Knowledge Enhanced Person-Job Fit for Talent Recruitment

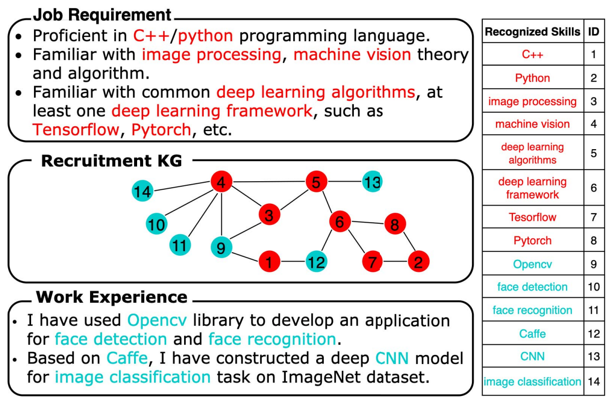
Kaichun Yao[†], Jingshuai Zhang[†], Chuan Qin[†], Peng Wang[†],
Hengshu Zhu[†*] and Hui Xiong[‡]
[†]Baidu Talent Intelligence Center, Baidu Inc, China
[‡]Artificial Intelligence Thrust, The Hong Kong University of Science and Technology (Guangzhou), China
[‡]Department of Computer Science & Engineering, The Hong Kong University of Science and Technology, China
yaokaichun@outlook.com, {zhangjingshuai, wangpeng40}@baidu.com,
{chuanqin0426, zhuhengshu, xionghui}@gmail.com

***Abstract*—As an essential task of talent recruitment, person-job fit aims to measure the matching degree between talent qualification and the job requirements of a position. Existing studies usually formulate this task as a long text matching problem with a focus on learning effective representations of both job postings and resumes. However, it is commonly known that there exists a semantic gap between textual job postings and textual resumes. Therefore, in this paper, we study how to improve person-job fit by bridging this semantic gap with the help of prior knowledge. To this end, we first design a distantly supervised skill extraction model to identify the skill entities from the given job postings and resumes using only unlabeled data and skill entity dictionaries. The identified skill entities will be used to construct a skill knowledge graph (KG) on the global corpus, which can provide the prior knowledge. Also, we propose a knowledge enhanced person-job fit approach for talent recruitment. Here, we model job postings and resumes as two graphs and fuse the prior external knowledge into the graph representation learning. Specifically, we first build the graphs from job posting and resume text. Then, we design a knowledge-aware graph encoder that can not only capture the contextual word relationships within each job posting or resume, but also incorporate the prior knowledge into node representation learning. In addition, we propose an interactive learning method to perform effective graph matching in both graph-level and node-level, respectively. Meanwhile, a multi-task learning strategy is introduced to facilitate the graph representation learning. Finally, extensive experiments conducted on real-world datasets have clearly validated the effectiveness of our approaches compared with state-of-the-art baselines.**

***Index Terms*—talent recruitment, person-job fit, graph representation learning, multi-task learning**

I. Introduction

With the rapid development of online recruitment services, such as Linkedin, Glassdoor and BossZhipin, a huge number of job application records have been accumulated. For example, it has been reported that there were more than 100 million job applications made on LinkedIn each month [1]. Indeed, these sufficient recruitment data have enabled a new paradigm of building intelligent talent recruitment systems. Along this line, an essential research direction, namely person-job fit, has been attracting a wide range of attention in both academia and industry. It aims to measure the matching degree between talent qualification and job requirements, in order to adapt the right candidates to the right positions [1, 2].



Job Requirement
- Proficient in C++/python programming language.
- Familiar with image processing, machine vision theory and algorithm.
- Familiar with common deep learning algorithms, at least one deep learning framework, such as Tensorflow, Pytorch, etc.

Work Experience
- I have used Opencv library to develop an application for face detection and face recognition.
- Based on Caffe, I have constructed a deep CNN model for image classification task on ImageNet dataset.

Recognized Skills	ID
C++	1
Python	2
image processing	3
machine vision	4
deep learning algorithms	5
deep learning framework	6
Tesorflow	7
Pytorch	8
Opencv	9
face detection	10
face recognition	11
Caffe	12
CNN	13
image classification	14

Fig. 1. A motivating example of person-job fit. The right table shows the recognized skill entities from a job requirement and a work experience with corresponding ID number. The node with an ID number will be connected in a skill knowledge graph shown in the left middle rectangle. Note that we do not distinguish the specific relations between two skill entities in this example.

In the literature, earlier studies of person-job fit mainly focus on utilizing collaborative filtering algorithms to recommend job for candidates or recommend resumes to recruiters [3, 4]. However, these approaches largely rely on the extensive handcrafted features so that easily lead to expensive labour cost and the inefficient, inaccurate, and subjective judgments. Recently, the neural network based approaches [1, 5, 6, 7, 8] are consequently proposed for person-job fit, formulating it as a long text semantic matching problem. These approaches mainly focus on learning effective representations of both job postings and candidate resumes from historical application records using hierarchical convolutional neural network (CNN) [1] or recurrent neural network (RNN) [5]. However, there still exist two open challenges for these approaches. First, they only process the text paragraphs including the working experience and job requirements, and fail to comprehend the involved prior knowledge in the text

* Hengshu Zhu is the corresponding author.
[1]https://www.businessofapps.com/data/linkedin-statistics

like the relations between skill entities. Figure 1 gives a motivating example to illustrate this problem. For instance, "*Machine Vision*" is required in the job posting while there is no mention for it in the candidate's work experience. In reality, many skill entities like "*Opencv*" and "*Face Recognition*" in the work experience are strongly related to "*Machine Vision*". Obviously, external knowledge can reduce the semantic gap between different types of descriptions. Second, Although these models adopt a variety of mechanisms to achieve the interpretability for person-job fit, they are still difficult to interpret from the perspective of explicit knowledge relevance.

To address the above challenges, here we propose a **k**nowledge enhanced **g**raph **r**epresentation learning **m**odel, namely *KGRM*, for person-job fit, which can not only capture the contextual word relationships within each job posting or resume, but also fuse the external prior knowledge into node representation learning. To be specific, we first design a distantly supervised skill extraction model to identify the skill entities from the given job postings and resumes using only unlabeled data and skill entity dictionaries. The identified skill entities will be used to construct a skill relation knowledge graph on the global corpus, which can provide the prior knowledge for certain skill entity in single document (i.e., a job posting or a resume). Then, instead of processing textual content, we construct a graph, consisting of a word subgraph and a skill knowledge subgraph, for each job posting or resume, and propose a knowledge aware graph encoder to fuse the two subgraphs and learn the graph representation. In addition, we design an interactive matching learning method to conduct more effective graph matching in both global graph-level and local node-level, which can achieve good interpretability in the matching process. Meanwhile, we utilize a multi-task learning strategy with classification task for facilitating the graph representation learning. Finally, we conduct extensive experiments on real-world datasets with two typical tasks, namely skill entity extraction and person-job fit. The experimental results clearly validate the effectiveness of our approaches compared with a number of state-of-the-art baselines.

Specifically, the contributions of this paper can be summarized as follows:

- We propose a new research direction of person-job fit, by designing a knowledge enhanced graph matching model that incorporates the prior knowledge into the graph representation learning process.
- We propose a novel knowledge aware graph encoder and a multi-task learning framework to effectively conduct the graph representation learning, and design an interactive matching approach to perform more effective graph matching learning.
- We evaluate our approaches with extensive experiments on real-world recruitment datasets, which consistently outperform a number of state-of-the-art baselines with a significant margin.

II. Related Work

The related works can be grouped into three categories, namely *skill entity extraction*, *graph representation learning* and *person-job fit*.

A. Skill Entity Extraction

As an fundamental task in talent recruitment, skill entity extraction plays an important role in many downstream tasks, such as person-job fit [5], interview assessment [9] and market trend analysis [10]. Actually, skill entity extraction [11] can be regarded as a task of Named Entity Recognition (NER). Recently, neural-based NER approaches have gained huge attention and achieved promising performances [12, 13, 14]. These approaches often leverage the neural network models such as BiLSTM [15] or Bert [16] to learning word-level semantic information and then apply a CRF layer to infer the NER tags. However, they are supervised learning based approaches, which require a large amount of human annotated NER training corpora. Unfortunately, the training corpora are usually expensive to obtain.

The above issue motivates a lot of work on distantly supervised NER. A typical line of effort focuses on introducing external knowledge or dictionary to automatically annotate partial entities in training data. For instance, Yang et al. [17], Shang et al. [18] and Cao et al. [19] applied Partial-CRFs [20] to model incomplete annotations for specific domains by leveraging seed annotations or a domain dictionary. Except partial-CRFs technique for incomplete annotations, Peng et al. [21] leveraged PU learning to perform NER task using only unlabeled data and named entity dictionaries. Different from their work, based on the PU learning, we proposed a distantly supervised skill entity extraction method with a transfer-learning technique to further improve the performance.

B. Graph Representation Learning

In recent years, graph neural networks (GNN) has gained more and more attention. GNN based approaches [22, 23, 24, 25] have been consequently proposed for graph-structured data and achieved great success on various real-world applications [26, 27, 28].

