Interview Choice Reveals Your Preference on the Market: To Improve Job-Resume Matching through Profiling Memories

Anonymous Submission

ABSTRACT

Online recruitment services are now rapidly changing the landscape of hiring traditions on the job market. There are millions of registered users with resumes, and tens of thousands of job postings available on the Web. Such big data result in information explosion and bring new challenges: how to match the job requirements with resumes in order to put the right talents into the right positions. Learning good *job-resume matching* for recruitment services is important: recruiters would identify possible candidates, and job seekers are able to find positions to have interviews from massive job postings. Existing studies on job-resume matching generally focus on learning good representations of job descriptions and resume texts with comprehensive matching structures. We assume that it would bring benefits to learn the preference of both recruiters and job-seekers from previous interview histories and expect such preference is helpful to improve job-resume matching. To this end, in this paper, we propose a novel matching network with preference modeled. The key idea is to explore the latent preference given the history of all interviewed candidates for a job posting and the history of all job applications for a particular talent. To be more specific, we propose a profiling memory module to learn the latent preference representation by interacting with both the job and resume sides. We then incorporate the preference into the matching framework as an end-to-end learnable neural network. Based on the real-world data from an online recruitment platform, the experimental results show that the proposed model could improve the job-resume matching performance against a series of state-of-the-art methods. In this way, we demonstrate that recruiters and talents indeed have preference and such preference can improve job-resume matching on the job market.

KEYWORDS

Job-resume matching, talent recruitment, profiling memory, neural networks

ACM Reference Format:

Anonymous Submission. 2019. Interview Choice Reveals Your Preference on the Market: To Improve Job-Resume Matching through Profiling Memories. In *KDD '19: The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 4–8, 2019, Alaska, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3220045

1 INTRODUCTION

Along the whole procedure of human resource managements for companies, talent recruitment stands at the very beginning of the long pipeline. This task plays an important role in the success of business. Talent recruitment aims at identifying the right talents for the right job positions through effectively linking talent competencies to job requirements.

However, talent recruitment is rather challenging: there are thousands of millions of candidates on the job market, while a large number of job vacancies are always calling for talents. Traditionally, talent recruitment largely depends on manual reviews conducted by human resource departments or even head-hunters. It is reported that on average, recruiters need to spend 52 days and 4,000 dollars for finding a right talent to fill an open job position [5]. Quoted from Forbes, corporations in the US spend about 72 billion dollars every year for recruiting related services, and the amount of money spent on such activities is estimated to be three times bigger all over the world [3]. In this way, we expect any improvement for matching jobs with talents results in bigger opportunity for business success, and importantly, it can also save real money.

With the fast growth of the Internet, online recruitment services are now rapidly changing the landscape of the hiring traditions on the job market. Now recruiters are able to have a broad outreach for candidate talents and talents are aware of various job opportunities. Generally, the online platform connects job providers and job seekers so as to serve as a marketplace for efficient and effective job-resume matching between job openings and candidate talents. For recruiters, they can search for relevant candidates and accept interview suggestions for the jobs. As to job seekers, they can also apply for the interviews of various job advertisements. As of October 2018, the global online recruitment service platoform LinkedIn[1] reports 590 million registered members from 200 countries, of which more than 250 million are active with job postings and/or applications [15]. With big data available, it indicates a good timing to establish data-driven methods for the matching task.

Regardless of the importance of job-resume matching, there is still a huge semantic gap between what the candidates describe on their resumes and what the recruiters request for the job openings. Previously, researchers have investigated various ways for learning to match talents with job requirements. A series of different perspectives have been studied such as job-oriented skill measurement [32] and matching metric modeling [17]. However, these efforts are basically built on manual inspections of feature engineering with expert experiences [21, 39]. Generally, human-based features are built with high costs with inaccurate (or even subjective) judgments. Furthermore, human features are hard to scale up with more and more data or to transfer from one recruiting domain to another.

With the surge of deep learning techniques, researchers find that the end-to-end learnable frameworks can largely improve the job-resume matching using the job descriptions and resume contents, both of which are represented as hidden vectors. Very recently, semantic representations are learned as embeddings to be fed into deep matching structures to rank candidates [22]. Moreover, resume texts can be modeled as a fine-grained representation with "ability" information incorporated [21]. In this way, the experiences of the candidates are encoded with the hierarchical structure so as to better match with the job descriptions with such job-oriented abilities.

These contributions are demonstrated to be useful for learning to match talents and job positions with embedded descriptions and

KDD '19, August 4–8, 2019, Alaska, USA

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220045

[1] https://www.linkedin.com/

resume texts via various deep matching structures. Yet, we find that another important clue has been overlooked: actually, people have preference on the job market! For recruiters, they may tend to hire candidates with particular experiences even if several candidates have similar skill sets. For job seekers, they may tend to work for certain positions despite that they are in fact competent for more other positions. Although such preference is latent, the hidden factor indeed has impacts on the matching results between job providers and job seekers.

To this end, in this paper, we propose to learn job-resume matching methods with the hidden preference information incorporated. We expect that with proper utilization of latent preference, the job-resume matching performance will be improved. The **first** challenge for this motivation is that how to model the preference information. We observe that the interview history for a particular job opening, or the interview application record of a specific candidate talent, indicate such preference. Note that the interview step is considered as one of the most useful tools and the final testing ground for evaluating potential candidates in the hiring process. Neither the recruiters nor the job seekers would like to waste time, money and energy for undesired choices. Through the profiling of history records, we are likely to mine the preference out for both recruiters and job seekers. Once the preference is learned, the **second** challenge is how to incorporate the information into the matching structure with the job description and resume representations to improve the job-resume matching.

Our contributions are manifold by tackling the mentioned challenges in the job-resume matching task:

• To learn the potential preference of recruiters and job seekers, we utilize the history record for the candidate talents and the job positions. To be more specific, we propose a profiling memory module to remember the preference information, sequentially updated by all previous choices from the history record, i.e, the interviewees and the applications.

• We propose an interactive schema between the *preference memory* and the job descriptions as well as the resume. For instance, the resumes in the interviewee history have impacts on the memory module and the memory has impacts on the job representation learning while vice versa. With the designed "reading-and-updating" operations for the memory, we aim at iteractively learn a better job representation with preferred interviewee profiles incorporated and similarly polishing a resume representation as well.

We run experiments using the real-world talent recruitment data from the largest online hiring service platform in China, which is called "*Boss-Hiring*". For each job posting and the candidate talent pair, i.e., (job, resume), we apply a classification function to decide whether the job-resume matching is a good fit. The matching scores are calculated with preference information incorporated and therefore, indicate the tastes of recruiters and job seekers. We examine the results to see how many ground truth pairs are correctly identified. We observe that the performance of job-resume matching is largely improved in terms of precision, recall, and AUC metrics, which demonstrates the effectiveness of our insights and the proposed job-resume matching model.

