CareerPathKG: Knowledge Graph Integrated Framework for Career Intelligence

Ngoc-Quang Le, Duc Duong Hoang, Thi-Hai-Yen Vuong[*], Mai Vu Tran
VNU University of Engineering and Technology
`{22024510,22028259,yenvth,vutm}@vnu.edu.vn, {quangln,duonghd}@dagoras.io`

Abstract

The labor market is experiencing rapid and continual shifts in required skills and competencies, driven by technological advancement and evolving industry structures. Within this dynamic environment, candidates increasingly face challenges in orienting their career development, requiring them to continuously update their knowledge and capabilities to meet contemporary job requirements; this need is particularly necessary for new entrants to the labor market, who must cultivate a comprehensive understanding of current labor-market conditions. To address these issues, this study proposes an enterprise recruitment framework grounded in a career path knowledge graph, capturing occupations, skill requirements, and career transitions using standardized taxonomies enriched with job-posting data. The framework integrates transformer-based embeddings, large language models, and knowledge-graph reasoning to support efficient and reliable CV assessment, CV-JD matching and career guidance. Data and resources are available at: https://github.com/lengocquanggit255/Tinix-CareerPathKG.

1 Introduction

Ongoing technological advancements are reshaping the labor market, narrowing or transforming many existing occupations while simultaneously creating new ones with evolving skill requirements. These shifts create new opportunities but also make career orientation more challenging, as individuals must continuously update their knowledge to meet emerging job requirements. The challenge is especially significant for new labor-market entrants, who often lack a clear overview of occupational trends, skill demands, and viable career pathways. Meanwhile, relevant information remains dispersed across job postings, training programs, competency frameworks, and labor-market reports, underscoring the need for a unified and interpretable representation to support both job seekers and employers.

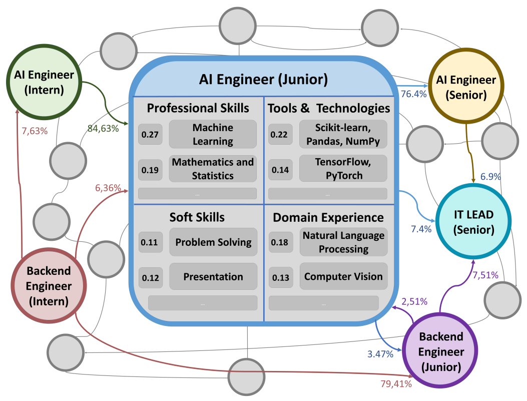

Figure 1: An illustrative example of the career path KG, where job-position nodes (e.g., AI Engineer (Junior)) are linked to weighted skill requirements and connected by edges that indicate career transition probabilities.

Although machine learning (ML) and large language models (LLMs)–based systems have shown promise in tasks such as curriculum vitae (CV) assessment and curriculum vitae–job description (CV-JD) matching, they still face several limitations in this domain. Traditional ML methods often rely on sparse representations, handcrafted features, or keyword centric similarity measures, rendering them brittle when facing semantically diverse job descriptions or heterogeneous skill expressions (Bian et al., 2020). LLMs-based approaches typically encode information in dense vectors without explicit relational structure, making their reasoning opaque and difficult to explain; recent studies also highlight issues of hallucination, inconsistency, and limited grounding when applied to recruitment-related tasks such as candidate ranking or skill extraction (Vaishampayan et al., 2025; Frazzetto et al., 2025). These limitations point to the need for structured, semantically rich representations that can capture explicit relationships between occupations, skills,

[*]Corresponding author.

and career trajectories.

In this study, we introduce the career path knowledge graph (career path KG), a unified framework that is designed to be task-agnostic and career-centric, integrating occupations, required skills, and career transitions into a coherent representation of the labor market. First, we construct a large-scale data acquisition and natural language processing (NLP) pipeline that consolidates standardized taxonomies with real-world job-posting data; this process produces a comprehensive and up-to-date KG capturing the current skill and occupation landscape, as illustrated in Figure 1. Second, we exploit the relational structure of the graph to support multiple downstream tasks, including CV assessment, CV–JD matching, and career guidance, thereby improving both interpretability and prediction accuracy. Moreover, the resulting models exhibit robust performance and computational efficiency suitable for real-world industrial deployment. Third, although our empirical analysis focuses on computer-science occupations to demonstrate feasibility and evaluation, the framework is readily extensible to other domains due to its modular data-integration workflow and domain-agnostic graph schema.

2 Related Work

KG effectively represent entities and their relationships in real-world data. A recent survey systematizes KG techniques into three core phases: extraction, using GNN and Transformer-based methods; learning, to refine embeddings; and evaluation, through intrinsic and extrinsic metrics (Choi and Jung, 2025). Further advancements enhance link prediction and semantic enrichment using methods such as text-based KG completion with constrained zero-shot LLMs (Yang et al., 2024) and relational rotation embeddings (Sun et al., 2019). These studies provide a foundation for building and enriching KG across diverse domains.