The main goal of GNN is to learn the representation of the graph in the node level or graph level. To learn more effective graph representations for various downstream tasks, a graph convolutional network (GCN) [29] is proposed to learn node representations by performing convolutional operations on graph-structured data. Although GCN achieves good performance, it takes a fixed edge connections as the input graph, and can not dynamically adjust the connectivity information during the graph learning process. Motivated by this observation, the graph attention networks (GAT) is proposed to attend different weights on different neighborhoods by leveraging masked self-attentional layers. In order to learn node representation more effectively on a large graph, GraphSAGE [23] is proposed to conduct inductive node embedding using a sample and aggregate method. However, these GNN models including GCN, GAT, and GraphSage are designed for homogeneous graphs. To capture the heterogeneity of nodes in graphs, HAN [30] is proposed to update node semantic feature based on both node-level aggregation and meta-path level aggregation. For a graph with various node types but single edge types, HGAT [31] is proposed for encoding heterogeneous graph from type-level learning and node-level learning. To achieve more accurate and explainable recommendation, a knowledge graph attention network (KGAT) [32] is proposed to employ knowledge-aware attention mechanism to learn the weight of each neighbor and then update a node's representation with the learned weighted sum over its neighbors. Different from above approaches, our graph model not only captures the contextual word relationships within each document, but also fuses external prior knowledge.

C. Person-Job Fit

As an important task in talent recruitment, person-job fit has been widely studied in the literature. Earlier methods focus more on job recommendation problem [3, 4, 33] on the online recruitment platforms like LinkedIn. Based on the user information including education and working experience and the interactive information such as clicking and browsing time for a user on each job post, some recommendation algorithms like collaborative filtering have been adapted to recommend jobs to users.

Recently, with the emerge of deep learning, deep neural networks based approaches [2, 5, 7] have been proposed for person-job fit. These approaches cast person-job fit as a long text matching problem and mainly focus on learning effective representations for jobs and resumes. To be specific, Zhu et al. [1] first designed a multi-layer of convolutional neural network to encode job postings and resumes. To reduce the dependence on human annotating data, Qin et al. [5] proposed a RNN based structure with hierarchical ability-aware attention mechanism to learn the semantic representation of texts. Different from the above approaches, Bian et al. [6] studied the domain adaptation problem for person-job fit and designed a deep global match network to capture the global semantic interactions between two sentences from a job posting and a resume respectively. In addition, to integrate different types of information, ResumeGAN [34] is proposed to leverage the adversarial learning to learn more expressive representations of job postings and resumes. Jiang et al. [2] aimed to learn comprehensive and effective representations of the resumes and job postings by fusing entity features. From the perspective of interpretability of person-job fit system. Le et al. [35] utilized a deep interactive representation learning to learn the interdependence between a resume and a job posting. However, these approaches generally focused on learning good representations of job descriptions and resume texts, but ignored the preference learning from previous interview histories. Motivated by this, Yan et al. [36] designed a preference-learning model which explores the latent preferences from the historical matching records. Considering that job-resume interaction data is sparse and noisy, Bian et al. [7] proposed a co-teaching framework, called MV-CoN, which uses a text-matching model and a relation-matching model to enhance the semantic representations and reduce the influence of noise in training data. However, all of these approaches ignored the external prior knowledge in job posts and resumes which could reduce the semantic gap between two types of texts. Therefore, the capability of these matching methods are usually limited.

III. Distantly Supervised Skill Extraction

To construct skill knowledge graph, we need to recognize skill entities in the recruitment data, i.e., resume and job posting. Actually, skill entity extraction can be viewed as a NER task, which is usually formulated as a sequence labeling problem and has achieved great success by utilizing either typical graph models [37, 38] or well-designed neural network based models [14, 39]. However, these NER approaches are mainly based on supervised learning, which requires lots of labor costs for sequence data annotation. This makes it hard to apply them to label-few domains such as recruitment or medical domains [40]. To this end, we propose a robust distantly supervised skill entity extraction method, which performs skill extraction only using unlabeled data and skill entity dictionary. Obviously, skill entity dictionary is relatively easier to obtain compared with the large scale fine-grained annotated data.

To be specific, our distantly supervised method includes three steps: 1) dictionary-based annotation, 2) binary classification learning, and 3) dictionary expansion. Formally, given a set of input text sentences $\mathcal{S} = \{S_1, S_2, ..., S_{|\mathcal{S}|}\}$ in job postings or resumes, where $S_i = \{w_{1,i}, w_{2,i}, ..., w_{|S_i|,i}\}$, we aim at extracting all skill entities by infering a sequence of labels $Y = \{y_1, ..., y_j, ..., y_{|S_i|}\}$ for the input sentence S_i, where $y_j \in \{0, 1\}$ is the binary label of the j-th word. In addition, we use $\mathcal{D}_e$, $\mathcal{D} \subseteq \mathcal{S}$, $\mathcal{D}^+$ and $\mathcal{D}^u$ to denote the skill entity dictionary, the unlabelled dataset, the set of skill entity words labeled by $\mathcal{D}_e$, and the rest unlabeled words, respectively.

A. Dictionary-based Annotation

In order to generate the training instances, we utilize a skill entity dictionary to automatically annotate data, which is constructed by both automatic and manual methods. To be specific, we first utilize a automatic phrase mining method, AutoPhrase [41], to extract high quality phrases from the corpus, and then we rank these phrases according to their frequency. Finally, we will manually filter some incorrect phrases to construct a more accurate skill entity dictionary. After constructing the dictionary, we adopt the binary label assignment mechanism for the NER task instead of following the common IOB or IOBES tagging scheme. More concretely, skill entity words are mapped to the positive class and non-skill entity words are mapped to the negative class.

B. Binary Classifier

In this section, we design a neural-network-based architecture to build a skill entity classifier f_s, which consists of a character-level encoder, a word-level encoder and a classification layer.

1) Character-Level Encoder: As inputs, we introduce the character information to enhance word representations. The character embedding can not only handle infrequent words better, but also be fit for mispelling words, new words and out-of-vocabulary words. Thus, it can improve the robustness to morphological and misspelling noise [42]. For each word w in the input sentence, we use the one-layer convolution network model [43] to extract the character-level features $e_c(w)$. To be specific, we assume word w consisting of a sequence of characters $c = \{c_1, c_2, ..., c_m\}$, where m is the number of characters. The embedding representation of characters is represented by $\{e(c_1), e(c_2), ..., e(c_m)\}$, which is initialized with the pretrained BERT model [16]. We apply convolution operations over character sequence $\{e(c_1), e(c_2), ..., e(c_m)\}$ followed by max pooling and dropout [44] techniques so that we obtain a fixed length of representation of the word $e_c(w)$.

2) Word-Level Encoder: Except the character-level representation $e_c(w)$ of w, we also learn the word-level semantic representation $e_w(w)$. Specifically, we train *Word2Vec* model [45] and EMLo [46] on the entire job posting and resume corpus to learn the embedding of the words, respectively. The difference between Word2Vec and EMLo is that the former learns a static local semantic representation of a word, while the latter learns a dynamic contextual semantic representation. We compare the experimental results of them in the Section VII. In addition, we find that most of skill entities are noun phrases or non-predicates, and some skill entities capitalize the first letter. For instance, given a job requirement of "Machine Learning Engineer": "Experience with big data analysis tools such as Apache Spark and Hadoop", we can observe that skill entities, such as "big data analysis tools", "Apache Spark" and "Hadoop", are noun phrases and non-predicates, and "Apache Spark" and "Hadoop" start with a capital letter. Therefore, we consider the part-of-speech (POS) tag, the dependency label and whether a word begins with a captial letter as the human designed features $e_h(w)$ to enhance the representation of w. Concretely, these features are processed as a multi-hot format and then concatenated as $e_h(w)$. Finally, we obtain the non-contextual word representation $e(w)$ by these three part of features:

$$e(w) = [e_c(w); e_w(w); e_h(w)], \tag{1}$$

where [;] denotes the concatenation operation.

To obtain the contextual information for a word w in the input sequence, we apply a bidirectional LSTM (Bi-LSTM) network [15] due to its superior performance, which has been widely used in modeling sequential words to capture both past and future input features. Bi-LSTM stacks a forward LSTM [47] and a backward LSTM so that the representation of word w_t can be denoted as the concatenation of hidden states of the forward and backward LSTM at the t step:

$$\begin{aligned}
\overrightarrow{h_t} &= \overrightarrow{\text{LSTM}}(e(w_t), \overrightarrow{h_{t-1}}),\\
\overleftarrow{h_t} &= \overleftarrow{\text{LSTM}}(e(w_t), \overleftarrow{h_{t+1}}),\\
e(w_t) &= [\overrightarrow{h_t}; \overleftarrow{h_t}].
\end{aligned} \tag{2}$$

3) Classification Layer: Given the word representation, $e(w)$, in the input sequence, we aim to fulfill a word-level classification. To build a robust binary classifier, we optimize our classifier with two steps: pre-training step and finetune step. For the pre-training step, we construct the training instances $\mathcal{D}_p$ by adopting skill entity dictionary matching to obtain the positive instances and a non-entity sampling strategy [48] to sample negative instances. After that, we use a fully connected layer followed by softmax to output a probability-like score:

$$p(y|w) = \text{softmax}(W_p e(w|s) + b_p), \tag{3}$$

where W_p is a trainable parameter and b_p is the bias term for the pre-training step. Finally, we minimize cross entropy of the predicted class and ground truth to optimize parameters:

$$\mathcal{L}_p = - \sum_{(w_j, y_j) \in \mathcal{D}_p} y_j logp(y_j|w_j). \tag{4}$$