The rest of the paper is organized as follows. In Section 2, we review the relevant literature. In Section 3 we introduce the task statement including problem formulation and model overview. The details of the model is elaborated in Section 4. We run experiments and evaluations in Section 5 and draw conclusions in Section 6.

2 RELATED WORK

In this section, we will briefly introduce some works related to this paper including talent science studies and matching algorithms.

Recruitment-oriented talent science studies always play a core function of human resource management to support the success of business. The newly available recruitment big data enables researchers to conduct hiring analysis through more quantitative ways [7]. Given these observations, researchers are aware that talents have circles which can be used for talent sourcing [31]. Interestingly, they can even offer career path development advice based on talent survival analysis [12] and predict is it a good time to switch career by job transition [30]. Besides, job-related information from various social media sources and the inter-company job-hopping network shows the flow of talents [4]. Job skills of talents are demonstrated to be valuable to measure their popularity on the job markets [9, 33]. Recently, researchers are devoted to analyze recruitment market from more novel hiring perspectives [14], using market trend analysis [38] and job interview assessment [28].

The emergence of various online hiring services provides a novel perspective for better recruitment process and also posts new requirements for this research area. In particular, the study of measuring the fitting degree between the talent qualification and the job requirements, namely job-resume matching [21], has become one hot topic catching on fire. Job-resume matching is the fundamental techniques for recruitment search and recommendation systems [25, 26]. The early research efforts of Person-Job Fit can be dated back to [16], where the authors built a bilateral person-job recommendation system using the profile information from both candidates and jobs, in order to find a good match between talents and jobs. In [36], Zhang *et al.* compared a number of user-based collaborative filtering and item-based collaborative filtering algorithms on recommending suitable jobs for job seekers. In [37], Zhang *et al.* generalized fine-grained linear mixed models (GLMix) at the user/item level in the LinkedIn recommendation system, and promoted job applications for job seekers.

Generally, the job-resume matching task is highly related to text mining and natural language processing techniques such as text classification [35] and similarity measurement [6]. Nowadays, deep neural networks (DNNs, also known as *deep learning*) have made significant improvements. DNNs are highly automated learning machines; they can extract underlying abstract features of data automatically by exploring multiple layers of non-linear transformation [2]. Compared with traditional methods that largely depend on the effective human-designed representations and input features, the deep learning based approaches can learn effective models for large-scale textural data without labor-intensive feature engineering [21]. Basically, these methods model sentences using convolutional [8, 27] or recurrent [20, 29] units to construct abstractive representations. Palangi *et al.* proposed sentence matching based on vector similarities [20]. Usually, sentences are compared in a pairwise matching style via word-by-word matchings, known as sentence pair modeling [8, 20]. The chain-based matching is also demonstrated to be useful by mixing sentence information as a chain sequence [11, 24, 34].

Till very recently, job-resume matching performance are advanced with the help of deep neural networks as well. In [39], the authors proposed to encode the job and resume respectively with two convolutional neural networks into a shared space and calculate their matching degree with cosine similarity. Qin *et al.* [21] leveraged hierarchical RNNs to encode the documents and incorporate the ability-aware correlation between the job and the resume via an attention mechanism. Researcher from LinkedIn introduced their search architecture with representation learning and sparse entity encoding with deep models [23]. These methods are built on deep architectures to learn job requirements and resume representations, but NO *preference* is modeled and utilized yet.

Preference for job-resume matching is not a completely new concept. As early as 2007, Lee *et al.* proposed a job recommendation system based on job preferences and interests [10]. The personalization model is quite simple and straightforward. In this paper, we propose to model the preference information through memory profiling. The learning paradigm is a human-like process which mimics human resource experts with deep learning. To the best of our knowledge, we are the **first** to model job-resume matching using deep neural networks with preference is profiled and incorporated. It is a novel insight and indicates unique contribution.

3 TASK STATEMENT

Here we target at dealing with the job-resume matching problem, which focuses on measuring the matching degree between job requirements and the resume of talents. We will first introduce the problem formulation and the model overview in this section.

3.1 Problem Formulation

To formulate the job-resume matching task, we use J to denote a job posting which contains several sentences of job requirements and/or job descriptions, namely $J = \{j_1, j_2, \dots\}$. Similarly, for each talent resume, we use R to denote the contents of the experience and skill elaborations of the candidate, i.e., $R = \{r_1, r_2, \dots\}$. For each sentence in job descriptions J and candidate resume R, it consists of multiple words. We will discuss how to represent the words in Section 4.1. To simplify our formulation, we denote that the job posting can be learned as a hidden vector and so as the candidate resume. Then, the matching between the job and the resume literally boils down to the matching of two hidden vectors.

In practice, the talent recruitment data naturally consist of interview/application records. For instance, we are able to keep track of what job postings that a candidate has applied. We are also aware that for a particular job opening, which candidates have been interviewed. A candidate can apply for multiple jobs, and of course a job can be applied by many candidates. Those application and interview records naturally provide labeled data for us to learn the latent preference on the job market both for the recruiter side and the job seeker side.

Formally, for a particular job opening J, the interview history of candidate talents maintained in the proposed "memory" is denoted as $M^J = \{R_1, R_2, \dots\}$. For the talent resume R, we are also aware of the job application record as $M^R = \{J_1, J_2, \dots\}$. The goal of job-resume matching is to: 1) identify a set of qualified talents and the job opening as pairs, and 2) classify the job-resume pairs according to the likelihood that a candidate talent would be a good fit to the taste of the job provider, using the matching function $Match(J, R)$ given M^J and M^R. The target is to learn a predictive model for the matching degree of R and J given the history M^J and M^R. For each pair of (J,R), we have the corresponding recruitment label $y \in \{0, 1\}$, which indicates whether the selected candidate indeed has the interview opportunity or not. The model ideally predicts as many job-resume pairs with positive labels as possible (i.e., y=1), which means improved job-resume matching with increased successful interview opportunities.

