is the batch size. The similarity matrix S contains dot product between queries (job descriptions) and passages (resumes). The term $\text{softmax}_{\text{row}}(S)$ applies the softmax function row-wise to normalize these similarities into probabilities.

The diagonal elements of the resulting matrix, represented by $\text{diag}(\text{softmax}_{\text{row}}(S))$, correspond to the probabilities of each query matching its correct passage. The loss function penalizes the model when it assigns low probabilities to these positive pairs by taking their logarithm. By averaging the negative log probabilities across the batch, the model learns to maximize the relevance of query-passage pairs, enhancing retrieval performance for the resume-matching task.

5 Evaluation

To evaluate the proposed architecture (BM-25 + DPR-FT2), we assess the model's performance in retrieving the most relevant (top-k) resumes against job descriptions from the test data. We use metrics such as Accuracy, Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), and Mean Normalized Discounted Cumulative Gain (MNDCG) to compare its effectiveness with traditional methods like BM-25, DPR-BASE, and the introduced versions like DPR-FT and DPR-FT2.

Evaluation Metrics

1. Accuracy

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{2}$$

2. Mean Average Precision (MAP)

$$\text{MAP} = \frac{1}{Q}\sum_{q=1}^{Q}\left(\frac{\sum_{i=1}^{n} P(i)\cdot \text{rel}(i)}{\text{Total Relevant Documents}}\right) \tag{3}$$

- Q is the number of queries,
- $P(i) = \frac{\text{Number of Relevant Documents up to Rank } i}{i}$,
- $\text{rel}(i)$ is the relevance score (1 if relevant, 0 otherwise) at rank i.

3. Mean Reciprocal Rank (MRR)

$$\text{MRR} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{\text{Rank}_i} \tag{4}$$

- Q is the number of queries,
- Rank_i: The position (rank) of the first relevant document for the i^{th} query.

4. Mean Normalized Discounted Cumulative Gain (MNDCG@k)

$$\text{MNDCG@k} = \frac{1}{Q}\sum_{q=1}^{Q}\left(\frac{\text{DCG@k}}{\text{IDCG@k}}\right) \tag{5}$$

$$\text{DCG@k} = \sum_{i=1}^{k}\frac{\text{rel}(i)}{\log_2(i+1)}, \quad \text{IDCG@k} = \sum_{i=1}^{k}\frac{1}{\log_2(i+1)}$$

- Q is the number of queries,
- k is the cut-off rank,
- $\text{rel}(i)$ is the relevance score (1 if relevant, 0 otherwise) at rank i.

Accuracy serves as a basic metric but does not account for the position of the retrieved results. Mean Reciprocal Rank (MRR) evaluates only the position of the first relevant document, whereas Mean Average Precision (MAP) considers all relevant documents by averaging precision across their positions. Mean Normalized Discounted Cumulative Gain (MNDCG) further incorporates both position and relevance, providing a more nuanced evaluation.

We primarily focus on MAP and MNDCG over MRR since they comprehensively evaluate retrieval performance by accounting for multiple relevant documents and their rankings. These metrics offer a more robust assessment of the model's retrieval effectiveness, particularly when multiple relevant resumes exist for a given job description.

6 Results & Inferences

The results presented in Table 1 below highlight the performance of various retrieval models on test data comprising 100 job descriptions. The baseline model, DPR-BASE, which employs default encodings from Facebook and is primarily trained for open-domain tasks, underperformed across all metrics when compared to the traditional BM-25 algorithm initially.

Accuracy measures the proportion of correct predictions made by the retrieval model, providing a straightforward indication of model effectiveness though does not account the position of retrieval. In this experiment, DPR-BASE achieved an accuracy of only 48.6%, while BM-25 demonstrated a significantly higher accuracy of 63.0%. This disparity indicates that the default DPR model, lacking domain specificity, struggles to effectively match resumes to job descriptions. While DPR-FT2 model showed a slight improvement, reaching 65.0%. The highest accuracy of 68.2% was achieved by combining BM-25 with the fine-tuned DPR model, emphasizing the benefit of hybridizing traditional (lexical) and neural (semantic) approaches.

MRR quantifies the rank position of the first relevant document. A higher MRR indicates that relevant documents are returned earlier in the ranked list .Though MRR only check for the first retrieved document in the top-k, it is not an appropriate measure to total retrieval relevance and performance. BM-25 achieved perfect MRR (1.000),

MODEL	ACCURACY	MRR	MAP	MNDCG
BM-25	63.0%	**1.0000**	0.9368	0.7160
DPR-BASE	48.6%	0.7597	0.7101	0.5158
DPR-FT	63.0%	0.9325	0.8549	0.6820
DPR-FT2	65.0%	0.9683	0.8966	0.7147
BM-25 + DPR-FT	**68.2%**	**1.0000**	**0.9510**	**0.7611**

Table 1: Comparative Results on Test Data of 100 Job Descriptions (on 2000 resume samples).

while the fine-tuned models (DPR-FT and DPR-FT2) showed significant improvements, reaching 0.9325 and 0.9683, respectively. The combination of BM-25 and DPR-FT again achieved the highest MRR of 1.0000.

