Learning to Match Jobs with Resumes from Sparse Interaction Data using Multi-View Co-Teaching Network

Shuqing Bian[1], Xu Chen[4], Wayne Xin Zhao[2,3*], Kun Zhou[1],
Yupeng Hou[2], Yang Song[5], Tao Zhang[5] and Ji-Rong Wen[2,3]
[1]School of Information, Renmin University of China
[2]Gaoling School of Artificial Intelligence, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
[4]Department of Computer Science, University College London
[5]BOSS Zhipin NLP Center
{bianshuqing, houyupeng, jrwen}@ruc.edu.cn, xu.chen@ucl.ac.uk, batmanfly@gmail.com,
francis_kun_zhou@163.com, {songyang, kylen.zhang}@kanzhun.com

ABSTRACT

With the ever-increasing growth of online recruitment data, *job-resume matching* has become an important task to automatically match jobs with suitable resumes. This task is typically casted as a supervised text matching problem. Supervised learning is powerful when the labeled data is sufficient. However, on online recruitment platforms, job-resume interaction data is sparse and noisy, which affects the performance of job-resume match algorithms.

To alleviate these problems, in this paper, we propose a novel multi-view co-teaching network from sparse interaction data for job-resume matching. Our network consists of two major components, namely text-based matching model and relation-based matching model. The two parts capture semantic compatibility in two different views, and complement each other. In order to address the challenges from sparse and noisy data, we design two specific strategies to combine the two components. First, two components share the learned parameters or representations, so that the original representations of each component can be enhanced. More importantly, we adopt a co-teaching mechanism to reduce the influence of noise in training data. The core idea is to let the two components help each other by selecting more reliable training instances. The two strategies focus on *representation enhancement* and *data enhancement*, respectively. Compared with pure text-based matching models, the proposed approach is able to learn better data representations from limited or even sparse interaction data, which is more resistible to noise in training data. Experiment results have demonstrated that our model is able to outperform state-of-the-art methods for job-resume matching.

KEYWORDS

Job-Resume Matching, Graph Neural Network, Co-Teaching

*Corresponding author.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411929

ACM Reference Format:
Shuqing Bian, Xu Chen, Wayne Xin Zhao, Kun Zhou, and Yupeng Hou, Yang Song, Tao Zhang and Ji-Rong Wen. 2020. Learning to Match Jobs with Resumes from Sparse Interaction Data using Multi-View Co-Teaching Network. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411929

1 INTRODUCTION

Nowadays, online recruitment platforms play an increasingly important role in connecting job seekers with employers. According to a report from LinkedIn[1], there were 660 million users and 20 million job listings on LinkedIn from about over 200 countries and territories all over the world as of late November 2019. With the huge amount of online recruitment data, it has become an essential task to automatically match jobs with suitable candidates, called *job-resume matching*, which aims to accelerate the recruitment process.

In online recruitment systems, employers publish the *job postings* (referred to *jobs* for short) describing what qualifications areas are essential to satisfactory performance in a position, and job seekers upload the *resumes* stating their skills, experience, and attributes. Basically, the two kinds of documents are mainly written in natural languages. Hence, a typical approach is to cast the job-resume matching task as a supervised text matching problem [20, 26]. In such a setting, these methods rely on a set of matched or unmatched job-resume pairs (termed as *job-resume interaction*) as training data. Apparently, the amount and quality of available job-resume interaction data directly affect the performance of job-resume matching algorithms.

However, on online recruitment platforms, job-resume interaction data is extremely sparse, and likely to contain noise. As shown in the report [21], on average, a job posting only has about three candidates for final interview. The major reason is that employers will carefully evaluate whether a job seeker is suitable for this job before sending an interview acceptation. While, other kinds of interactive behaviors (*e.g.,* chatting online and requesting profiles) are not accurate to reflect the final status for job-resume matching, so that it is not reliable to utilize interaction data as training

[1]https://www.businessofapps.com/data/linkedin-statistics/

data. Especially, negative feedbacks (*i.e.,* explicit rejection) are more difficult to obtain in practice. Many employers may not actively send an explicit rejection notification to unqualified candidates. In order to obtain sufficient negative samples, a simple way is to sample random pairs without acceptation status [29], called *negative sampling*. Such a way is problematic since it will incorporate noisy labeled data. For example, although online interaction has been finished without an explicit success, an employer and job seeker may actually schedule the interview via phones in an offline way.

