Universal Library of Engineering Technology **Research Article**

ISSN: 3064-996X | Volume 2, Issue 4

Open Access | PP: 22-28

DOI: https://doi.org/10.70315/uloap.ulete.2025.0204004

Recent Advances in Machine Learning Algorithms for Candidate–Job Matching

Roman Ishchenko, PhD

Raised Networks Inc., USA.

Abstract

The candidate–job matching (CJM) problem, central to high-skill recruitment in domains like technology, management, and finance, has seen rapid progress through machine learning (ML) since 2021. Modern systems move beyond simple keyword matching, leveraging advanced natural language processing (NLP), graph representations, and hybrid recommender methods. Transformer-based models (e.g. BERT and derivatives) now embed resumes and job descriptions into semantic spaces, enabling nuanced similarity comparisons. Graph neural networks (GNNs) capture rich relationships among candidates, skills, and jobs, often outperforming traditional neural models in screening tasks. Classical ML approaches (e.g. support vector machines, tree ensembles) remain useful for structured feature matching but are complemented by deep models for unstructured text. Recommender-system techniques – including collaborative filtering, content-based filtering, and hybrid designs – incorporate contextual signals (experience, industries, user behaviors) to improve personalization. Reviewed benchmarks report that fine-tuned transformers and GNNs can significantly boost ranking accuracy (e.g. ~15% NDCG improvements [1]) and screening sensitivity (e.g. GNN balanced accuracy 65.4% vs 55.0% for a plain MLP [2]). These gains come with challenges: neural approaches often act as black boxes, raising interpretability concerns, and large models incur high computational costs that demand scalable architectures (e.g. bi-encoder retrieval with cross-encoder re-ranking in multi-stage pipelines). Bias mitigation has become critical; domain-specific models have been shown to yield fairer outcomes than off-the-shelf large language models. This review surveys recent (2021–2025) peer-reviewed work on CJM, covering algorithmic approaches (SVMs, ensemble trees, Siamese and cross-encoder transformers, GNNs, and hybrid recommenders), model architectures, input representations (resumes, job text, skill ontologies), and evaluation methods. We synthesize experimental findings from academic studies, discussing strengths and limitations of each approach, including accuracy, robustness, interpretability, and fairness. Finally, we highlight open challenges and directions for making CJM more transparent and equitable while maintaining scalability in practice.

Keywords: *Candidate–Job Matching; Graph Neural Networks; Machine Learning; Recommendation Systems; Transformer Models.*

INTRODUCTION

Recruitment in various sectors (technology, management, finance, etc.) increasingly relies on automated systems to filter large applicant pools. Traditionally, candidate screening and job matching depended on manual resume review or simple keyword matching in Applicant Tracking Systems (ATSs). However, manual screening is time-consuming and prone to human bias [2]. The massive growth of online job postings and digital resumes has motivated advanced machine learning (ML) approaches for candidate–job matching (CJM). Early automated methods used rule-based or keyword-based matching, which often miss latent semantic alignments and poorly handle varied language in resumes and job descriptions [1]. By contrast, modern ML and AI techniques can learn to interpret unstructured text and structured profile data jointly, yielding more accurate and scalable matching.

Recent breakthroughs in NLP, especially transformer architectures (BERT, RoBERTa, etc.), have enabled deeper semantic understanding of textual data. Systems now encode resumes and job descriptions into embedding vectors in a shared latent space, allowing cosine-similarity ranking or learned matching scores. For example, the *CareerBERT* system fine-tuned a sentence-BERT (SBERT) model in a Siamese network to project resumes and standardized job titles into the same space, outperforming traditional keyword methods. Beyond text, researchers are constructing richer data representations: graph-based models capture

Citation: Roman Ishchenko, "Recent Advances in Machine Learning Algorithms for Candidate–Job Matching", Universal Library of Engineering Technology, 2025; 2(4): 22-28. DOI: https://doi.org/10.70315/uloap.ulete.2025.0204004.

relationships among candidates, skills, and roles, enabling candidates to match to jobs via multi-hop semantic paths [2]. Hybrid recommender approaches integrate content similarity with collaborative signals (e.g. historical application patterns), sometimes augmented with contextual features like geolocation or industry trends [3].