In the human resource management domain, prior work has explored machine learning and large language models for CV assessment and CV–JD matching. Beyond traditional keyword-based or handcrafted-feature approaches (Bian et al., 2020), recent studies increasingly investigate knowledge graph–based representations to capture structured relationships between jobs, skills, and learning resources. Job-posting-enriched KGs have been proposed to model skill–job dependencies for skill-based candidate matching (de Groot et al., 2021). JobEdKG integrates job postings with online course data to forecast emerging skill demands and recommend personalized learning pathways (Fettach et al., 2024). More recently, HRGraph leverages LLM-based entity extraction combined with KG reasoning to support job and employee recommendation tasks (Wasi, 2024). These studies demonstrate the effectiveness of structured KGs for modeling evolving labor-market information, but they typically focus on isolated tasks (e.g., matching or recommendation) rather than providing a unified framework that jointly supports CV assessment, CV–JD matching, and career guidance within a single coherent representation.

3 Methodology

Figure 3 illustrates the overall framework, comprising three main components: (1) data collection, (2) career path KG construction, (3) downstream tasks.

3.1 Career Path KG Construction

3.1.1 Data Collection

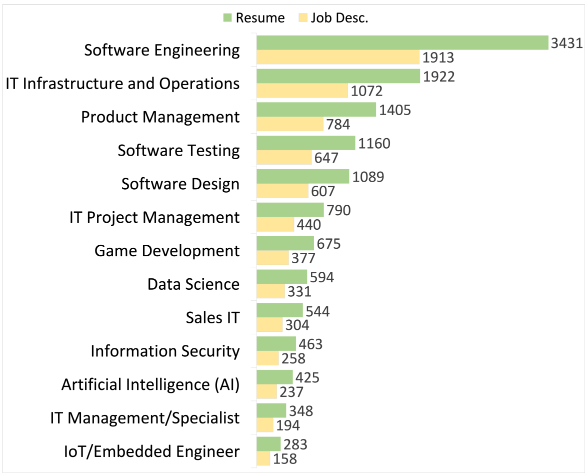

Figure 2: Distribution of CVs and JDs across 13 technology-related job categories.

CVs and JDs were collected from major Vietnamese recruitment platforms and mapped to predefined categories for cross-source consistency. The dataset covers computer-science jobs across 13 categories (Figure 2). CVs were processed using a template-based schema to extract titles, skills, and experience, while JDs were processed to obtain job titles, required skills, and detailed requirements. To ensure privacy, all CVs were manually anonymized by removing personal and educational information, whereas publicly available company information in JDs was retained. Job titles were standardized

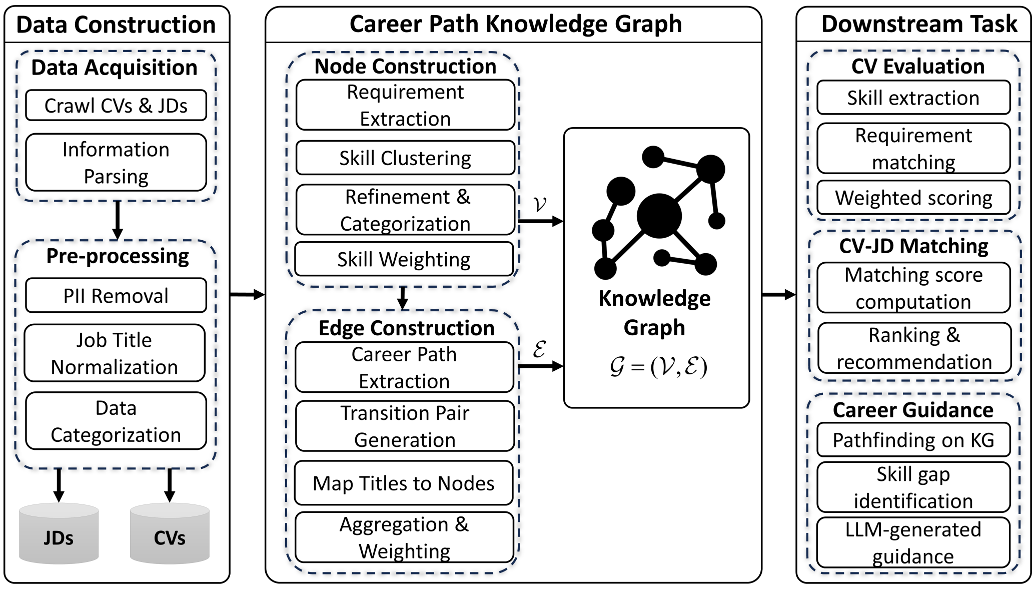

Figure 3: Overview of the proposed framework.

using an LLM-assisted normalization pipeline that maps heterogeneous title expressions to a unified taxonomy by resolving lexical variants, abbreviations, and semantically equivalent roles. Standardized positions were then assigned to one of five predefined career levels (`Intern`, `Junior`, `Middle`, `Senior`, and `Expert`) based on role descriptions, responsibilities, and experience indicators. The resulting dataset comprises 13,129 CVs and 7,322 job descriptions.