To address the above issues, we consider capturing multi-view data signals for alleviating the sparsity of explicit interaction data. Inspired by collaborative filtering algorithms in recommender systems [12, 22], our idea is to mine underlying correlations among jobs and resumes for enhancing the representations by aggregating useful evidence from similar jobs or resumes. Since job-resume interaction data itself is sparse, we do not directly learn the correlation characteristics from such interaction data. Instead, we notice that job postings or resumes are usually written in a skill-oriented way and associated with a specified position. Containing the same skill keywords or position labels is an important indicator for the correlation between two documents. Figure 1 has presented a motivating example for our idea. As we can see, job j_1 is a minor position with few historical matched cases. It is relatively difficult to directly match it with suitable candidates. While, j_1 shares some same skill requirements with two other jobs of j_2 and j_3. Interestingly, j_2 and j_3 have historical matched resumes r_2 and r_3, respectively. Furthermore, we can indeed find that resume r_1 contains composite skills that are contained in r_2 and r_3. Although these skills might not be exactly the same as those required by j_1, it is likely that r_1 is a good candidate for j_1. In this case, we can see that relation-based semantic signal is potentially useful to improve a pure text-based matching model. Hence, we aim to develop a multi-view learning approach that is able to effectively integrate both kinds of data signals for deriving a more capable, robust matching model.

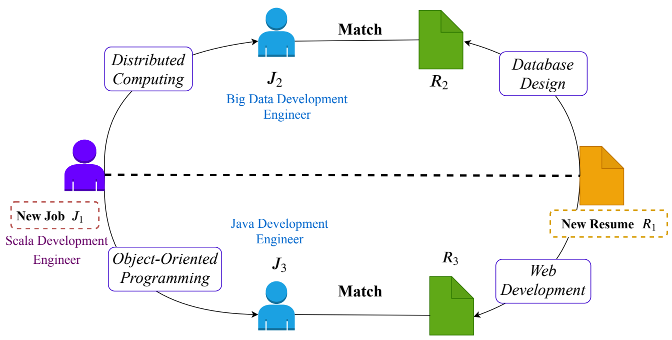


Figure 1: A motivating example for relation-based match. Here, we present three jobs $\{j_1, j_2, j_3\}$ and three resumes $\{r_1, r_2, r_3\}$, where $\langle j_2, r_2\rangle$ and $\langle j_3, r_3\rangle$ are matched pairs in training data.

To this end, in this paper, we propose a novel multi-view co-teaching network from sparse interaction data for job-resume matching. Our network consists of two major components, namely text-based matching model and relation-based matching model. The two parts capture semantic compatibility in two different views, and are designed to complement each other. The text-based matching component is implemented by a hierarchical self-attention text encoder, using the Transformer architecture [27] and the pre-trained BERT [3] model. In this way, we can obtain the corresponding text representations of jobs and resumes. For the relation-based matching component, we first construct the job-resume relation graph, in which jobs or resumes are considered as nodes and their relation-specific connections are considered as links. Furthermore, we develop a relational graph neural network for learning node representations based on the job-resume relation graph. Having both components, the key point is how to integrate them into a unified approach. We design two strategies for combining the merits of the two models. First, we let the two parts share the learned parameters or representations, so that the original representations of each part can be enhanced. More importantly, we adopt a co-teaching mechanism to reduce the influence of noise in training data. The core idea is to let the two components help each other by selecting more reliable training instances. The two strategies focus on *representation enhancement* and *data enhancement*, respectively. Compared with pure text-based matching models, the proposed approach is able to learn better representations from limited or even sparse interaction data, which is more resistible to noise in training data.

To evaluate the effectiveness of our model, we conduct extensive experiments on three real-world datasets. Experimental results demonstrate that our model outperforms several state-of-the-art methods for job-resume matching. To the best of our knowledge, it is the first time that both text- and relation-based matching models are integrated into a unified approach for the job-resume matching task. Our approach specially considers the sparsity issue of training data and is also resistible to noisy data.

2 RELATED WORK

This paper aims to learn an effective job-resume matching model with noisy interaction data. Thus we review the related work in the fields of job-resume matching and learning with noisy labels respectively.

