Enhancing Job-Skill Matching with LLM-Driven Data Augmentation and Fine-Tuned Bi-Encoders

Working notes paper for the TalentCLEF Lab at CLEF 2025

Mohab Ali[1,*]

[1]*Computer and Systems Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt*

Abstract
Recent advances in large language models (LLMs) have enabled a wide range of applications across different domains, including human resources (HR). A key challenge in this space is identifying relevant skills for a given job title. This paper details our approach developed for the TalentCLEF 2025 shared task on job–skill matching. We explore various AI-based strategies, with a significant focus on leveraging LLM-driven data augmentation to enhance model training. Our methodology evaluates cross-encoder models and sentence similarity architectures utilizing diverse pretrained models, analyzing their trade-offs in mean Average Precision (mAP) and computational efficiency. Our final system achieved a mAP of 0.345 on the test set. We also discuss insights from contrasting experiments, including those less successful, to highlight modeling limitations and offer recommendations for future improvements in this domain. This work contributes to the TalentCLEF 2025 shared task's aim to efficiently align occupational titles with relevant professional skills.

Keywords
Job-Skill Matching, Large Language Models, Data Augmentation, Sentence Embeddings, Bi-encoder, TalentCLEF, Human Resources

1. Introduction

1.1. Motivation

Job-skill matching is a fundamental task in modern human resources (HR) and talent management. With job postings often attracting thousands of applicants, there is a growing demand for automated systems that can efficiently and accurately match candidates to job requirements. A key component of this automation is identifying and ranking the most relevant skills for a given job title, a challenge actively being addressed by research and shared tasks in the field. Recent advances in natural language processing (NLP), particularly large language models (LLMs), have opened new avenues for building intelligent HR systems. These technologies have the potential to power applications such as skill-based candidate ranking, personalized up-skilling recommendations, and early detection of emerging skill gaps. However, deploying such systems in real-world settings involves critical challenges related to scalability, semantic ambiguity, and fairness, which this work aims to explore.

1.2. Task Definition

The Job Title–Based Skill Prediction task, a part of the TalentCLEF 2025 lab within the Conference and Labs of the Evaluation Forum (CLEF) [1], requires participants to rank a predefined set of candidate skills for a given job title. The goal is to assign a relevance score (ranging from 0, not relevant, to 1, highly relevant) to each skill, ensuring that more appropriate skills are placed higher in the resulting ranked list.

This task utilizes a dataset derived from the ESCO (European Skills, Competences, Qualifications, and Occupations) database [2]. The training data consists of three main components:

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain
*Corresponding author.
es-mohab.ali2019@alexu.edu.eg (M. Ali)
0009-0006-0372-7602 (M. Ali)

- Job–skill relations file: Maps job IDs to skill IDs, providing a label (essential/optional) for each job-skill pair.
- Job titles file: Provides multiple names and aliases for each job ID.
- Skill names file: Provides multiple names and aliases for each skill ID.

Each job and skill ID corresponds to a unique URL in the ESCO taxonomy. For validation and testing, the data is structured as follows:

- A skills corpus file: Lists all candidate skill entries (with their names and aliases).
- A job queries file: Provides a list of job titles for which skills need to be ranked. These job titles were specifically selected by domain experts, which may result in a data distribution different from the training set.
- A ground truth file: Contains the relevance labels, represented as pairs of job and skill indices that reference their respective positions in the queries and corpus files.

This formulation simulates a real-world retrieval scenario where a system must match a job title query against a predefined skills corpus to identify the most relevant skills.

1.3. Challenges

This task presents several key challenges. First, scalability is a major concern: matching hundreds of job titles with hundreds of candidate skills quickly becomes computationally expensive, particularly when using cross-encoder models that require evaluating every possible pair, resulting in quadratic complexity. Second, accessibility must be considered. High-performing solutions should ideally run on affordable hardware to ensure practical adoption in real-world HR settings. Third, semantic ambiguity and lack of context make it difficult for models to make accurate predictions based only on short job titles or skill names. This issue affects both humans and machines. Finally, while LLM augmentation can effectively provide additional context, its deployment is often constrained by significant computational costs due to the need for high-end GPUs or reliance on expensive APIs. Designing a solution that balances contextual reasoning and computational efficiency is therefore non-trivial.