3.1.2 Graph Schema

We define the career path KG as a weighted directed graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_n\}$ is the set of nodes, $\mathcal{E} = \{e_1, \dots, e_m\}$ is the set of directed weighted edges. Each node $v_i = (t_{v_i}, l_{v_i}, S_{v_i})$ consists of a job title t_{v_i}, a career level l_{v_i}, and an associated set of representative requirements $S_{v_i} = \{(r^k_{v_i}, c^k_{v_i}, w^k_{v_i})\}_{k=1}^{K_{v_i}}$, each triplet contains a requirement phrase $r^k_{v_i}$, a predefined category $c^k_{v_i} \in \mathcal{C}$ derived from a data-driven analysis of common requirement types observed across a large corpus of CVs, JDs, and a normalized importance weight $w^k_{v_i} \in [0, 1]$ (see Section 3.1.3). Here, K_{v_i} denotes the number of requirement items associated with node v_i. A directed edge represents a weighted transition $e_j = (v_a, v_b, w_{ab})$, where $v_a, v_b \in \mathcal{V}$ denote a career transition from v_a to v_b, and $w_{ab} \in [0, 1]$ quantifies its empirical strength or frequency (see Section 3.1.4).

3.1.3 Node Construction

For each job node v_i with a job title t_{v_i} and career level l_{v_i} taken from a predefined list, we construct its requirement set S_{v_i} as follows. We first retrieve all JDs in the corpus matching (t_{v_i}, l_{v_i}). From this subset, we then extract a set of requirement phrases $\mathcal{J}_{v_i} = \{j_1, \dots, j_{n_i}\}$, where n_i is the number of extracted phrases. JDs for the same title and career level often contain overlapping, redundant, and unclear requirement phrases, making the extracted set $\mathcal{J}_{v_i}$ noisy and repetitive. To reduce this redundancy, we cluster semantically similar phrases and derive a representative requirement for each cluster. First, we embed each phrase j_k into a vector $\mathbf{e}_k \in \mathbb{R}^d$ and apply Agglomerative Clustering (Müllner, 2011) to group these embeddings into requirement clusters $C_{v_i} = \{c^1_{v_i}, \dots, c^{K'_{v_i}}_{v_i}\}$, where $K'v_i$ is automatically determined by a distance threshold. For each cluster $c^p_{v_i} \in C_{v_i}$, We then apply an encoder–decoder summarization model to generate a concise representative requirement $r^p_{v_i}$ from the cluster phrases, yielding the set $R_{v_i} = \{r^1_{v_i}, \dots, r^{K'_{v_i}}_{v_i}\}$, where $r^p_{v_i}$ represents the requirement for cluster $c^p_{v_i}$. To further refine this representation, we employ a LLM to filter out requirements $r^p_{v_i} \in R_{v_i}$ that are irrelevant or redundant for node v_i, and to assign each remaining requirement $r^q_{v_i}$ to one of the predefined requirement categories in the category set $\mathcal{C} = \{c_1, \dots, c_M\}$. We denote the resulting categorized requirement set $S_{v_i} = \{(r^k_{v_i}, c^k_{v_i}, w^k_{v_i})\}_{k=1}^{K_{v_i}}$, where $c^k_{v_i} \in \mathcal{C}$ is the category label of requirement $r^k_{v_i}$ and $K_{v_i} \leq K'_{v_i}$.

We estimate the importance weights $w_{v_i}^k$ from historical CV data. Let $\mathcal{U}_{v_i}$ denote the set of CVs mapped to node v_i based on its job title t_{v_i} and career level l_{v_i}. For each CV $u \in \mathcal{U}_{v_i}$, let $\mathcal{R}_u = \{s_1, \ldots, s_{m_u}\}$ be its extracted skill set. We use a transformer-based binary classifier $f_{\text{match}}(r_{v_i}^k, s) \in \{0,1\}$ to determine whether a skill $s \in \mathcal{R}_u$ satisfies requirement $r_{v_i}^k$. The importance weight is then defined as the empirical frequency with which $r_{v_i}^k$ is satisfied across CVs in $\mathcal{U}_{v_i}$:

$$w_{v_i}^k = \frac{1}{|\mathcal{U}_{v_i}|} \sum_{u \in \mathcal{U}_{v_i}} \mathbb{I}\Big(\exists\, s \in \mathcal{R}_u : f_{\text{match}}(r_{v_i}^k, s) = 1\Big)$$

where $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition inside is true and 0 otherwise, yielding $w_{v_i}^k \in [0,1]$ for all k.

