



JobFormer: Skill-Aware Job Recommendation with Semantic-Enhanced Transformer

ZHIHAO GUAN, Nanjing University of Science and Technology, China

JIA-QI YANG, State Key Laboratory for Novel Software Technology, Nanjing University, China

YANG YANG*, Nanjing University of Science and Technology, China

HENGSHU ZHU, Career Science Lab, BOSS Zhipin, China

WENJIE LI, The Hong Kong Polytechnic University, China

HUI XIONG, the Artificial Intelligence Thrust, The Hong Kong University of Science and Technology, China

Job recommendation aims to provide potential talents with suitable job descriptions (JDs) consistent with their career trajectory, which plays an essential role in proactive talent recruitment. In real-world management scenarios, the available JD-user records always consist of JDs, user profiles, and click data, in which the user profiles are typically summarized as the user's skill distribution for privacy reasons. Although existing sophisticated recommendation methods can be directly employed, effective recommendation still has challenges considering the information deficit of JD itself and the natural heterogeneous gap between JD and user profile. To address these challenges, we proposed a novel skill-aware recommendation model based on the designed semantic-enhanced Transformer to parse JDs and complete personalized job recommendation. Specifically, we first model the relative items of each JD and then adopt an encoder with the local-global attention mechanism to better mine the intra-job and inter-job dependencies from JD tuples. Moreover, we adopt a two-stage learning strategy for skill-aware recommendation, in which we utilize the skill distribution to guide JD representation learning in the recall stage, and then combine the user profiles for final prediction in the ranking stage. Consequently, we can embed rich contextual semantic representations for learning JDs, while skill-aware recommendation provides effective JD-user joint representation for click-through rate (CTR) prediction. To validate the superior performance of our method for job recommendation, we present a thorough empirical analysis of large-scale real-world and public datasets to demonstrate its effectiveness and interpretability.

CCS Concepts: • **Information systems** → **Data mining**.

Additional Key Words and Phrases: Skill-Aware Representation, Transformer, Job Recommendation

*Yang Yang (Corresponding Author) is with PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, and Jiangsu Key Lab of Image and Video Understanding for Social Security, School of Computer Science and Engineering, Nanjing University of Science and Technology.

Authors' addresses: Zhihao Guan, Nanjing University of Science and Technology, Nanjing, China, 210094, zhguan@njust.edu.cn; Jia-Qi Yang, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, 210023, yangjq@lamda.nju.edu.cn; Yang Yang, Nanjing University of Science and Technology, Nanjing, China, 210094, yyang@njust.edu.cn; Hengshu Zhu, Career Science Lab, BOSS Zhipin, Beijing, China, 100028, zhuhengshu@gmail.com; Wenjie Li, The Hong Kong Polytechnic University, Hong Kong, China, wenjie.li@polyu.edu.hk; Hui Xiong, the Artificial Intelligence Thrust, The Hong Kong University of Science and Technology, Guangzhou, China, xionghui@ust.hk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Copyright held by the owner/author(s).

ACM 1556-472X/2024/10-ART

<https://doi.org/10.1145/3701735>

1 INTRODUCTION

Job recommendation aims at providing the right jobs to the right job seekers. In recent years, online recruitment data has experienced explosive growth. According to the report from The Insight Partners [35], the global online recruitment market size is expected to grow from \$29.29 billion in 2021 to \$47.31 billion by 2028. As a result, it is crucial for recruitment platforms to develop effective job recommendation systems that not only help companies quickly recruit candidates for specific positions, but also meet the needs of job seekers for an efficient and personalized job search experience.

In real-world recruitment scenarios, the available recruitment records typically only include job descriptions (JDs), user profiles, and click data. It is worth noting that users generally do not upload complete user profiles, primarily because complete user profiles may involve sensitive information, which could lead to privacy concerns or identity theft in case of improper use. Therefore, user profiles are usually summarized as personal skill distribution. With the rapid development of deep learning (DL), intelligent recommendation systems have revolutionized the recruitment field [3, 53–55]. A natural and straightforward idea is to extract available information from JDs and user profiles, and rely on many off-the-shelf sophisticated recommendation methods to accomplish job recommendation. For example, content-enriched recommendation approaches can model semantic relevance between user profiles and JDs from two aspects: the general features of user profiles and JDs, as well as the textual content information [47]. The former focuses on the feature interactions of user profile and JD, e.g., [4, 13, 40] explore the possibility of adopting neural models to automatically discover complex higher-order feature interactions for click-through rate (CTR) prediction and recommendation. The latter focuses more on multi-level automatic representation learning of textual content, e.g., [32, 59] employ the specially designed neural network to model the talent resumes and job descriptions respectively, which are jointly trained as a binary classification problem (i.e., consistent or inconsistent). Considering that these approaches cannot effectively mine users' interest preferences, a few approaches also explore generating recommendation lists in a two-stage manner, which contains a recall stage for forming a candidate set and a ranking stage for ranking candidate items based on their relevance to user interests [31, 46]. However, these approaches are problematic since the JD itself is insufficiently informative, and there is a natural heterogeneous gap between JD (represented by short texts in Figure 1 (b)) and user profiles (represented by skill distributions in Figure 1 (c)), which inevitably leads to a decline in recommendation effectiveness.

To this end, in this paper, we propose a skill-aware job recommendation method that incorporates a semantic-enhanced Transformer and a two-stage learning strategy. We first consider enhancing the semantic representation of JD by aggregating complementary information from neighbor JDs, which provides a more comprehensive understanding of the duties and requirements for a specific position, since neighbor JDs typically contain information related to the same domain or position. Specifically, we first model the relative items of each JD, and then adopt an encoder with the local-global attention mechanism to better mine the intra-job and inter-job dependencies from JD tuples. Moreover, in order to better mitigate the heterogeneous gap between JD and user profiles, as well as effectively discover users' interest preferences, our idea is to employ a two-stage learning strategy for skill-aware job recommendation. To be specific, as shown in Figure 1 (a), we first utilize user profiles (i.e., personal skill distributions) to guide the representation learning of JDs in the recall stage, which enables to recall a set of candidate JDs in the embedding space according to the relevance metric function (i.e., cosine similarity). Then, in the ranking stage, we further predict the click-through rate between candidate JDs and the user for a personalized job recommendation.

The main contributions of this paper can be summarized as follows:

- We design a semantic-enhanced Transformer, which employs the local-global attention mechanism to effectively explore JD information for rich contextual semantics;

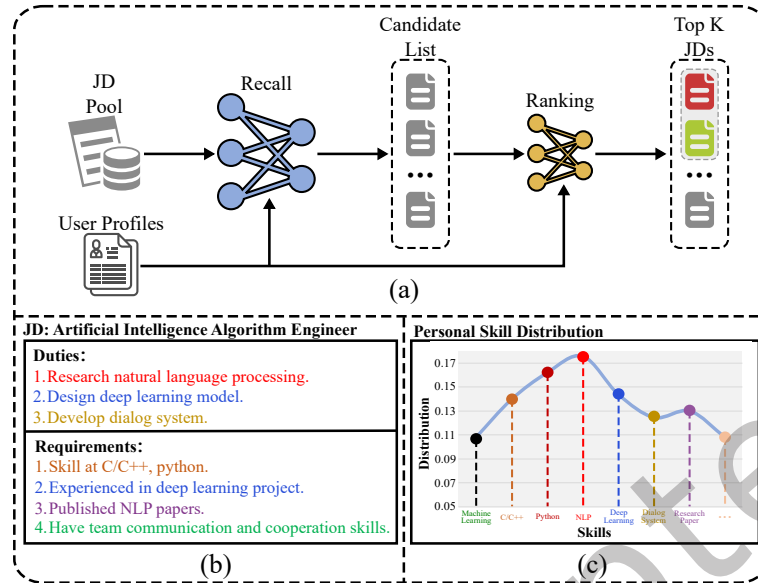


Fig. 1. A motivating example of two-stage skill-aware job recommendation (a). The JD contains multiple items including duties and requirements (b), and the corresponding user can be represented with personal skill distribution (c). Note that actually the items in JD correspond to the skill labels, and the degrees of demands correspond to the skill distributions.

- We propose JobFormer, a two-stage learning-based method for job recommendation, which leverages skill-aware JD representation to bridge the heterogeneous gap between JDs and user profiles, as well as to enhance recommendation performance;
- Our method achieves state-of-the-art performance across multiple real-world datasets and provides better interpretability for job recommendation.

2 RELATED WORK

In this paper, we aim to learn an effective job recommendation method with skill-aware JD representation. Therefore, our study is related to job recommendation systems and JD representation learning.