For the finetune step, we follow Positive-unlabeled (PU) learning [21] to optimize parameters, which can train a classifier by using only labeled positive instances and a set of unlabeled data. Actually, PU learning is suitable for our distantly supervised scenario since we do not have labeled data of all classes. Finally, the probability-like score of positive class are modeled by:

$$f_s(w) = \text{softmax}(W_f e(w) + b_f), \tag{5}$$

where W_f is a trainable parameter and b_f is the bias term for the finetune step. We then compute the prediction loss on w given label y as follows:

$$l(f_s(w), y) = |y - f_s(w)|. \tag{6}$$

The empirical training loss for PU learning is estimated by a bounded non-negative positive-unlabeled learning (bnPU) algorithm [21]:

$$\mathcal{R}_l(f_s) = \gamma \cdot \pi_p R_p^+(f_s) + \max(0, \mathcal{R}_u^-(f_s) - \pi_p \mathcal{R}_p^-(f_s)). \tag{7}$$

where

$$\begin{aligned}
R_p^+(f_s) &= \frac{1}{|\mathcal{D}^+|} \sum_{w|s \in \mathcal{D}^+} l(f_s(w), 1),\\
R_p^-(f_s) &= 1 - R_p^+(f_s),\\
R_u^-(f_s) &= \frac{1}{|\mathcal{D}^u|} \sum_{w|s \in \mathcal{D}^u} l(f_s(w), 0),
\end{aligned} \tag{8}$$

and π_p is the ratio of entity words within $\mathcal{D}^u$, and γ is a weight value for the positive class.

C. Dictionary Expansion

PU learning utilizes a domain-specific dictionary to perform data labeling and uses the empirical risk on labeled positive data to estimate the expectation risk of positive data [21]. It is hard to satisfy a requirement that the positive instances x_i^p draw identically independent from the distribution $P(S|Y = 1)$. To alleviate this problem, we expand the skill entity dictionary iteratively. To be specific, after pre-training the binary classifier, we continue to finetune the classifier with PU learning and use it to predict the unlabeled dataset. Within the unlabeled dataset, we extract all of the predicted skill entities and count their number. If it occurs over k times for a predicted skill entity, we will add it into the skill entity dictionary in the next iteration. This iteration performs several times until the skill dictionary does not change.

IV. Skill Knowledge Graph

In order to discover the global prior knowledge information, we construct a skill knowledge graph from a series of large scale real-world datasets, including resumes, job postings and encyclopedia data. Different from existing supervised learning based entity relation extraction methods [49, 50], we have no large scale fine-grained annotation data for recruitment domain. Therefore, we utilize statistical approaches and heuristic rules to build a global skill knowledge graph. We assume that the skill entities with a high relevance exist a certain type of relation which can reflect a dependency attribute and coordinating relation, and can provide an extra supplementary information for person-job fit. Actually, we consider the relevance including co-occurrence and hyponymy relations between skill entities. Taking Figure 1 for example, we can easily discover a hyponym-hypernym relation (i.e., *Tensorflow / Pytorch* isA *deep learning framework*) in the job requirement. In addition, *Opencv*, *face detection* and *face recognition* co-occur within a sentence in the work experience. Although it is hard to distinguish what the specific relations are among these skill entities, the global co-occurrences on the entire corpus can provide useful information to make them achieve a mutual complementation.

Specifically, we first utilize the proposed distantly supervised skill extraction model to extract skill entities from job postings and resumes. Then, to discover the high relevance between skill entities, we recognize the hyponymy relation and co-occurrence relation, respectively. For the former, we construct them with two parts: (1) We design certain *lexico-syntactic patterns* to detect hyponym-hypernym relations in text. For instance, patterns like "$Skill_A$ such as $Skill_B$ and $Skill_C$", or "$Skill_A$ including $Skill_B$ and $Skill_C$" often indicate hypernymy relations of the form "$Skill_B$ isA $Skill_A$" and "$Skill_C$ isA $Skill_A$"; (2) We utilize a external computer science ontology [51] to supplement the hypernymy relations. For the latter, we first compute a co-occurrence matrix for all skill entities on the global datasets, where each element in the matrix denotes a co-occurrence frequency between two skill entities within one sentence, and then we rank the skill entities for each row according to the normalized frequency value. Finally, we build the co-occurrence relations for two skills if their co-occurrence frequency is more than a threshold value. With this global co-occurrences discovery, it is able to supplement extra useful information for a skill entity, which can not obtain in a single job posting or resume.

V. Graph Construction

In this section, we introduce how to construct a graph from a job posting or resume text. Different from the previous text

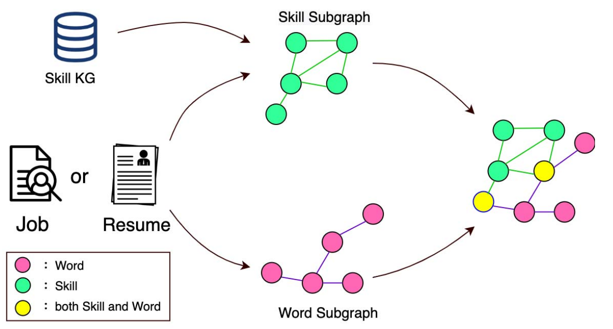


Fig. 2. The graph construction process.

matching based models [1, 2, 5, 7], we adopt a graph-matching way for person-job fit. In our setting, *job posting* or *resume* will be regarded as a graph. To be specific, the structure of the graph is defined as follow: for a graph $\mathcal{G}$ = (V, E), where V and E denote the nodes and edges in the graph. The nodes V consist of the set of words and skill entities. The edges E have two types: *Word-Word* edges E_{ww} that reflect the relations between words in text, and *Skill-Skill* edges E_{ss} that denote the relations between skill entities in the constructed skill knowledge graph. According to two types of edges, the graph can be divided into two subgraphs: word subgraph and skill subgraph. The graph construction process is shown in Figure 2.

A. Word Subgraph Construction

Given a text T (i.e., job posting or resume) with m words, i.e., $T = \{w_1, w_2, ..., w_m\}$, we regard all words w_i appearing in the text as the nodes of the graph. Each edge starts from a word in the text and ends with its adjacent words. To be specific, we define word subgraph $\mathcal{G}_{ww}$ of text T as:

$$\begin{aligned} V_w &= \{w_i | i \in [1, m]\}, \\ E_{ww} &= \{e_{ij} | i \in [1, m]; j \in [i, i+l]\}, \end{aligned} \tag{9}$$

where V_w and E_{ww} are the node set and the edge set of the word subgraph. l is the window size, which denotes the number of adjacent words connected to each word in the graph.

B. Skill Subgraph Construction

In order to build skill subgraph, we first identify all skill entities in the job posting or resume text using the proposed distantly supervised method in Section III, and then the identified skill entities will be linked to the skill knowledge graph. For each identified skill entity, we sample k-hop neighborhood in the skill knowledge graph to build our skill subgraph $\mathcal{G}_{ss}$. In this way, we can fuse more prior knowledge into the graph. It is noted that we do not distinguish the different type of relations between skill entities in the skill knowledge graph. The two skill entities will be linked if they have a certain type of relation. We will consider modeling the different types of relations for skill entities in the future work.

Thus, we construct a graph $\mathcal{G} = \mathcal{G}_{ww} \cup \mathcal{G}_{ss}$ for each job posting or resume.

VI. Knowledge Enhanced Person-Job Fit

In this section, we first formulate person-job fit problem. Then we introduce our proposed *KGRM* in detail, which is a graph matching network framework with four key modules: *Knowledge Aware Graph Encoder* (KAGE), *Graph Matching Learning*(GML), *Multi-task Learning* (MTL), and *Similarity Learning* (SL). For a job-resume matching pair ($\mathcal{G}_j$, $\mathcal{G}_r$), KAGE module first not only captures the contextual word relationships within each job posting or resume, but also fuses external prior knowledge into node representation learning. Then, GML module learns the feature interactions between $\mathcal{G}_j$ and $\mathcal{G}_r$ in both global graph-level and local node-level, respectively. Moreover, we introduce a MTL strategy with auxiliary task to facilitate graph representation learning. Finally, SL module completes the computation of similarity score.

A. Graph Matching for Person-Job Fit

Different from the previous methods [1, 2, 5], we formulate person-job fit as a graph matching task. Given a set of job posting and resume texts, we first convert them into the graph format, and then we have a set of jobs $\mathcal{G}_j = \{G_j^1, G_j^2, ..., G_j^n\}$ and a set of resumes $\mathcal{G}_r = \{G_r^1, G_r^2, ..., G_r^k\}$, where n and k are the total numbers of jobs and resumes, respectively. Based on these graph data, we construct a training matching set $\mathcal{D}_{jr}$ = $\{<G_j, G_r, y_{j,r}>|G_j \in \mathcal{G}_j, G_r \in \mathcal{G}_r\}$, where $y_{j,r}$ is a binary label denoting the final match result between job j and resume r: success or failure. Our aim is to learn a function f_m: (G_j, G_r) $\rightarrow \mathcal{R}$, which maps a pair of graphs into a similarity score.

B. Knowledge Aware Graph Encoder

The constructed graph for job posting or resume is heterogeneous with multiple types of nodes and edges. Hence, it is difficult to directly apply the traditional homogeneous GNN (such as GCN and GAT) to learn the graph representation. To address this problem, we propose a knowledge aware graph encoder to perform the graph representation learning. It consists of two parts: the Input Embedding Layer and the Knowledge Encoding Layer. The overall architecture for the graph encoder is shown in Figure 3.