3.1.4 Edge Construction

Statistic	Count
Number of CVs	13129
Number of Job Descriptions	7322
Avg. Skills per CV	15
Avg. Requirements per JD	8
Distinct Job Titles	71
Career Levels	5
Graph Nodes ($\|\mathcal{V}\|$)	355
Graph Edges ($\|\mathcal{E}\|$)	6254
Average Out-degree	8.16
Average Skills per Node	15.10

Table 1: Dataset and graph statistics.

Each CV lists a chronological sequence of work experiences, with each entry mapped to a node $v \in \mathcal{V}$. The career trajectory is thus represented as an ordered sequence $\mathcal{P} = (u_1, u_2, \ldots, u_m)$, where each $u_k \in \mathcal{V}$ and u_1 denotes the earliest position while u_m is the most recent one. A directed edge (u_k, u_{k+1}) is created whenever a transition between two consecutive positions occurs within a CV. Aggregating all such transitions across the entire collection yields the total number of observed transitions between any two nodes:

$$N_{ij} = \big|\{\, \mathcal{P} \mid \exists k : (u_k, u_{k+1}) = (v_i, v_j) \,\}\big|.$$

The transition weight for edge (v_i, v_j) is computed by row-normalizing the transition counts:

$$w_{ij}^{(t)} = \frac{N_{ij}}{\sum_{k=1}^{|\mathcal{V}|} N_{ik}},$$

The overall statistics of the constructed career graph are summarized in Table 1.

3.2 Downstream Task

Our framework supports three downstream tasks– CV assessment, CV–JD matching, and career guidance—using methods specifically optimized for real–time inference.

3.2.1 CV Assessment

Given a target job node v_i, we assess the suitability of a CV u based on its extracted skill set $\mathcal{R}_u = \{s_1, \ldots, s_{m_u}\}$. The assessment procedure is summarized in Algorithm 1.

Algorithm 1 CV Assessment Algorithm

```
Input: CV skill set $\mathcal{R}_u$, requirement set $S_{v_i}$
Output: $\text{Score}(\mathcal{R}_u, v_i)$
 1: $\text{Score} \leftarrow 0$
 2: for each requirement triplet $(r_{v_i}^k, c_{v_i}^k, w_{v_i}^k) \in S_{v_i}$ do
 3:     $f_{\text{req}}(r_{v_i}^k) \leftarrow \max_{s \in \mathcal{R}_u} f_{\text{match}}(r_{v_i}^k, s)$
 4: end for
 5: for each category $c_i \in \mathcal{C}$ do
 6:     $S_{v_i}^{(c_i)} \leftarrow \{(r_{v_i}^k, c_{v_i}^k, w_{v_i}^k) \in S_{v_i} \mid c_{v_i}^k = c_i\}$
 7:     $\text{Score}_{c_i} \leftarrow \sum_{(r_{v_i}^k, c_{v_i}^k, w_{v_i}^k) \in S_{v_i}^{(c_i)}} w_{v_i}^k f_{\text{req}}(r_{v_i}^k)$
 8:     $\text{Score} \leftarrow \text{Score} + \alpha_{c_i} \cdot \text{Score}_{c_i}$
 9: end for
10: return $\text{Score}(\mathcal{R}_u, v_i)$
```

For each requirement triplet $(r_{v_i}^k, c_{v_i}^k, w_{v_i}^k) \in S_{v_i}$, the skill-matching model $f_{\text{match}}(r_{v_i}^k, s)$ measures the semantic similarity between the requirement $r_{v_i}^k$ and a CV skill $s \in \mathcal{R}_u$. The best alignment score for requirement $r_{v_i}^k$ is then:

$$f_{\text{req}}(r_{v_i}^k) = \max_{s \in \mathcal{R}_u} f_{\text{match}}(r_{v_i}^k, s).$$

Let $\mathcal{C} = \{c_1, \ldots, c_M\}$ denote the predefined requirement categories, and let α_{c_i} be the importance weight assigned to category $c_i \in \mathcal{C}$. For each category $c_i \in \mathcal{C}$, let $S_{v_i}^{(c_i)} = \{(r_{v_i}^k, c_{v_i}^k, w_{v_i}^k) \in S_{v_i} \mid c_{v_i}^k = c_i\}$ denote the subset of requirements in S_{v_i} belonging to category c_i, with node-level requirement weights $w_{v_i}^k$ obtained during node construction. The overall matching score between the CV u and the target job node v_i is:

$$\text{Score}(\mathcal{R}_u, v_i) = \sum_{c_i \in \mathcal{C}} \alpha_{c_i} \sum_{(r_{v_i}^k, c_{v_i}^k, w_{v_i}^k) \in S_{v_i}^{(c_i)}} w_{v_i}^k f_{\text{req}}(r_{v_i}^k).$$

3.2.2 CV–JD Matching

This component links the information extracted from CVs and JDs to produce job and candidate recommendations. Given a candidate CV u with its extracted skill set $\mathcal{R}_u$ and a target job node v_i, their compatibility is measured by the matching score $\text{Score}(\mathcal{R}_u, v_i)$ as defined in the previous section.