2.1 Job Recommendation Systems

Job recommendation is a core component of recruitment platforms, and it has been extensively studied in the literature [6, 20]. Early approaches treated this problem as a job-resume matching problem [8, 26], and obtained matching capabilities based on the collaborative filtering assumption. However, this approach overemphasizes the interaction between user profiles and job postings, which can result in limited recommendation performance when interaction data is sparse. To mitigate this challenge, recent studies have focused more on utilizing intelligent techniques to mine textual information, aiming to enhance the semantic representation of job and user profiles. Around this problem, [7] employed TF-IDF statistical approach to encode JDs and resumes. [58] developed a generalized linear mixed model (GLMix), a fine-grained model at the user or item level, in the LinkedIn job recommendation system, and generated 20% to 40% more job applications for job seekers. Thanks to the advances in deep neural networks (DNNs) that are extensively used in the field of natural language processing (NLP), DNN-based approaches are proposed and have demonstrated state-of-the-art results. For instance, [59] developed

a novel end-to-end neural model, which projects both job postings and candidate resumes onto a shared latent representation for joint representation learning. [32] designed an ability-aware neural network, which extracts the ability-aware representations for job postings and resumes simultaneously by hierarchical representation structures. These methods emphasize the significance of considering effective representations of multi-modal input (i.e., job descriptions and resumes) in job recommendation tasks. Nevertheless, previous job recommendation approaches usually only consider the relevance between JDs and user profiles, without paying more attention to users' interest preferences for JDs. Moreover, due to the complex heterogeneity between JDs and user profiles (e.g., personal skill distribution), it remains challenging to improve job recommendation performance.

2.2 JD Representation with Deep Learning

Generally, the problem of JD representation based on textual data can be categorized as the tasks of text mining, which is highly relevant to Natural Language Processing techniques, such as text classification [25, 36], machine translation [42, 48], and multi-modal learning [51, 52]. Recently, due to the advanced performance and flexibility of deep learning, more and more researchers have attempted to leverage deep neural networks to address text mining problems. In contrast to traditional approaches that heavily rely on effective manually designed representations and input features (e.g., N-gram model [39], Bag-of-words model [22] and parse trees [5]), the deep learning-based approaches can automatically learn effective feature representations from a large-scale text corpus.

Among various deep learning models, traditional deep learning models (e.g., convolutional neural network (CNN) [23] and recurrent neural network (RNN) [9]) and Transformer-based models are two representative and extensively used approaches, which can provide practical ways for JD representation from different perspectives. Specifically, CNNs can effectively extract local semantics and hierarchical relationships in textual data. For instance, [33] proposed to encode the job based on CNN. [59] have shown that the power of CNN on person-job fit tasks, even only using a few one-dimensional convolutional layers to learn JD representation. Furthermore, RNN-based models have also achieved remarkable performance. For example, [32] designed a word-level semantic representation for both job requirements and job seekers' experiences based on the Recurrent Neural Network. Similarly, [49] adopt the RNNs with GRU units to propagate information along the word sequence of job posting. Compared with traditional deep learning models, Transformer-based methods more naturally model sequential textual data and learn global semantic representation using the self-attention mechanism. For example, [2] developed a hierarchical self-attention text representation model for developing the semantic matching model, in which a BERT-based encoder is first adopted to represent jobs, followed by a Transformer-based encoder that processes the entire text document using learned sentence embeddings. Although Transformers process textual data efficiently, the information deficit of JD itself reduces the parsing of JD semantics.

In this paper, we follow some outstanding ideas in the above works according to the properties of job recommendation and propose a two-stage method JobFormer based on the semantic-enhanced Transformer with skill-aware JD representation. Therefore, JobFormer can not only improve the performance of job recommendation, but also enhance the model interpretability in practical scenarios.

3 PRELIMINARIES

In company talent management, the available JD-user records usually include job descriptions, user profiles (i.e., personal skill distribution), and click data. Specifically, a job description contains multiple items with short text forms, which describe duties and requirements. Without any loss of generality, we utilize $\mathbf{j} = \{j^1, j^2, \dots, j^M\}$ to denote the items (e.g., the JD about artificial intelligence algorithm engineer as shown in Figure 1 (b)), M denotes the total number of duties and requirements, and we fix M as the maximum number of items for all JDs with padding mask as [37]. For user profiles, we adopt all the skills summarized as global label space (i.e., C skills totally), and then acquire the skill ratings from experts. Lastly, we normalize the summarized skill ratings as

the label distribution with softmax operator according to [11], i.e., $\mathbf{y} = \{y^1, y^2, \dots, y^C\}$, satisfying the constraint $y^c \in [0, 1]$ and $\sum_c y^c = 1$. To simplify our problem, we assume that a job can be represented by its duties and requirements, and the skill distribution of a candidate can mainly reflect his competency.

As a matter of fact, as shown in Figure 1 (b), the duties and requirements in JD correspond to the personal skills, e.g., “Design deep learning model” indicates that the user needs to have “deep learning” skill. The degree of demands corresponds to the skill distributions, e.g., considering that “deep learning” is repeatedly mentioned in the JD, and the demand is high (i.e., “Design deep learning model” in duties and “Experienced in deep learning project” in requirements), the degree of skill “deep learning” should be high. A straightforward approach is to utilize label distribution learning (LDL) to predict the skill distribution of the input JDs and match it with the user profiles (i.e., personal skill distribution) according to the cosine similarity. However, this approach only recommends JDs to users from the perspective of relevance, which lacks the mining of users’ interest preferences. Therefore, our idea is first to recall a set of candidate JDs and then utilize the click data to optimize the ranking ability of the model on the candidate JDs for a personalized job recommendation. Along this line, we can formally define the problems:

DEFINITION 1. (Skill-Aware Representation Learning for JD Recall). Given a set of successful person-job records \mathcal{D} , each record $(\mathbf{j}, \mathbf{y}) \in \mathcal{D}$ is the corresponding JD and skill distribution. The target of JD recall can be formulated as learning a predictive model f for predicting the skill distribution of the input \mathbf{j} , and then recalling a set of candidate JDs from a large-scale JD pool based on their relevance to user profiles.

DEFINITION 2. (Click-Through Rate Prediction for JD Ranking). Given a set of candidate sets \mathcal{D}' with their corresponding click data, $\mathcal{D}' \subset \mathcal{D}$. JD ranking aims to further discover user-interesting JDs from candidate JDs, which can be defined as a CTR prediction task, i.e., predicting the probability of a user clicking on the candidate JDs.

4 JOBFORMER

As shown in Figure 2, following a widely used paradigm in real-world recommendation systems, JobFormer contains a recall stage for candidate JDs generation and a ranking stage for candidate JDs ranking. Specifically, in the recall stage, we first leverage the TextCNN [19] to encode the JD for diverse item-level representations, which are fed into the local-global Transformer to capture rich contextual semantics. The learned JD representations are further calculated similarity scores with user profiles (e.g., personal skill distribution) to recall candidate JDs from a large-scale JD pool. Finally, in the ranking stage, candidate JDs are combined with user profiles for CTR prediction with a click predictor. Next, we will describe each component of our JobFormer in detail.

4.1 Item-Level Encoder

As shown in Figure 1 (b), each job description \mathbf{j} includes a set of items, including the duties and requirements. It is notable that these items are in short text forms and have no contextual information, which may lead to semantic confusion if we directly concatenate these items into long sentences for representation learning. For example, the word “Design” in “Design deep learning model” only describes the duty of deep learning model, and has no contribution to other duties or requirements, so there is no contextual relationship. Based on this idea, we need to model the items separately to obtain the item-level representations.

Without any loss of generality, we adopt a shared TextCNN to process the items separately. In detail, given an item \mathbf{j}^m with S words, the corresponding matrix can be represented as: $\mathbf{j}^m = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_S] \in \mathbb{R}^{S \times d}$. Then we apply two one-dimensional convolutional layers on the input layer considering that one-dimensional convolution can deal with an unfixed-length sequence [19]. To reduce the training cost, we apply Batch Normalization [16] followed by a Rectified Linear Unit (ReLU) layer [30] and a one-dimensional max-pooling layer on the outputs of