1) Input Embedding Layer: In our setting, the input is the embeddings of nodes (Word and Skill) derived from the heterogeneous graph. For the embeddings of words, we utilize the *Word2Vec* model [45] to learn the embeddings of the word in the entire job posting and resume corpus. For the embeddings of skill entities, in order to better capture the semantic relations of different skill entities and knowledge relevance, we utilize the *transD* [52], instead of Word2Vec, to learn the embedding representation on the skill knowledge graph. Finally, the learned embedding representations will be the initialized features of nodes, denoted as $\boldsymbol{h} \in \mathcal{R}^{|\mathcal{V}|\times d}$ where d is the embedding dimension.

2) Knowledge Encoding Layer: We build individual graphs for each document (i.e., job posting or resume), which contain a word subgraph $\mathcal{G}_{ww}$ and a knowledge subgraph $\mathcal{G}_{ss}$. Word subgraph represents the internal information about a document itself while knowledge subgraph reflects external prior knowledge. For each subgraph, the feature information of node is propagated and aggregated contextually by the node interaction. But how to incorporate the different information between subgraphs? Although the classical heterogeneous graph models such as HAN [30] can capture the heterogeneity of different subgraphs, it is not suitable for them to apply in our setting. Hence, we propose a new solution to incorporate prior knowledge into graph representation learning. To be specific, we update the node feature with four steps: *information propagation*, *knowledge gate*, *information merge* and *global self-attention aggregation*.

Information Propagation: The node could receive the information from its adjacent neighbours. The process can be formulated as:

$$a_w^t = \text{MLP}^t(D_w^{-1}A_w h^{t-1}), \tag{10}$$

$$a_s^t = \text{MLP}^t(D_s^{-1}A_s h^{t-1}), \tag{11}$$

where A_w and $A_s \in \mathcal{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ are the adjacency matrixes of word subgraph and knowledge subgraph, respectively. $\boldsymbol{h} \in \mathcal{R}^{|\mathcal{V}|\times d}$ contains the feature of nodes ($|\mathcal{V}|$ denotes the number of all nodes including words and skill entities, and d is the embedding dimension.). Note that A_w and A_s are zero diagonal matrixes since we do not consider the feature of the node itself but that of its neighbors to update its embedding representation. MLP denotes a multi-layer perceptron. a_w^t and $a_s^t \in \mathcal{R}^{|\mathcal{V}|\times d}$ are the intermediate message matrixs. D_w and $D_s \in \mathcal{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ are the diagonal in-degree matrixs of A_w and A_s, respectively, where $D_{ii} = \sum_j A_{ij}$.

Knowledge Gate: In order to perform a joint learning for the knowledge subgraph and the word subgraph, we propose a *knowledge gate* mechanism to better fuse prior knowledge. The underlying motivation is that we want to use a gate mechanism to filter out noise and highlight the useful information to the representation learning of node in a graph. Formally, for a node i co-occuring in word subgraph and knowledge subgraph [2], let $a_{w=i}^t \in \mathcal{R}^d$ denote the embedding of the word node i in word subgraph G_{ww}, and $a_{s=i}^t \in \mathcal{R}^d$ denote the representation of the skill node i in knowledge subgraph G_{ss}. Knowledge gate g_k is defined as

$$g_k = \sigma(W_k a_{w=i}^t + U_k a_{s=i}^t), \tag{12}$$

where σ is the sigmoid function, and $W_k \in \mathcal{R}^{d\times d}$, $U_k \in \mathcal{R}^{d\times d}$ are learning parameters. With knowledge gate g_k, the knowledge enhanced representation for node i can be defined as follows:

$$a_i^t = g_k \odot a_{w=i}^t + (1-g_k) \odot a_{s=i}^t, \tag{13}$$

where $\odot$ is an element-wise multiplication operation. Equation (13) means that prior knowledge is fused into node representation by a combination of the word representation and the knowledge representation. In the combination, the knowledge gate element-wisely controls how much information from word node $a_{w=i}^t$ is preserved, and how much information

[2]The node is both a word and a skill. i.e., the yellow node in Figure 3.

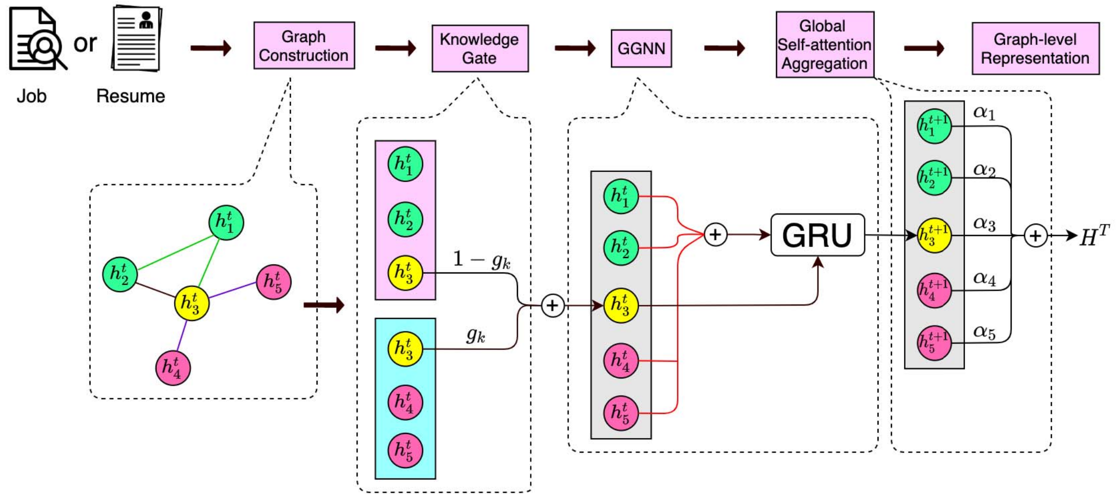


Fig. 3. The framework overview of knowledge aware graph encoder.

from prior knowledge $a^t_{s=i}$ flows in. The advantage of the element-wise operation is that it offers a way to precisely control the contributions of prior knowledge and words in node representation learning. Entries of g_k lie in [0, 1]. The larger entry of g_k is, the more information from the corresponding entry of $a^t_{w=i}$ will be kept in a^t_i. In contrast, the smaller an entry of g_k is, the more information from the corresponding entry of $a^t_{s=i}$ will flow into a^t_i. Since g_k is determined by both $a^t_{w=i}$ and $a^t_{s=i}$ and learned from training data, it will keep the useful parts in the representations of word node and skill node (i.e., prior knowledge) and at the same time filter out noise from them.

Information Merge: After incorporating prior knowledge into node representation learning, the node then merge with its own representation to update. Actually, for each graph, we employ the Gated Graph Neural Networks [53] to update the features of the nodes. As the graph layer operates on the first-order neighbours, we can stack such layer t times to achieve high-order feature interactions, where a node can reach another node t hops away. The formulas of the interaction are:

$$\begin{aligned} z^t &= \sigma(W_z a^t + U_z h^{t-1} + b_z), \\ r^t &= \sigma(W_r a^t + U_r h^{t-1} + b_r), \\ \tilde{h}^t &= tanh(W_h a^t + U_h(r^t \odot h^{t-1}) + b_h), \\ h^t &= \tilde{h}^t \odot z^t + h^{t-1} \odot (1 - z^t) \end{aligned} \tag{14}$$

where all W, U and b are trainable weights and biases not shared cross time steps. z and r function as the update gate and reset gate respectively to determine to what degree the neighbour information contributes to the current node embedding. The reset gate controls the amount of information from the previous time step (in h^{t-1}) that should propagate to the candidate representations, $\tilde{h}^t$. The new representations h^t are finally obtained by linearly interpolating between the previous and the candidate ones, using the coefficients returned by the update gate.

Global Self-attention Aggregation: After the word nodes and skill nodes are sufficiently updated for T iterations, we obtain a matrix $h^T \in \mathcal{R}^{|\mathcal{V}|\times d}$ containing the final node representations, which can be aggregated to a graph-level representation for the document. To be specific, we apply a global self-attention mechanism [54] to h^T to obtain the final document representation. At first, the self-attention coefficient e^T for each node is computed by non-linear transformation of h^T, and then the attention coefficient will be normalized with softmax to obtain an alignment vector α. Finally, the representation of graph H^T is computed as a weighted sum of the final node representations h^T. The whole process can be formulated as:

$$\begin{aligned} e^T &= tanh(h^T W_a) \cdot v_a, \\ \alpha^T_i &= \frac{exp(e^T_i)}{\sum_{j=1}^{|\mathcal{V}|} exp(e^T_j)}, \\ H^T &= \sum_{i=1}^{|\mathcal{V}|} \alpha^T_i h^T_i, \end{aligned} \tag{15}$$

where $W_a \in \mathcal{R}^{d\times d}$ and $v_a \in \mathcal{R}^d$ are learning parameters.

C. Graph Matching Learning

Given two heterogeneous graphs $\mathcal{G}_{g1}$ and $\mathcal{G}_{g2}$, we obtain the graph-level embeddings $H_{g1} \in \mathcal{R}^d$ and $H_{g2} \in \mathcal{R}^d$ and two sets of node-level embeddings $[h^k_{g1}]^m_{k=1}$ and $[h^k_{g2}]^n_{k=1}$ by knowledge aware graph encoder in Section VI-B, where m and n denote the number of nodes in $\mathcal{G}_{g1}$ and $\mathcal{G}_{g2}$, respectively. In order to perform effective graph matching, we design two interactive matching mechanisms to better capture the feature interactions in both graph-level and node-level.