MAP aggregates the precision at each relevant document across multiple queries. It is a robust measure of retrieval quality. BM-25 achieved a MAP of 0.9368, whereas the DPR-BASE scored lower at 0.7101. Fine-tuning the DPR model (DPR-FT) led to a MAP of 0.8549, and further fine-tuning (DPR-FT2) improved it to 0.8966. The best MAP was attained by the BM-25 + DPR-FT, which scored 0.9510.

MNDCG assesses the ranking quality, giving higher weight to relevant documents appearing earlier in the ranked list. BM-25 achieved an MNDCG of 0.7160, which was higher than DPR-BASE (0.5158), indicating that BM-25 provides better ranking of relevant documents. The fine-tuned DPR models (DPR-FT and DPR-FT2) showed improvements in MNDCG, with scores of 0.6820 and 0.7147, respectively. But BM-25 + DPR-FT hybrid achieved the highest MNDCG at 0.7611 among all.

Conclusion

The results highlight the importance of model fine-tuning and hybridization in improving retrieval performance. While BM-25 performs well in traditional information retrieval tasks, fine-tuning the DPR model (DPR-FT) enhances its ability to capture domain-specific nuances, improving all metrics significantly. The BM-25 + DPR-FT hybrid model further reinforces the advantage of combining traditional and neural models, achieving the highest scores across all evauation metrics accuracy, MRR, MAP, and MNDCG. The improvements observed with the DPR-FT models indicate that further refinements, including larger batch sizes and extended training (using more training samples), could yield even better results.

7 Future Scope

Building upon the enhancements made to the DPR-BASE model through fine-tuning, we anticipate that further fine-tuning using more training samples will yield even greater improvements in accuracy, Mean Average Precision (MAP), and Mean Maximum Average Precision (MMAP). The below graph from the DPR paper (Karpukhin et al., 2020) hints that training on more samples that are of good quality will tend to increase the model performance.

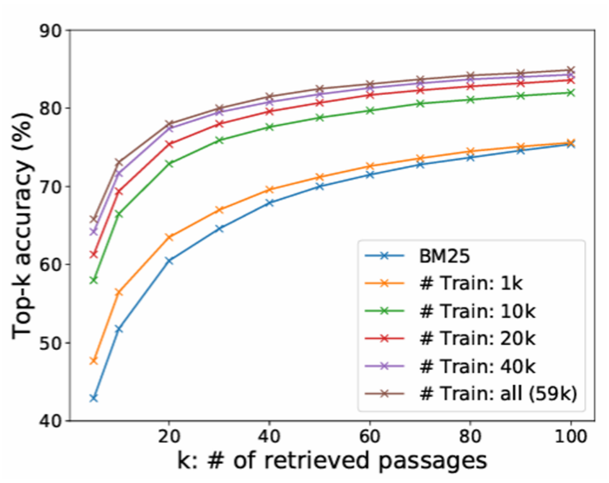

Figure 5: Accuracy by increasing Train Data (ref).

Increasing the batch size is expected to improve performance by providing more samples for robust training, especially when leveraging contrastive learning. This will likely enhance the model's ability to learn meaningful representations. Another limitation is that DPR is constrained by a context size of 512 tokens. In the future, we could explore converting decoder-only models into strong encoders, following the approach by BehnamGhader et al. (2024), as their context window is much larger than the 512-token limit.

In addition to these advancements, we aim to explore hybrid search model called BM-42[2] proposed

[2]https://qdrant.tech/articles/bm42/

by Qdrant that potentially outperforms the BM-25. This innovative approach is projected to surpass the performance of the existing models, including BM-25 and DPR-FT2, thereby setting a new benchmark in retrieval accuracy.

The current training process might include queries from the same class label within a batch, given the batch size of 16 and only 10 classes. For example, if Query 1 and Query 11 belong to the same class, we should avoid considering Passage 11 as a negative pair for Query 1, and vice versa. Incorporating in-batch positives, where queries of the same class are paired with relevant passages, is expected to improve training efficiency and model performance, resulting in better outcomes.

References

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Xiaoyu Chen, Haoyu Zhang, and Yuchao Liu. 2022. Comparative analysis of dense passage retrieval methods. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1234–1245. ACM.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Kamgang Fabrice Franklin Jiechieu and Noah Tsopze. 2021. Skills prediction based on multi-label resume classification using cnn with model predictions explanation. *Neural Computing and Applications*, 33(10):5069–5087.

Vladislav Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladislav Karpukhin, Naman Goyal, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Lijing Xiong, Zongyi Liu, and Haoyu Zhang. 2021. Enhancing dense passage retrieval with a novel reranking framework. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1507–1510. ACM.

A Models and Code Repository

To facilitate reproducibility and further research, we have made our final models and code available at the following link:

DPR-FT and DPR-FT2

The repository includes:

- DPR Fine-Tuned Models and Parameters
- Code used to train and evaluate the models.

B Link to the DPR Base Models

- **Question Encoder (Query)**
- **Context Encoder (Passage)**