This review critically surveys ML methods developed between 2021 and 2025 for candidate–job matching. We focus on high-skill domains but note many techniques apply broadly. We detail algorithmic approaches (traditional ML, deep learning, recommender systems), model architectures (e.g. Siamese encoders vs cross-encoders, GNN variants), data inputs and outputs (resume text, job descriptions, skills), and evaluation protocols (ranking and classification metrics). Key performance results from academic benchmarks (peer-reviewed journals and conferences) are summarized. We also discuss interpretability and ethical considerations – notably how ML choices affect bias and fairness – and the scalability of these models in real-world recruiting scenarios. The survey highlights emerging strengths (semantic matching, graph reasoning) and remaining challenges (data quality, opacity, computational cost) to guide future research on fair, efficient, and accurate candidate–job matching.

MATERIALS AND METHODS

This study employs a systematic review approach to examine recent advances (2021–2025) in machine learning algorithms applied to candidate–job matching (CJM). The objective was to synthesize peer-reviewed research addressing algorithmic innovations, model architectures, and empirical outcomes in automated recruitment systems. The review followed a PRISMA-inspired conceptual structure, ensuring methodological transparency and reproducibility while emphasizing analytical depth and technical relevance to high-skill sectors such as technology, management, and finance.

Each selected paper was reviewed for methodological rigor and categorized according to algorithmic type — including traditional ML, transformer-based NLP, graph-based, recommender, and hybrid architectures. Key data extracted from each study included the model architecture, type of input data (e.g., resumes, job descriptions, or skills ontologies), evaluation methodology, and reported performance metrics such as accuracy, F1-score, AUC, NDCG, and MAP. This categorization enabled cross-study comparison of strengths and limitations across algorithm families, particularly regarding interpretability, scalability, and bias handling.

Data synthesis employed a comparative and narrative analysis framework rather than meta-analysis, due to dataset heterogeneity. Reported results were grouped by model category and benchmarked against traditional baselines to highlight relative improvements. Additional validation was conducted by verifying publication credibility, cross-referencing open-source implementations, and reviewing industrial-scale systems. This integrative methodology ensured a balanced perspective between academic research and applied AI systems in recruitment.

Key Algorithms and Frameworks

This section summarizes the core algorithms and data representations used in modern CJM systems. We organize the methods into major categories and describe typical architectures.

Inputs and Outputs: CJM systems generally take as input a candidate profile (often a resume or CV) and a job description (JD). Resumes and JDs are primarily unstructured text but may include structured fields (skills, education, experience). Preprocessing often involves parsing resumes (e.g. skill/experience extraction), and text normalization (tokenization, lemmatization). Inputs are encoded into feature vectors or embeddings. The output is a compatibility score or ranking: either a match score between one candidate and one job, or a list of top-k job recommendations for a candidate (or vice versa).

Traditional Machine Learning Approaches

Classical ML models have been adapted for CJM by engineering features from resumes and job specs. For example, bag-of-words or TF-IDF vectors of skill keywords can feed into linear models or tree ensembles. Supervised classifiers (e.g. Support Vector Machines, Random Forests, XGBoost) have been trained to predict candidate suitability. These models rely on handcrafted features such as skill overlaps, similarity of standardized attributes, or applicant metadata. Due to their transparency, methods like decision trees offer some interpretability. However, they struggle with the rich semantics of free text and require extensive feature engineering. Still, gradient-boosted trees (e.g. XGBoost) and LSTMs (for sequence modeling) are cited among common methods in job recommender research [3]. Some systems apply ranking SVMs or pairwise ranking forests to score candidates for a given job. These methods typically use metrics such as precision/recall or AUC for evaluation.