1) Graph-level Matching: With two graph level representations H_{g1} and H_{g2}, we can jointly learn the representations for both job postings and resumes. To measure the matching degree between them, we finally treat them as input and apply a comparison mechanism based on a fully-connected network to learn the overall Person-Job Fit representation Q for predicting the label by a logistic function.

$$Q_g = tanh(W_q[H_{g1}; H_{g2}; H_{g1} \odot H_{g2}; |H_{g1} - H_{g2}|] + b_q) \tag{16}$$

2) Node-level Matching: The graph-level representations may lose some node-level information such as the node feature distribution and graph size. We argue that, in many cases, the differences between two graphs lie in small substructures and are hard to be reflected by the graph level embedding. An analogy is that, in Natural Language Processing, the performance of sentence matching based on one embedding per sentence can be further enhanced through using fine-grained word-level information [15, 17]. To this end, we consider using node-level matching to enhance graph-level matching. Specifically, for node-level matching, we compute the similarities between each node in $\mathcal{G}_{g1}$ and each node in $\mathcal{G}_{g2}$. First, we use a match matrix $M \in \mathcal{R}^{m\times n}$ to model the feature interactions. Formally, to compute the node similarity in M, we perform a linear transformation as

$$M_{i,j} = h^i_{g1} W_n h^j_{g2}, \tag{17}$$

where $W_n \in \mathcal{R}^{D\times D}$ denotes the match matrix. M is a $m \times n$ matrix that can be viewed as a image, following previous work [55], we use a hierarchical convolution with two stacked 2-D convolutional layers and 2-D max-pooling layers to capture important feature interactions from the node-level matching. Finally, the match representation $Q_{g1,g2}$ is derived as

$$Q_n = ConvNet(M; \theta), \tag{18}$$

where Q_n summarizes the match information from feature interactions in the node-level, and θ are all the involved parameters in our hierarchical convolutional neural networks.

D. Multi-task Learning with Auxiliary Task

To learn a more effective graph representation, we employ a multi-task learning framework which combines the main task of graph matching with an auxiliary task of graph classification for job postings and resumes through parameter sharing strategy. Our main task and the auxiliary task share a knowledge enhanced graph encoder network. To be specific, after obtaining H_{g1} and H_{g2} for the graph embedding pairs, we first pass them through a feedforward neural network and a softmax layer to obtain the probability distribution of graph classification, i.e.. $P_{g_1}(c_1|H_{g1})$ and $P_{g_2}(c_1|H_{g2})$. Then, the loss function $\mathcal{L}_c$ can be formulated as

$$\begin{aligned} &P_{g_1}(c_1|H_{g1}) = softmax(W_1 H_{g1} + b_1),\\ &P_{g_2}(c_2|H_{g2}) = softmax(W_2 H_{g2} + b_2),\\ &\mathcal{L}_c(g_1, g_2) = logP_{g_1} + logP_{g_2}, \end{aligned} \tag{19}$$

where W_1, W_2, b_1 and b_2 are learnable parameters.

E. Similarity Learning

After obtaining graph matching representations in the graph-level and in the node-level, respectively, we concatenate them (i.e., Q_g and Q_n) and then a two-layer feedforward neural network with the sigmoid nonlinearity is employed to predict a similarity score s_{g_1,g_2}. Then, s_{g_1,g_2} is compared against the ground-truth similarity score y_{g_1,g_2}using the following mean squared error loss function. In the end, we combine multi-task learning and similarity learning to compute the total loss:

$$\begin{aligned} &s_{g1,g2} = W_4(W_3[Q_g; Q_n] + b_3) + b_4,\\ &\mathcal{L} = \frac{1}{|\mathcal{D}_{jr}|} \sum_{(g_1^i, g_2^j)\in\mathcal{D}} [(s_{g_1^i,g_2^j} - y_{g_1^i,g_2^j})^2 - \lambda\mathcal{L}_c(g_1^i, g_2^j)], \end{aligned} \tag{20}$$

where W_3, W_4, b_3 and b_4 are learnable parameters. $\mathcal{D}_{jr}$ is the set of training graph pairs, and y_{g_1,g_2} is the ground-truth similarity label between g_1 and g_2. λ is a hyperparameter to balance the matching task and the classification task.

VII. Experiments

In this section, we will introduce the experimental results on real-world datasets for validating the effectiveness of the proposed methods.

A. Evaluation on Skill Entity Extraction

1) Datasets: The recruitment data contains 21,000 job postings and 86,000 resumes provided by a high-tech company. Based on the collected recruitment data, we have constructed two datasets to evaluate our method, i.e., job posting and resume datasets. Specifically, as we introduced before, we first collected the sorted high-quality phrases produced by AutoPhrase [41]. Actually, in our experiments, we selected top 20,000 frequent phrases as candidate vocabulary and manually checked whether each phrase is a skill entity. Thus, we obtained a dictionary including 12,325 skill entities. Then we could utilize the skill entity dictionary to get training instances by string matching method. Moreover, for both datasets, we randomly selected 1,500 samples and invited domain experts to label them. Among the 1,500 labeled samples, 1,000 samples were selected as the testing set and 500 samples were selected as the validation set.

2) Implementation Details: We selected top 5,000 and 15,000 most frequent words as the vocabularies for job posting dataset and resume dataset, respectively. The out-of-vocabulary words will be replace by the "<*unk*>" during the training process. For character-level encoder, the dimension of character vector was set to 50 and the character vectors were initialized by the pre-trained BERT model [16]. For word-level encoder, the dimension of word vector was set to 200 and the word vectors were initialized by the pre-trained skip-gram model [45] and kept updated during the training process. The sizes of all the LSTM hidden units were set to 256 and their hidden states are initialized as zeros. The learning rate was set to 1e-3 during the training period. Moreover, we chose the mini-batch size to be 32, and the entity ratio π is set to 0.3 and 0.1 for job posting dataset and resume dataset, respectively. The positive class weight γ is set to 2.0 and 5.0

TABLE I
THE PERFORMANCE OF SKILL ENTITY EXTRACTION.

Dataset	Method	Precision	Recall	F1
job posting	Dictionary Match	**0.9521**	0.4216	0.6869
	Fuzzy-BiLSTM-CRF	0.8801	0.9032	0.8916
	PU learning	0.8742	0.9059	0.8901
	Our method	0.9113	**0.9317**	**0.9215**
	- w/o dictionary expansion	0.8778	0.8827	0.8803
	- w/o one-hot features	0.8849	0.9063	0.8956
	- w/o char_emb	0.8812	0.9116	0.8964
	- w/o word_emb(EMLo)	0.8657	0.8983	0.8817
	- w/o word_emb(W2V)	0.8721	0.9056	0.8885
resume	Dictionary Match	**0.9256**	0.3048	0.6152
	Fuzzy-BiLSTM-CRF	0.7932	0.8427	0.8180
	PU learning	0.7913	0.8411	0.8162
	Our method	0.8178	**0.8662**	**0.8420**
	- w/o dictionary expansion	07920	0.8373	0.8147
	- w/o one-hot features	0.8058	0.8562	0.8310
	- w/o char_emb	0.8047	0.8536	0.8292
	- w/o word_emb(EMLo)	0.7984	0.8382	0.8178
	- w/o word_emb(W2V)	0.8034	0.8427	0.8226

TABLE II
THE STATISTICS OF OUR SKILL KNOWLEDGE GRAPH.

# of skill entities	# of co-occurrence relations	# of hyponymy relations	
		Pattern Match	External Knowledge Base
50,744	463,450	45,303	20,304

TABLE III
SOME IMPORTANT STATISTICS OF JOB-RESUME MATCHING DATASETS.

Datasets		# of positive samples	# of negative samples	# of samples
Technology	train	75,614	75,614	151,228
	validation	9,451	9,451	18,902
	test	9,452	9,452	18,904
Product	train	24,910	24,910	49,820
	validation	3,113	3,113	6,226
	test	3,114	3,114	6,228
User Interface	train	2,315	2,315	4,630
	validation	289	289	578
	test	289	289	578
Others	train	21,523	21,523	43,046
	validation	2,690	2,690	5,380
	test	2,690	2,690	5,380

for job posting dataset and resume dataset, respectively. For dictionary expansion, we set k[3] to 8 according to the empirical verification in our preliminary experiments, which means the predicted skill entity will be added into the dictionary if it occurs over 8 time in the corpus.

3) Baselines: Here we chose several methods as the baseline methods, including (1) Dictionary Match [18], which adopts an exact string matching strategy to obtain skill entity mentions with exactly the same surface name as in the dictionary; (2) Fuzzy-BiLSTM-CRF, which used a distantly supervised LSTM-CRF model [18] without annotated training data; (3) PU learning [21], which is same as our method without using non-skill entity sampling to produce negative samples to pre-train the classifier; (4) our method w/o (one-hot features / char embedding / word embedding (EMLo or Word2Vec), which removed human designed features $e_h(w)$, char embedding $e_c(w)$, and word embedding $e_w(w)$, respectively.

4) Evaluation Metrics: Same as the most NER tasks, we adopted the evaluation metrics including Precision, Recall, F1 to evaluate the performance of proposed methods. The precision returns a positive predictive rate, and the recall gives a true positive rate. F1 measure is a combination of precision and recall, and gives the harmonic mean of them.

