Candidate Matching System

Majd Alsayari*[1]
TAHAKOM
Riyadh, Saudi Arabia
alsayarimajd@gmail.com

Lina Alsayari*[1]
TAHAKOM
Riyadh, Saudi Arabia
lenasayari@gmail.com

Lena Alkhodhairi*[1]
TAHAKOM
Riyadh, Saudi Arabia
lalkhodhairi@gmail.com

Abstract

Recruitment processes increasingly rely on automated systems to identify qualified candidates efficiently. Yet, traditional keyword-based methods often miss the deeper semantic alignment between candidate CVs and job descriptions (JDs). The Candidate Matching System developed in this project combines hybrid retrieval, re-ranking, and knowledge graph reasoning to improve matching accuracy and interpretability. Large-scale CV and JD data are collected, normalized, and embedded using the BGE-M3 (Chen et al., 2024) and MB25s (Lù, 2024) models. Dense and sparse retrieval are integrated to balance lexical precision with semantic understanding, while a cross-encoder re-ranker refines top results for higher relevance. A domain-specific knowledge graph enriches contextual links between skills, experience, and job requirements. An AI-based reasoning component further explains why selected CVs best fit each JD, enhancing transparency and trust. Overall, the system advances interpretable and semantically aware candidate–job matching for automated recruitment.

1 introduction

The increasing reliance on automated recruitment platforms highlights the need for systems capable of understanding more than just keywords. Traditional matching methods in applicant tracking systems often depend on lexical overlap, overlooking the semantic and contextual alignment between a candidate's qualifications and a job's requirements. This limitation motivated the development of an end-to-end candidate matching system that applies modern natural language processing (NLP) techniques to represent and compare CVs and job descriptions (JDs) more intelligently.

The project was undertaken as a hands-on learning initiative, allowing the team to apply concepts in information retrieval, representation learning, and model evaluation to a real-world problem. The objective was to design a system that can process unstructured CV and JD data, extract meaningful representations, and determine the most relevant candidate profiles for each job posting with improved accuracy and transparency.

To achieve this, the system incorporates multiple stages: large-scale data collection and normalization, hybrid retrieval combining sparse and dense representations, cross-encoder re-ranking for refined similarity scoring, and integration of a knowledge graph to enrich contextual relationships between skills and experience. An additional explainability component uses AI-based reasoning to clarify why certain CVs are selected, promoting interpretability and user trust.

*Both authors contributed equally to this research.
[1]Supervised by Dr.Tanveer Hussain

Evaluation plays a central role in the project. Appropriate metrics were selected to measure retrieval quality and overall system performance, ensuring the results reflect both semantic accuracy and practical usefulness.

Ultimately, the project delivers a complete candidate matching system that demonstrates how hybrid retrieval, knowledge integration, and explainable reasoning can collectively enhance automated candidate selection while deepening understanding of modern NLP techniques.

2 Methodology

This section outlines the main components and workflow of the Candidate Matching System. As illustrated in Figure 1, the process begins with data collection and preprocessing where raw CV and job description data are normalized, scored, and standardized before progressing through hybrid retrieval, re-ranking, and explainability stages. Each step contributes to accurately identifying and interpreting the best candidate matches for a given job description.

2.1 Data

The data forms the foundation of the Candidate Matching System. This subsection explains the datasets used, the preprocessing and standardization applied, and the steps taken to prepare the data for evaluation and model integration.

2.1.1 Data Sources The data used in this project was obtained from publicly available resources on the Hugging Face platform. Two datasets were initially considered for building and testing the Candidate Matching System. Both datasets contained paired records of candidate CVs and job descriptions (JDs), each labeled to indicate whether the pair represented a suitable match.

The first dataset we used is by (Namuangtoun, 2024) it has a three-class labeling scheme with the categories No Fit, Potential Fit, and Good Fit, while the second dataset (Bayu, 2024) employed a binary labeling format, where 0 denoted not fit and 1 denoted fit.

After an initial review and comparison, the second dataset was found to be more consistent and better aligned with the project's objectives. Its simpler binary labels and cleaner structure made it a more reliable foundation for the subsequent stages of development, including normalization, matching, and evaluation.

To further verify the reliability of the dataset and ensure that the labeling aligned with the actual semantic similarity between CVs and JDs, the Qwen-8B (Team, 2025) model was used to assign a 0–100 similarity score for each pair. This scoring process served as an additional validation layer, allowing a more nuanced view of candidate–job relationships beyond the binary labels.