Transformer-based NLP Models

A major trend is using transformer neural networks to encode text. Bidirectional Encoder Representations (BERT) and its variants (RoBERTa, DistilBERT) are fine-tuned on CJM tasks. Two architectural patterns prevail:

- Siamese neural networks (two-tower bi-encoders) project job descriptions and résumés into a shared embedding space so that semantically compatible pairs lie close together while incompatible pairs are far apart. In a practical CJM setup, each tower comprises (i) a multilingual sentence-transformer backbone that converts segments of the input document into contextual vectors, followed by (ii) a sequential head that models dependencies across segments before producing a single fixed-length representation for the job or candidate. We consider three interchangeable heads on the same backbone: GRU, LSTM, and a lightweight Transformer

encoder. Training uses triplet loss over (anchor job, positive résumé, negative résumé) tuples, encouraging the model to reduce the distance between true matches and enlarge it for non-matches by a margin. At inference, cosine similarity between the two tower outputs serves as the match score; in large candidate pools this enables efficient ANN retrieval (e.g., FAISS) with optional re-ranking [4].

- **Cross-encoder models:** Alternatively, a single transformer takes the concatenated resume and job description as input, outputting a compatibility score. This "cross-attention" allows richer interaction between texts, often improving accuracy at the cost of speed. Cross-encoders can consider fine-grained alignments but require re-encoding the pair each time (scaling quadratically with pair count). As a result, many systems use a two-stage pipeline: a fast bi-encoder to retrieve a small set of candidate matches, followed by a cross-encoder ranker for final scoring. This architecture is analogous to modern document retrieval systems. While powerful, cross-encoders are generally used when only a limited number of pairs must be scored or in offline analysis.

Recent studies employ both strategies. For example, the *Resume2Vec* framework uses multiple transformer encoders (BERT, RoBERTa, DistilBERT) to embed documents [1]. It reported large gains in ranking metrics (up to ~15% NDCG improvement) over keyword-based ATS systems . In practice, domain-adapted BERT models (e.g. pre-trained on job corpora) often boost performance further.

Graph Neural Network (GNN) Models

Graph-based methods model CJM as a graph problem. One approach is to construct a **bipartite graph** connecting candidate nodes and job nodes via edges representing relationships. Nodes may also include skill or attribute entities. Edge features can encode semantic similarities (e.g. cosine similarity of embeddings). GNNs then propagate information across this graph to predict match outcomes. For instance, Frazzetto *et al.* built small candidate–job graphs (14 nodes: candidate, job, and attributes) with edges weighted by embedding similarities [2]. They trained GNNs (e.g., Graph Convolutional Networks, Gated Graphs, Graph Attention Networks) on binary screening labels. In experiments, GNNs significantly outperformed an MLP baseline: a GCN achieved 65.4% balanced accuracy vs 55.0% for an MLP, and detected nearly half of qualified candidates versus under 10% for the MLP . This highlights GNN strength in leveraging relational structure and catching minority outcomes. Beyond bipartite designs, large-scale industrial systems create vast heterogeneous graphs. LinkedIn's *STAR* system (2025 KDD) integrates an industry-scale job–candidate graph (billions of nodes/edges) with a transformer+GNN pipeline [5]. The STAR model trains an LLM encoder on long profile/job text and a GNN over the graph to augment signals, achieving improved recommendation quality in A/B tests. In summary, GNNs are favored for their ability to capture multi-hop dependencies (e.g. shared skills or social connections) and to mitigate cold-start by graph connectivity. Architectures include GCN, GraphSAGE, GAT, and heterogeneous GNNs, often combined with learned embeddings from text.

Beyond applied pipelines, there is also a rich graph-theoretic foundation relevant to how we represent and reason over CJM data. In automata theory and formal languages, labeled directed graphs encode state transitions, and their structural properties—such as uniqueness and bounds for admissible edge labelings—affect the information capacity of a graph. For example, one study [6] analyzed the number of possible labelings in definite automata graphs, proving uniqueness for strongly connected graphs over binary alphabets and exponential upper bounds as the alphabet grows. These insights extend naturally to applied ML: candidate–job graphs are likewise labeled, directed, and multi-relational, and their label sets (skills, roles, outcomes) govern the graph's ability to disambiguate paths and support expressive message passing. Situating GNN design within such labeling constraints provides a formal basis for using heterogeneous graphs with typed edges, relation-specific attention mechanisms, or edge-weight priors derived from domain ontologies, ensuring the learned representations remain both structured and interpretable.