5) Results: Table I shows the overall performance of our proposed distantly supervised skill entity extraction method as well as the baseline methods. We can clearly observe that our model performs the best F1-score and recall score on both job posting dataset and resume dataset. Meanwhile, we also observe that dictionary match based method achieves very high precision score but low recall score, resulting in a worst F1 score on the two datasets. Compared with Fuzzy-BiLSTM-CRF, our model improves 2.99% F1-score on the job posting dataset and 2.4% F1-score on the resume dataset. For PU learning, our method improves 3.14% on the job posting dataset and 2.56% on the resume dataset that validates the effectiveness of the pre-train strategy. In addition, we also conduct ablation test to validate the effectiveness of each module in Table I. We can find that if we remove each kind of features, i.e., human designed features, char embedding and word embedding, the performance will decrease more or less. For word embedding representation, Word2Vec based word embedding decreases more than EMLo based word embeding, indicating EMLo can capture the semantic representation better. Moreover, when removing dictionary expansion and pre-training strategy , F1-score decreased by 4.12% on the job posting dataset and 2.73% on the resume dataset that validates the effectiveness of our strategy.

[3]Actually, the performance of skill entity extraction decreased more or less when k is set to a larger or smaller values than 8.

B. Skill Knowledge Graph

We constructed the skill knowledge graph from multi-source datasets. Except the job posting dataset and resume dataset in Section VII-A, we also collected more than 80,000,000 job positngs and large scale encyclopedia data from various websites. By the above-mentioned skill extraction and relation extraction methods, we finally obtain 50,744 skill entities, 65,607 hyponymy relations and 463,450 co-occurrence relations. Actually, we do not distinguish the specific relations between two skill entities for person-job fit in our setting. In the future work, we will explore to incorporate multiple type of relations for skill entities into person-job fit task to further improve the performance. The statistics of our skill knowledge graph are summarized in in Table II.

C. Evaluation on Job-Resume Match

1) Datasets: Our experiments were conducted on a large real-world dataset provided by a high-tech company, which ranges from 2016 to 2020. Since resume data usually contain some sensitive information, the identity information, such as name, phone number and email, will be removed from the records. The dataset contains 21,000 job postings and 860,000 resumes. For the classification auxiliary task, the collected job postings have four categories, namely *Technology*, *Product*, *User Interface* and *Others*. We set a candidate's resume as the same category with the applied job's category. For the matching main task, based on this dataset, we construct a job-resume match dataset containing a large number of job-resume matching pairs. Specifically, we set the job-resume pair with the successfully interviewed record as positive instances. For negative instances, we consider two types of samples: (1) a candidate applies for a job posting but without receiving notifications for interview (strong negative instances); (2) we randomly sample the job-resume pair without any application records between them (weak negative instances). Finally, we randomly split each dataset into a training set, a validation set and a test set. Some important statistics are summarized in Table III.

2) Implementation Details: We set the dimension of node representation for word and skill as 100. For word nodes, its initialized vectors were pre-trained by the skip-gram model [45] on the entire resumes and job postings corpus. For skill nodes, we obtain the initialized vectors by using transD [52] tool pre-trained on our skill knowledge graph. Window size l discussed in Section V is set to 2. We initialized network parameters such as MLP parameters with uniform distribution in [-$\sqrt{6/(n_{in}+n_{out})}$, $\sqrt{6/(n_{in}+n_{out})}$], where n_{in} and n_{out} denote the number of the input and output units, respectively, while the parameters for the embeddings are initialized with a standard normal distribution. For the gated GNN, we stack 2 graph layers to perform feature interactions. For optimizing the parameters of models, we used the Adam [56] optimizer with an initial learning rate of 1e-3. Moreover, we set batch size as 64 for training, and used the dropout layer with a keep probability of 0.2 in order to prevent overfitting. The hop size was set to 2 when we build skill-skill graph, and λ used in the training loss is set to 0.2. We used early stop training if the validation loss on validation set does not decrease for 8 consecutive epochs.

3) Baselines: To achieve the comprehensive and comparative analysis of our KGRM, we deployed several state-of-the-art baselines which fit our problem setting:

PJFNN [1]: It applies a CNN based method to the joint representation of job-resume pairs, and uses the cosine similarity to compute the matching degree.

APJFNN [5]: It proposes a hierarchical recurrent neural networks (RNN) with hierarchical ability-aware attention to process the job-resume pairs.

DGMN [6]: It is a hierarchical attentional RNN based match network for capturing the global semantic interactions between two sentences from a job-resume pairs.

MV-CoN [7]: It combines text matching model and relation-matching model to learn an enhanced representations for job postings and resumes.

simGNN [57]: It is graph matching model, which do not consider the heterogeneity of graph.

HAN [30]: It is heterogeneous graph matching model, which consider the heterogeneity of graph with heterogeneous graph attention network.

In addition, we also compared several variants of our **KGRM** on the job-resume fit dataset to examine the relative influences of different modules on performance:

KGRM(w/o Gate) is a variant of KGRM without using knowledge gate mechanism.

KGRM(w/o K) is a variant of KGRM without using external knowledge.

KGRM(w/o Class) is a variant of KGRM without using classification task to optimize parameters of model.

KGRM(w/o Conv) is a variant of KGRM without using node-level interaction in the matching layer.

4) Evaluation Metrics: Following previous work [5, 7], except Precision, Recall and F1, we also adopted Accuracy and AUC as the evaluation metrics to evaluate the performance of proposed methods. The accuracy denotes the percent of the correct predictions. The AUC represents how much the model is capable of distinguishing between classes, and a high AUC indicates a good result.

5) Results: The overall performance is shown in Table IV. Clearly, we observe that our KGRM achieves the best performance on all four test datasets, which verifies the effectiveness of our method by incorporating external knowledge into the graph representation learning. It is noted that DGMN, APJFNN, PJFNN and Mv-CoN are text matching based neural network methods, while simGNN and HAN are graph matching based neural network methods. Compared with text matching based methods, KGRM improves 2.49%, 2.55%, 4.45%, and 2.5% AUC score than the best baseline MV-CoN on four datasets, respectively. Among graph matching methods, we can find that the heterogeneous graph matching method HAN achieves better performance than the homogeneous graph matching method simGNN on all four datasets, which demonstrates its importance for combining inner text information with the prior knowledge. Although HAN can incorporate external knowledge well with heterogeneous graph representation learning, KGRM still improves 2.22%, 1.44%, 3.23%, and 2.18% AUC score on four datasets respectively, which validates the superiority of the proposed knowledge fusion mechanism (i.e., the knowledge gate) in person-job fit. Due to the effectiveness of our KGRM, we have deployed it on Baidu Cloud as a solution for intelligent talent recruitment [4].

6) Ablation Study: To evaluate the relative influences of different modules on performance of our model, we compared with several variants as we mentioned before on four test datasets. The results are shown in Table IV. It is observed that KGRM achieves the best performance on the all metrics. It is noted that the performance for KGRM(w/o K) decreases the most (1.46%, 1.67%, 2.72% and 1.87% AUC score on four datasets, respectively) that denotes the importance of fusing external knowledge into person-job fit. In addition, after removing knowledge gate mechanism, its AUC score

[4]https://cloud.baidu.com/solution/recruitment

TABLE IV
THE OVERALL PERFORMANCE OF KGRM AND BASELINES ON FOUR TEST DATASETS.

Dataset	Metric	DGMN	APJFNN	PJFNN	Mv-CoN	simGNN	HAN	KGRM	-w/o Gate	-w/o K	-w/o Conv	-w/o Class
Technology	AUC	0.7025	0.7346	0.7475	0.7578	0.7203	0.7605	**0.7827**	0.7714	0.7681	0.7691	0.7769
	ACC	0.6544	0.6782	0.6919	0.7049	0.6636	0.7113	**0.7148**	0.7102	0.7061	0.7014	0.7112
	P	0.6674	0.6888	0.7019	0.7073	0.6742	0.7111	**0.7166**	0.7195	0.7086	0.7027	0.7218
	R	0.6542	0.6780	0.6918	0.6945	0.6691	0.7104	**0.7147**	0.7100	0.706	0.7013	0.7110
	F1	0.6474	0.6734	0.6880	0.6933	0.6592	0.7092	**0.7141**	0.7069	0.7052	0.7009	0.7076
Product	AUC	0.6577	0.6979	0.7068	0.7206	0.6762	0.7316	**0.7461**	0.7410	0.7294	0.7327	0.7402
	ACC	0.6188	0.6505	0.6505	0.6723	0.6354	0.6815	**0.6921**	0.6890	0.6726	0.6779	0.6811
	P	0.6261	0.6537	0.6508	0.6658	0.6431	0.6793	**0.6937**	0.6905	0.6754	0.6896	0.6843
	R	0.6187	0.6504	0.6505	0.6654	0.6392	0.6751	**0.6940**	0.6889	0.6725	0.6778	0.6804
	F1	0.6131	0.6486	0.6503	0.6643	0.6337	0.6732	**0.6925**	0.6883	0.6712	0.6728	0.6810
User Interface	AUC	0.7148	0.7006	0.7004	0.7242	0.7061	0.7364	**0.7687**	0.7561	0.7415	0.7495	0.7612
	ACC	0.6654	0.6602	0.6619	0.6758	0.6612	0.6879	**0.7213**	0.6984	0.6843	0.7072	0.7123
	P	0.6764	0.6705	0.6786	0.6841	0.6706	0.6896	**0.7227**	0.6993	0.6953	0.7074	0.7148
	R	0.6663	0.6610	0.6630	0.6739	0.6632	0.6880	**0.7215**	0.6986	0.6849	0.7073	0.7136
	F1	0.6609	0.6557	0.6547	0.6723	0.6601	0.6841	**0.7210**	0.6982	0.6802	0.7072	0.7122
Others	AUC	0.6825	0.7419	0.7636	0.7721	0.7067	0.7753	**0.7971**	0.7852	0.7784	0.7950	0.7901
	ACC	0.6307	0.6797	0.6928	0.7075	0.6458	0.7078	**0.7296**	0.7154	0.7097	0.7248	0.9225
	P	0.6313	0.6805	0.6933	0.7101	0.6535	0.7175	**0.7317**	0.7169	0.7139	0.7249	0.7281
	R	0.6306	0.6797	0.6929	0.7034	0.6529	0.7039	**0.7294**	0.7152	0.7096	0.7248	0.7212
	F1	0.6302	0.6793	0.6926	0.7003	0.6521	0.7023	**0.7289**	0.7148	0.7083	0.7248	0.7201

TABLE V
THE PERFORMANCE OF KGRM AND ITS VARIANTS ON JOB-RESUME MATCH TEST SET.