1.4. Contribution

In this study, we investigate both cross-encoder and bi-encoder (sentence similarity) approaches for the job–skill matching task, evaluating their trade-offs in performance and efficiency. To address the lack of input context, we experiment with augmenting job descriptions using lightweight LLMs that do not require high-end hardware. Our experiments include successful and unsuccessful attempts, with all results documented to inform future work in this area.[1]

2. Related work

2.1. Encoder models

Encoder models are central to job–skill matching tasks, as they transform textual input, such as job titles and skills, into dense vector representations that capture semantic similarity. In our experiments, we evaluate several pretrained encoder models, each with different domain characteristics and training strategies.

Zhang et al. fine-tuned XLM-RoBERTa [3], a multilingual transformer model pretrained on 2.5 TB of data, using the same ESCO database [2] provided for this shared task. Their goal was to adapt the model to the HR domain by performing domain-adaptive pretraining on the ESCO taxonomy, which spans 27 languages. The resulting model, ESCOXLM-R, was trained using masked language modeling

[1]The code for this study is available at https://github.com/mawhab/Job-Skill-Matching.

and additional multilingual objectives tailored to the ESCO structure. We selected this model for its domain specificity and strong alignment with the dataset used in the task.

Wang et al. introduced the E5 model family [4], a set of general-purpose sentence encoders with four variants: small, base, large, and large-instruct. The models were pretrained using contrastive learning on one billion multilingual text pairs, then fine-tuned on supervised tasks. The "instruct" variant was further tuned using instruction-following objectives and achieved strong results across multiple benchmarks. We include these models to explore trade-offs between performance and efficiency.

Reimers and Gurevych developed JobBERT [5], a domain-specific sentence encoder built on top of all-mpnet-base-v2 [6]. It was fine-tuned on a large corpus of job titles and their associated skills and requirements, with the goal of enabling semantic matching in HR and recruitment settings. The model maps job-related text into 1024-dimensional embeddings and is optimized for job title similarity and job–skill alignment tasks. We experimented with this model for its extensive pretraining on jobs and job descriptions.

2.2. Large Language Models

LLMs can complement encoder-based systems by generating additional context or augmenting inputs with richer descriptions. While many LLMs are resource-intensive, recent open-weight models have improved accessibility. In our experiments, we use LLaMA 3.1 8B [7], a compact but capable model that fits on a single GPU. We use it to generate descriptions and contextual data that improve the performance of embedding-based methods in similarity scoring.

3. Initial Approaches

3.1. Architectural Decisions

3.1.1. Cross-Encoder

The first approach centered on a cross-encoder architecture. In this setup, a job title and a candidate skill were concatenated, typically separated by a special [SEP] token, and fed as a single sequence into a pretrained transformer encoder. For extracting a final representation for classification, we experimented with two common strategies: using the final hidden state of the [CLS] token, and applying mean pooling over the last hidden layer's outputs (respecting attention masks).

This pooled representation was then passed through a linear projection layer followed by a sigmoid activation to predict a relevance score between 0 and 1. The core idea was that the cross-attention mechanisms within the encoder would allow for deep contextual interaction between the job and skill, enabling the model to learn different nuanced relevance signals. The intended advantage for this design was better modeling of nuanced job-skill interactions. However, while conceptually powerful, this approach suffered critically from computational inefficiency. Evaluating each job-skill pair independently resulted in quadratic inference complexity ($O(J \cdot S)$ where J is the number of jobs and S is the number of skills). With even a moderately sized test set, this led to prohibitive processing times. Despite showing some promise in capturing relevance, achieving an initial mean Average Precision (mAP) of approximately (0.23-0.26) after 1-5 epochs of fine-tuning with cross-entropy loss across different encoders (ESCOXLM-R, E5-Large, and E5-Instruct), its lack of scalability made it impractical as a standalone solution for the full task.