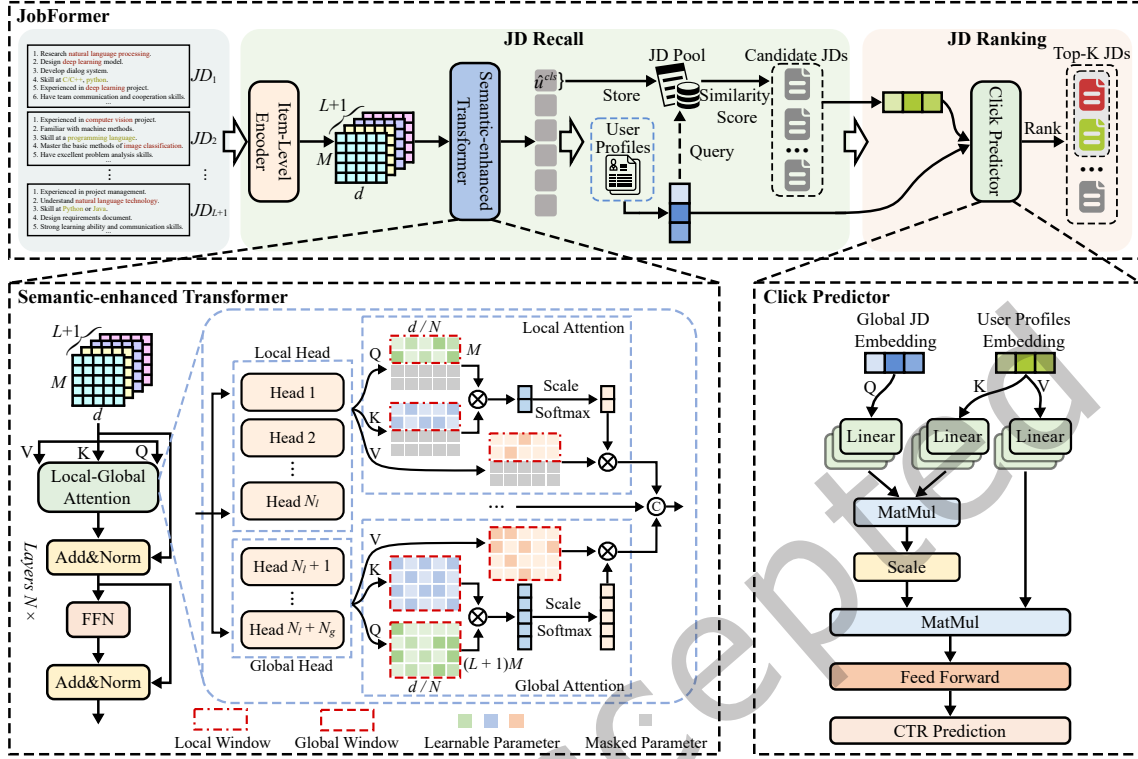


Fig. 2. An illustration of the proposed JobFormer, which includes a recall stage for candidate JDs generation and a ranking stage for candidate JDs ranking. In the recall stage, the JD and its neighbors constitute the JD tuple, and the item-level encoder aims for the item representation, which acts as the input token for the semantic-enhanced Transformer. Then the designed semantic-enhanced Transformer encodes both the intra-job and inter-job information to acquire more discriminative JD representation, which is further recalled as candidate JDs according to the JD-user cosine similarity. Lastly, in the ranking stage, recalled candidate JDs are combined with user profiles for CTR prediction via a click predictor and ranked for a personalized job recommendation.

one-dimensional convolutional layers. Therefore, each item is inputted into the shared TextCNN:

$$\mathbf{u}^m = \text{TextCNN}(j^m)$$

where $\mathbf{u}^m \in \mathbb{R}^d$ denotes the representations of m -th item.

4.2 Semantic-Enhanced Transformer

Based on the item-level representations, the key challenges to encode the JD are: 1) Various importance. Different duties and requirements carry varying levels of importance within a job description. Take the JD in Figure 1 (b) as an example, “Design deep learning model” and “Experienced in deep learning project” are more important than “Priority for published papers” considering that deep learning skill is repeatedly mentioned and has a high demand (e.g., “Design” and “Experienced”), while “published papers” is a supplementary condition. 2) Deficient information. Single JD may ignore some derived information. Take the JD in Figure 1 (b) as an example, many items are related to “machine learning” skill, which is needed for the post (i.e., the successfully accepted user is

considered for this skill). To overcome these problems, we design the Transformer with local-global attention heads to measure the importance of intra-job items and integrate the inter-job information.

Local Encoder. To comprehensively encode each JD by considering the dependencies between items, we employ the Transformer encoder [37] as the backbone, which can encode the relationships among items by adopting the self-attention mechanism. Specifically, as shown in Figure 2, with the item representations, a job can be denoted as $U_l = [\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^M] \in \mathbb{R}^{M \times d}$, where d is the dimension of hidden states. The identical block contains two sub-layers: 1) The first sub-layer utilizes multi-head attention to learn the correlated representations. 2) The second sub-layer adopts a position-wise feed-forward network (FFN). In multi-head attention layer, the input representations can be used to compute three matrices: Q , K , and V corresponding to queries, keys, and values. The dot-product similarity between queries and keys determines attention distributions:

$$\begin{aligned} Q_l &= U_l W_{Q_l}, \quad K_l = U_l W_{K_l}, \quad V_l = U_l W_{V_l}, \\ A_l &= \frac{Q_l K_l^\top}{\sqrt{d_{N_l}}} \quad \text{Att}(U_l) = \sigma(A_l) V_l, \end{aligned} \quad (1)$$

where $Q_l \in \mathbb{R}^{M \times d_{N_l}}$, $K_l \in \mathbb{R}^{M \times d_{N_l}}$, $V_l \in \mathbb{R}^{M \times d_{N_l}}$, and $W_{Q_l} \in \mathbb{R}^{d \times d_{N_l}}$, $W_{K_l} \in \mathbb{R}^{d \times d_{N_l}}$, $W_{V_l} \in \mathbb{R}^{d \times d_{N_l}}$ are learnable matrices. N_l denotes the number of local heads. The activation function σ can be used as softmax here. It is notable that multi-head attention is defined as the local attention here in literature, which aims to encode the intra-job information.

Global Encoder. To introduce the extra neighbor JDs as complementary information, we further propose the joint modeling strategy with local-global attention. In detail, we first select L neighbors for \mathbf{j} , i.e., $\mathcal{N}(\mathbf{j}) = \{\mathbf{j}_1, \mathbf{j}_2, \dots, \mathbf{j}_L\}$. Without any loss of generality, we adopt the Euclidean distance according to the global embedding $\hat{\mathbf{u}}^{cls}$ and title of JD (e.g., the ‘‘Artificial Intelligence Algorithm Engineer’’ as shown in Figure 1 (b)), i.e., $\|\hat{\mathbf{u}}^{n_1, cls} - \hat{\mathbf{u}}^{n_2, cls}\|_2^2$, $t(\mathbf{u}^{n_1}) = t(\mathbf{u}^{n_2})$, where $t(\cdot)$ represents the title representation with one-hot form, n_1 and n_2 represent the index of $\hat{\mathbf{u}}$. Thereby, we can concatenate $L + 1$ JDs as input with special token [SEP] for separation, i.e., $U_g = [U_l, \mathcal{N}(U_l)]^\top \in \mathbb{R}^{(L+1)M \times d}$. As shown in Figure 2, we add N_g parallel heads as global attention. The dot-product similarity can be reformulated as:

$$\begin{aligned} Q_g &= U_g W_{Q_g}, \quad K_g = U_g W_{K_g}, \quad V_g = U_g W_{V_g}, \\ A_g &= \frac{Q_g K_g^\top}{\sqrt{d_{N_g}}} \quad \text{Att}(U_g) = \sigma(A_g) V_g, \end{aligned} \quad (2)$$

where $Q_g \in \mathbb{R}^{(L+1)M \times d_{N_g}}$, $K_g \in \mathbb{R}^{(L+1)M \times d_{N_g}}$, $V_g \in \mathbb{R}^{(L+1)M \times d_{N_g}}$, and $W_{Q_g} \in \mathbb{R}^{d \times d_{N_g}}$, $W_{K_g} \in \mathbb{R}^{d \times d_{N_g}}$, $W_{V_g} \in \mathbb{R}^{d \times d_{N_g}}$ are learnable matrices. The activation function σ can be used as softmax here. Finally, local-global attention is composed of $N = N_l + N_g$ parallel heads, and $d_{N_l} = d_{N_g} = d/N$. For each JD, the results of local head and corresponding global head are concatenated, and the FFN can be reformulated as:

$$\begin{aligned} \text{MultiAtt}(U) &= [\text{Att}(U_l)_1, \dots, \text{Att}(U_l)_{N_l}, \\ &\quad \text{Att}(U_g)_1[id(U_l)], \dots, \text{Att}(U_g)_{N_g}[id(U_l)]] W_U \\ \text{FFN}(\text{MultiAtt}(U)) &= \max(0, \text{MultiAtt}(U) W_1 + b_1) W_2 + b_2, \end{aligned} \quad (3)$$

where $id(U_l)$ denotes the corresponding index of local U_l in $\text{Att}(U_g)$. W_1 , W_2 , and W_U are matrices for linear transformation, b_1 and b_2 are the bias terms. Meanwhile, each sub-layer is followed by dropout [34], shortcut connection [14], and layer normalization [1]. We can also add the special token [CLS] for each JD to learn the global representations, and the attention value depicts the importance of each item. Finally, for each JD, we can acquire the global JD embedding, i.e., $\hat{\mathbf{u}}^{cls}$, from the [CLS] token, and individual embedding from other tokens.

Therefore, the local head attentions are responsible for capturing local dependencies based on local details, i.e., the intra-job items, and global head attentions are designed to model the long-term dependencies between

JDs, i.e., the inter-job items. The combination of local and global attention enables our JobFormer to dynamically model local items and capture the global dependencies of similar JDs. Consequently, for each JD \mathbf{j} , we can acquire both the JD-level and item-level representations, i.e., $\hat{\mathbf{j}} = [\hat{\mathbf{u}}^{cls}, \hat{\mathbf{u}}^1, \hat{\mathbf{u}}^2, \dots, \hat{\mathbf{u}}^M] \in \mathbb{R}^{(M+1) \times d}$, $\hat{\mathbf{u}}^{cls} \in \mathbb{R}^d$ is the JD global representation.