Job-resume matching. Matching jobs and resumes stands at the core of a recruitment platform. As an important task in recruitment data mining [13, 25], person-job fit has been extensively studied in the literature. Early methods cast this problem as a recommendation task [4, 17], and the matching capability is obtained based on the collaborative filtering assumption. To alleviate this problem, recent research mostly focused on text matching technology, where the basic problem is how to represent the document. Around this problem, Shen *et al.* [26] proposed to encode the job and resume based on CNN. Qin *et al.* [20] leveraged RNN and BiLSTM to capture textual sequential information for more accurate semantic representation. In order to discriminate different sentence importances so as to achieve better encoding accuracy, Yan *et al.* [30] proposed a profiling memory module to learn the latent preference representation by interacting with both the job and resume sides. Bian *et al.* [2] used hierarchical attention-based RNN to match the jobs and resumes. In addition, Luo *et al.* [18] studied the effectiveness of adversarial training for the job-resume matching problem. Unlike these models, which focus on either the relations or the semantics of the jobs and resumes, we combine both of these information by a unified framework to alleviate the data sparsity and noisy problems.

Learning with noisy labels. Basically, learning with noisy labels aims to solve the problem when some of the training data is unreliable in supervised learning. Early methods, such as the curriculum learning [1] and self-paced learning [16], alleviate this problem by reordering the training samples. Easy instances will be learned before the harder ones. Following this idea, many variants have been proposed, such as the methods based on deep reinforcement learning [5], MentorNet [11] and UBD [19]. In parallel to the curriculum learning, several studies leverage different weighting mechanisms to lower the impact of the noisy instances [28].

Our paper is inspired from a recent work called co-teaching network (CTN) [7]. The co-teaching network adopts a *learning to teach* strategy for dealing with noise [6] and unlabeled data [31]. However, there are several significant differences with our work. First, CTN identifies the noisy samples within the same view, while our model leverages different information (*i.e.,* relation and text) to examine noisy instances. In addition, we design a "re-weighting" mechanism tailored for one-class classification problem, and apply it into the job-resume matching task.

3 PROBLEM DEFINITION

Suppose that we have a set of jobs $\mathcal{J} = \{j_1, j_2, \cdots, j_n\}$ and a set of resumes $\mathcal{R} = \{r_1, r_2, \cdots, r_m\}$, where n and m are the total numbers of jobs and resumes, respectively. Each job or resume is represented by a text document describing the position's requirements or the applicant's skills, respectively. We are also given an observed (training) matching set $\mathcal{Y} = \left\{\langle j, r, y_{j,r}\rangle | j \in \mathcal{J}, r \in \mathcal{R}\right\}$, where $y_{j,r}$ is a binary label indicating the final match result between job j and resume r: *accept* (YES) or *reject* (NO). Based on the above notations, our task is to learn a predictive function $f(j', r')$ based on the matching set $\mathcal{Y}$, so that it can accurately estimate the matching degree for an unseen (testing) job-resume pair $\langle j', r'\rangle$. In practice, the interaction data for the job-resume matching task is usually extremely sparse (*i.e.,* $n \times m \gg |\mathcal{Y}|$), and the training data may be also noisy (*e.g.,* the randomly sampled negative instances). Previous methods mainly focus on learning effective text representations, and cast the task as a supervised text matching task. Especially, they seldom consider the influence of the quality of training data on the model performance. In what follows, we present our multi-view co-teaching network for addressing these issues, where we characterize the matching patterns from different views, and leverage their complementary features to improve the training instances.

4 THE PROPOSED APPROACH

In this section, we introduce the proposed approach for the job-resume matching task in detail. The overall framework is presented in Figure 2. On one hand, jobs and resumes are described in text documents. We adopt a hierarchical self-attention text encoder for capturing text semantics, called *text-based matching model*. On the other hand, we construct a relation graph, regarding the jobs and resumes as nodes and their underlying correlations as links. We cast this task as link prediction and develop a graph neural network based model, called *relation-based matching model*. As motivated in Section 1, the two models have their own merits, so we further integrate them into a unified multi-view co-teaching network. Figure 2 presents the overall architecture of our proposed approach. Next, we describe each part in detail.

4.1 Text-based Matching Component

For text-based matching approaches, they seek to represent job text and resume text in a suitable way, and then construct the matching model based on semantic similarity. A key difficulty lies in how to effectively represent the job and resume documents.