3.2 Model Overview

In general, the proposed job-resume matching model can be divided into two sides: the job side and the resume side. Once we learn the representations of the job description and the resume texts, we concatenate them together through a deep neural networks for job-resume match. A matching network with multi-layer perceptron (MLP) is a standard way to mix multi-dimensions of the utterance

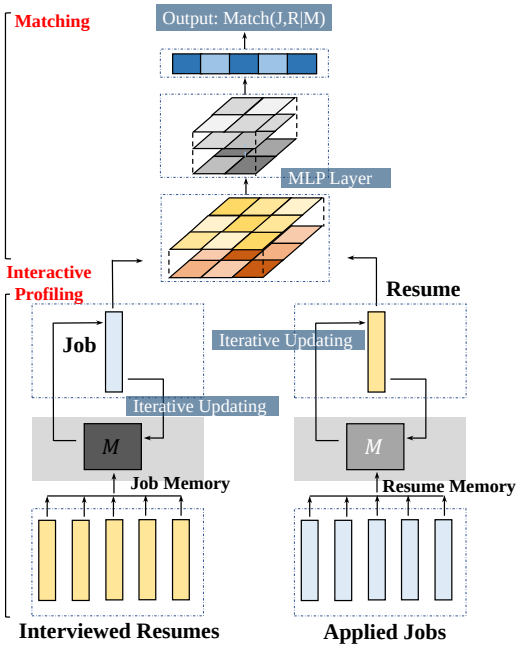

Figure 1: Model overview. For the job posting, the model memorizes interviewed candidates, and update the job representation iteratively. For the talent resume, the model memorizes the job application histories, and update the resume representation accordingly. The final representations of job posting and resume are fed into the matching network of multi-layer perceptrons (MLP).

information [2, 27]. Here we conduct a classification function to identify the matched pairs.

As to the representation learning of job requirements (as well as the resume), we propose a profiling memory to "*remember*" the preference on the job market. Inspired from human recruiters with professional expertise, we understand recruiters choose candidate talents with specific preference for interviews. They hold some latent criteria but the preference may be subject to change because the expectation towards a suitable candidate might be adjusted from time to time. Any updates of the memorized preference literally result in new candidates for interviews.

Given all interviewed candidates, we establish a memory module to learn hiring preference from the interviewee records. The memory module remembers what descriptions are relevant to the job position, forget about the irrelevant ones, and finally write the preference signals into the job representation. In this way, we establish a job embedding with both job requirements and the characteristics of expected resumes. In other words, the job texts and interviewed candidate resumes are interacted through the memory module and the memory learns the preference through the interview records. We mimic the hiring process of human recruiters to make the learning model more human-like. Intuitively, the memory will be updated as a newly interviewed candidate is recorded, and the job representation shall be synchronized with the updated preference accordingly. To this end, the job representation learning is actually an iterative updating process.

We have a symmetric way to learn the resume representation as well. A job seeker holds the work preference, and updates the

preference by applying new positions for interview. We estimate the preference of what is likely to be interested in from the job application records of the talents, and incorporate that into one's resume representation through the proposed profiling memory. With resume information enhanced by job preference, we are expected to find better matched jobs that the candidate is inclined to apply.

The model overview is illustrated in Figure 1, with both sides of the job representation and resume representation *interactively* learned and *iteratively* updated by profiling memories. An MLP network is built upon the job-resume matching. In the following section, we will elaborate the technical details of proposed matching model with preference memories incorporated.

4 JOB-RESUME MATCHING MODEL

To be self-contained, we first give a very quick overview of word embedding and the basic neural network units.

4.1 Preliminaries

4.1.1 Word Embeddings. In text-related models, a word generally acts as an atomic unit; thus, the internal relation between similar words might lose. A typical approach is to map a discrete word to a dense, low-dimensional, real-valued vector, called an *embedding* [19]. Each dimension in the vector captures some (anonymous) aspect of underlying word meanings. This process is known as vectorization. Given enough data, word embeddings can make highly accurate guesses about the meaning of a particular word. Embeddings can equivalently be viewed that a word is first represented as a one-hot vector and multiplied by a look-up table [19].

In our model, we first vectorize all words using their embeddings. Word embeddings are initialized randomly, and then tuned during training as part of model parameters.

4.1.2 Gated Recurrent Units (GRU). We use the recurrent neural networks (RNNs) with GRU units to propagate information along the word sequence. RNNs keep a hidden state vector, which changes according to the input at each time step. GRU is a gating mechanism in recurrent neural networks (RNN), introduced by Cho *et al.* [1]. Their performance was found to be better than the vanilla RNN by addressing the gradient vanishing/explosion problems. The GRU cell consists of an update gate vector $\mathbf{z}_i$, a reset gate vector $\mathbf{r}_i$, and an output vector $\mathbf{h}_i$. For each time step i with the input word embedding $\mathbf{x}_i$ and the previous hidden state $\mathbf{h}_{i-1}$, the updated hidden state $\mathbf{h}_i = \text{GRU}(\mathbf{x}_i, \mathbf{h}_{i-1})$ is computed by:

$$\begin{aligned} \mathbf{z}_i &= \sigma(W_{\mathbf{z}} \cdot \mathbf{x}_i + U_{\mathbf{z}} \cdot \mathbf{h}_{i-1}) \\ \mathbf{r}_i &= \sigma(W_{\mathbf{r}} \cdot \mathbf{x}_i + U_{\mathbf{r}} \cdot \mathbf{h}_{i-1}) \\ \hat{\mathbf{h}_i} &= \tanh(W \cdot \mathbf{x}_i + U \cdot (\mathbf{r}_i \odot \mathbf{h}_{i-1})) \\ \mathbf{h}_i &= \mathbf{z}_i \odot \hat{\mathbf{h}_i} + (1 - \mathbf{z}_i) \odot \mathbf{h}_{i-1} \end{aligned} \tag{1}$$

$\odot$ denotes element-wise multiplication. $\sigma(\cdot)$ is a known as a sigmoid function. All $\mathbf{W}$'s and all $\mathbf{U}$'s are weighted matrices. Bias items are omitted in Equation (1).

To further study the interactions and information exchanges between sentences, we establish a Bi-directional GRU (Bi-GRU) network taking the sentence representation and generating two independent hidden state vectors respectively. In general, the hidden states of a bi-directional GRU are concatenations of the hidden states of both the forward GRU and the backward GRU.

We regard the job descriptions and the resume as a piece of text document with multiple sentences. For each sentence in the document, we use the standard Bi-GRU network of Equation (1) to encode the input words. For all sentences within the job document, we use a standard GRU network to encode all the sentence embeddings, i.e., using GRU $(\mathbf{j}_{i-1}, \mathbf{h}_i^j)$ for sentence-level encoding [13], where the inputs of the GRU unit are the job sentence representation $\mathbf{j}$ and the corresponding hidden states $\mathbf{h}^j$. We can conduct the same operations to resumes using GRU $(\mathbf{r}_{i-1}, \mathbf{h}_i^r)$. In this way, the job requirements and the resumes can be encoded as a single hidden semantic vector with all intermediate sentence-level representations available during the encoding process.

4.2 Memory Profiling

We illustrate the interactive memory profiling using the job posting side information (M^J) as an example. For the talent resume side, the memory learning (M^R) is exactly the same. We omit the superscript of M when there is no ambiguity, since we only introduce the description of job posting memory profiling to avoid redundancy. The details of the memory profiling are illustrated in Figure 2.