4.3 Skill-Aware Representation Learning for JD Recall

With the JD-level representations $\hat{\mathbf{u}}^{cls}$, we can directly predict the skill distribution with a simple classifier, e.g., a fully connected network g with softmax operator. For simplicity, we can utilize the KL-divergence [27] between ground-truth and prediction:

$$\ell = KL(\mathbf{y}, g(\hat{\mathbf{u}}^{cls})) = \sum_c \mathbf{y}^c \log\left(\frac{\mathbf{y}^c}{g^c(\hat{\mathbf{u}}^{cls})}\right) \quad (4)$$

$KL(a, b) = a \log \frac{a}{b}$ is the KL-divergence that penalizes difference. However, the direct prediction ignores the correlations between different skills, i.e., a skill can help to learn another skill under certain conditions. For example, as shown in Figure 1, when the user has a high description degrees on skills “Deep Learning” and “Dialog System”, the skill “Natural Language Processing” is more likely to have a higher description degree than skill “Software Engineering”. Because “Deep Learning” and “Dialog System” are usually related to “Natural Language Processing”. Therefore, taking skill correlations into consideration can include more data information and achieve better performance [15, 17, 50], we consider two aspects in the recall stage: 1) skill correlation enhancement. 2) relation consistency.

Skill correlation enhancement. Based on the above analysis, we collect additional information about the accepted users to assist in predicting the skill distribution, including the position name (which can reflect the skills the user has mastered) and position level (which can reflect the user’s skill proficiency). In detail, we first encode this additional user information \mathbf{e}^{aux} with a fully connected layer and then adopt a bidirectional ranking loss [10] with margin α to match JD (i.e., $\hat{\mathbf{u}}^{cls}$) and additional user information (i.e., \mathbf{e}^{aux}):

$$M = [\alpha - s(\mathbf{e}^{aux}, \hat{\mathbf{u}}^{cls}) + s(\mathbf{e}^{aux}, \hat{\mathbf{u}}_*^{cls})]_+ + [\alpha - s(\mathbf{e}_*^{aux}, \hat{\mathbf{u}}^{cls}) + s(\mathbf{e}_*^{aux}, \hat{\mathbf{u}}^{cls})]_+ \quad (5)$$

where $[x]_+ \equiv \max(x, 0)$ and s is a similarity score function (i.e., cosine similarity). $\hat{\mathbf{u}}_*^{cls}$ and \mathbf{e}_*^{aux} represent the corresponding hardest negative JD and hardest negative additional user information within a mini-batch, respectively.

Relation consistency. To constrain the structure of neighbors, we define a relation consistency constraint using a metric learning-based constraint, inspired by the linguistic structuralism [29] that relations can better present the knowledge than individual examples. Specifically, each JD and its neighbors can be denoted as a bag of $L + 1$ instances, i.e., $\Psi(\mathbf{j})$, and the pairwise similarity between JD representations and predictions should be consistent. Therefore, we constrain the KL divergence of the similarity vectors calculated by the predictions and representations. The JD representations can be denoted as $\hat{\mathbf{u}}^{i,cls}$ and the predictions can be formulated as $g(\hat{\mathbf{u}}^{i,cls})$, where $\hat{\mathbf{u}}^{i,cls} = SE - Transformer(TextCNN(\mathbf{j}_i))$, $SE - Transformer$ denotes the semantic-enhanced Transformer. Therefore, the objective of relation consistency can be formulated as:

$$R = KL(\Phi(g(\hat{\mathbf{u}}^{1,cls}), \dots, g(\hat{\mathbf{u}}^{L+1,cls})), \Phi(\hat{\mathbf{u}}^{1,cls}, \dots, \hat{\mathbf{u}}^{L+1,cls})), \quad (6)$$

Φ is a relation prediction function with softmax operator, which measures the relation energy of the given tuple. In detail, Φ aims to measure the similarities, using the predictions as an example:

$$\begin{aligned} \Phi(g(\hat{\mathbf{u}}^{1,cls}), \dots, g(\hat{\mathbf{u}}^{L+1,cls})) &= [q_{n_1, n_2}] \quad n_1, n_2 \in [1, \dots, L+1] \\ q_{n_1, n_2} &= \frac{\exp(d_{n_1, n_2})}{\sum \exp(d.)} \end{aligned}$$

where $d_{n_1, n_2} = KL(g(\hat{\mathbf{u}}^{n_1, cls}), g(\hat{\mathbf{u}}^{n_2, cls}))$ measures the distance. q_{n_1, n_2} denotes the relative instance-wise similarity. Finally, we pull the $[q_{n_1, n_2}]$ into vector form. $\Phi(\hat{\mathbf{u}}^{1, cls}, \dots, \hat{\mathbf{u}}^{L+1, cls})$ is calculated in the same way, with $d_{n_1, n_2} = \|\hat{\mathbf{u}}^{n_1, cls} - \hat{\mathbf{u}}^{n_2, cls}\|_2$. Since the structure has higher-order properties than a single output, it can transfer knowledge more effectively and is more suitable for consistency measures. We define the total loss by combining the Eq. 4, Eq. 5, and Eq. 6:

$$L = \sum_j \ell(j) + \lambda M(j) + \mu R(j). \quad (7)$$

where λ and μ is the balance parameter. To effectively minimize the target function with the guidance of auxiliary neighbors, each training batch consists of groups of JDs. Each group contains a central JD together with its neighbor JDs as [18]. During inference, each test JD can be conditioned by a set of neighbors from training JDs to construct JD tuples. Subsequently, we compare the relevance degree of predicted skill distribution and personal ground-truth, i.e., $sc(g(\hat{\mathbf{u}}^{cls}), \mathbf{y})$. In this paper, sc denotes the cosine similarity function (other metrics such as KL-divergence and Euclidean distance can also be utilized). Finally, we can recall candidate JDs from a large-scale JD pool according to the similarities.

4.4 Click-Through Rate Prediction for JD Ranking

The ranking stage aims to further discover user-interesting JDs from a small number of candidate JDs. As shown in Figure 2, the candidate JD and user profiles (i.e., personal skill distribution) are used to compute a click score via click predictor for personalized JD ranking. More specifically, we first train a cross-attention module to obtain joint embedding using candidate JD global embedding $\hat{\mathbf{u}}^{cls} \in \mathbb{R}^d$ and user profiles $\mathbf{s} \in \mathbb{R}^n$, where d and n represent the dimension of embedding and the total number of skills, respectively:

$$\begin{aligned} Q_c &= (\hat{\mathbf{u}}^{cls})^\top W_{Q_c}, \quad K_c = \mathbf{s}^\top W_{K_c}, \quad V_c = \mathbf{s}^\top W_{V_c}, \\ A_c &= Q_c^\top K_c, \quad \mathbf{e}^{joint} = \sigma(A_c) V_c^\top, \end{aligned} \quad (8)$$

where $Q_c \in \mathbb{R}^{1 \times d}$, $K_c \in \mathbb{R}^{1 \times n}$, $V_c \in \mathbb{R}^{1 \times n}$, and $W_{Q_c} \in \mathbb{R}^{d \times d}$, $W_{K_c} \in \mathbb{R}^{n \times n}$, $W_{V_c} \in \mathbb{R}^{n \times n}$ are learnable matrices. The activation function σ can be used as softmax here. Then, the joint embedding $\mathbf{e}^{joint} \in \mathbb{R}^d$ is fed into two fully connected layers with sigmoid function to output the predicted click probability:

$$\hat{y}_{click} = \text{sigmoid}(FC(\mathbf{e}^{joint})) \quad (9)$$

Furthermore, we define the click-through rate prediction task as a binary classification problem, where a JD-user click interaction is assigned a target value of 1, otherwise 0. Specifically, we use the cross-entropy as the loss function:

$$L = - \left(\sum_{(jd, user) \in \mathbb{R}^+} \log(\hat{y}_{click}) + \sum_{(jd, user) \in \mathbb{R}^-} \log(1 - \hat{y}_{click}) \right) \quad (10)$$

where \mathbb{R}^+ and \mathbb{R}^- are the positive and negative click records. Finally, the click scores of candidate JDs are used for personalized ranking.

5 EXPERIMENTS

In this section, we demonstrate the effectiveness of JobFormer by verifying the following problems:

- The recall and ranking performance compared with state-of-the-art baselines;
- The performance compared with various variants;
- Sensitivity analyses of parameters;
- Interpretability of JobFormer.

5.1 Data Description

We conduct our validation on a real-world talent recruitment dataset, which is provided by a high-tech company in China. To protect the privacy of candidates, all the application records are anonymized by deleting personal information. The dataset contains 37540 successful job applications. Indeed, the low acceptance ($\approx 1\%$) clearly validates the importance of job recommendation in online recruitment. In detail, we analyze some statistics of our dataset. We find that the number of applications is relatively steady. Besides, it is notable that each job posting may accept multiple users. We find that the number of job postings with respect to the number of their successfully accepted users roughly follows a long-tail distribution, and the vast majority of acceptances are controlled within 3 users. Thereby, we randomly select a user to constitute the person-job pair considering that the skill distributions of these users are similar. Finally, 15046 person-job pairs are kept in total.