Method	AUC	ACC	P	R	F1
KGRM	**0.791**	**0.772**	**0.671**	**0.686**	**0.678**
KGRM(w/o classification)	0.786	0.763	0.662	0.683	0.672
KGRM(w/o context-atten)	0.776	0.74	0.642	0.683	0.652
KGRM(w/o node-intera)	0.769	0.736	0.637	0.676	0.646
KGRM(w/o node-atten)	0.769	0.737	0.635	0.672	0.645
KGRM(w/o path-atten)	0.753	0.715	0.621	0.661	0.628

decreases by 1.13%, 0.51%, 1.26% and 1.19% on four dataset respectively, demonstrating its effectiveness of knowledge gate mechanism. Meanwhile, after removing the node-level interaction in the interactive matching learning, its AUC score decreases by 1.36%, 1.34%, 1.92% and 2.1%, respectively. These results verify the effectiveness of node-level feature interaction. Moreover, the performance for KGRM (w/o Class) decreases by 0.59%, 0.59%, 0.75% and 0.7% respectively that demonstrates the multi-task learning strategy with classification task can slightly improve the performance in our setting.

7) Parameter Sensitivity: To explore the impact of some parameter settings on performance, we conducted additional experiments on the four testing datasets. The detailed experimental results are shown in Table VI. Clearly, we can observe that our model achieves best performance under conditions: indirected, $w = 2$ and hop = 2 which means using an indirected graph, the window size for word is set to 2 and the hop size for skill sampling from skill knowledge graph is set to 2, respectively. Its performances decreased more or less when we adopt other parameter settings. In addition, we also present the performance trends shown in Figure 4 and Figure 5. Figure 4 exhibits the performance of KGRM with a varying window size on the four datasets. The result reveals that the word graph density is vital to node representation learning and the best AUC / F1 score occurs when the window size is 2. Figure 5 illustrates the performance of KGRM with a varying hop size on four testing datasets. It presents a similar trend as the varying window size when the skill graph density grows.

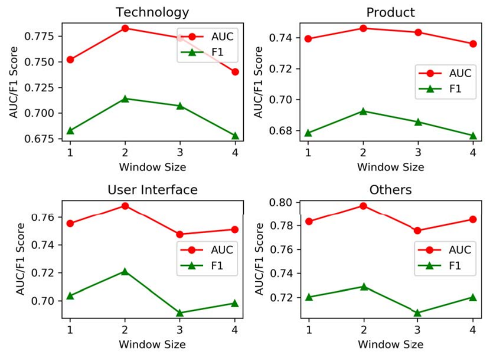


Fig. 4. AUC and F1 score with a varying window size.

8) Case Study: To further illustrate the effectiveness of our KGRM, we present an example of feature interaction process for skill entities in a job posting and a candidate's resume. The interactive process for skill entities is shown in the top right of Figure 6. It is noted that, for simplicity, we only present the node-level feature interaction for the partial skill entities. We find that there are some external skill entities such as "*tcp*", "*ip*" and "*socket communication*" in the figure except for the skill entities in the job posting and the resume. As

TABLE VI
THE OVERALL PERFORMANCE OF KGRM ON DIFFERENT PARAMETER SETTINGS.

Dataset	Metric	indirected							directed
		hop = 2				hop = 1	hop = 3	hop = 4	hop = 2
		w = 1	w = 2	w = 3	w = 4	w = 2			
Technology	AUC	0.7521	**0.7827**	0.7734	0.7402	0.7708	0.7606	0.7585	0.7753
	ACC	0.6844	**0.7148**	0.7083	0.6807	0.7112	0.7112	0.7018	0.7142
	P	0.6874	**0.7166**	0.7119	0.6864	0.7218	0.7101	0.7004	0.7219
	R	0.6843	**0.7147**	0.7082	0.6805	0.7110	0.7074	0.7013	0.7140
	F1	0.6830	**0.7141**	0.7070	0.6781	0.7076	0.7012	0.6952	0.7116
Product	AUC	0.7393	**0.7461**	0.7435	0.7361	0.7312	0.7305	0.7268	0.7328
	ACC	0.6802	**0.6921**	0.6859	0.6784	0.6796	0.6702	0.6685	0.6732
	P	0.6842	**0.6937**	0.6868	0.6821	0.6817	0.6804	0.6757	0.6735
	R	0.6801	**0.6940**	0.6859	0.6783	0.6731	0.6708	0.6647	0.6732
	F1	0.6784	**0.6925**	0.6855	0.6767	0.6695	0.6659	0.6613	0.6731
User Interface	AUC	0.7554	**0.7687**	0.7476	0.7511	0.7572	0.7481	0.7411	0.7381
	ACC	0.7143	**0.7213**	0.6949	0.7019	0.7163	0.6975	0.6905	0.6808
	P	0.7522	**0.7227**	0.7051	0.7132	0.7541	0.7072	0.7004	0.6815
	R	0.7153	**0.7215**	0.6955	0.7025	0.7159	0.6981	0.6913	0.6806
	F1	0.7037	**0.7210**	0.6914	0.6983	0.7047	0.6943	0.6896	0.6803
Others	AUC	0.7833	**0.7971**	0.7756	0.7850	0.7881	0.7771	0.7701	0.7726
	ACC	0.7218	**0.7296**	0.7072	0.7269	0.7245	0.7091	0.7010	0.7042
	P	0.7258	**0.7317**	0.7072	0.7321	0.7271	0.7093	0.7013	0.7042
	R	0.7216	**0.7294**	0.7072	0.7267	0.7237	0.7080	0.7015	0.7042
	F1	0.7204	**0.7289**	0.7072	0.7203	0.7219	0.7079	0.7010	0.7042

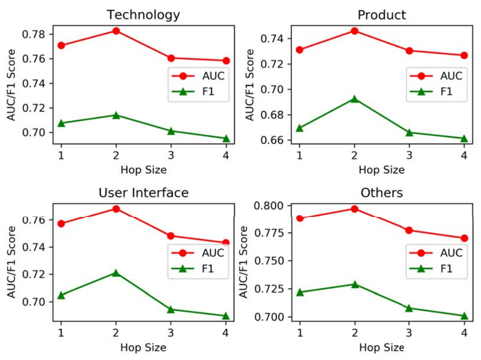
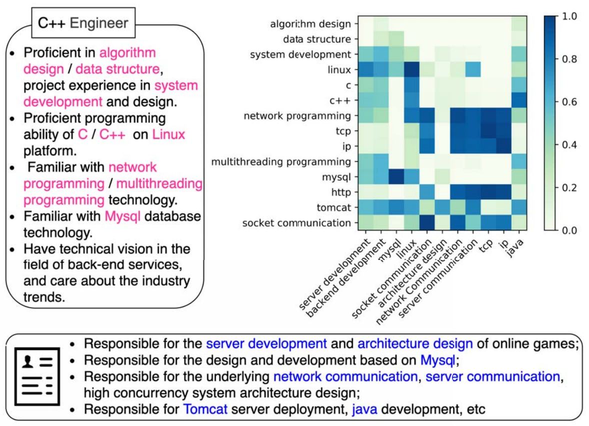


Fig. 5. AUC / F1 score with a varying hop size.


C++ Engineer

- Proficient in algorithm design / data structure, project experience in system development and design.
- Proficient programming ability of C / C++ on Linux platform.
- Familiar with network programming / multithreading programming technology.
- Familiar with Mysql database technology.
- Have technical vision in the field of back-end services, and care about the industry trends.

- Responsible for the server development and architecture design of online games;
- Responsible for the design and development based on Mysql;
- Responsible for the underlying network communication, server communication, high concurrency system architecture design;
- Responsible for Tomcat server deployment, java development, etc

Fig. 6. The case study of KGRM in our real-world dataset. The top left rectangle shows a job position for C++ engineer. The bottom rectangle is a candidate's resume. Skill entities in them are highlighted in red and in blue, respectively. The top right picture presents the skill entities feature interactions for the matching learning.

shown in the figure, these external skill entities bring obvious feature interaction that indicates fusing the prior knowledge can reduce the semantic gap between a job posting and a resume. Meanwhile, the interaction between skill entities also explains why the candidate fits this job position (the similarity score predicted by our KGRM is 0.86).