Many text representation models have been explored in this task, including LSTM [32] and CNN [14]. More recently, self-attention mechanisms (*e.g.,* Transformer [27]) and its extensions on pre-trained model (*e.g.,* BERT [3]) have made great progress in various natural language processing tasks. However, BERT usually has a length limit on the input text, *e.g.,* 512 words, which prevents the accurate modeling of long documents. Besides, we believe that the sentence boundary should be useful signal to text representations, *e.g.,* a sentence corresponds to a skill requirement. Based on above considerations, we develop a hierarchal self-attention text representation model for developing the semantic matching model, in which a BERT-based encoder is first adopted to represent sentences, and then a Transformer-based encoder is used to represent the overall document based on learned sentence embeddings.

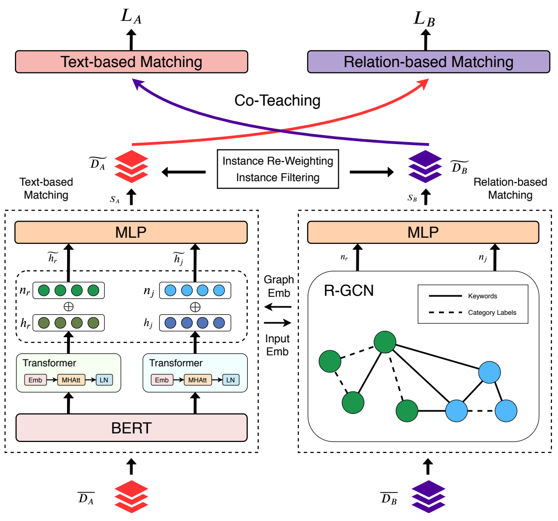


Figure 2: The overall architecture of our proposed approach.

4.1.1 *BERT-based Sentence Encoder.*

The first layer of our model is a sentence encoder implemented by a standard BERT model, which is a bidirectional Transformer with multiple layers. Given a job or resume sentence, a special token "*CLS*" has been inserted into the beginning of the sentence. For each token in the input sentence, two kinds of embeddings are considered as input, including token embeddings indicate the meaning of each token, and position embeddings indicate the position of each token within the text sequence. These two embeddings are summed to a single input vector and fed to the BERT encoder. The learned representation for the "*CLS*" symbol is treated as the sentence representation.

4.1.2 Hierarchical Transformer Encoder. The document encoder is developed on top of the BERT-based sentence encoder. Given a job or resume, it takes as input the sentence embeddings and outputs the overall document representation. Using a hierarchical architecture, our encoder is able to model very long documents, and also keep the sentence boundaries. Formally, the update formulas for our document encoder are given as follows:

$$\tilde{\boldsymbol{h}}_r^{(l)} = \mathrm{LN}\left(\boldsymbol{h}_r^{(l-1)} + \mathrm{MHAtt}\left(\boldsymbol{h}_r^{(l-1)}\right)\right), \quad (1)$$

$$\tilde{\boldsymbol{h}}_j^{(l)} = \mathrm{LN}\left(\boldsymbol{h}_j^{(l-1)} + \mathrm{MHAtt}\left(\boldsymbol{h}_j^{(l-1)}\right)\right), \quad (2)$$

$$\boldsymbol{h}_r^{(l)} = \mathrm{LN}\left(\tilde{\boldsymbol{h}}_r^{(l)} + \mathrm{FFN}\left(\tilde{\boldsymbol{h}}_r^{(l)}\right)\right), \quad (3)$$

$$\boldsymbol{h}_j^{(l)} = \mathrm{LN}\left(\tilde{\boldsymbol{h}}_j^{(l)} + \mathrm{FFN}\left(\tilde{\boldsymbol{h}}_j^{(l)}\right)\right), \quad (4)$$

where j denotes a job document, r denotes a resume document, $\boldsymbol{h}_r^{(l)}$ and $\boldsymbol{h}_j^{(l)}$ are the l-th layer input resume and job vectors, *LN* is the layer normalization operation, and *MHAtt* is the multi-head attention operation.