5.2 Implementations

The settings of our experiments include the word embedding of item encoder, structure of the semantic-enhanced Transformer, and training details for the recall and ranking stages. For the item encoder, we first employ TextCNN with two one-dimensional convolutions (i.e., kernel sizes are 2 and 3) as the item-level encoder. The dimension of word is $d = 512$. Note that TextCNN can process an unfixed-length sequence [19]. The semantic-enhanced Transformer is with 4 layers and 8 heads, i.e., $N = 8$, including local heads $N_l = 6$ and global heads $N_g = 2$. The classifier g is with two fully connected layers. For the semantic-enhanced Transformer, we set the maximum number of items in each JD as $M = 40$, the excessive parts are removed. The number of neighbor is $L = 2$, the parameter $\lambda = 0.2, \mu = 0.4$. Following [12], we initialized all the parameters in JobFormer with uniform distribution in $[-\sqrt{6/(n_{in} + n_{out})}, \sqrt{6/(n_{in} + n_{out})}]$, where n_{in}, n_{out} denote the number of input and output units, respectively. The number of negative samples associated with positive one is 200 and 3 for the recall and ranking stages, respectively. In all experiments, the batch size is set to 32. The optimization method is Adaptive Moment Estimation (Adam), the learning rate is searched in 0.5, 0.1, 0.05, 0.01, 0.005, 0.001 to find the best settings for each task and annealed by 0.8 every 3 epochs. Finally, we set the learning rate as 0.001. The ratio of dropout is 0.1 and the maximal number of epochs is 30. We run the following experiments with the implementation of an environment on NVIDIA Tesla V100 SXM2 GPUs. The code is available at <https://github.com/njustkmg/TKDD24-JobFormer>.

5.3 Baseline Approaches

To verify the effectiveness of JobFormer, we compare it with various baseline methods: 1) Traditional word embedding-based recommendation methods, i.e., NPA [44], NAML [43], NRMS [45], and CNE-SUE [28]. 2) BERT-based pre-training recommendation methods, i.e., UNBERT [57] and MINER [24]. 3) CTR ranking methods, i.e., MaskNet [41] and FRNet [38]:

- **NPA**: a news recommendation method which adopts the personalized attention mechanism to model text semantic information;
- **NAML**: a news recommendation method with attentive multi-view learning to obtain news representation and attentive pooling to learn user representation;

Table 1. Experimental results of different approaches on JD recall task. Higher Recall and NDCG rates mean better performance.

Methods	Metric									
	Recall@20	Recall@40	Recall@60	Recall@80	Recall@100	NDCG@20	NDCG@40	NDCG@60	NDCG@80	NDCG@100
NPA	61.76	77.68	81.78	87.39	92.39	28.18	31.46	32.18	33.09	33.86
NAML	62.46	78.68	87.19	91.89	95.20	32.29	36.33	37.96	38.02	38.72
NRMS	67.67	81.18	86.69	88.79	93.09	33.12	35.45	37.88	38.33	38.72
CNE-SUE	61.66	76.68	87.99	93.99	96.60	27.70	30.53	31.50	31.84	32.50
UNBERT	51.43	62.14	69.34	76.95	85.36	23.77	27.66	29.71	30.76	31.27
MINER	59.36	76.08	85.29	92.89	96.60	27.94	31.34	32.97	34.21	34.78
JobFormer	69.98	83.98	90.97	95.01	97.29	34.30	37.21	38.42	39.14	39.47

- **NRMS**: a news recommendation method which utilizes multi-head self-attention networks to extract fine-grained representations from the news title and user history respectively;
- **CNE-SUE**: a news recommendation framework consisting of collaborative news encoding and structural user encoding to enhance news and user representation learning;
- **UNBERT**: a BERT-based approach which leverages the pre-trained model to enhance textual representation and capture multi-grained signals at both word-level and news-level;
- **MINER**: a BRET-based pre-training model which employs a poly attention scheme to learn multiple interest vectors for each user;
- **MaskNet**: a CTR ranking framework which takes advantage of instance-guided mask to solve the inefficiency of the feedforward neural network in CTR prediction;
- **FRNet**: a CTR prediction model with a feature refinement module to learn context-aware feature representations by integrating the original and complementary feature representations with bit-level weights.

Given the person-job data, the comparison methods for the recall stage consider the JD as input and personal skill distribution as ground-truth for training. For the ranking stage (i.e., CTR prediction task), we utilize the candidate JDs along with the click data to predict the probability of positive feedback, i.e. click, taking place on a JD. To measure the performances, we consider two aspects of evaluations: 1) JD recall, which aims to verify that the personal skill distribution can find the candidate JDs from a large-scale JD pool. 2) JD ranking, which aims to discover user-interesting JDs from candidate JD pool according to click scores.

5.4 JD Recall Performance

To verify the effectiveness of JobFormer in the JD recall task, we first adopt the widely-used metrics in recommendation systems [21], i.e., Recall@ K and NDCG@ K . Recall@ K counts the ratio of successfully predicted JDs among top- K JDs to all positive JDs for each user query; NDCG@ K represents Normalized Discounted Cumulative Gain at K , which places an inverse log reward on all positions that hold a relevant JD. We repeat every experiment on each method 5 times and record the average results in Table 1. Referring to this table, we have several findings: 1) Traditional word embedding-based recommendation methods outperform BERT-based pre-training recommendation methods. This indicates that word embedding can better encode the JD, mainly due to the JD structure and the relatively small scale of our dataset. Thereby, we introduce the word embedding and TextCNN to encode JD texts in model design. 2) NRMS performs the best among all the word embedding-based methods on most criteria. The reason lies in that NRMS adopts deep models such as Transformer to learn contextual representations of JD items by capturing their interactions. This phenomenon indicates that multi-head self-attention can more effectively mine JD semantic information. 3) JobFormer achieves the best performance compared with other baselines on all criteria, e.g., JobFormer exceeds NRMS 2.31% on Recall@20, 1.18% on NDCG@20. The results validate the effectiveness of designed modules (e.g., local-global Transformer) for processing JD.

Table 2. Experimental results of different approaches on JD ranking task. Higher AUC and MRR mean better performance.

Methods	Metric	
	AUC	MRR
NPA	75.52	28.53
NAML	75.48	28.39
NRMS	76.88	29.11
CNE-SUE	74.87	28.06
UNBERT	68.16	19.45
MINER	73.21	26.33
FRNet	76.54	29.18
MaskNet	74.93	27.92
JobFormer	77.58	29.82

Table 3. Ablation results on our JobFormer.

Methods	Metric									
	Recall@20	Recall@40	Recall@60	Recall@80	Recall@100	NDCG@20	NDCG@40	NDCG@60	NDCG@80	NDCG@100
LSTM+	62.11	79.45	86.71	91.05	93.02	29.17	31.51	33.11	34.28	34.81
TextCNN+	67.74	83.36	90.12	93.95	96.11	32.44	35.69	36.94	37.57	37.90
BERT+	67.79	83.08	89.86	94.18	96.14	31.87	35.04	36.25	36.94	37.39
local-global+	68.85	83.66	90.62	94.95	96.98	33.06	35.92	37.56	37.92	38.29
w/o M	69.45	83.26	90.52	94.65	97.17	33.21	36.81	38.11	38.52	39.12
w/o R	69.65	82.86	90.52	95.16	97.27	34.10	36.94	38.14	38.70	39.17
JobFormer	69.98	83.98	90.97	95.01	97.29	34.30	37.21	38.42	39.14	39.47

5.5 JD Ranking Performance

Next, we further carry out the JD ranking task to discover user-interesting JDs from candidate JDs. In detail, we rank the candidate JDs according to the predicted click scores between candidate JD global representation and user profiles (i.e., personal skill distribution) for a personalized job recommendation. Following [21], we adopt AUC and MRR to evaluate JD ranking performance, and the comparison results are shown in Table 2. We acquire a similar conclusion to the JD recall task that JobFormer achieves the best performance compared with other baselines. Note that for CTR ranking methods (i.e., FRNet and MaskNet), we utilize the raw samples of candidate JDs recalled by JobFormer to predict the click scores. JobFormer continues to perform best, mainly because JD representations obtain rich intra-job and inter-job information via the semantic-enhanced Transformer in the recall stage.