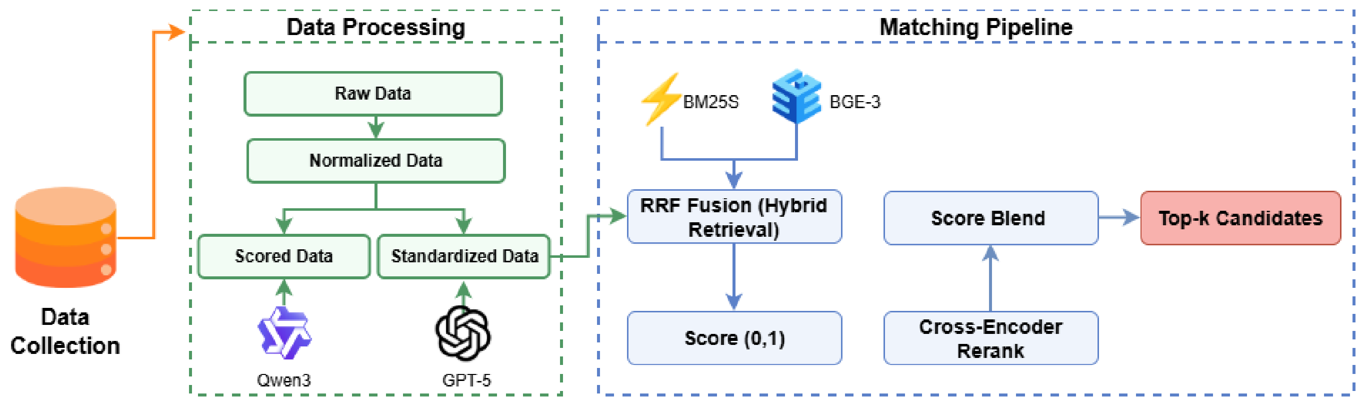

Figure 1: Overview of the Candidate Matching System workflow, showing data preprocessing, hybrid retrieval (BM25S + embedding), and cross-encoder re-ranking leading to top-k candidate selection.

2.1.2 Data Normalization and Standardization The dataset underwent several processing stages to ensure consistency, interpretability, and readiness for downstream analysis. Each stage refined the data to progressively move from unstructured raw text toward a structured and comparable representation.

Raw Data. This represents the dataset in its original form as collected from Hugging Face sources. At this stage, the CV and JD pairs contained unprocessed, free-form text that often included redundant information, inconsistent formatting, and varied naming conventions. Although labeled, the data required refinement before being suitable for similarity scoring and retrieval.

Normalized Data. A light preprocessing stage was applied to clean the raw data while preserving its semantic meaning. This involved removing unnecessary symbols, punctuation artifacts, and irregular spacing. Additionally, equivalent terms and naming variations were unified to ensure a consistent representation across all records. For example, different textual forms referring to the same concept or tool were standardized under a single canonical form. This step ensured that both CV and JD entries followed a uniform writing style, enabling models to focus on actual content rather than surface-level differences.

Standardized Data. Because the normalized data still consisted of unstructured free text, a further transformation stage was introduced using GPT-5 (OpenAI, 2025). The model was prompted to extract and organize essential professional details—such as job titles, education, years of experience, key skills, and tools—into a consistent JSON-like schema. Separate structured formats were defined for CVs and JDs to maintain alignment while respecting their unique content characteristics. This produced a machine-readable dataset that could be directly leveraged for similarity evaluation and retrieval.

Overall, this multi-stage process established a validated, clean, and interpretable dataset, bridging the gap between raw textual inputs and the structured representations required by the Candidate Matching System.

2.1.3 Preparation for Evaluation Once the data was cleaned and standardized, it was prepared for use in the system's evaluation stage. Each job description (JD) was compared against all available candidate CVs to assess their degree of fit. This scoring process generated a large matrix of CV–JD similarity values, which later served as the basis for ranking and performance measurement.

To maintain consistency with the project's overall evaluation framework, every pair was assigned a numeric score representing how strongly the candidate matched the job requirements. These scores were stored alongside the binary labels already present in the dataset, allowing direct comparison between model predictions and ground-truth annotations. This alignment ensured that subsequent experiments—such as retrieval, re-ranking, and explainability, could be conducted on a coherent, well-structured dataset.