Our key takeaway from this experiment was that while the cross-encoder approach effectively captures relevance through deep job-skill interactions through attention mechanisms, its significant computational expense at both training and inference makes it more suitable for re-ranking a smaller set of candidates rather than for initial large-scale retrieval. Due to time and resource limitations, we decided not to move forward with this approach.

3.1.2. Bi-Encoder

The second exploratory direction utilized a bi-encoder architecture, focusing on sentence similarity. Here, job titles and skills were encoded into dense vector embeddings independently using a shared pretrained encoder. The relevance score for a job-skill pair was then computed using the cosine similarity between their respective embeddings.

The intended advantage for this approach was its high computational efficiency, as job and skill embeddings could be pre-computed and stored, allowing for fast nearest-neighbor searches or similarity calculations. Although significantly more efficient, this approach often struggled with the lack of explicit interaction context. When relying solely on the often concise raw job titles and skill names, the encoders found it challenging to disambiguate meanings or infer relevance accurately without the direct comparison offered by cross-attention. This was reflected in its performance, resulting in a lower mAP of approximately (0.13-0.22) across different encoders (ESCOXLM-R, E5-Large, E5-Instruct). It is important to note that these bi-encoder results reflect zero-shot performance using the base pretrained encoders without any task-specific fine-tuning at this exploratory stage.

Our key takeaways from this experiment are that bi-encoders are efficient but require either highly effective base encoders adept at capturing nuanced semantics from short texts or, more critically, richer input representations that provide more context for the individual job and skill texts.

3.1.3. Summary of model architectures

These initial explorations highlighted a key trade-off: the deeper contextual understanding of cross-encoders came at a steep computational price, while the efficiency of bi-encoders was often undermined by a deficit in contextual awareness when using their base pretrained representations and basic textual inputs. This understanding directly motivated our subsequent efforts to develop a more balanced and effective methodology, focusing on enhancing the input to bi-encoder systems, as detailed in the following section.

3.2. Data Formatting and Augmentation

To address the context limitations of the bi-encoder design and improve its performance, we experimented with several approaches to enrich job and skill representations, as detailed below.

3.2.1. ESCO Descriptions

ESCO's taxonomy database contains not only information about jobs, skills, and their relations, but also detailed descriptions for each job and skill entry. These descriptions represent an excellent opportunity for contextual enrichment, being reliable, accurate, and directly relevant to the entities the encoder processes.

However, this approach quickly encounters a challenge with the validation and test sets. While the skills in these sets are from ESCO (and thus have descriptions), the job titles are not directly from ESCO. Instead, they are selected by domain experts and lack a direct mapping or URL to an equivalent ESCO occupation (even if such an equivalent might conceptually exist).

To overcome this, our strategy involved training models with the available ESCO descriptions and then, for validation and testing, attempting to find the closest matching job title from the training set (which has ESCO descriptions) and using its description as a proxy.

We used two distinct methods to find these close matches. First, we tried Python's get_close_matches function from the Difflib library. This function performs sequence-based text matching, identifying the closest n matches for a given string (a validation/test job title) from a list of strings (training set job titles) based on a similarity cutoff score. This approach, while effective, was relatively slow but could be performed offline and prepared in advance.

The second method utilized JobBERT. This was a conceptually strong approach, as JobBERT is specifically pretrained for job title similarity. We first generated embeddings for all job titles from both the training set and the validation/test sets using JobBERT, then for each validation/test job title, we identified the most similar training set job title via cosine similarity ranking. This method also benefited from being significantly faster due to efficient batch processing and parallelization capabilities for embedding generation.

We evaluated this proxy-description augmentation technique with various encoders. A key observation across both Difflib and JobBERT-based matching methods was that the model had the highest score when using a very high similarity cutoff (around 0.97). This strict cutoff meant that approximately 95% of the validation set job titles did not receive a proxy description from the training set, as no sufficiently close match could be identified. However, using descriptions only for the matched 5% still outperformed using no descriptions at all or using a lower cutoff (which would introduce more, potentially less accurate, descriptions). This finding strongly signaled that while accurate descriptions were crucial, having no description was preferable to having an inaccurate one. This insight provided a strong motivation for exploring LLM-based description generation, as discussed next.