5.6 Ablation Study

To verify the effectiveness of each module, we conduct more ablation studies, including: 1) LSTM+, TextCNN+, and BERT+, we utilize LSTM/TextCNN/BERT as item-level encoder, and then input the item representations to the Transformer encoder, which is trained with KL divergence as the loss function. 2) local-global+, we adopt the TextCNN as item-level encoder, and then input the item representations to the semantic-enhanced Transformer encoder with local-global attention, which is trained with KL divergence as the loss function. 3) w/o M , we remove the skill correlation enhancement $M(\cdot)$ (see Eq. 5) in JobFormer. 4) w/o R , we remove the relation consistency $R(\cdot)$ (see Eq. 6) in JobFormer. Table 3 records the results, which reveal that: 1) TextCNN+ outperforms LSTM+,

Table 4. Experimental results of different local-global heads and neighbors on JD recall task. (JF: JobFormer)

Methods	Metric									
	Recall@20	Recall@40	Recall@60	Recall@80	Recall@100	NDCG@20	NDCG@40	NDCG@60	NDCG@80	NDCG@100
JF ($N_l = 4, N_g = 4$)	68.34	83.16	90.02	94.45	96.47	33.60	36.50	37.64	38.36	38.59
JF ($N_l = 5, N_g = 3$)	69.15	84.03	90.32	94.75	97.31	33.83	36.93	37.94	38.73	39.05
JF ($N_l = 6, N_g = 2$)	69.98	83.98	90.97	95.01	97.29	34.30	37.21	38.42	39.14	39.49
JF ($N_l = 7, N_g = 1$)	68.85	83.77	90.24	94.64	97.09	34.04	36.92	38.03	38.79	39.07
JF ($L = 1$)	69.25	83.26	90.02	94.15	96.27	33.02	36.07	37.26	37.89	38.29
JF ($L = 2$)	69.98	83.98	90.97	95.01	97.29	34.30	37.21	38.42	39.14	39.49
JF ($L = 3$)	69.95	84.23	90.52	95.03	97.07	33.65	36.62	38.22	38.28	39.03
JF ($L = 4$)	68.75	83.56	90.51	94.45	96.87	32.89	35.97	37.22	37.73	38.19
JF ($L = 5$)	67.84	83.16	90.22	94.05	96.91	32.27	35.42	36.56	37.22	37.70

confirming that TextCNN more effectively models JD items as it can process unfixed-length sequences to capture more information [19], whereas LSTM typically requires fixed-length sequences. Additionally, the pre-trained model BERT does not yield substantial performance improvements and actually falls behind TextCNN in NDCG metrics, mainly due to the item-based JD structure and the relatively small scale of our dataset. 2) local-global+ can improve performance, which indicates that local-global attention can effectively model the intra-job and inter-job information. 3) w/o M performs superior to the local-global+, which shows that the relation consistency can further improve the recall performance. 4) w/o R performs best in variants, which indicates that it is crucial to consider skill correlations. 5) JobFormer performs best compared with all the variants, which reveals that all the designed modules can promote the prediction of skill distribution in the recall stage.

5.7 Parameter Analysis

Influence of Local-Global Heads. Considering that JobFormer with various local heads and global heads can have different emphasis on intra-job information and inter-job information, we fix the total heads as $N = 8$ and tune the local-global heads in $\{(N_l = 4, N_g = 4), (N_l = 5, N_g = 3), (N_l = 6, N_g = 2), (N_l = 7, N_g = 1)\}$, to empirically investigate the impact on JD recall performance. The top section of Table 4 depicts the results, the performance (including Recall and NDCG) of JobFormer firstly increases and then decreases. The JobFormer acquires best performance when local-global is $(N_l = 6, N_g = 2)$. This phenomenon confirms that the intra-job information is essential for learning skill-aware representation, but the neighbor JD can also provide additional supplementary information, i.e., the performance of $(N_l = 7, N_g = 1)$ is worse than $(N_l = 6, N_g = 2)$. Therefore, we need proper attention to neighbor information.

Number of Neighbors. To validate the influence of neighbors on the JD representation learning, we incorporate neighbors with different numbers (i.e., $L \in \{1, 2, 3, 4, 5\}$) to empirically investigate the impact of neighbors on JD recall task. Note that we fix the local-global heads as $(N_l = 6, N_g = 2)$. The bottom section of Table 4 depicts the results, the performance of JobFormer also increases firstly and then decreases on various criteria. The reason is that more neighbors can even bring noise and the over-smooth problem.

Influence of Balance Parameters. To explore the influence of hyper-parameters, we tune the $\lambda, \mu \in \{0.1, 0.2, 0.4, 0.6, 0.8\}$ to conduct more experiments. The top section of Figure 3 depicts the performance of different λ . With the increase of λ , the results of JD recall first increase and then decrease. This shows that skill correlation enhancement has a promoting effect, but over-considering skill correlation may introduce bias for prediction. The bottom section of Figure 3 depicts the performance with different μ . The results of JD recall first increase with the increase of μ , and then decrease after $\mu > 0.4$. This shows that relation consistency actually has a promoting effect, but the label distribution prediction loss (i.e., KL-divergence) still has more contributions to model learning.

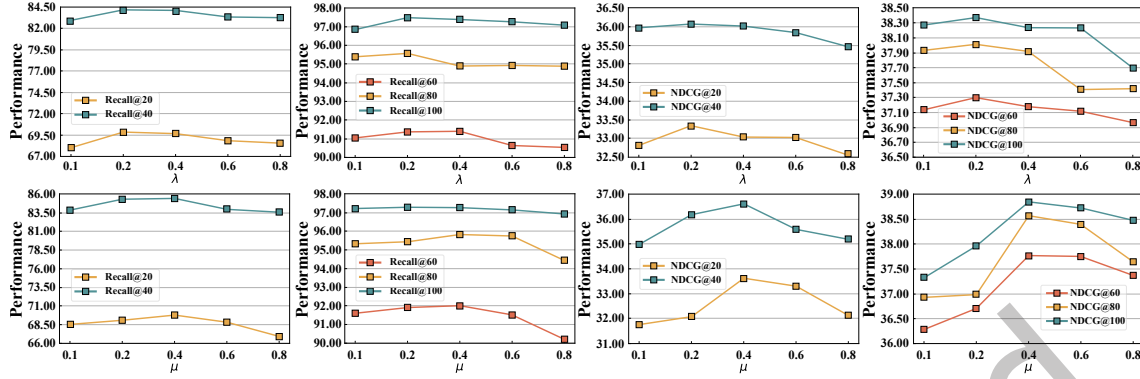


Fig. 3. Influence of Balance Parameters. The figures in the first row are the results of λ , and the second row gives the results of μ .

Table 5. Experimental results of different approaches on public dataset.

Methods	Metric									
	Recall@20	Recall@40	Recall@60	Recall@80	Recall@100	NDCG@20	NDCG@40	NDCG@60	NDCG@80	NDCG@100
NPA	18.21	28.84	37.64	50.86	60.46	7.04	9.13	10.68	12.83	14.39
NAML	18.49	29.26	36.83	49.27	60.95	6.55	8.73	10.07	12.08	13.85
NRMS	18.97	26.88	38.83	47.27	60.59	7.48	9.26	11.33	12.69	14.47
CNE-SUE	17.68	28.44	36.85	48.96	61.01	7.12	9.23	10.78	12.73	14.32
UNBERT	12.89	24.08	35.21	43.85	56.03	4.53	6.78	8.77	10.34	12.07
MINER	15.20	27.65	37.87	48.40	60.83	5.63	8.15	10.12	11.68	13.58
JobFormer	19.96	32.71	41.96	52.67	64.16	7.53	10.05	11.79	13.83	15.04

5.8 Results on Public Dataset

To validate the generalization of JobFormer, we conduct more experiments on a commonly used public dataset, i.e., SBU 3DFE dataset [56]. Specifically, the SBU 3DFE dataset contains 2,500 facial expression images. A 243-dimensional feature vector is extracted from each image by the method of Local Binary Patterns (LBP). Each image is scored by 23 persons on the 6 basic emotion labels (i.e., happiness, sadness, surprise, fear, anger, and disgust) with a 5-level scale. We score and normalize them into a label distribution over all the 6 emotion labels following [56]. Table 5 records the results, and we can acquire similar analyses as the private person-job dataset that JobFormer can also achieve the best recall performance, which validates the effectiveness of JobFormer on the recommendation task.

5.9 Case Study

JD Recall Visualization. Firstly, we evaluate whether JobFormer could effectively recall user-related JDs from a large-scale JD pool. Figure 4 shows the recall results of top-3 JDs given the corresponding user query. We find that most of the recalled candidate JDs according to the cosine similarity are exactly matched (title displayed in red font) with the ground-truth. Other outputs are also reasonable. Using the first row of Figure 4 as an example, the 2nd and 3rd candidate JDs also have corresponding keywords such as “front-end”, “W3C standards” are related to skill “HTML5”, and “Spring MVC”, “MyBatis” are related to skill “Java”. Meanwhile, other skills (e.g., “MySQL”) are also required for Front-end Development Engineer and Java R&D Engineer. Besides, we also exhibit