4.2.1 Memory Initialization. For the job requirement representation, as mentioned, we have a series of last hidden states of each and every sentence embedded as $\mathbf{j}_n$, we initialize the profiling memory as follows:

$$M = \{\mathbf{m}_1^{(0)}, \mathbf{m}_2^{(0)}, \ldots, \mathbf{m}_n^{(0)}\} \quad \text{where} \quad \mathbf{m}_n^{(0)} = \mathbf{j}_n^{(0)}$$

n denotes the ending time step for job document encoding. The memory blocks are initialized as the first representation of job requirements. Note that the proposed memory is used to store preference information which is updated when new interviewed candidates are added into the record.

4.2.2 Memory Reading. Suppose now we have memory blocks with the hiring preference with k interviewed candidates recorded, denoted as $M^{(k)} = \{\mathbf{m}_1^{(k)}, \mathbf{m}_2^{(k)}, \ldots, \mathbf{m}_n^{(k)}\}$. How to read the information to update the job side representation using the memory block? We design a relevance vector for memory reading (denoted as $\mathbf{l}^{\text{read}}$), which can be computed by the soft attention mechanism:

$$\mathbf{l}_i^{\text{read}} = \sum_j \alpha_{ij} \mathbf{m}_j^{(k)} \tag{2}$$

where the signal α denotes the sentence-level attention distribution over the preference memory blocks.

The attention signal can be calculated as:

$$\alpha_{ij} = \text{softmax}\big(\phi(\mathbf{m}_j^{(k)}, \mathbf{h}_{i-1}^{(k-1)}, \mathbf{j}_i^{(k-1)})\big) \tag{3}$$

where

$$\phi(\mathbf{m}_j^{(k)}, \mathbf{h}_{i-1}^{(k-1)}, \mathbf{j}_i^{(k-1)}) = \mathbf{v}_r^T \tanh(W_l^{\text{read}} \cdot \left[\mathbf{m}_j^{(k)}, \mathbf{h}_{i-1}^{(k-1)}, \mathbf{j}_i^{(k-1)}\right]) \tag{4}$$

W_l^{read} are weight matrices. $\mathbf{v}_r$ is a weight vector. $\mathbf{v}_r^T$ denotes its transpose. We omit the bias item in Equation (4). We use superscripts k to denote the memory/job representation vectors from the k-th iteration. The calculated score is based on how to control the information fusion from the memory-to-job document relevance. The attention schema is parametrized as a neural network which is jointly trained with all the other components [24, 34].

It is known that in the original version of GRU in Equation (1), the update gate $\mathbf{z}_i$ is used to decide how much of hidden state should be retained and how much should be updated.

However, due to the way $\mathbf{z}_i$ is calculated, it is insensitive to the information control from the memory blocks. Yet, we aim at incorporating such information for job representation iterations.

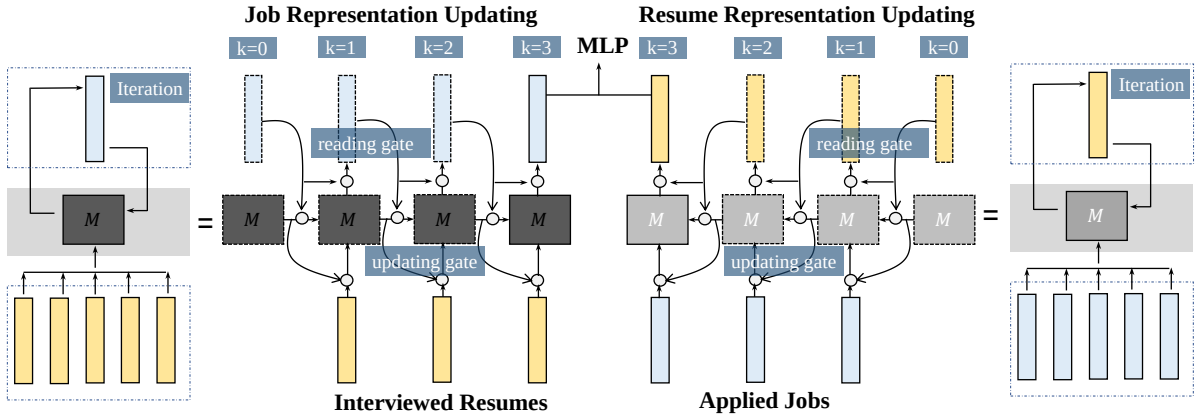

Figure 2: The details of memory module functions. The iterative updating procedure of job representation and resume representation is unfolded as a sequential process. The memory blocks have three operations: 1) memory initializaton, 2) memory updating (gated by job-resume relevance), and 3) memory reading to finalize the job and resume embeddings.

To this end, we modify the GRU cell which uses a new reading gating $\mathbf{g}_i^{\text{read}}$ instead of $\mathbf{z}_i$. The new cell takes in two inputs, the sentence representation and the relevance information, which could be regarded as a **selective reading-and-gating** process. The selection will be based on both the input sentence embeddings and the relevance vector in order to highlight the input texts which are heavily correlated with memorized preference. For each sentence, the selective network generates an update gate vector $\mathbf{g}_i^{\text{read}}$ to update the sentence-level hidden state $\mathbf{h}_i$ in Equations (1):

$$\mathbf{h}_i^{(k)} = \mathbf{g}_i^{\text{read}} \odot \hat{\mathbf{h}}_i^{(k)} + (1 - \mathbf{g}_i^{\text{read}}) \odot \mathbf{h}_i^{(k-1)} \tag{5}$$

where

$$\begin{aligned} \mathbf{f}_i^{\text{read}} &= \tanh(\mathbf{W}^{\text{read}} \cdot [\mathbf{j}_{i-1}^{(k-1)}; \mathbf{l}^{\text{read}}; \mathbf{j}_{i-1}^{(k-1)} \odot \mathbf{l}^{\text{read}}]) \\ \mathbf{g}_i^{\text{read}} &= \frac{\exp(\mathbf{f}_i^{\text{read}})}{\sum_j \exp(\mathbf{f}_j^{\text{read}})} \end{aligned} \tag{6}$$

The gating function is to calculate the relevance between the input sentence embedding $\mathbf{j}$ and the memorized preference based on the relevance vector $\mathbf{l}^{\text{read}}$. We use this gated reading module to automatically decide to which extent the information of the embeddings should be incorporated based on the memory module. In other words, the modified GRU network can be modeled with more accurate, and relevant, information from the learned preference. Here the modified GRU of Equation (5) are applied to the job representation to finalize the job hidden vector after k times of iterations according to k interviewed candidates in the history record.