To recommend jobs for a given CV u, we compute $\text{Score}(\mathcal{R}_u, v_i)$ for all job nodes $v_i \in \mathcal{V}$ and rank these nodes in descending order of their scores; the top-ranked nodes are returned as job recommendations for u. Conversely, to recommend candidates for a given job node v_i, we compute $\text{Score}(\mathcal{R}_u, v_i)$ for all CVs in the candidate pool and rank them in descending order, returning the highest-scoring CVs as recommendations for v_i.

3.2.3 Career Guidance

Given a candidate with current position v_c and target position v_t, we first identify a feasible transition path in the career path KG. The optimal path is:

$$Q^* = \arg\max_{P} \prod_{(v_i,v_j)\in\text{Edges}(P)} w_{ij},$$

where P ranges over all valid paths from v_c to v_t and w_{ij} denotes the transition weight between nodes v_i and v_j. Let S_{v_c} and S_{v_t} be the refined requirement sets of v_c and v_t, respectively. The skill gap is defined as the set of requirements in the target role that are not covered in the current role

$$\Delta S = S_{v_t} \setminus S_{v_c}.$$

Finally, the LLM produces a guidance output:

$$G = G(\Delta S, Q^*),$$

which explains missing requirements, recommends skill development, and justifies the suggested transitions toward the target role v_t. The prompting template is provided in Appendix C.3.

4 Evaluation & Discussion

4.1 Experimental Setup

We evaluate our method on a set of 100 anonymized CVs collected from real recruitment data. All models share the same preprocessing pipeline. Ground-truth annotations and qualitative feedback are provided by 5 HR experts with at least five years of experience in recruiting and evaluating technical candidates. Additional evaluation details are reported in Appendix B.

The transformer-based binary classifier f_{match} is implemented by fine-tuning PhoBERT (Nguyen and Nguyen, 2020). Training data are constructed by pairing CV skills with JD requirements. Each skill–requirement pair is first labeled by a LLM and then reviewed and, if necessary, corrected by HR experts[1]. The encoder–decoder summarization model used in the node construction step is based on T5 (Raffel et al., 2020). To obtain training data, we group semantically similar requirement phrases into clusters, use a LLM to generate concise summaries for each cluster, and then ask HR experts to validate and refine these summaries. We also release this summarization dataset.[2] All LLM-based components in our pipeline are implemented using Qwen2.5-14B (Qwen et al., 2025). For completeness, detailed configurations of all models and prompts used in our system are provided in Appendix A.2 and Appendix C.

4.2 Results

We evaluate our framework across three downstream tasks–CV assessment, CV-JD matching, and career guidance–to capture its overall capability in understanding, recommending, and advising within real-world recruitment contexts. All results are compared against strong LLM baselines, as summarized in Table 2.

Model	RMSE (↓)	Recall@10 (↑)	Satisfaction (↑)
GPT-5	12.4	68.9	4.2
Gemini-2.5-Pro	11.7	70.4	4.3
Claude 4.5	12.1	67.8	4.1
Career path KG	**8.9**	**78.3**	**4.6**

Table 2: Quantitative comparison across three evaluation tasks. Lower RMSE and higher Recall@10/Satisfaction indicate better performance.

4.2.1 CV Assessment

This task assesses the degree to which each model aligns with human judgment. HR experts assign quality scores to CVs on a 0–100 scale. To examine the consistency and variability of these expert evaluations, Figure 4 illustrates the score distribution across the 100 assessed CVs. Model performance is measured using the RMSE between predicted scores and expert ratings. GPT-5, Gemini-2.5-Pro, and Claude 4.5 achieve RMSE values of 12.4, 11.7, and 12.1, respectively, whereas our

[1] https://huggingface.co/datasets/lengocquangLAB/Tinix-req-skill-matching

[2] https://huggingface.co/datasets/lengocquangLAB/Tinix-req-sum

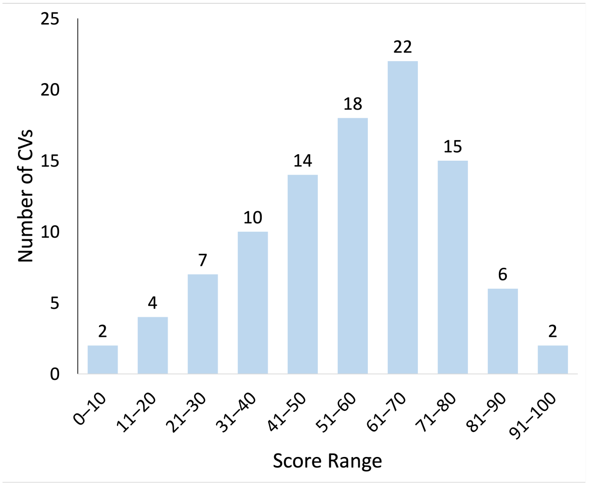

Figure 4: Score distribution for the CV assessment task across all evaluated CVs.

framework attains the lowest RMSE of 8.9, demonstrating the closest alignment with expert assessments. As reported in Table 3, our model also exhibits significantly lower inference time per CV than large commercial LLMs, while simultaneously delivering superior accuracy.