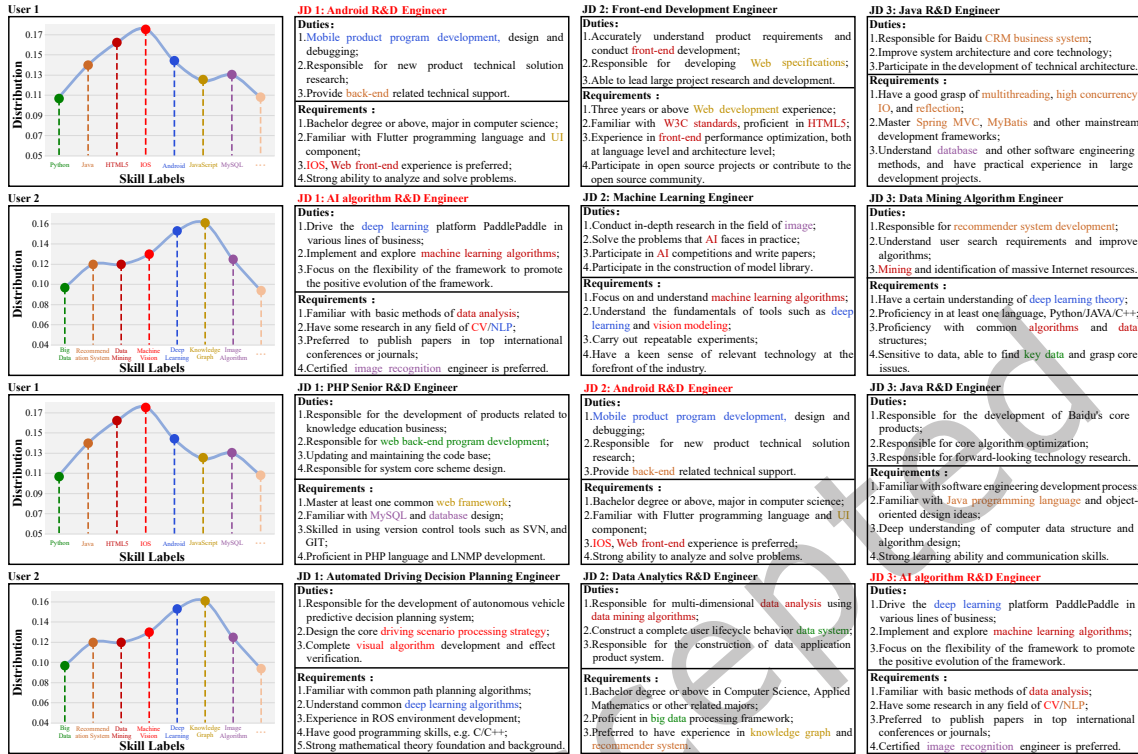


Fig. 4. (Best viewed in color when zoomed in.) Qualitative success results of JD recall given user queries. For each user query, we show the top-3 ranked JD text. The first two rows exhibit the results of JobFormer, and the last two rows give the results of the state-of-the-art NRMS model. We observe that our JobFormer can find the correct results (i.e., red marked) in the first-ranked JDs, and NRMS is inferior to JobFormer.

the results of the state-of-the-art recommendation model NRMS, which is inferior to JobFormer in that it only matches the correct JD at the 2nd or 3rd location.

Interpretable CTR Prediction. To verify whether JobFormer could highlight the most critical skills that have strong contributions to CTR prediction, we present the attention weights of three different JDs with their corresponding user profiles, the results are recorded in Figure 5. In detail, we take the cumulative attention weights of each dimension of [CLS] token as the level of emphasis the entire JD places on personal skills. Considering the page limitation, we only show the top-10 skills for interpretability analysis.

From the figures, we find that: 1) the skill-aware JD representation can accurately capture most of the relevant skills. For example, the skills “Machine Learning”, “Automatic Driving”, “Data Mining”, “Image Algorithm” and “Deep Learning” directly correspond to the skills of user1, which have the higher attention weights (i.e., 9.61, 9.54, 7.94, 7.19, 6.78). Other skills are also highly relevant to Vision Algorithm Engineer. Furthermore, the predicted click score (0.87) means that JobFormer is more likely to recommend JD1 to user1. 2) JobFormer can further efficiently identify JDs that mismatch with the current user. For example, the predicted click score in Case3 (0.38) is much lower than Case1 and Case2, because JD1 is intended for a computer vision position, while user2 is involved in the software development industry. Therefore, the obtained skill attention weights are relatively low.

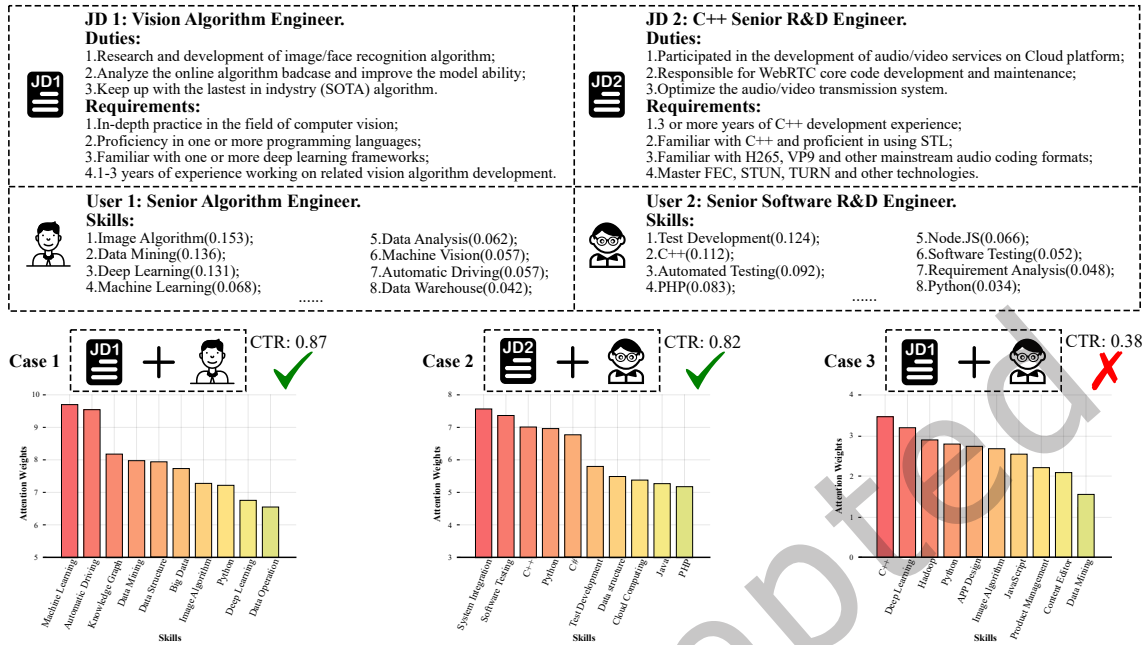


Fig. 5. (Best viewed in color when zoomed in.) The example of interpretable CTR prediction.

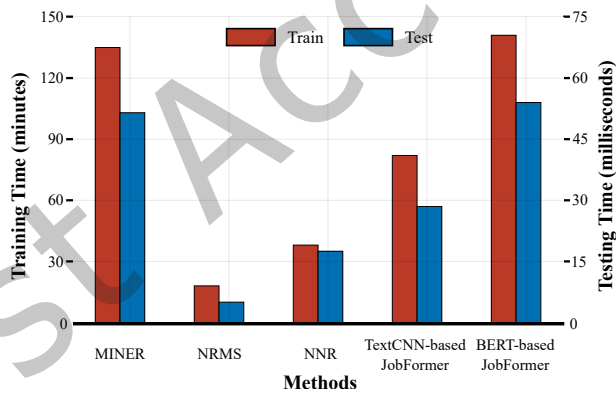


Fig. 6. The training and testing efficiency of JobFormer and compared deep models on JD recall task.

5.10 Computational Efficiency

To evaluate the efficiency, we present the computational times of JobFormer alongside other deep models on the JD recall task. Specifically, all experiments are conducted on a server with a 2-core CPU@2.40GHz, 160GB RAM, and an NVIDIA Tesla V100 SXM2 GPU. As shown in Figure 6, the training time of TextCNN-based JobFormer is 82 minutes, significantly shorter than the comparison BERT-based recommendation method MINER. In addition, employing TextCNN as a text encoder enhances computational efficiency without compromising recall performance. Although the training time of our method is slightly higher than NRMS and NNR, our performance

outperforms these two methods. Furthermore, after the training process, the average cost of each instance in the testing set is 28ms. This validates that our model can be effectively used in the real-world management analysis system.

6 CONCLUSION

In this paper, we propose a skill-aware representation method (JobFormer) that can utilize job descriptions and user profiles (personal skill distribution) to accomplish personalized job recommendation. Our method contains a recall stage and a ranking stage. In the recall stage, we first leverage the semantic-enhanced Transformer to parse JDs and guide the representation learning of JD via personal skill distribution. In detail, we design an encoder with the local-global attention mechanism to mine the intra-job and inter-job dependencies from JD tuples. With the skill-aware JD representation, we can recall a portion of JDs relevant to the user as a candidate set from a large-scale JD pool according to the JD-user cosine similarity. In the ranking stage, candidate JDs are further computed with user profiles for CTR prediction and ranked for personalized JD recommendation. Experiments on real-world and public datasets can well demonstrate the effectiveness and interpretability of JobFormer.

7 ACKNOWLEDGMENTS

This work is partially supported by National Key RD Program of China (2022YFF0712100), NSFC (62276131), Natural Science Foundation of Jiangsu Province of China under Grant (BK20240081), the Fundamental Research Funds for the Central Universities (No.30922010317, No.30923011007). Zhihao Guan and Jia-Qi Yang contributed equally to this work.