4.2.3 Memory Updating. There are two information sources for the memory updating: the memory needs to update with the (revised) job description information and also update with the new interviewee resume information. For the relevance between the job requirements from the last iteration, we propose another soft attention mechanism to obtain an attention weight vector, which indicates the probability of emphasizing the j-th block from the memory.

$$\beta_{ij} = \text{softmax}\big(\pi(\mathbf{m}_j^{(k)}, \mathbf{j}_i^{(k)})\big) \tag{7}$$

Given $\mathbf{v}_u$ and W_l^{update} as parameters, we define

$$\pi(\mathbf{m}_j^{(k)}, \mathbf{j}_i^{(k)}) = \mathbf{v}_u^T \tanh(W_l^{\text{update}} \cdot \left[\mathbf{m}_j^{(k)}, \mathbf{j}_i^{(k)}\right]) \tag{8}$$

Similarly, for the relevance between the memory and the input new resume information, we can also have:

$$\gamma_{ij} = \text{softmax}\big(\pi(\mathbf{m}_j^{(k)}, \mathbf{r}_i^{(k+1)})\big) \tag{9}$$

Given $\mathbf{v}_u$ and W_l^{update} as parameters, we define

$$\pi(\mathbf{m}_j^{(k)}, \mathbf{r}_i^{(k+1)}) = \mathbf{v}_u^T \tanh(W_l^{\text{update}} \cdot \left[\mathbf{m}_j^{(k)}, \mathbf{r}_i^{(k+1)}\right]) \tag{10}$$

Here, for the alignments between the memory blocks and the job sentences as well as the alignments between the memory blocks and the candidate resume sentences, we use the shared parameters W_l^{update} and $\mathbf{v}_u$. In this way, we have the relevance vector for memory updating according to the previous representation of job requirement texts and the new candidate resume sentences, calculated by:

$$\mathbf{l}_i^{\text{update}} = \sum_n \beta_{ni} \mathbf{m}_i^{(k)} + \sum_m \gamma_{mi} \mathbf{m}_i^{(k)} \tag{11}$$

Finally, we apply the gating functions to update the input of the memory module

$$M^{(k+1)} = \mathbf{g}_i^{\text{update}} \odot M^{(k)} \tag{12}$$

where

$$\begin{aligned} \mathbf{f}_i^{\text{update}} &= \tanh(\mathbf{W}^{\text{update}} \cdot [\mathbf{m}_i^{(k)}; \mathbf{l}^{\text{update}}; \mathbf{m}_i^{(k)} \odot \mathbf{l}^{\text{update}}]) \\ \mathbf{g}_i^{\text{update}} &= \frac{\exp(\mathbf{f}_i^{\text{update}})}{\sum_j \exp(\mathbf{f}_j^{\text{update}})} \end{aligned} \tag{13}$$

Till now, we have introduced the representation learning method for job descriptions and the reading-and-updating schema of the memory module. The job encoding now interacts with the resume information through the profiling memory, which fully utilizes the interviewed candidate history from the record. Please note that the resume encoding on the candidate side would be exactly the same by using the job application track record to obtain the resume

Table 1: Statistics about sample size and details about documents of job descriptions and resumes.

Statistics	Values
# of job postings	78,107
# of resume	87,208
# of positive samples (in job-resume pairs)	91,119
# of negative samples (in job-resume pairs)	132,651
avg # of resumes in history for a job posting	3.89
avg # of job application in history for a candidates	6.01
avg words # per job posting	151.04
avg sentence # per job posting	12.65
avg words # per resume	105.36
avg sentence # per resume	7.64

representation. The preference-aware job and resume representation learning is based on the symmetric neural network structures, which is shown in Figure 2.

4.3 Matching Network

We concatenate last sentence-level hidden states of the job representation and resume representation together. Then we feed the concatenated vector to an ensuing network for further information mixing. Vector concatenation for matching is known to be effective [27, 34].

The vector is passed through a multi-layer, fully-connected, feed-forward neural network, also known as *multi-layer perceptron* (MLP) [2], which allows rich interactions. The network enables to extract features automatically, starting from lower-level representations to higher-level ones, till the system provides an overall judgment of the job-resume matching degree. A single neuron outputs the matching score. We follow previous studies for job-resume matching [21, 39] and formulate the task as a classification problem. The outputs are the probabilities of the different classes, which are computed by the softmax function on the matching vectors. In general, the performance becomes better as the number of interaction layers increases but it takes more computational resources. In this work, we empirically build a 3-layer MLP network.

We apply the cross-entropy loss for classification to train the proposed networks. Given a positive sample $(\mathbf{j}^+, \mathbf{r}^+)$ in the training set, we randomly sample a negative instance $\mathbf{r}^-$ and/or $\mathbf{j}^-$. The objective is to maximize the scores of positive samples while minimizing that of the negative samples.

5 EXPERIMENTS AND EVALUATION

In this section, we will introduce the experimental results based on a real-world recruitment data set. In the meanwhile, some case studies are demonstrated for revealing interesting findings obtained by our proposed model.

5.1 Dataset

In this paper, we conducted experiments on a real-world dataset provided by the largest online recruiting platform named "Boss-Hiring" (Boss Zhipin)[2] in China. To protect the privacy of candidates, all the user records were anonymized by deleting identity information.

The raw dataset consists of 131,568 job postings and 129,353 resumes from job seekers. Our work aims at finding appropriate talents for suitable job positions, which will be of practical values in the recruitment process. Among all the data obtained, we conduct data filtering and cleaning by removing incomplete resumes as well as job postings without any applicants for a given period of time. We summarize the statistics of the dataset in Table 1. For document preprocessing, we tokenize each sentence into words with the benchmark Chinese tokenizer toolkit named JieBa [3]. Please note that for different job positions, the interviewee records may share some common candidates while for different job seekers, they may also share some job interview histories: the candidate talent may apply for several job positions during the same time period.

5.2 Experimental Setups

Here, we introduce the detailed settings of our experiments, including the technique of word embedding, parameters for our proposed model, as well as the details of training stage.

We conducted the task of job-resume matching based on the real-word data set, i.e., we used the interviews as positive samples, and then used the job-resume pairs without interviews as the negative instance to train the models. The batch size is set to 128 with each sample as a job-resume pair. Each sentence in the resumes and job postings is clipped with a max length of 100. Meanwhile, each document of a resume and a job posting is clipped with 30 and 25 sentences respectively.

We represent the words in job postings and resumes with 100-dimension pretrained skip-gram vectors which are fixed during training [19]. The dimension of the memory block vector is also set to 100. The model is trained with Adam optimizer and the learning rate is set to 0.0005. In order to tackle the imbalanced data problem, we used the under-sampling method to randomly select negative instances that are equal to the number of positive instances for each job posting to evaluate our model. Both the size of validation set and testing set is set to 1,000 and the training will be early stopped if the evaluation results on the validation set does not increase for 5 successive epochs.

5.3 Evaluation Metrics

We follow the same evaluation paradigms in the job-resume matching studies proposed recently [21, 39], and we also regard the job-resume matching task as a classification problem. In the real-world process of talent recruitment, people usually decide a latent threshold to filter qualified candidates from the suggested ones. Empirically, the threshold is highly relevant to professional experiences and personalized tastes. Thus, we comprehensively validate the performance using the *AUC index* to measure the performance under different situations [18]. AUC can reflect model performance within different boundary values between classes and thus is widely used in two-class classification problems [21].