The resulting prepared dataset therefore functioned as both a training and benchmarking resource, enabling the system to be evaluated on accuracy, interpretability, and ranking quality in later sections of the report.

2.2 Hybrid Retrieval

The retrieval stage is responsible for identifying a broad pool of potentially relevant CVs for a given Job Description (JD). To balance lexical precision and semantic generalization, we employ a hybrid retrieval framework that integrates both sparse and dense representations. We utilize the *standardized* (summarized) versions of both Job Descriptions (JDs) and Candidate CVs as inputs to the retrieval stage. These standardized summaries are concise, normalized textual representations of the original documents that preserve key skills, roles, qualifications, and experience signals. Empirically, the standardized forms demonstrated superior retrieval accuracy compared to raw or merely normalized texts. Quantitative evidence for this choice is provided in Section 5.

2.2.1 Sparse Retrieval (BM25) (Lù, 2024) The sparse retrieval branch is based on the BM25 ranking function, a traditional probabilistic model that rewards exact token overlaps between the JD and CV summaries. After normalization and tokenization, each CV summary is represented as a bag of weighted terms. The BM25 scorer computes the relevance of a JD query to each CV document based on term frequency, inverse document frequency, and document length normalization. This component excels at capturing exact keyword matches and domain-specific terminology (e.g., "financial analysis", "data visualization", "regression modeling"), ensuring that obvious lexical correspondences are preserved.

2.2.2 Dense Retrieval (Sentence Transformer) (Chen et al., 2024) In parallel, the dense retrieval branch encodes both JDs and CVs using a Sentence Transformer model that converts text into high-dimensional semantic vectors. These embeddings capture contextual and paraphrased relationships beyond surface-level word overlap. Cosine similarity between the JD vector and all precomputed CV vectors is computed efficiently using a FAISS index for large-scale retrieval.

This approach retrieves CVs that are semantically related even when they use different wording (e.g., "forecasting revenue" vs. "sales prediction"), improving recall for paraphrased or contextually similar phrases.

2.2.3 Reciprocal Rank Fusion (RRF) (Cormack et al., 2009)

To merge the outputs of both branches, we use Reciprocal Rank Fusion (RRF), which combines the BM25 and embedding-based rankings into a single unified score:

$$\text{RRF Score}(d) = \sum_{m \in \{\text{BM25,Embedding}\}} \frac{1}{k + \text{rank}_m(d)} \tag{1}$$

where k is a smoothing constant (typically k=60).

RRF effectively balances the strengths of both models, producing stable, high-recall retrieval results. Each retrieved CV is assigned a normalized score in the 0–1 range, which is later used as the prior probability for reranking.

The hybrid retriever thus produces a candidate pool that is both lexically faithful and semantically rich, serving as the foundation for the subsequent reranking phase.

2.3 Reranker

While the hybrid retriever achieves high recall by combining lexical and semantic search, its scoring functions remain relatively shallow—relying on token overlap or independently computed embeddings. To refine this candidate list into a precise, contextually ranked output, we employ a cross-encoder reranking model. **Throughout the retrieval and reranking stages, we use the standardized (summarized) versions of both Job Descriptions (JDs) and candidate CVs.**

2.3.1 Cross-Encoder Architecture (Xiao et al., 2023) The reranking stage uses a cross-encoder transformer that jointly encodes the JD and each candidate CV into a single sequence:

$$\texttt{[CLS]}\ \text{JD}\ \texttt{[SEP]}\ \text{CV}\ \texttt{[SEP]} \tag{2}$$

This representation allows the model to attend over both texts simultaneously, capturing fine-grained token-level interactions that dual-encoder embeddings cannot. As a result, the cross-encoder can recognize nuanced semantic alignments (e.g., "managed accounting systems" $\leftrightarrow$ "implemented SAP for finance automation") and contextual relevance that depend on cross-text relationships.

2.3.2 Scoring and Normalization Each JD–CV pair is passed through the cross-encoder to produce a relevance score, typically via a regression or classification head applied to the `[CLS]` representation. To ensure comparability across candidates, the raw scores are min–max normalized to $[0, 1]$:

$$\hat{s}_{\text{ce}}(d) = \frac{s_{\text{ce}}(d) - \min_{d'} s_{\text{ce}}(d')}{\max_{d'} s_{\text{ce}}(d') - \min_{d'} s_{\text{ce}}(d')}. \tag{3}$$

This normalized score reflects the model's confidence in the semantic alignment between the JD and the CV.