3.2.2. LLM Generated Descriptions

To generate informative descriptions, particularly for potentially ambiguous job titles, and ensure system accessibility, we needed a high-performing LLM capable of running on a single consumer-grade GPU. We opted to use LLaMA 3.1 8B instruct. We engineered the prompt with the following objectives: conciseness (to fit encoder context windows), clarity (to avoid superfluous tokens or hallucinations), inclusion of 2-3 relevant skills (to directly enhance job-skill similarity signals), direct output of only the description (omitting conversational preambles), and avoidance of redundant information already present in the job title. We worked with the following prompt:

> You are an expert HR assistant specialized in writing job descriptions. Generate a highly concise, professional, and factual job description based ONLY on the provided job title. Focus on typical key responsibilities. **Crucially, explicitly list 2-3 common, essential skills associated with the role within the description, often towards the end.** Do not add information not directly implied by the job title. The description must be brief, ideally 2-3 sentences maximum, to fit within processing limits. Output only the description text, without any preamble or introductory phrases.

We generated descriptions for all job titles in the training, validation, and test sets. For the training set, this was done despite the availability of official ESCO descriptions to ensure consistency in the style and focus of descriptions across all data splits; LLM-generated descriptions might emphasize different aspects than the ESCO originals, and training on a consistent distribution was deemed important. Furthermore, we generated a unique description for each job title alias, even those mapping to the same underlying ESCO job ID. This was to prevent the encoder from potentially learning to disregard the specific alias text and overly rely on identical descriptions if they were shared.

3.2.3. Summary of Input Formatting

Our exploration into enriching context, initially with official ESCO descriptions and then more effectively with LLM-generated content, aimed to provide richer inputs for the encoders. We used LLMs to generate descriptions for jobs for training, validation, and test sets. For skills, we used the available ESCO descriptions as they were available across training, validation, and test sets. The final input text to the encoders would be:

- For jobs: "Job: (job title) [SEP] Description: (LLM generated description)"
- For skills: "Skill: (skill name) [SEP] Description: (official ESCO description)"

For E5-Instruct, we include an instruction prompt to make use of the model's instruction tuning. The input was adjusted to the following:

- For jobs: "Instruct: Given a job title and its description, retrieve relevant skills based on their descriptions. Query: Job: (job title) [SEP] Description: (LLM generated description)"
- For skills: "Skill: (skill name) [SEP] Description: (official ESCO description)"

4. Methodology

4.1. Overall System Architecture

Our final system uses a bi-encoder architecture. Pretrained transformer models serve as the backbone to independently encode job and skill textual representations. These encoders are fine-tuned using a contrastive learning approach. At inference, the relevance of a skill to a job is determined by the cosine similarity between their respective vector embeddings, allowing for efficient ranking. This architecture was selected to balance predictive performance with the computational efficiency required for potentially large-scale matching, informed by our initial explorations in Section 3.1.

4.2. Input Representation

The textual inputs for jobs and skills were combined with their descriptions to provide rich contextual information to the encoders, as detailed in Section 3.2.3.

4.3. Encoder Models and Embedding Extraction

We evaluated three distinct pretrained transformer models as the backbone for our bi-encoder: ESCOXLM-R, E5-Large, and E5-Instruct. These models were fully fine-tuned. For each input text, the dense vector embedding was derived by mean pooling over the last hidden layer's token embeddings, respecting attention masks.

4.4. Fine-tuning setup

The bi-encoder models were fine-tuned using Multiple Negatives Ranking Loss (MNRL) following Henderson et al. [8]. MNRL is a contrastive loss function designed to train retrieval models by ensuring that positive (relevant) pairs have higher similarity scores than negative (irrelevant) pairs. For constructing training batches, we processed our list of training jobs. For each job, one of its aliases was randomly sampled, along with one skill associated with it and a randomly sampled alias for that skill. This formed a positive (anchor, positive) pair. This sampling strategy ensured that each unique job ID appeared at most once per batch, which is crucial for the correctness of using in-batch negatives and aids training efficiency on the large dataset. All other skill texts within the same batch served as in-batch negatives for each anchor job text.