Model	**CV Assessment (s)**	**CV-JD Matching (s)**
GPT-5	15.3	13.9
Gemini-2.5-Pro	32.2	22.4
Claude 4.5	26.2	3.2
Career path KG	**4.7**	**0.24**

Table 3: Average inference time per query for CV assessment and CV–JD matching.

4.2.2 CV-JD Matching

We measure the ability of each system to retrieve suitable job descriptions for a given CV using `Recall@10`. GPT-5, Gemini-2.5-Pro, and Claude 4.5 achieve `Recall@10` scores of 68.9, 70.4, and 67.8, respectively, while our framework attains the highest score of 78.3, demonstrating superior capacity in modeling career-level compatibility via graph-based representations. As shown in Table 3, our model achieves lower inference time (0.24s per query) than all LLM-based baselines. These results highlight the benefit of integrating graph-structured knowledge, which guides the retrieval process toward semantically coherent roles and reduces unnecessary exploration over irrelevant job categories.

4.2.3 Career Guidance

In the interactive guidance task, experts rate the usefulness and accuracy of career suggestions on a 1–5 satisfaction scale. GPT-5, Gemini-2.5-Pro, and Claude 4.5 receive average satisfaction scores of 4.2, 4.3, and 4.1, respectively, whereas our model achieves the highest score of 4.6, showing that it provides more realistic and actionable recommendations for career development.

4.3 Error Analysis

Our framework supports three downstream tasks, namely CV assessment, CV–JD matching, and career guidance, each of which demonstrates characteristic error patterns arising from different sources of model limitations. Table 4 provides a systematic summary of these errors, detailing their underlying causes and illustrating them with representative real-world examples.

For the CV assessment task, errors primarily result from semantic over-matching and imbalanced weighting across requirement categories. Specifically, the model may rely excessively on surface-level keyword overlap, leading to inflated similarity scores for skills that are lexically related but semantically distinct. In addition, frequent or tool-centric skill categories may dominate the final evaluation score, thereby overshadowing less common but semantically critical requirements, such as architectural or design-related competencies.

In the CV-JD matching task, failures are mainly attributed to role ambiguity and insufficient modeling of soft skills. Certain technical skills are inherently transferable across multiple job families, which can cause the system to assign similar relevance scores to fundamentally different roles. Moreover, soft skills are often described implicitly through activities or responsibilities rather than explicit terminology, making them difficult to align accurately using text-based matching alone.

For the career guidance task, errors typically stem from oversimplified transition patterns and abstract competency requirements. The system tends to favor statistically plausible career paths without fully verifying hidden prerequisites, such as leadership experience or cross-functional exposure. Furthermore, high-level requirements are often weakly grounded in concrete skill clusters, preventing the model from identifying precise skill gaps between a candidate's current profile and a target role.

Overall, these issues highlight the difficulty

Table 4: Error analysis and representative cases of downstream tasks

Error Type	Cause	Representative Examples
CV assessment		
Semantic over-matching	Model relies on shared keywords instead of true meaning.	Requirement: "Experience with **CI/CD pipelines**." CV: "Worked with data **pipeline** processing." → Overlap "pipeline" leads to incorrect high similarity.
Category imbalance	High weights for some categories overshadow important rare skills.	Requirement: "Knowledge of **backend API design**." CV: "**RESTful API design**" listed among tool skills. → Tool-heavy categories dominate the final score.
CV-JD Matching		
Role ambiguity	A skill can match multiple job families, causing unstable ranking.	CV: "Good with **Data modeling.**" Job Titles: Backend Engineer / Data Analyst. → Assigns similar scores to unrelated roles.
Missing soft-skill matching	Soft skills in JD are vague or implicit, reducing match accuracy.	Requirement: "Have good **communication with teams**." CV: "Have **managed weekly sprint meetings**." → Fails to link the expressions.
Career Guidance		
Simplified career path	System favors probable transitions without checking hidden prerequisites.	Suggested: Intern → Data Engineer → Product Manager. Context: Lacks leadership or product experience. → Path ignores real managerial requirements.
Missing skill-gap details	Abstract requirements do not map clearly to skill clusters.	Target: "**Lead architectural decisions**." Current: "Implemented API modules." → Cannot infer leadership and architecture gaps.

of aligning heterogeneous human-written content with structured knowledge representations, suggesting that text-only signals are insufficient to capture nuanced professional contexts.