2.3.3 Hybrid Blending with Prior Scores To balance the precision of the cross-encoder with the robustness of the hybrid retriever, we linearly combine the normalized cross-encoder score with the prior retrieval score:

$$S_{\text{final}} = \alpha\, \hat{s}_{\text{ce}} + (1 - \alpha)\, S_{\text{retrieval}}, \qquad \alpha \in [0, 1], \tag{4}$$

where:

$S_{\textbf{retrieval}}$ is the normalized hybrid (BM25 + embedding) score;
$\hat{s}_{\textbf{ce}}$ is the normalized cross-encoder score; and
α controls the contribution of each component.

This blended strategy yields a ranked list that balances semantic precision and retrieval consistency. In our experiments, values of α between 0.6 and 0.8 typically offered a strong trade-off. *Unless otherwise stated, subsequent experiments use the standardized data mode with* $\alpha = 0.5$, *a candidate pool of 200, and top-*$k = 10$*; Section 5 shows this configuration yields the best overall performance.*

2.4 Explainability

The system incorporates an explainability component powered by an LLM (GPT-5). After the reranking stage produces the top-k CVs for each JD, the LLM receives the JD together with each candidate CV and generates a brief explanation describing why that candidate was ranked highly.

3 Knowledge Graph

Candidate CVs are typically composed of unstructured text, making it difficult to explicitly identify relationships among the various elements they contain. To overcome this limitation, we explored a new approach by using a Knowledge Graph (KG) representation designed to capture the semantic and contextual relationships among entities within each CV. This structured format transforms unorganized textual content into a graph data representation that reflects how different aspects of a candidate's background interact.

3.1 CV Knowledge Graph Construction

Each CV is converted into a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $\mathcal{V}$ represent different types of entities and edges $\mathcal{E}$ describe the relationships among them. The graph is constructed using a LLM (GPT-5), guided by a carefully designed prompt that defines the entity types, specifies the possible relationships between them, and provides normalization rules to ensure consistent structure and terminology across all generated graphs.

The system defines seven primary node types to ensure full coverage of common CV components:

- **Skills** — technical and soft skills.
- **Tools** — software, programming languages, or frameworks.
- **Job Roles** — professional titles or occupational positions.

- **Education Degrees** — academic qualifications.
- **Certifications** — professional certificates or credentials.
- **Awards** — honors or recognitions received.
- **Institutions** — universities, companies, or organizations associated with the candidate.

Directed edges connect these nodes to represent interactions between them. For example, CV – [: HAS_SKILL] → Skill and CV – [: HAS_TOOL] → Tool indicate that the candidate has certain skills and tools. In addition, the Knowledge Graph also models more detailed interactions, such as which specific skills or tools are associated with each job role or educational record.

The KG representation provides a comprehensive and detailed structured view of the candidate's CV, offering a deeper semantic understanding rather than relying solely on textual similarity.

3.2 CV Embedding

Each CV graph is encoded into a vector representation using a combination of **Graph Neural Network (GNN)** propagation and an **Attention–IDF pooling** mechanism. This stage transforms structured graph information into context-aware embeddings.

3.2.1 Embedding Cache An embedding cache is created for each unique token within every entity type using the **BGE-M3** model. This caching step accelerates the process by embedding each unique term once and reusing it across different CVs, ensuring both efficiency and consistency.

3.2.2 Graph Neural Network (GNN) The objective of this stage is to make each node representation not only reflect its own content but also incorporate contextual information from connected nodes. Node embeddings are initialized from the precomputed BGE-M3 embedding caches, and the graph undergoes two hops of message passing. During this process, information flows along directed edges whose strengths are determined by predefined relation weights, while edges in the opposite direction carry half of that weight. The node update rule is defined as:

$$H_{\text{dst}}^{(l+1)} = 0.5\,H_{\text{dst}}^{(l)} + 0.5\,\text{MeanNeighbors}(w_r\,H_{\text{src}}^{(l)}),$$

where w_r represents the weight assigned to each relation type. This propagation allows each node to integrate structural and semantic information from its neighboring entities, resulting in context-enriched node embeddings.