Still, we also include the classic metrics for the classification task using *accuracy*, *precision*, *recall* and $F1$ scores [18]. We regard the matching score of 0.5 as the threshold. Hence job-resume pairs with a matching score over 0.5 will be counted as the positive result while pairs under 0.5 as the negative results. For fairness, we adopt the same evaluation standard for all methods in our experiments.

5.4 Competing Methods

We include several algorithms as baselines to compare the performance. For completeness, we include the classic classification methods as well as state-of-the-art job-resume matching models based on deep neural networks.

For the traditional classification methods, we can run *Logistic Regression* (**LR**), *Decision Tree* (**DT**), *Naive Bayes* (**NB**), *Random Forests* (**RF**), and *Gradient Boosting Decision Tree* (**GBDT**). These methods are typical and representative. For these methods, we treat

[2] https://www.zhipin.com

[3] https://github.com/fxsjy/jieba

Table 2: Overall performance of all methods. '⋆' indicates that we accept the improvement hypothesis of our model over the best baseline at a significance test level of 0.01.

Methods	Accuracy	Precision	Recall	F1	AUC
LR	0.552	0.551	0.558	0.554	0.569
DT	0.549	0.546	0.591	0.567	0.559
NB	0.539	0.540	0.538	0.539	0.550
RF	0.570	0.578	0.588	0.583	0.587
GBDT	0.579	0.585	0.589	0.587	0.594
HRNNM [13, 20]	0.596	0.603	0.561	0.581	0.629
PJFNN [39]	0.620	**0.631**	0.581	0.604	0.640
AAPJF [21]	0.605	0.607	0.601	0.604	0.637
JRMPM	**0.634⋆**	0.620	**0.692⋆**	**0.652⋆**	**0.671⋆**

the mean vector of all word vectors in a resume (or a job posting) as its latent vector. Then we regard the latent vectors of a candidate resume and the corresponding job posting together as the input of all baseline methods.

We also include the deep neural network baselines, which could be regarded as state-of-the-art models for job-resume matching.

• *Hierarchical RNN Matching* (**HRNNM**). Hierarchical modeling for document embedding was at first introduced by Li *et al.,* [13]. In this way, the resume and the job requirements are encoded through the hierarchical structure of both word-level RNN and the sentence-level RNN. The last hidden states from the hierarchical RNN sequence are matched by the cosine similarity in pairs, which is a simple and effective matching model [20].

• *Person-Job Fit Neural Network* (**PJFNN**). PJFNN was originally proposed to regard the Person-Job Fit problem as a classification task [39]. It takes a job-resume pair as input and predicts the probability that they reach an interview agreement. The authors proposed a joint representation learning approach which encodes the resumes and the job postings independently with two convolutional neural networks (CNNs) and calculates the cosine similarity as the matching degree.

• *Ability-Aware Person-Job Fit* (**AAPJF**). AAPJF also formulates the Person-Job Fit problem as a classification task [21]. The authors proposed a hierarchical recurrent neural networks (RNNs) to encode the resume and the job posting which also incorporates ability information. The new insight is to model ability-aware representation with job requirement and resume learning.

• *Job-Resume Matching with Profiling Memory* (**JRMPM**). The unique contribution of our proposed model is that we match job and resume with latent preference. In JRMPM model, we utilize the profiling memories to learn the preference on the job market for both the recruiters and the job seekers. The interview history of the job position and job application record of the candidate talents have mutual impact on representation learning of each other, which is an interactive process through the profiling memories.

5.5 Overall Performance

The overall performance is shown in Table 2. Among all classic classification methods, *NB* shows the weakest result while *GBDT* performs relatively well. It is interesting to see that even though these standard machine learning models take the pre-trained word vectors as the input features, their results are still not as good as deep neural network based methods. We conclude that such a phenomenon may indicate that the pre-trained word vectors are not enough to characterize the semantic features of the recruitment texts. This should be the reason of why the end-to-end deep neural networks can extract more accurate semantic word representations and outperform the classic classification methods.

For the deep neural network algorithms, the simple *HRNNM* model shows better results than GBDT. Not surprisingly, the talent recruitment oriented person-job fit models, i.e., *PJFNN* and *AAPJF*, show prominent improvement over the results of HRNNM, which concurs the observations from previous literature [21, 39] according to our recruitment data. The improvement indicates that the ability-aware models as well as the job/resume representation learning are beneficial for this particular task. Generally, we conclude that the performance of *PJFNN* and *AAPJF* are to some extent comparable in terms of the $F1$ scores and the AUC scores. Still, *PJFNN* shows slightly better results in the *accuracy* score. The advantage might be resulted from the different ways of encoding job/resume sentences.

We are delighted to find that the results of our proposed *JRMPM* method have the overall advantages over the performance of *PJFNN* and *AAPJF* in terms of *accuracy*, $F1$ scores, as well as *AUC* scores. The *precision* score is slightly worse than that of *PJFNN*. It is due to the situation that precision and recall scores are balanced in trade-off and the *recall* score of *JRMPM* is much higher than that of *PJFNN*. The improvement over the best baseline has passed the significance test and we demonstrate that using the preference memory for job-resume matching indeed facilitates the performance.

Table 3: Ablation studies: the impacts of job and/or resume memories by adding to the model framework.

Model Variants	Accuracy	Precision	Recall	F1	AUC
No Memory	0.611	0.615	0.587	0.601	0.641
+Job Memory	0.622	0.612	0.645	0.628	0.655
+Resume Memory	0.621	0.613	0.659	0.635	0.657
Full Model	0.634	0.620	0.692	0.652	0.671

Table 4: Hyperparaemter (k) tuning: how many job applications/interviewed candidates to remember in the memory.

	Accuracy	Precision	Recall	F1	AUC
k=1	0.622	0.617	0.645	0.631	0.664
k=2	0.627	0.614	0.689	0.649	0.669
k=3	0.625	0.607	0.710	0.655	0.670
k=4	0.634	0.620	0.692	0.652	0.671
k=5	0.631	0.618	0.682	0.649	0.671

5.6 Analysis and Discussions

5.6.1 Ablation Study of Profiling Memories. Actually, it is not surprising to see that *JRMPM* shows the best results, but can we credit the improvement to the latent preference information used in our model? It would be interesting to conduct ablation studies to examine the effectiveness of the proposed preference memories. The first model variant is that we remove the memory blocks from both the recruiter side and the job seeker side (denoted as "No Memory"). The model essentially degenerates into the job representation learning and resume representation learning and matching with an MLP layer after the job-resume concatenation. The other two model variants are to add the memory from either side of the matching network in separate, i.e., job representation with interviewee memory (denoted as "+Job Memory") and resume representation with application memory (denoted as "+Resume Memory").