A comprehensive list of hyperparameters can be found in Table 1. We used the AdamW optimizer [9] with a linear learning rate decay schedule and a 6% warmup period. The final model weights were selected from the epoch that achieved the highest mAP on the validation set. All training was conducted on a P100 GPU via the Kaggle platform.

Table 1
Hyperparameters used for fine-tuning

Hyperparameter	Value
Batch size	8
Learning rate	1e-5
Betas	0.9, 0.98
Gradient accumulation steps	16
Epochs	20

4.5. Inference

During inference, all job texts and skill texts along with their generated descriptions are first encoded using the fine-tuned bi-encoder to generate their respective dense vector embeddings. For a given query job, its embedding is compared against all skill embeddings in the target corpus using cosine similarity. Skills are then ranked for that job in descending order of their similarity scores.

5. Results and Discussion

Our best model achieved a mAP score of 0.345 on the official TalentCLEF test set. In this section, we present different encoder models' results on the validation set.

5.1. Results

Table 2 presents a comprehensive comparison of the performance of the three evaluated encoder backbones on the validation set. These results cover both zero-shot performance and performance after fine-tuning with MNRL under different input augmentation conditions, as detailed in the table caption. Across all fine-tuned settings, a clear trend of improvement was observed with the addition of richer contextual information. When specifically considering our most effective augmentation strategy (LLM-generated job descriptions combined with ESCO skill descriptions), E5-Instruct achieved the highest validation mAP of 0.313. The standard E5-Large was a close second, reaching a mAP of 0.3089. Somewhat surprisingly, ESCOXLM-R, under these same conditions, achieved a mAP of 0.298. The full set of experimental results in Table 2 further shows the impact of zero-shot versus fine-tuned performance and the varying effects of the different description types on each encoder.

Table 2
Results from different encoders and different experiments. In experiments where descriptions were used, skill descriptions are always taken from ESCO's database as explained in Section 3.2.3. The first column shows the zero shot results of each model with no descriptions, indicating the baseline results. All other columns are fine-tuned results.

	No Descriptions (Zero-Shot)	No Descriptions	ESCO Descriptions	LLM Descriptions
ESCOXLM-R	0.1317	0.2707	0.2689	0.298
E5-Large	0.2022	0.2902	0.3081	0.3089
E5-Instruct	**0.2197**	**0.2953**	**0.3099**	**0.313**

5.2. Discussion

Our experiments reveal several key insights into building effective job-skill matching systems. The final fine-tuned bi-encoder architecture, particularly when using E5-Instruct with LLM-generated job descriptions, demonstrated strong performance, achieving 0.313 mAP on our validation set and generalizing well to the official TalentCLEF test set with a mAP of 0.345.

5.2.1. Interpreting Encoder Performance

The E5 model family, especially E5-Instruct, consistently outperformed ESCOXLM-R in our fine-tuned setups (Table 2). The E5 models' robust performance, even with minimal context (as seen in the "No Descriptions" fine-tuned column), suggests their large-scale diverse pretraining provides a strong foundation. The instruction tuning of E5-Instruct likely further aided its ability to focus on the retrieval task when presented with augmented inputs. The comparatively lower scores for ESCOXLM-R, despite its ESCO-specific pretraining, were notable. The largest performance gap between ESCO-proxied descriptions and LLM-generated descriptions was observed for ESCOXLM-R (an increase from 0.2689 to 0.298 mAP). This could imply that ESCOXLM-R is highly sensitive to the style and completeness of contextual information; when most validation job titles lacked a high-quality ESCO proxy description, its performance suffered more acutely than the E5 models. Conversely, it benefited more when comprehensive LLM descriptions were provided for all titles. This suggests that while domain pretraining is valuable, adaptability to varied descriptive styles, or a consistent supply of rich context, is equally crucial.