Hybrid and Recommender Systems Approaches

Recommender-system techniques are widely applied to CJM. Systems combine **content-based filtering** (matching resume content to job content) with **collaborative filtering** (using applicant-job interaction history) in hybrid models. For example, hybrid designs might weight content similarity alongside a user–item matrix factorization or k-nearest-neighbors on historical applications. Others incorporate knowledge graphs or ontologies (e.g. ESCO taxonomy) to enrich profiles. Hybrid systems often include contextual features: one review notes that adding geographic and industry context to CF+content models improves match accuracy . Common algorithms used include matrix factorization, k-NN, and knowledge-based inference. The systematic review by Ertuğrul and Bitirim found that content-based, collaborative, hybrid, and knowledge-based filters dominate the literature [3]. Evaluation metrics are drawn from recommender and IR domains: precision@k, recall, F1, NDCG, MAP, AUC, etc . Notably, hybrid models have been reported to reduce false positives in candidate ranking and to leverage both text embeddings and collaborative signals for cold-start alleviation.

Collectively, the models reviewed in this section represent the core algorithmic paradigms that define modern candidate–job matching research. While each approach differs in architecture, training objectives, and scalability, they share the same ultimate goal—deriving a robust semantic representation of both candidates and roles to enable accurate and explainable matching.

Table 1 provides a comparative overview of the principal model families identified in the literature, summarizing their input representations, architectural properties, advantages, limitations, and common evaluation metrics.

Table 1. Comparative overview of key algorithms and frameworks for candidate–job matching (CJM).

Model Category	Core Input Representation	Architecture Highlights	Main Advantages	Limitations / Trade-offs	Typical Eval Metrics
Traditional ML	TF-IDF, bag-of-words, handcrafted features	Linear/SVM, Random Forest, XGBoost	Transparent, easy to deploy	Poor semantic coverage, heavy feature engineering	Accuracy, F1, AUC
Transformer-based NLP	Contextual text embeddings (BERT, SBERT, RoBERTa)	Bi-encoder / Cross-encoder	Strong semantic understanding, multilingual	High compute cost, low interpretability	NDCG, MAP, MRR
Siamese Networks (GRU/LSTM /Transformer heads)	Sentence embeddings + sequential modeling	Dual-tower with triplet loss	Efficient retrieval, captures document structure	Sensitive to margin tuning; two-stage design may add latency	Top-K Accuracy, MRR
Graph Neural Networks (GNNs)	Node/edge embeddings (candidate–job–skill graphs)	GCN, GraphSAGE, GAT, heterogeneous GNN	Models multi-hop dependencies, cold-start resilience	Needs labeled relations; limited scalability on dense graphs	Balanced Accuracy, Recall@K
Hybrid / Recommender Systems	Text + interaction logs	Content-based + Collaborative filtering	Personalized, leverages behavioral data	Cold-start for new entities, data sparsity	Precision@K, Recall@K, NDCG
LLM / Hybrid GNN–LLM Systems	Long text, graph structure	LLM encoder + graph propagation	Integrates global context, flexible zero-shot use	Expensive inference, bias risk	AUC-ROC, Fairness metrics

RESULTS

Several recent studies report quantitative gains from modern ML models over traditional baselines. We summarize key findings:

- **Transformer Embedding Models:** Fine-tuned transformer encoders significantly outperform keyword or static-embedding methods. For example, Kurek et al. (2024) [8] evaluated a zero-shot MiniLM-based recommendation model and achieved Top-100 accuracy of 55.45% and Top-500 accuracy of 81.11%, surpassing conventional ATS baselines. Similarly, the Resume2Vec system (MDPI 2023) reported up to 15.85% higher NDCG and 15.94% higher RBO compared to keyword search [1].
- **Siamese vs. Cross-Encoder Models:** Empirical evidence continues to favor Siamese (bi-encoder) architectures for large-scale recruitment pipelines due to their strong balance of accuracy and scalability. Recent work [4] introduced a modular Siamese framework that systematically compares GRU, LSTM, and Transformer sequential heads atop a multilingual Sentence Transformer backbone, trained end-to-end with triplet loss on real-world recruitment data. Among the tested variants, the Transformer-based Siamese model achieved a Mean Reciprocal Rank (MRR) of 0.979 and a Top-100 accuracy of 87.2%, outperforming both traditional baselines and static embedding approaches. Visualization of embedding spaces using t-SNE further confirmed that self-attention mechanisms produced tighter clustering of matching job–résumé pairs and clearer separation of irrelevant ones. By contrast, cross-encoder architectures—where job and résumé texts are jointly encoded with cross-attention—typically yield marginally higher per-pair accuracy but at a significant computational cost, as they require re-encoding every candidate–job combination. For large candidate pools, this leads to quadratic scaling, making real-time retrieval impractical. Consequently, most modern CJM systems adopt a two-stage pipeline: an efficient Siamese bi-encoder for initial retrieval, followed by a cross-encoder ranker applied to the top-N matches. This hybrid design preserves the high precision of cross-attention models while maintaining the speed necessary for production-scale candidate search.
- **Graph Neural Networks:** The GNN-based approaches report marked improvement in screening efficacy, especially for the critical minority class of qualified candidates. Frazzetto *et al.* (2025) found that a GCN achieved 65.4% balanced accuracy on a resume screening task, compared to 55.0% for a feed-forward MLP [2]. The GNN identified 48.9% of qualified candidates versus only 8.5% for the MLP. These gains illustrate that modeling the relational graph structure can uncover subtler signals of fit that flat models miss. Industrial-scale results (e.g. LinkedIn) are generally proprietary but allude to improved candidate engagement and match relevance when integrating GNNs [5].

- **Hybrid Recommenders:** Studies show that combining content and collaborative information mitigates typical recommender problems. For example, adding geographical context and skill-skill relationships into a content-based model yields higher recommendation relevance [7]. The SLR by Ertuğrul and Bitirim noted extensive use of hybrid and knowledge-based techniques, and reported that hybrid models tended to reduce false positives in ranking compared to pure content filters [3]. Specific accuracy numbers vary by dataset, but hybrid systems generally achieve higher precision and recall metrics (e.g. 90%+ F1 on certain curated benchmarks) than single-method baselines.
- **Large Language Models (LLMs):** While powerful, off-the-shelf LLMs show mixed performance in CJM. An internal study (Eightfold AI, 2025) compared various LLMs (GPT variants, Claude, etc.) against a proprietary supervised CJM model ("Match Score") on 10,000 real candidate–job pairs. The domain-specific model achieved ROC AUC of 0.85, outperforming the best general LLM at AUC 0.77 . The LLMs also had larger disparities in fairness metrics across demographic groups. This suggests that while LLMs capture broad language patterns, specialized fine-tuning on hiring data yields more accurate and equitable matching. Nonetheless, LLM-derived embeddings and outputs can enhance existing pipelines if carefully managed.

Taken together, these results indicate that advanced ML methods can substantially improve the quality of candidate–job matching. Transformer embeddings and graph-based models often lead to double-digit improvements in ranking metrics . However, performance depends on data quality and representation. Most studies emphasize that rigorous evaluation (including human expert assessment) is necessary to validate gains in real hiring scenarios.

Quantitative outcomes from recent studies are summarized below. Figure 1 visualizes headline results across model families on a common percentage scale to facilitate comparison.