Comparing the performance of the model variants against the full model in Table 3, we have the following observations for the impacts of memories. All model variants which remove memory modules show prominent performance drop. In particular, the model variant

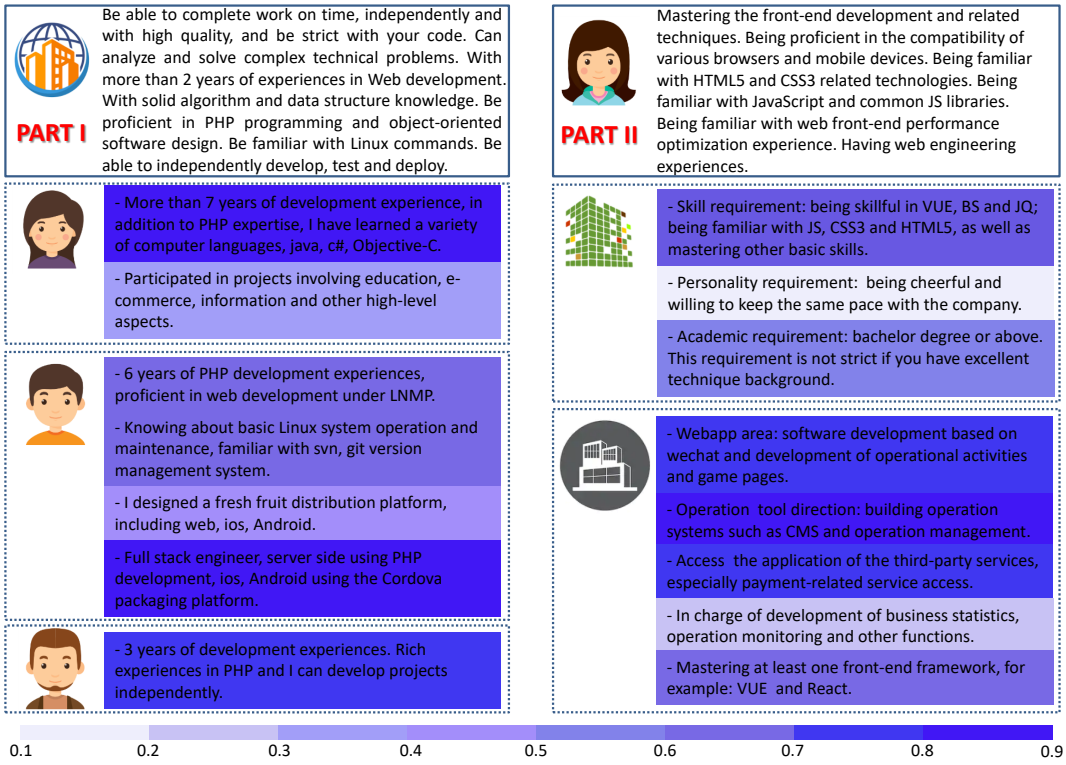

Figure 3: Case studies of visualization: 1) on the *left* side (Part I), we show the job requirements and the interviewed candidates; 2) on the *right* side (Part II), we show the resume with the job application history. The color indicates the attention signal weights from the memory: darker color indicates larger weights. The memory indeed remembers the correlated parts between the resume and the job requirements, and could be used as the enhancement with the profiling information as *preference*.

No Memory has the worst performance and some results are not as good as that of *PJFNN*. The phenomenon indicates that without the preference modeling, the proposed model will not be stronger than baselines. It is good to see that with the incorporation of memory vectors from either the job side or the resume side, the model variants (i.e., *+Job Memory* and *+Resume Memory*) have better performance than *No Memory* and *PJFNN*. The impacts of the preference module are quite prominent. Furthermore, when the preference of both sides (recruiters and job seekers) added, the full model yields the best performance. From the ablation studies, we conclude the preference memory from both sides indeed help the job-resume matching task to improve the performance.

5.6.2 Shall We Have a LONG Memory? There is a hyperparameter k of how many recent records shall be recorded into the memory module. Are longer memories always preferred? We assume not. For humans, it is natural to adjust the expectations towards candidate talents (or dream jobs). It is natural to assume that the most recent records are likely to be more useful. We tune the hyperparameter k from 1 to 5, which means that we use only the most recent k (k=1,2,3,4,5) records in the memory for preference learning. Note that k=0 actually means no memory.

From the results in Table 4, we can see when k is small, the performance generally increases as k increases. The margin for the improvement gains become smaller and smaller when k is large. This phenomenon indicates that using several recent records would be sufficient to learn the preference information. On the contrary, using too many records as a *long* memory may bring in noise because preference might have been changed, and the noise might hurt the performance. In our case, k=4 shall be the best setting.

5.6.3 Case Visualization. We visualize how the preference guides the representation learning of the *JRMPM* model. Given a job opening, *JRMPM* scans the preference history and updates its representation with the resumes of interviewed candidates. At each updating step, the attention mechanism of *JRMPM* incorporates the correlation between the job requirements and the sentences of the interviewed resume. We visualize the sentence-level attention signal between the job and the candidate resume which captures the correlation in the preference memory. The principle is the same for both the job and the talent sides.

The left side of Figure 3 (Part I) shows job posting of a PHP-based software developer and the interviewed candidates. All of the applicants have introduced about the software developing experiences of several years. Therefore, the memory modules learns about the preference of the recruiter towards the real experience in details of software development.

The right side of Figure 3 (Part II) shows a resume of a development engineer and the job application history. The *JRMPM* memory learns to distinguish the importance of each sentence. The model assigns a large weight towards the skill requirement and "remembers" the relevant text information to enhance the the original resume information. Besides, we find the memory module learns in a way of reducing redundancy. The last sentence in the second recent

job application also contains similar job requirements as the first sentence describes in the most recent job. The information from the most recent record gains more attention weights.

6 CONCLUSION

In this paper, we proposed a novel end-to-end deep neural network based framework for the job-resume matching task. We expect the model will improve the online talent recruitment process by reducing human labors and providing better matching results. The key idea is to learn the job requirement and resume text representations with latent preference information incorporated. To be more specific, we design a memory module to "remember" records in interview/application history. The memory reads the records of preferred candidate resumes and update the job representation with resume-aware information and similarly, the memory reads the job application history and update the resume representation with job-oriented information. In this way, the job and resume representations are learned in an interactive way through the profiling memories. The whole procedure mimics hiring behaviors and is more human-like.