VIII. CONCLUSION

In this paper, we formulated the task of person-job fit as a graph matching problem and proposed a knowledge enhanced person-job fit approach for talent recruitment. Along this line, we first combined pre-training strategy with PU learning to achieve distantly supervised skill entities extraction from the job posting or resume texts. Then, we developed a graph representation learning model which has the ability in bridging the semantic gap between different job descriptions of the similar positions by fusing external prior knowledge. Also, we designed a knowledge aware graph encoder to fulfill an effective graph representation learning, and proposed an interactive learning method to conduct graph matching at the graph-level and the node-level respectively. Moreover, we introduced a multi-task learning strategy with a classification task to facilitate the graph representation learning process. Finally, extensive experiments on the real-world datasets clearly demonstrated the effectiveness of our approaches.

ACKNOWLEDGEMENT

This research was partially supported by the Natural Science Foundation of China under Grant No. 61836013.

References

[1] C. Zhu, H. Zhu, H. Xiong, C. Ma, F. Xie, P. Ding, and P. Li, "Person-job fit: Adapting the right talent for the right job with joint representation learning," *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 3, pp. 1–17, 2018.

[2] J. Jiang, S. Ye, W. Wang, J. Xu, and X. Luo, "Learning effective representations for person-job fit by feature fusion," in *CIKM*, 2020, pp. 2549–2556.

[3] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel, "Matching people and jobs: A bilateral recommendation approach," in *HICSS'06*, vol. 6. IEEE, 2006, pp. 137c–137c.

[4] S. Yang, M. Korayem, K. AlJadda, T. Grainger, and S. Natarajan, "Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach," *Knowledge-Based Systems*, vol. 136, pp. 37–45, 2017.

[5] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, and H. Xiong, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 25–34.

[6] S. Bian, W. X. Zhao, Y. Song, T. Zhang, and J.-R. Wen, "Domain adaptation for person-job fit with transferable deep global match network," in *EMNLP-IJCNLP*, 2019, pp. 4812–4822.

[7] S. Bian, X. Chen, W. X. Zhao, K. Zhou, Y. Hou, Y. Song, T. Zhang, and J.-R. Wen, "Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network," in *CIKM*, 2020, pp. 65–74.

[8] C. Qin, H. Zhu, T. Xu, C. Zhu, C. Ma, E. Chen, and H. Xiong, "An enhanced neural network approach to person-job fit in talent recruitment," *ACM Transactions on Information Systems (TOIS)*, vol. 38, no. 2, pp. 1–33, 2020.

[9] C. Qin, H. Zhu, C. Zhu, T. Xu, F. Zhuang, C. Ma, J. Zhang, and H. Xiong, "Duerquiz: A personalized question recommender system for intelligent job interview," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2165–2173.

[10] C. Zhu, H. Zhu, H. Xiong, P. Ding, and F. Xie, "Recruitment market trend analysis with sequential latent variable models," in *SIGKDD*, 2016, pp. 383–392.

[11] K. Yao, C. Qin, H. Zhu, C. Ma, J. Zhang, Y. Du, and H. Xiong, "An interactive neural network approach to keyphrase extraction in talent recruitment," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2383–2393.

[12] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 357–370, 2016.

[13] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *NAACL-HLT*, K. Knight, A. Nenkova, and O. Rambow, Eds. The Association for Computational Linguistics, 2016, pp. 260–270.

[14] Y. Zhang and J. Yang, "Chinese NER using lattice LSTM," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, Eds. Association for Computational Linguistics, 2018, pp. 1554–1564.

[15] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *CoRR*, vol. abs/1508.01991, 2015.

[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186.

[17] Y. Yang, W. Chen, Z. Li, Z. He, and M. Zhang, "Distantly supervised NER with partial annotation learning and reinforcement learning," in *COLING*, E. M. Bender, L. Derczynski, and P. Isabelle, Eds. Association for Computational Linguistics, 2018, pp. 2159–2169.

[18] J. Shang, L. Liu, X. Gu, X. Ren, T. Ren, and J. Han, "Learning named entity tagger using domain-specific dictionary," in *EMNLP*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Association for Computational Linguistics, 2018, pp. 2054–2064.

[19] Y. Cao, Z. Hu, T. Chua, Z. Liu, and H. Ji, "Low-resource name tagging learned with weakly labeled data," in *EMNLP-IJCNLP*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 261–270.

[20] O. Täckström, D. Das, S. Petrov, R. T. McDonald, and J. Nivre, "Token and type constraints for cross-lingual part-of-speech tagging," *Trans. Assoc. Comput. Linguistics*, vol. 1, pp. 1–12, 2013.

[21] M. Peng, X. Xing, Q. Zhang, J. Fu, and X. Huang, "Distantly supervised named entity recognition using positive-unlabeled learning," in *ACL*, A. Korhonen, D. R. Traum, and L. Màrquez, Eds. Association for Computational Linguistics, 2019, pp. 2409–2419.

[22] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NeurIPS*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 3837–3845.

[23] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1024–1034.

[24] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 4805–4815.

[25] S. Wang, Z. Chen, D. Li, Z. Li, L.-A. Tang, J. Ni, J. Rhee, H. Chen, and P. S. Yu, "Attentional heterogeneous graph neural network: Application to program reidentification," in *ICDM*. SIAM, 2019, pp. 693–701.

[26] S. Wang, L. He, B. Cao, C.-T. Lu, P. S. Yu, and A. B. Ragin, "Structural deep brain network mining," in *SIGKDD*, 2017, pp. 475–484.

[27] Y. Gao, L. Wu, H. Homayoun, and L. Zhao, "Dyngraph2seq: Dynamic-graph-to-sequence interpretable learning for health stage prediction in online health forums," in *ICDM*. IEEE, 2019, pp. 1042–1047.

[28] Y. Chen, L. Wu, and M. J. Zaki, "Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension," in *IJCAI*, C. Bessiere, Ed. ijcai.org, 2020, pp. 1230–1236.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*. OpenReview.net, 2017.

[30] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. ACM, 2019, pp. 2022–2032.

[31] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," in *EMNLP-IJCNLP*, 2019, pp. 4821–4830.

[32] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *SIGKDD*, 2019, pp. 950–958.

[33] M. Diaby, E. Viennet, and T. Launay, "Toward the next gener-

ation of recruitment tools: an online social network-based job recommender system," in *ASONAM*. IEEE, 2013, pp. 821–828.

[34] Y. Luo, H. Zhang, Y. Wen, and X. Zhang, "Resumegan: An optimized deep representation learning framework for talent-job fit via adversarial learning," in *CIKM*, 2019, pp. 1101–1110.

[35] R. Le, W. Hu, Y. Song, T. Zhang, D. Zhao, and R. Yan, "Towards effective and interpretable person-job fitting," in *CIKM*, 2019, pp. 1883–1892.

[36] R. Yan, R. Le, Y. Song, T. Zhang, X. Zhang, and D. Zhao, "Interview choice reveals your preference on the market: To improve job-resume matching through profiling memories," in *SIGKDD*, 2019, pp. 914–922.

[37] A. McCallum, D. Freitag, and F. C. N. Pereira, "Maximum entropy markov models for information extraction and segmentation," in *ICML*. Morgan Kaufmann, 2000, pp. 591–598.

[38] B. Settles, "Biomedical named entity recognition using conditional random fields and rich feature sets," in *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications, NLPBA/BioNLP 2004, Geneva, Switzerland, August 28-29, 2004*, N. Collier, P. Ruch, and A. Nazarenko, Eds., 2004.

[39] L. Liu, J. Shang, X. Ren, F. F. Xu, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," in *AAAI*, S. A. McIlraith and K. Q. Weinberger, Eds. AAAI Press, 2018, pp. 5253–5260.

[40] L. Deléger, R. Bossy, E. Chaix, M. Ba, A. Ferré, P. Bessières, and C. Nedellec, "Overview of the bacteria biotope task at bionlp shared task 2016," in *Proceedings of the 4th BioNLP Shared Task Workshop, BioNLP 2016, Berlin, Germany, August 13, 2016*. Association for Computational Linguistics, 2016, pp. 12–22.

[41] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1825–1837, 2018.

[42] X. Ma and E. H. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *ACL*. The Association for Computer Linguistics, 2016.

[43] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1746–1751.

[44] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[45] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2013.

[46] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018, pp. 2227–2237.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] Y. Cao, Z. Hu, T. Chua, Z. Liu, and H. Ji, "Low-resource name tagging learned with weakly labeled data," in *EMNLP-IJCNLP*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds. Association for Computational Linguistics, 2019, pp. 261–270.

[49] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 2124–2133.

[50] S. Zheng, Y. Hao, D. Lu, H. Bao, J. Xu, H. Hao, and B. Xu, "Joint entity and relation extraction based on a hybrid neural network," *Neurocomputing*, vol. 257, pp. 59–66, 2017.

[51] F. Osborne and E. Motta, "Klink-2: Integrating multiple web sources to generate semantic topic networks," in *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 9366. Springer, 2015, pp. 408–424.

[52] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL-IJCNLP*, 2015, pp. 687–696.

[53] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[54] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *ICLR*. OpenReview.net, 2017.

[55] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng, "Text matching as image recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.

[57] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 384–392.