3.2.3 Attention–IDF Pooling and Normalization To derive a single vector representation for each node type, an **Attention–IDF pooling** mechanism is applied. This method aggregates node embeddings by assigning an importance weight to each node based on two factors: (1) the **Inverse Document Frequency (IDF)**, which emphasizes rare entities across all CVs, and (2) **meta-based attribute weights**, which incorporate contextual priors derived from the candidate's background—such as job type (e.g., full-time weighted higher than part-time) and academic degree level (e.g., PhD weighted higher than bachelor).

Each node's attention weight a_i is computed as:

$$a_i = \frac{\exp(\beta \cdot \text{IDF}_i + \gamma \cdot \text{Attr}_i)}{\sum_j \exp(\beta \cdot \text{IDF}_j + \gamma \cdot \text{Attr}_j)},$$

where IDF_i and Attr_i represent the rarity and contextual importance of node i, respectively, and β and γ are balancing coefficients that control the contribution of each factor. The final type-level embedding is then obtained through a weighted sum of all node embeddings:

$$v_{\text{type}} = \sum_i a_i E_i,$$

where E_i denotes the GNN-updated embedding of node i. Finally, all resulting per-type vectors are **L2-normalized** to ensure consistent scaling across CV representations.

The resulting representation for each CV can be expressed as:

$$\text{CV} = [v_{\text{skills}}, v_{\text{tools}}, v_{\text{roles}}, v_{\text{programs}}, v_{\text{institutions}}, v_{\text{certifications}}, v_{\text{awards}}],$$

where each v_{type} represents the embedding corresponding to a specific node category. Together, these vectors form a structured and semantically rich representation of the candidate's CV.

3.3 Job Description Embedding

Each Job Description is transformed into a structured vector representation using the **Standardized Data**, which provides separate fields for each JD. Four fields are used: *job title*, *skills*, *tools*, and *education*. Each field is embedded individually using the **BGE-M3** model, forming a structured multi-vector representation:

$$\text{JD} = [v_{\text{title}}, v_{\text{skills}}, v_{\text{tools}}, v_{\text{education}}],$$

where each v_{section} represents the embedding of a specific field. This structured representation aligns with the CV embeddings, enabling semantic correspondence between job requirements and candidate profiles.

4 Similarity Scoring

After constructing structured embeddings for both Job Descriptions and candidate CVs, the system performs a matching stage inspired by cross-attention. Each JD is represented by four vectors corresponding to its key components:

$$\text{JD} = \left[v_{\text{skills}}^{JD}, v_{\text{tools}}^{JD}, v_{\text{education}}^{JD}, v_{\text{title}}^{JD}\right].$$

Although the CV representation contains additional information, only the four vectors that directly align with JD fields are used during matching:

$$\text{CV} = \left[v_{\text{skills}}^{CV}, v_{\text{tools}}^{CV}, v_{\text{programs}}^{CV}, v_{\text{roles}}^{CV}\right].$$

This ensures a one-to-one correspondence between JD and CV components. In this formulation, the JD acts as the *query*, while the CV provides *key–value* information, following the structural idea of cross-attention but without any learnable parameters.

4.1 Per-Type Similarity Computation

For each JD–CV pair, the similarity between corresponding components is computed using the dot product of their L2-normalized vectors:

$$s_t(JD, CV) = \left\langle v_t^{JD},\ v_t^{CV} \right\rangle, \qquad t \in \{\text{skills, tools, programs, roles}\}.$$

This produces a four-dimensional similarity vector:

$$s = \left[s_{\text{skills}}, s_{\text{tools}}, s_{\text{programs}}, s_{\text{roles}} \right].$$

This vector forms the fundamental matching signal for all subsequent aggregation heads.

4.2 Non-Learned Multi-Head Aggregation

To model different ways a JD may implicitly emphasize its requirements, the system applies multiple fixed attention-style heads. Each head represents a different weighting pattern over the four JD–CV similarity components.

A single head h computes a weighted combination of the per-type similarities:

$$\text{Head}_h(JD, CV) = \sum_{t=1}^{4} w_t^{(h)}\ s_t(JD, CV),$$

where the weights satisfy:

$$\sum_{t=1}^{4} w_t^{(h)} = 1.$$

The final matching score is the average across all H heads:

$$\text{Score}(JD, CV) = \frac{1}{H} \sum_{h=1}^{H} \text{Head}_h(JD, CV).$$

This mechanism acts as a non-learnable cross-attention module, in which each head simulates a different focus on the JD requirements. The system applies several predefined multi-head aggregation types. Three head types were evaluated:

4.2.1 Fixed-Weight Heads (H = 1) This type applies a static weighting scheme across the four similarity components. Two variants are included.