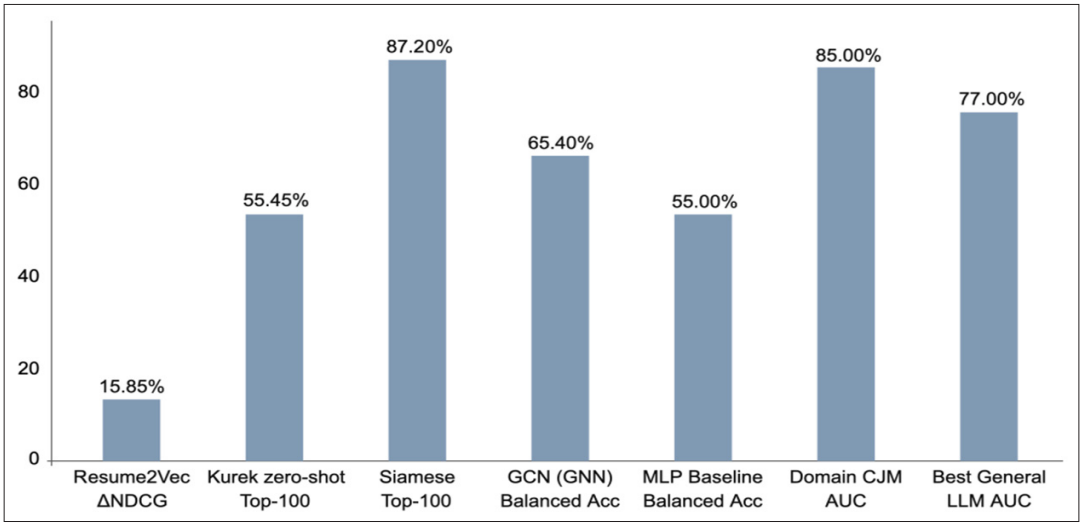

Figure 1. CJM results from recent studies (2023–2025)

DISCUSSION

The surveyed methods exhibit complementary strengths and weaknesses in the CJM context:

- **Semantic Understanding (Transformer-based models):** Deep NLP models excel at capturing context and meaning in text. They can detect synonymy and related skills automatically, greatly improving matching over keyword filters [1]. Their embeddings encode candidate expertise holistically, accommodating varied resume formats. However, they are large and opaque: BERT-based models have millions of parameters, making them slow to train and requiring GPU resources. Inference can be costly for pairwise scoring. Interpretability is a challenge – understanding why a model matched a resume to a job often requires post-hoc explanation techniques (e.g. attention visualization). Additionally, fine-tuning requires substantial labeled data, and models can inherit biases from training corpora . Mitigation strategies (balanced training, fairness constraints) are advised, as these systems will influence high-stakes hiring outcomes.
- **Efficiency and Scalability:** In large organizations, scalability is critical. Siamese encoder models alleviate this by precomputing embeddings; a bi-encoder can score millions of candidate–job pairs with simple vector operations. Cross-encoders, despite higher accuracy on single pairs, are usually restricted to filtering among a few candidates. Industrial systems (e.g. the STAR framework [5]) combine offline training of embeddings with nearline serving for fresh data, achieving real-time matching. GNN training on massive graphs can be resource-intensive; approaches like inductive learning and mini-batching of graph data are employed. The LinkedIn STAR and other industry reports highlight that a decoupled training scheme (train LLM and GNN

separately, then merge signals) provides a practical trade-off .

- **Graph Structured Learning:** GNNs offer rich relational reasoning but introduce complexity. Constructing meaningful graphs (defining nodes and edges) requires domain knowledge (e.g. skill ontologies) and sometimes external data sources (social links, job taxonomies). While GNNs can improve minority candidate detection [2], they make interpretability harder: understanding which graph connections led to a high score is nontrivial. Nonetheless, messages passing in GNNs can be analyzed or visualized to some extent, and the structure itself (e.g. shared skill neighbors) provides intuitive cues. Moreover, GNNs can mitigate cold-start by linking new candidates through attribute nodes (skills, education) to existing job/people networks.
- **Hybrid Recommenders:** Combining multiple signal types often yields robustness. For example, adding content similarity (resumes ↔ jobs) to collaborative signals (co-application patterns) can improve recall without sacrificing precision [7]. Hybrid models can be more interpretable if built with components like weighted linear combinations of scores. They also allow modular updates: one can swap in a new text model or a new collaborative module independently. However, tuning and integrating heterogeneous methods can be complex. Cold-start remains a concern if neither content nor history is available (e.g. brand-new skills or roles).
- **Bias and Fairness:** Recruitment systems must ensure non-discrimination. The surveyed literature notes that generic LLMs can inadvertently perpetuate gender or racial bias learned from data . In contrast, supervised models trained on curated hiring data (as in Match Score) achieved near-equal impact ratios across demographics. Interpretability tools (feature importance in tree models, attention weights) can help diagnose bias sources. Fairness-aware learning algorithms and bias audits are recommended. Additionally, hybrid systems can incorporate fairness constraints explicitly. Ethical deployment guidelines suggest combining quantitative fairness evaluation with human oversight.
- **Interpretability:** Simple models (logistic regression, decision trees) allow direct attribution of matches to skill overlaps or historical evidence. Deep models require auxiliary explanation methods (e.g. LIME, SHAP). Explainability is particularly important in HR to justify decisions to stakeholders. Recent work on explainable person–job recommendation (outside our date range) points to generating natural-language rationales or highlighting key resume sections as interpretability aids. Incorporating such explainers into CJM pipelines is an open area for future work.