We conduct experiments on real-world data from the online recruitment service platform. Extensive experimental results clearly validate the effectiveness of our proposed job-resume matching method in terms of *accuracy*, *AUC*, and $F1$ scores, etc. Especially, we conduct additional ablation studies to verify that the memories are rather useful and the preference information is the key factor for improvement. We also notice that it may not be a good idea to remember too many records in the memory because once the preference changes, unnecessary records may become noises.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR'15*.
[2] Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2, 1 (2009), 1–127.
[3] Josh Bersin. 2013. https://www.forbes.com/sites/joshbersin/2013/05/23/corporate-recruitmenttransformed-new-breed-of-service-providers/
[4] Yu Cheng, Yusheng Xie, Zhengzhang Chen, Ankit Agrawal, Alok Choudhary, and Songtao Guo. 2013. JobMiner: A Real-time System for Mining Job-related Patterns from Social Media. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. ACM, New York, NY, USA, 1450–1453. https://doi.org/10.1145/2487575.2487704
[5] Facebook. 2017. Talent Acquisition. Retrieved December 19, 2017 from http://blog.bersin.com/benchmarking-talent-acquisition-increasing-spend-cost-per-hireand-time-to-fill/
[6] Wael H. Gomaa and Aly A. Fahmy. 2013. A survey of text similarity approaches. *International Journal of Computer Applications* 13, 68 (2013), 13–18.
[7] Christopher G Harris. 2017. Finding the Best Job Applicants for a Job Posting: A Comparison of Human Resources Search Strategies. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*. IEEE, 189–194.
[8] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*. 2042–2050.
[9] Faizan Javed, Phuong Hoang, Thomas Mahoney, and Matt McNair. 2017. Large-Scale Occupational Skills Normalization for Online Recruitment.. In *AAAI'17*. 4627–4634.
[10] Danielle H. Lee and Peter Brusilovsky. 2007. Fighting information overflow with personalized comprehensive information access: A proactive job recommender. *Autonomic and autonomous systems, ICAS07. Third international conference on* 2, 1 (2007), 21–21.
[11] Chaozhuo Li, Yu Wu, Wei Wu, Chen Xing, Zhoujun Li, and Ming Zhou. 2016. Detecting Context Dependent Messages in a Conversational Environment. In *COLING'16*. 1990–1999.
[12] Huayu Li, Yong Ge, Hengshu Zhu, Hui Xiong, and Hongke Zhao. 2017. Prospecting the Career Development of Talents: A Survival Analysis Perspective. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, New York, NY, USA, 917–925. https://doi.org/10.1145/3097983.3098107
[13] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2016. A hierarchical neural autoencoder for paragraphs and documents. In *ACL'16*.
[14] Hao Lin, Hengshu Zhu, Yuan Zuo, Chen Zhu, Junjie Wu, and Hui Xiong. 2017. Collaborative Company Profiling: Insights from an Employee's Perspective.. In *AAAI*. 1417–1423.
[15] LinkedIn. 2018. LinkedIn Wiki. https://en.wikipedia.org/wiki/LinkedIn
[16] J. Malinowski, T. Keim, T. Weitzel, and O. Wendt. 2006. Matching People and Jobs: A Bilateral Recommendation Approach. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)(HICSS)*, Vol. 06. 137c. https://doi.org/10.1109/HICSS.2006.266
[17] Jochen Malinowski, Tobias Keim, Oliver Wendt, and Tim Weitzel. 2006. Matching people and jobs: A bilateral recommendation approach. In *null*. IEEE, 137c.
[18] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*.
[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[20] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2015. Deep Sentence Embedding Using the Long Short Term Memory Network: Analysis and Application to Information Retrieval. *arXiv preprint arXiv:1502.06922* (2015).
[21] 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 *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 25–34.
[22] Rohan Ramanath, Hakan Inan, Gungor Polatkan, Bo Hu, Qi Guo, Cagri Ozcaglar, Xianren Wu, Krishnaram Kenthapadi, and Sahin Cem Geyik. 2018. Towards Deep and Representation Learning for Talent Search at LinkedIn. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2253–2261.
[23] Rohan Ramanath, Hakan Inan, Gungor Polatkan, Bo Hu, Qi Guo, Cagri Ozcaglar, Xianren Wu, Krishnaram Kenthapadi, and Sahin Cem Geyik. 2018. Towards Deep and Representation Learning for Talent Search at LinkedIn. In *CIKM '18*. ACM, 2253–2261.
[24] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *ICLR*.
[25] RÃącz, GÃąbor, Attila Sali, and Klaus-Dieter Schewe. 2016. Semantic matching strategies for job recruitment: A comparison of new and known approaches. In *International Symposium on Foundations of Information and Knowledge Systems*. Springer, 149–168.
[26] Tomoki Sekiguchi. 2004. Person-organization fit and person-job fit in employee selection: A review of the literature. *Osaka keidai ronshu* 6, 54 (2004), 179–196.
[27] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, New York, NY, USA, 373–382. https://doi.org/10.1145/2766462.2767738
[28] Hengshu Zhu Chen Zhu Tong Xu Chao Ma Shen, Dazhong and Hui Xiong. 2018. A Joint Learning Approach to Intelligent Job Interview Assessment. In *IJCAI'18*. 3542–3548.
[29] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. In *AAAI'16*. 2835–2841.
[30] Jian Wang, Yi Zhang, Christian Posse, and Anmol Bhasin. 2013. Is It Time for a Career Switch?. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13)*. ACM, New York, NY, USA, 1377–1388. https://doi.org/10.1145/2488388.2488509
[31] Huang Xu, Zhiwen Yu, Jingyuan Yang, Hui Xiong, and Hengshu Zhu. 2016. Talent circle detection in job transition networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 655–664.
[32] Tong Xu, Hengshu Zhu, Chen Zhu, Pan Li, and Hui Xiong. 2018. Measuring the popularity of job skills in recruitment market: A multi-criteria approach. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[33] Tong Xu, Hengshu Zhu, Chen Zhu, Pan Li, and Hui Xiong. 2018. Measuring the Popularity of Job Skills in Recruitment Market: A Multi-Criteria Approach. In *AAAI'18*. AAAI, 655–664.
[34] Rui Yan, Dongyan Zhao, et al. 2017. Joint Learning of Response Ranking and Next Utterance Suggestion in Human-Computer Conversation System. In *SIGIR'17*. 685–694.
[35] Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML'17*. 412–420.
[36] Cheng Yang Zhang, Yingya and Zhixiang Niu. 2014. A research of job recommendation system based on collaborative filtering. In *Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on*. IEEE, 533–538.
[37] 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 (KDD '16)*. ACM, New York, NY, USA, 363–372. https://doi.org/10.1145/2939672.2939684
[38] Chen Zhu, Hengshu Zhu, Hui Xiong, Pengliang Ding, and Fang Xie. 2016. Recruitment Market Trend Analysis with Sequential Latent Variable Models. In *KDD '16*. 383–392.
[39] 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 Transactions on Management Information Systems (TMIS)* 9, 3 (2018), 12.