- **Equal-weight head**: all four components receive the same weight ($\frac{1}{4}$ each).
- **Single-type heavy heads**: one component receives full weight, and the remaining components receive zero.

These configurations provide simple baselines for assessing the effect of uniform versus exclusive emphasis on JD requirements.

4.2.2 Pair-Based Heads (H = 4) This type models interactions between JD components by assigning each head to one pair of similarity components and giving them equal weights of 0.5 and 0.5. There are six possible JD–CV type pairs, arranged here in two rows for clarity:

(skills, tools), (skills, programs), (skills, roles),
(tools, programs), (tools, roles), (programs, roles).

A single model selects four out of these six pairs, producing four heads in total. The number of possible 4-pair selections from six is

$$6C4 = 15,$$

and therefore fifteen distinct 4-head configurations are evaluated. Each configuration captures a different pattern of pairwise interactions between JD components.

4.2.3 All-Pairs Heads (H = 6) This type incorporates all six JD component pairs. Each head corresponds to one pair and applies equal weights of 0.5 and 0.5 to its two similarity components. This creates a complete pairwise interaction structure between the JD and CV representations.

5 Evaluation

This section presents the quantitative evaluation of the Candidate Matching System, highlighting how different text preparation modes, blending parameters, and retrieval settings influence overall ranking performance. The evaluation was conducted on 452 Job Descriptions (JDs) and 637 Candidate CVs, with each JD compared against all CVs, resulting in a 452×637 similarity matrix. Each pair was assigned a GPT-5–based relevance score ranging from 0 to 100.

5.1 Evaluation Setup

A relevance threshold of $t = 75$ was used to define a “good fit,” labeling pairs with scores above this value as relevant. This threshold offered a balanced trade-off between precision and recall, ensuring that moderately suitable candidates were retained.

Three text modes were evaluated, reflecting the progressive data refinement stages introduced in Section 2.1:

- **Raw:** minimally processed text containing the original wording after basic punctuation and whitespace cleaning.
- **Normalized:** lightly cleaned text with standardized symbols, punctuation, and unified terminology to align synonyms and abbreviations between JDs and CVs.
- **Standardized:** GPT-5–generated structured summaries organizing professional information—such as job titles, experience, education, and skills—into a consistent schema.

For each mode, the hybrid retrieval (BM25 + embedding) produced a ranked candidate pool, which was reranked by a cross-encoder model. Final relevance scores were computed as a linear combination of retrieval and reranker outputs (Equation 4), with $\alpha \in \{0.5, 0.7, 0.8\}$ controlling their relative weight. Candidate pool sizes $\{100, 200, 400\}$ were tested and results evaluated at top-$k \in \{5, 10\}$.

5.2 Evaluation Metrics

The system was assessed using standard information retrieval metrics to capture both ranking accuracy and semantic alignment:

- **Precision@k:** fraction of top-k retrieved CVs judged relevant (≥ 75), measuring shortlist quality.
- **Recall@k:** proportion of all relevant CVs retrieved within the top-k, indicating coverage.
- **MAP (Mean Average Precision):** measures how early relevant CVs appear in the ranked list, emphasizing ranking quality.

- **gNDCG (Generalized Normalized Discounted Cumulative Gain):** accounts for graded relevance (0–100), rewarding correctly ordered highly relevant results.
- **MeanScore:** the average GPT-5 relevance of the top-k CVs, reflecting semantic consistency.

5.3 Results and Discussion

Table 5.3 reports the top-performing configurations under the relevance threshold $t = 75$. Results show clear improvements from text standardization and hybrid reranking.

Mode	α	Cand.	Top-k	gNDCG	MeanScore	Precision	Recall	MAP
Standardized	**0.5**	**200**	**10**	**0.853**	**70.60**	**0.587**	**0.305**	**0.527**
Standardized	0.5	400	10	0.852	70.69	0.583	0.295	0.519
Standardized	0.5	200	5	0.849	72.68	0.631	0.170	0.568
Normalized	0.5	400	10	0.733	61.31	0.463	0.220	0.374
Raw	0.5	100	10	0.727	60.82	0.453	0.218	0.375

Table 1: Top Configurations by Mode (Relevance Threshold = 75)