5 Conclusion

This study presents a unified recruitment framework built on a career path knowledge graph, which provides a structured, transparent, and interpretable representation of occupations, required skills, and career transitions within the labor market. By explicitly modeling career-related entities and their relationships, the proposed framework bridges the gap between unstructured recruitment data and structured reasoning, enabling more reliable and explainable decision-making in downstream recruitment tasks. We develop a standardized skill ontology and integrate it with transformer-based embeddings to capture both semantic similarity and explicit relational structure, thereby supporting effective CV–JD matching. Beyond matching, the framework naturally extends to practical functionalities such as CV assessment and career guidance, demonstrating its flexibility across multiple recruitment scenarios. Extensive experimental results show that our approach consistently improves the alignment between candidate profiles and job requirements across these downstream tasks, while maintaining robustness and computational efficiency. Overall, the proposed framework offers a practical, extensible, and interpretable solution for intelligent recruitment systems. Its task-agnostic design and modular construction allow straightforward adaptation to other occupational domains and evolving labor market conditions, highlighting its potential for real-world industrial deployment and future research on structured, knowledge-driven recruitment technologies.

Limitations

The proposed framework is subject to certain limitations. Our empirical evaluation is primarily based on end-to-end comparisons with LLMs using a relatively small dataset of 100 CVs. While informative, this setup does not fully quantify the contributions of individual components such as the KG construction or LLM-based normalization. Additionally, ablation studies and comparisons with strong graph-based baselines are not yet performed. Future work will expand the evaluation to larger and more diverse datasets, incorporate ablation studies to assess each pipeline component, and benchmark against both LLM and non-LLM strong baselines.

Our framework has not yet been systematically compared to prior approaches that combine KGs with contextual embeddings, leaving the distinct advantages of LLM-enabled reasoning underexplored. In future work, we plan to conduct controlled comparisons to isolate the added value of dynamic KG updates and LLM-based normalization relative to previous embedding-based methods.

The reliance on LLMs for preprocessing introduces challenges in controlling for biases in skill extraction and in quantifying error propagation across pipeline stages. Small inconsistencies in extracted skills can accumulate, affecting both KG quality and downstream reasoning. Future work will explore bias-aware extraction techniques and methods to measure and mitigate error propagation, such as confidence scoring, cross-validation among extractors, and constraints derived from the KG taxonomy.

The current methodology lacks extensive evaluation on out-of-domain jobs, novel skills, and emerging career paths. Consequently, its ability to generalize across industries and to suggest previously unconsidered career directions remains uncertain. Further investigation is needed into incremental KG expansion and long- and short-term career path recommendations beyond candidates' initial expectations.

References

Shuqing Bian, Xu Chen, Wayne Xin Zhao, Kun Zhou, Yupeng Hou, Yang Song, Tao Zhang, and Ji-Rong Wen. 2020. Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network. *Preprint*, arXiv:2009.13299.

Seungmin Choi and Yuchul Jung. 2025. Knowledge graph construction: Extraction, learning, and evaluation. *Applied Sciences*.

Maurits de Groot, Jelle Schutte, and David Graus. 2021. Job posting-enriched knowledge graph for skills-based matching. *Preprint*, arXiv:2109.02554.

Yousra Fettach, Adil Bahaj, and Mounir Ghogho. 2024. Jobedkg: An uncertain knowledge graph-based approach for recommending online courses and predicting in-demand skills based on career choices. *Engineering Applications of Artificial Intelligence*, 131:107779.

Paolo Frazzetto, Muhammad Uzair Ul Haq, Flavia Fabris, and Alessandro Sperduti. 2025. Graph neural networks for candidate-job matching: An inductive learning approach: P. frazzetto et al. *Data Science and Engineering*, pages 1–18.

Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. Phobert: Pre-trained language models for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1037–1042, Online. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *Preprint*, arXiv:1902.10197.

Swanand Vaishampayan, Hunter Leary, Yoseph Berhanu Alebachew, Louis Hickman, Brent A Stevenor, Weston Beck, and Chris Brown. 2025. Human and llm-based resume matching: An observational study. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4808–4823.

Azmine Toushik Wasi. 2024. Hrgraph: Leveraging llms for hr data knowledge graphs with information propagation-based job recommendation. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, page 56–62. Association for Computational Linguistics.

Rui Yang, Jiahao Zhu, Jianping Man, Li Fang, and Yi Zhou. 2024. Enhancing text-based knowledge graph completion with zero-shot large language models: A focus on semantic enhancement. *Preprint*, arXiv:2310.08279.

A Training Environment and Hyperparameter Configurations

This appendix provides the hardware setup and key hyperparameter configurations used to train and evaluate all models in this study.