5.2.2. Impact of Augmentation and Fine-tuning

An important finding is the significant improvement from zero-shot performance (e.g., 0.2197 for E5-Instruct) to fine-tuned performance with LLM descriptions (0.313 for E5-Instruct). This suggests that while pretrained encoders have general semantic understanding, task-specific fine-tuning is essential for optimal retrieval. The LLM-generated descriptions, which explicitly included 2-3 relevant skills as per our prompt, likely created stronger semantic anchors for the bi-encoder, directly facilitating better job-skill similarity assessments. The guarantee of having a description for every job title via LLM generation, regardless of its presence in ESCO, was a key advantage over the proxy-based ESCO description approach. Our final fine-tuned bi-encoder system also surpassed the performance of our initial cross-encoder explorations (0.23-0.26 mAP), demonstrating a more scalable and ultimately more effective solution.

5.2.3. Limitations

Despite these results, limitations exist. Our choice of LLaMA 3.1 8B for generating job descriptions balanced performance with accessibility; larger LLMs might offer further improvements at a higher computational cost. The prompt guiding this LLaMA 3.1 8B generation process, though iterated upon, could be further refined. Similarly, the instruction prompt prepended to the input for E5-Instruct, while simple and effective, was not exhaustively optimized and could be a subject for further experimentation. Additionally, our hyperparameter search for fine-tuning, while guided by common practices, was not exhaustive due to resource constraints.

6. Conclusion and Future Work

In this work, we explored various approaches for the TalentCLEF job-skill matching task, systematically evaluating different system architectures, encoder models, input augmentation strategies, and fine-tuning techniques. Our findings highlight the significant impact of rich contextual input (particularly through LLM-generated job descriptions) and task-specific fine-tuning (using MNRL) on the performance of bi-encoder models. Our most effective system utilized a fine-tuned E5-Instruct bi-encoder, augmented with LLM-generated job descriptions and official ESCO skill descriptions. This approach achieved a mAP of 0.345 on the official TalentCLEF test set, demonstrating a robust and scalable solution.

For future research, several avenues warrant exploration. Investigating the use of larger or more specialized LLMs for job description generation, as well as experimenting with LLM-generated descriptions for skills, could further enhance input representations. Additionally, more extensive hyperparameter tuning for the fine-tuning process and dedicated experimentation with more sophisticated instruction prompts for E5-Instruct may unlock additional performance. Finally, exploring a two-stage hybrid architecture, where our efficient fine-tuned bi-encoder retrieves an initial set of candidates subsequently re-ranked by a more computationally intensive cross-encoder, could offer a compelling balance of efficiency and accuracy. These directions could build upon the insights gained in this study to further advance the state-of-the-art in job-skill matching.

Declaration on Generative AI

During the preparation of this work, the author used Gemini 2.5 Pro in order to: Paraphrase and reword, Grammar and spelling check. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] L. Gasco, H. Fabregat, L. García-Sardiña, P. Estrella, D. Deniz, A. Rodrigo, R. Zbib, Overview of the TalentCLEF 2025: Skill and Job Title Intelligence for Human Capital Management, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2025.

[2] M. le Vrang, A. Papantoniou, E. Pauwels, P. Fannes, D. Vandensteen, J. De Smedt, Esco: Boosting job matching in europe with semantic interoperability, Computer 47 (2014) 57–64.

[3] M. Zhang, R. Van Der Goot, B. Plank, Escoxlm-r: Multilingual taxonomy-driven pre-training for the job market domain, arXiv preprint arXiv:2305.12092 (2023).

[4] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, F. Wei, Multilingual e5 text embeddings: A technical report, arXiv preprint arXiv:2402.05672 (2024).

[5] J.-J. Decorte, J. Van Hautte, T. Demeester, C. Develder, Jobbert: Understanding job titles through skills, arXiv preprint arXiv:2109.09605 (2021).

[6] Nils Reimers and Sentence Transformers contributors, all-mpnet-base-v2, https://huggingface.co/sentence-transformers/all-mpnet-base-v2, 2022. Accessed: April 2025. Hugging Face Model Card.

[7] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., The llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024).

[8] M. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, R. Kurzweil, Efficient natural language response suggestion for smart reply, arXiv preprint arXiv:1705.00652 (2017).

[9] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, arXiv preprint arXiv:1711.05101 (2017).