The system shows significant performance gains from data standardization and hybrid retrieval with re-ranking. Using the standardized mode ($\alpha = 0.5$, 200 candidates, top-10), it achieved gNDCG = 0.85, MAP = 0.53, and Precision@10 = 0.59, demonstrating strong semantic alignment with GPT-5 relevance scores. These metrics respectively show that the model ranks highly relevant candidates near the top (gNDCG), does so consistently across all jobs (MAP), and that roughly six of every ten top-10 results are truly relevant (Precision@10). Compared to normalized and raw modes, standardization improved ranking coherence by 15–30

In summary, the Candidate Matching System demonstrates reliable and explainable performance in large-scale JD–CV matching. The integration of hybrid retrieval, cross-encoder reranking, and GPT-5–based standardization yields a semantically aware framework capable of producing accurate, interpretable, and practically useful candidate rankings.

5.4 Knowledge Graph Evaluation

This section evaluates the performance of the matching models that use Knowledge Graph (KG)–derived CV embeddings together with the similarity scoring approach described in Section 4.2. Three aggregation sizes are examined—$H = 1$, $H = 4$, and $H = 6$—corresponding to fixed-weight, pair-based, and all-pairs configurations. Each setting represents a different structural hypothesis about how Job Description (JD) components interact with the CV representation. The objective of this evaluation is to determine which aggregation structure yields the strongest ranking performance across all JDs.

Model (Description)	H	gNDCG@10	MeanScore@10	Precision@10	Recall@10	MAP@10
Fixed-Weight Heads (H = 1)						
Skills-only	1	0.05784	3.14159	0.03142	0.03142	0.24407
Tools-only	1	0.06390	4.00443	0.04004	0.04004	0.25711
Programs-only	1	0.06238	3.84956	0.03850	0.03850	0.25380
Roles-only	1	0.05702	3.23009	0.03230	0.03230	0.23938
Equal weights	1	0.06182	3.76106	0.03761	0.03761	0.25241
Pair-Based Heads (H = 4)						
Best 4-head pair configuration	4	**0.07230**	**4.53540**	**0.04535**	**0.04535**	**0.28437**
All-Pairs Heads (H = 6)						
All-pairs (equal weights)	6	0.05961	3.36283	0.03363	0.03363	0.25074

Table 2: Knowledge Graph–Based Aggregation Types Results

Among the single-head configurations ($H = 1$), the tools-only head provides the strongest overall performance, outperforming all other single-type heavy heads across every metric.

Within the pair-based group ($H = 4$), the best-performing configuration is corresponds to the following four JD–CV type pairs:

$$(\text{skills, tools}),\ (\text{skills, programs}),\ (\text{skills, roles}),\ (\text{programs, roles}).$$

This configuration achieves the highest performance among all 15 possible 4-pair combinations and also ranks as the best model overall across all experiments. However, despite being the strongest variant, its absolute performance remains relatively low, indicating that even the best structured pair interactions remain insufficient for reliable matching. In addition, we investigated whether KG-based matching could serve as an additional candidate source for the reranking stage. For this experiment, the top 200 candidates retrieved by the hybrid retriever were combined with the top 200 candidates obtained from the KG-based matching (using its best-performing configuration). The union of these two sets was then passed to the reranker and evaluated using the best setting ($\alpha = 0.5$, top-$k = 10$).

The resulting performance remained almost identical to the baseline configuration, with: $\text{gNDCG}_{\text{mean}} = 0.8523$, $\text{MeanScore}_{\text{mean}} = 70.76$, $\text{Precision}_{\text{mean}} = 0.5834$, $\text{Recall}_{\text{mean}} = 0.2950$, and $\text{MAP}_{\text{mean}} = 0.5200$. These values are effectively unchanged from the hybrid-only pipeline, indicating that adding KG-derived candidates did not improve reranking effectiveness.

Therefore, KG outputs were not incorporated into the final production system.

6 Challenges and Limitations

Despite the promising results achieved by the Candidate Matching System, several challenges were encountered during development and evaluation.

1. Data Quality and Alignment. Publicly available CV–JD datasets often contain incomplete or noisy text, inconsistent labeling, and limited domain diversity. Although normalization and standardization mitigated many of these issues, the system's accuracy remains influenced by data heterogeneity and labeling bias.

2. Computational Cost. The hybrid retrieval and reranking pipeline, especially the cross-encoder stage, requires substantial GPU memory and compute time. This limited batch sizes and slowed large-scale evaluation, suggesting the need for more efficient architectures or approximate inference techniques.