REFERENCES

- [1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016).
- [2] 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. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 65–74.
- [3] Shuqing Bian, Wayne Xin Zhao, Yang Song, Tao Zhang, and Ji-Rong Wen. 2019. Domain adaptation for person-job fit with transferable deep global match network. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 4810–4820.
- [4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishikesh Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.
- [5] Colin Cherry and Chris Quirk. 2008. Discriminative, syntactic language modeling through latent svms. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas: Research Papers*. 65–74.
- [6] Yi-Chi Chou and Han-Yen Yu. 2020. Based on the application of AI technology in resume analysis and job recommendation. In *2020 IEEE International Conference on Computational Electromagnetics (ICCEM)*. IEEE, 291–296.
- [7] Ketki V Deshpande, Shimpei Pan, and James R Foulds. 2020. Mitigating demographic Bias in AI-based resume filtering. In *Adjunct publication of the 28th ACM conference on user modeling, adaptation and personalization*. 268–275.
- [8] Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. 2013. Toward the next generation of recruitment tools: an online social network-based job recommender system. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 821–828.
- [9] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.
- [10] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612* (2017).
- [11] Xin Geng and Rongzi Ji. 2013. Label Distribution Learning. In *ICDM*. TX, USA, 377–383.
- [12] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, Vol. 9. Sardinia, Italy, 249–256.
- [13] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. Las Vegas, NV, 770–778.
- [15] Sheng-Jun Huang and Zhi-Hua Zhou. 2012. Multi-Label Learning by Exploiting Label Correlations Locally. In *AAAI*, Vol. 26. Toronto, Canada, 949–955.
- [16] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*. Lille, France, 448–456.
- [17] Xiuyi Jia, Weiwei Li, Junyu Liu, and Yu Zhang. 2018. Label Distribution Learning by Exploiting Label Correlations. In *AAAI*. New Orleans, Louisiana, 3310–3317.
- [18] Xiuyi Jia, Zechao Li, Xiang Zheng, Weiwei Li, and Sheng-Jun Huang. 2021. Label Distribution Learning with Label Correlations on Local Samples. *IEEE TKDE* 33 (2021), 1619–1631.
- [19] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *ACL*. Baltimore, MD, 655–665.
- [20] Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized job recommendation system at linkedin: Practical challenges and lessons learned. In *Proceedings of the eleventh ACM conference on recommender systems*. 346–347.
- [21] Walid Krichene and Steffen Rendle. 2021. On Sampled Metrics for Item Recommendation (Extended Abstract). In *IJCAI*. Virtual Event, 4784–4788.
- [22] Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. 2008. Supervised and traditional term weighting methods for automatic text categorization. *IEEE transactions on pattern analysis and machine intelligence* 31, 4 (2008), 721–735.
- [23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
- [24] Jian Li, Jieming Zhu, Qiwei Bi, Guohao Cai, Lifeng Shang, Zhenhua Dong, Xin Jiang, and Qun Liu. 2022. MINER: multi-interest matching network for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*. 343–352.
- [25] Guangquan Lu, Jiangzhang Gan, Jian Yin, Zhiping Luo, Bo Li, and Xishun Zhao. 2020. Multi-task learning using a hybrid representation for text classification. *Neural Computing and Applications* 32 (2020), 6467–6480.
- [26] Yao Lu, Sandy El Helou, and Denis Gillet. 2013. A recommender system for job seeking and recruiting website. In *Proceedings of the 22nd International Conference on World Wide Web*. 963–966.
- [27] David JC MacKay and David JC Mac Kay. 2003. *Information theory, inference and learning algorithms*. Cambridge university press. 34–36 pages.
- [28] Zhiming Mao, Xingshan Zeng, and Kam-Fai Wong. 2021. Neural news recommendation with collaborative news encoding and structural user encoding. *arXiv preprint arXiv:2109.00750* (2021).
- [29] Peter Matthews. 2001. *A Short History of Structural Linguistics*. Cambridge University Press.
- [30] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*. Haifa, Israel, 807–814.
- [31] Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2021. Uni-FedRec: A unified privacy-preserving news recommendation framework for model training and online serving. *arXiv preprint arXiv:2109.05236* (2021).
- [32] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing Person-Job Fit for Talent Recruitment: An Ability-aware Neural Network Approach. In *SIGIR*. Ann Arbor, MI, 25–34.
- [33] Dazhong Shen, Hengshu Zhu, Chen Zhu, Tong Xu, Chao Ma, and Hui Xiong. 2018. A joint learning approach to intelligent job interview assessment. In *IJCAI*, Vol. 18. 3542–3548.
- [34] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15 (2014), 1929–1958.
- [35] The Insight Partners. 2023. *Online Recruitment Market*. <https://www.theinsightpartners.com/reports/online-recruitment-market>
- [36] Bing Tian, Yong Zhang, Jin Wang, and Chunxiao Xing. 2019. Hierarchical Inter-Attention Network for Document Classification with Multi-Task Learning. In *IJCAI*. 3569–3575.
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. Long Beach, CA, 5998–6008.
- [38] Fangye Wang, Yingxu Wang, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Enhancing CTR prediction with context-aware feature representation learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 343–352.
- [39] Sida I Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 90–94.
- [40] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 world wide web conference*. 1543–1552.
- [41] Zhiqiang Wang, Qingyun She, and Junlin Zhang. 2021. MaskNet: Introducing feature-wise multiplication to CTR ranking models by instance-guided mask. *arXiv preprint arXiv:2102.07619* (2021).

- [42] Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. 2020. Acquiring knowledge from pre-trained model to neural machine translation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 9266–9273.
- [43] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. Neural news recommendation with attentive multi-view learning. *arXiv preprint arXiv:1907.05576* (2019).
- [44] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. NPA: neural news recommendation with personalized attention. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2576–2584.
- [45] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019. Neural news recommendation with multi-head self-attention. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 6389–6394.
- [46] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Two birds with one stone: Unified model learning for both recall and ranking in news recommendation. *arXiv preprint arXiv:2104.07404* (2021).
- [47] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2022. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2022), 4425–4445.
- [48] Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. 2019. Tied transformers: Neural machine translation with shared encoder and decoder. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5466–5473.
- [49] Rui Yan, Ran Le, Yang Song, Tao Zhang, Xiangliang Zhang, and Dongyan Zhao. 2019. Interview choice reveals your preference on the market: To improve job-resume matching through profiling memories. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 914–922.
- [50] Yang Yang, Zhao-Yang Fu, De-Chuan Zhan, Zhi-Bin Liu, and Yuan Jiang. 2021. Semi-Supervised Multi-Modal Multi-Instance Multi-Label Deep Network with Optimal Transport. *IEEE TKDE* 33 (2021), 696–709.
- [51] Yang Yang, Hongchen Wei, Hengshu Zhu, Dianhai Yu, Hui Xiong, and Jian Yang. 2024. Exploiting Cross-Modal Prediction and Relation Consistency for Semisupervised Image Captioning. *IEEE Transactions on Cybernetics* 54, 2 (2024), 890–902.
- [52] Yang Yang, De-Chuan Zhan, Yi-Feng Wu, Zhi-Bin Liu, Hui Xiong, and Yuan Jiang. 2021. Semi-Supervised Multi-Modal Clustering and Classification with Incomplete Modalities. *IEEE Transactions on Knowledge and Data Engineering* 33, 2 (2021), 682–695.
- [53] Yang Yang, Chubing Zhang, Xin Song, Zheng Dong, Hengshu Zhu, and Wenjie Li. 2023. Contextualized Knowledge Graph Embedding for Explainable Talent Training Course Recommendation. *ACM Trans. Inf. Syst.* 42, 2 (2023), 33:1–33:27.
- [54] Kaichun Yao, Chuan Qin, Hengshu Zhu, Chao Ma, Jingshuai Zhang, Yi Du, and Hui Xiong. 2021. An interactive neural network approach to keyphrase extraction in talent recruitment. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2383–2393.
- [55] Kaichun Yao, Jingshuai Zhang, Chuan Qin, Peng Wang, Hengshu Zhu, and Hui Xiong. 2022. Knowledge Enhanced Person-Job Fit for Talent Recruitment. In *ICDE. Virtual*, 3467–3480.
- [56] Lijun Yin, Xiaozhou Wei, Yi Sun, Jun Wang, and Matthew J Rosato. 2006. A 3D facial expression database for facial behavior research. In *FGR*. Southampton, UK, 211–216.
- [57] Qi Zhang, Jingjie Li, Qinglin Jiá, Chuyuan Wang, Jieming Zhu, Zhaowei Wang, and Xiuqiang He. 2021. UNBERT: User-News Matching BERT for News Recommendation. In *IJCAI*. 3356–3362.
- [58] XianXing Zhang, Yitong Zhou, Yiming Ma, Bee-Chung Chen, Liang Zhang, and Deepak Agarwal. 2016. Glmix: Generalized linear mixed models for large-scale response prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 363–372.
- [59] Chen Zhu, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li. 2018. Person-Job Fit: Adapting the Right Talent for the Right Job with Joint Representation Learning. *ACM TMIS*. 9 (2018), 12:1–12:17.

Received 17 October 2023; revised 21 July 2024; accepted 21 October 2024