Challenges: Despite advancements, gaps remain. No gold-standard public datasets exist, hampering benchmark comparisons. Many systems are proprietary or tested on specific corpora, so reported results may not generalize. Data privacy concerns also limit sharing of real resumes. Moreover, evolving job markets mean that models must adapt to new skills and roles; continuous learning pipelines are needed. Finally, candidate preferences and multi-objective matching (culture fit, career goals) are still underexplored dimensions beyond raw skill alignment.

CONCLUSION

In the last five years, candidate–job matching has transitioned from rule-based systems to sophisticated ML-driven pipelines. Transformer encoders, GNNs, and hybrid recommenders now underpin the state-of-the-art. These methods offer markedly better matching accuracy and relevance than traditional approaches . They capture semantic and relational nuances in ways that keyword matching cannot. At the same time, these gains must be weighed against practical and ethical considerations. Ensuring model interpretability, fairness, and scalability is essential for responsible deployment. Domain-specific fine-tuning and bias auditing are particularly important when using large language models in hiring contexts . Going forward, we expect continued integration of pre-trained language models and graph techniques, alongside hybrid designs that leverage diverse data sources. Open challenges include creating shared evaluation frameworks, improving transparency, and balancing personalization with equity. Overall, the recent advances make automated CJM a powerful tool to assist recruiters – provided its limitations are recognized and mitigated.

REFERENCES

1. Bevara, R.V.K., Mannuru, N.R., Karedla, S.P., Lund, B., Xiao, T., Pasem, H., & Dronavalli, S.C. (2023). Resume2Vec: Transforming Applicant Tracking Systems with Intelligent Resume Embeddings for Precise Candidate Matching. *Electronics*, 14(4):794. https://doi.org/10.3390/electronics14040794 .
2. Frazzetto, P., Ul Haq, M.U., Fabris, F., & Sperduti, A. (2025). Graph Neural Networks for Candidate–Job Matching: An Inductive Learning Approach. *Data Science and Engineering*. https://doi.org/10.1007/s41019-025-00293-y
3. Ertuğrul, D.Ç., & Bitirim, S. (2025). Job recommender systems: A systematic literature review, applications, open issues, and challenges. *Journal of Big Data*, 12:140. https://doi.org/10.1186/s40537-025-01173-y .
4. Łępicki, M., Latkowski, T., Antoniuk, I., Bukowski, M., Świderski, B., Baranik, G., Nowak, B., Zakowicz, R., Dobrakowski, Ł., Act, B., & Kurek, J. (2025). Comparative Evaluation of Sequential Neural Network (GRU, LSTM, Transformer) Within Siamese Networks for Enhanced Job–Candidate Matching in Applied Recruitment Systems. Applied Sciences, 15(11), 5988. https://doi.org/10.3390/app15115988

5. Liu, P., He, S., Shen, J., Borisyuk, F., Hewlett, D., & others (2025). A Scalable and Efficient Signal Integration System for Job Matching. In *KDD 2025: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining* .

6. Ishchenko, R.A. Number of Labelings of Definite Automata Graphs. Moscow Univ. Math. Bull. 77, 102–107 (2022). https://doi.org/10.3103/S0027132222020048

7. Yap, R.E., Haw, S.C., & Al-Juboori, S. (2025). A Comprehensive Review on Machine Learning-Based Job Recommendation Systems. *International Journal on Robotics, Automation and Sciences*, 7(2), 36–55. (Open Access) .

8. Kurek, J., Latkowski, T., Bukowski, M., Świderski, B., Łępicki, M., Baranik, G., et al. (2024). Zero-Shot Recommendation AI Models for Efficient Job–Candidate Matching in Recruitment Process. *Applied Sciences*, 14(6):2601. https://doi.org/10.3390/app14062601.