3. Domain Adaptation. The pretrained language models (BGE-M3 and BM25s) were not specifically trained on recruitment-related data. Consequently, some domain-specific nuances—such as soft skills, company-specific terminology, or regional job market phrasing—were underrepresented.

4. Evaluation Granularity. The relevance threshold ($t = 75$) and GPT-5–based similarity scores provided a robust evaluation baseline, but human-labeled relevance judgments would offer a more rigorous and subjective validation.

5. Threshold Sensitivity and Metric Choice. While gNDCG/MAP capture graded ranking quality, the binary notion of "relevant" depends on t. Sensitivity analyses across $t \in [70, 85]$, confidence intervals (bootstrap), and significance tests (e.g., paired randomization) would strengthen claims.

7 Conclusion

This work introduced an end-to-end Candidate Matching System that integrates standardized data processing, hybrid retrieval, cross-encoder reranking, and an LLM-based explainability component to produce accurate JD–CV matching results. The multi-stage data preprocessing pipeline—progressing from raw text to normalized forms and finally to GPT-5–standardized summaries—played a crucial role, with the standardized representations consistently producing the strongest results across all evaluation metrics.

The hybrid retrieval module successfully combined sparse BM25 signals with dense BGE-M3 embeddings using Reciprocal Rank Fusion, enabling the system to retrieve a high-recall pool of candidates. The cross-encoder reranker then refined these candidates by modeling detailed interactions between each JD and CV. Together, these components achieved strong performance, with the best configuration (standardized mode, $\alpha = 0.5$, 200 candidates, top-10) reaching gNDCG ≈ 0.85, MeanScore ≈ 70.60, Precision@10 ≈ 0.59, Recall@10 ≈ 0.31, and MAP ≈ 0.53, showing high consistency with GPT-5 relevance scores.

The project also explored a Knowledge Graph–based approach for representing CVs through node-type embeddings, GNN propagation, and non-learned multi-head similarity aggregation. While this approach provided a structured semantic view of candidate information, its ranking performance was significantly lower than that of the hybrid retrieval pipeline. Even when KG-derived candidates were added to the hybrid candidate pool, no measurable improvements were observed. Therefore, KG-based matching was not included in the final system.

Overall, the results demonstrate that combining standardized text representations with hybrid retrieval and cross-encoder reranking forms a reliable and interpretable framework for large-scale candidate–job matching. Although the knowledge graph approach did not improve ranking quality, its exploratory evaluation lays the groundwork for more advanced methods in future work.

8 Future Work

Future extensions of this work can further enhance both performance and interpretability.

1. Domain-Specific Fine-Tuning. Fine-tuning the retrieval and reranking models on recruitment corpora (e.g., industry-specific CVs and JDs) could improve semantic precision, particularly for specialized roles.

2. Dynamic Knowledge Graph Reasoning. Expanding the knowledge graph to include hierarchical skill relationships and company-specific ontologies would enhance contextual inference and job–skill matching.

3. Efficiency and Scalability. Deploying techniques such as approximate nearest neighbor (ANN) search, mixed precision inference, or distillation-based rerankers can reduce latency and computational overhead.

4. Interactive Explainability. Future systems can provide human-in-the-loop interfaces that allow recruiters to query the reasoning process, modify constraints, or simulate alternative candidate rankings.

5. Continuous Learning. Integrating feedback loops where recruiter interactions or hiring outcomes update the model's priors could transform the system into an adaptive, data-driven recommendation engine.

References

B. W. Bayu. 2024. Job-CV Supervised Dataset. https://huggingface.co/datasets/bwbayu/job_cv_supervised. Accessed: 2025-11-11.

C. Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216 [cs.CL]

V. Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal Rank Fusion outperforms Condorcet and individual Rank Learning Methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 758–759. doi:10.1145/1571941.1572114

L. Xing Han Lù. 2024. BM25S: Orders of magnitude faster lexical search via eager sparse scoring. arXiv:2407.03618 [cs.IR] https://arxiv.org/abs/2407.03618

N. Namuangtoun. 2024. Resume-Job-Description-Fit Dataset. https://huggingface.co/datasets/cnamuangtoun/resume-job-description-fit. Accessed: 2025-11-11.

O. OpenAI. 2025. GPT-5 Model. https://openai.com. Accessed: 2025-11-11.

Q. Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

X. Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]