A.1 Hardware and Software Environment

Experiments are conducted on the following environment:

- CPU: 2x vCPU Intel Xeon (2.20GHz)
- GPU: NVIDIA GeForce RTX 2080 Ti (12GB VRAM)
- RAM: 32 GB

A.2 Model Settings

A.2.1 Embedding Model

- Pretrained checkpoint: `all-MiniLM-L6-v2`
- Number of parameters: 22M
- Maximum sequence length: 256
- Embedding dimension: 384

A.2.2 LLM Reasoning Model

- Pretrained checkpoint: `Qwen2.5-14B-Instruct`
- Number of parameters: 14B
- Maximum context length: 32,768 tokens
- Usage: Zero-shot and few-shot reasoning for summarization refinement, matching justification, and career guidance

A.2.3 Requirements Summarization Model

- Pretrained checkpoint: `google-t5/t5-large`
- Number of parameters: 770M
- Learning rate: 2×10^{-4}

A.2.4 Matching Model

- Pretrained checkpoint: `vinai/phobert-large`
- Number of parameters: 135M
- Learning rate: 2×10^{-5}

A.3 Training Time

All models were trained on a single NVIDIA GeForce RTX 2080 Ti with gradient accumulation to fit large-batch updates. The requirements summarization model required approximately 4 hours for 8 epochs, while the matching model also completed in about 4 hours. Overall, the full training pipeline took roughly 8 hours of compute time.

B Evaluation Details

B.1 Evaluation Protocol

B.1.1 CV Assessment

For the CV assessment task, each CV is processed by the proposed framework to produce a continuous suitability score that captures the candidate's overall alignment with the targeted career level. In parallel, HR experts independently assign ground-truth scores using a standardized evaluation rubric that accounts for experience, skill coverage, and role relevance. The model's predictions are evaluated against the averaged expert scores using the RMSE.

B.1.2 CV–JD Matching

In the CV–JD matching task, each CV is used as a query to retrieve a ranked list of job descriptions from the candidate pool. Relevant job descriptions are identified based on expert validation, taking into account role suitability and career-level compatibility. The system's performance is evaluated using `Recall@10`, which measures whether at least one expert-validated relevant job description appears among the top ten retrieved results. This protocol reflects realistic recruitment scenarios in which recruiters typically review only a limited number of top-ranked candidates or positions.

B.1.3 Career Guidance

For the career guidance task, the framework generates personalized career recommendations and potential transition paths for each CV, conditioned on the candidate's current role, skills, and inferred career stage. These recommendations are presented to HR experts in an interactive setting. Each expert independently evaluates the usefulness and accuracy of the suggested guidance using a 1–5 satisfaction scale, where higher scores indicate better alignment with the candidate's profile and more actionable career advice. Final satisfaction scores are obtained by averaging ratings across experts, providing a qualitative yet structured assessment of the framework's practical value in career guidance scenarios.

C Prompts

C.1 Matching Requirement–Skill

You are an expert in analyzing IT-related skills. Your task is to compare a job requirement from a job description (jd_requirement) with a CV skill (cv_skill), and classify their relationship into one of the following labels.

0 - Disjoint:
The two items belong to different domains, share no meaningful overlap, and cannot substitute for each other.

1 - Related:

- Partial semantic overlap.
- One concept is broader and the other is a subtype.
- Semantic containment in either direction.

Input (five pairs):

- Pair 1: "{jd_req_1}" vs "{cv_skill_1}"
- Pair 2: "{jd_req_2}" vs "{cv_skill_2}"
- Pair 3: "{jd_req_3}" vs "{cv_skill_3}"
- Pair 4: "{jd_req_4}" vs "{cv_skill_4}"
- Pair 5: "{jd_req_5}" vs "{cv_skill_5}"

C.2 Requirements Summarization

You are a professional text analysis expert. Given a list of skill-related sentences from the same cluster, your task is to generate an abstractive summary that captures the core representative skill.
Instructions:

- Produce an abstractive summary; do not copy text verbatim.
- Select a single core skill that best represents the cluster.
- If multiple synonymous expressions appear, keep only the most common form.
- Standardize software tools using the pattern "use [tool]".
- Do not use parentheses, explanations, comments, or additional notes.
- Output only one concise line.

Input skill list: {skill_list}
Output format: a single concise line.

C.3 Career Guidance

You are an expert IT career advisor using a Career Path Knowledge Graph (CP-KG). Your task is to generate a complete and actionable career-guidance output based on the following inputs:

- current_role: the candidate's current job title and level.
- target_role: the desired job title and level.
- optimal_path: a list of role transitions recommended by the CP-KG.
- skill_gap: a list of missing requirements needed to qualify for the target role.

Instructions:

- **Career Path Interpretation** Convert the list in optimal_path into a natural-language explanation. Describe why each transition is realistic and what the candidate gains at each step.
- **Skill-Gap Analysis** Group the items in skill_gap (technical skills, tools, soft skills, domain knowledge, etc.). Explain why each group matters for the target_role.
- **Recommended Learning Roadmap** For each missing skill, propose concrete learning actions such as topics to study, practice tasks, recommended resources, certification options, or project ideas.
- **Final Guidance Summary** Provide a short, clear summary describing what the candidate should focus on next and a realistic timeline for progressing from current_role to target_role.

Use a professional, concise, and supportive tone.