Let L denote the number of layers in the Transformer network. The final output layer is a sigmoid classifier, defined as

$$\hat{y}_{j,r} = \sigma\left(\boldsymbol{W}_1\left[\boldsymbol{h}_j^{(L)}; \boldsymbol{h}_r^{(L)}\right] + b_1\right), \quad (5)$$

where $\boldsymbol{h}_j^{(L)}$ and $\boldsymbol{h}_r^{(L)}$ are the representations at the final layer (*i.e.,* the L-th layer) for job document j and resume document r respectively, $\boldsymbol{W}_1$ is a parameter matrix for transforming the concatenated job and resume document representations, b_1 is a bias, and $\hat{y}_{j,r} \in (0,1)$ indicates the matching degree between job j and resume r.

4.2 Relation-based Matching Component

In the above model, jobs and resumes are matched according to their text content. The matching results are derived from their semantic compatibility. Here, we take a new perspective to study the job-resume matching task. Intuitively, there exist implicit correlations among jobs and resumes. For example, two similar jobs will be attractive to the same applicant, and a company may have a few comparable resumes for a job position. Since explicit matching interaction (no matter *success* or *failure*) is sparse, mining underlying implicit correlation will be useful to extract additional signals to complement semantic compatibility. For this purpose, we first construct a job-resume relation graph for capturing implicit correlations, and then develop a matching model using relational graph neural networks.

4.2.1 Construction of the Job-Resume Relation Graph. Before presenting the specific construction algorithms, we first formally define the job-resume relation graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the node set and the $\mathcal{E}$ is the link set. For our task, the node set is the union between job set and resume set, *i.e.,* $\mathcal{V} = \mathcal{J} \cup \mathcal{R}$, and the link set contains all the interaction links between two nodes on the graph, *i.e.,* $\mathcal{E} = \{\langle v_1, v_2, t_{v_1,v_2}\rangle | v_1, v_2 \in \mathcal{V}, t_{v_1,v_2} \in \mathcal{T}\}$, where $\mathcal{T}$ is the relation set and t_{v_1,v_2} is the relation label for the link between v_1 and v_2. Since we have two kinds of nodes, there are three possible types of links to consider in our task, *i.e., job-to-resume*, *job-to-job* and *resume-to-resume*. We consider two kinds of data signals to create implicit links between two nodes, either *category label* and *keyword*.

Link Creation using Category Labels. In an online recruitment platform, in order to organize the job and resumes, there is usually a job taxonomy, in which a higher (or lower) level indicates a more coarse-grained (or fine-grained) kind of job positions. We only consider the category labels at the bottom layer, which has good discrimination capacity on specifying some kind of jobs, *e.g., JAVA software engineer* or *accountant*. Here, we create links between two job nodes or two resume nodes if they share the same category label. A formed link is further attached with the corresponding category label as the relation label.

Link Creation using Keywords. In both job and resume documents, not all the keywords are equally informative for predicting the final match results. Some words may contain more important semantics (*e.g.,* the description words for skill). Here, we would like to extract a list of keywords from both job and resume documents, and identify underlying correlation through such indicators. The overall algorithm can be described as follows. First, we use the classic *tf-idf* term weighting method to select a list of candidate words. Then, we construct a word co-occurrence graph by counting the frequency that two words co-occur in a job or resume document. Third, we run a standard PageRank algorithm on the word co-occurrence graph and obtain the PageRank scores of each word. Finally, we sort the words according to their PageRank scores, and only keep the top K words as the *keywords*. When the text of two nodes (either a job or a resume) contains a same keyword, we create a link between them and attach the keyword as relation label.

4.2.2 Learning Job and Resume Representations. Based on the job-resume relation graph, we further learn effective representations for capturing the underlying semantics reflected by the graph for job-resume match. Recently, graph neural networks have become one popular class of models for learning node characteristics from the graph-structured data, *e.g.,* graph convolutional networks (GCN) [15]. However, traditional GCN mainly deals with homogeneous links, which cannot effectively characterize different types of links. Inspired by the recent progress made in knowledge graph completion [23], we adopt the Relational Graph Convolutional Network (RGCN) [24] to model the relation-specific links. Based on GCN, the core difference of RGCN lies in the aggregation step, where we collect the incoming information from neighbors and perform the aggregation operation according to different types of relations.

Formally, the node representation at the l-th layer is derived by its neighbors' embeddings at the previous layer as:

$$\boldsymbol{n}_v^{(l+1)} = \sigma\big(\underbrace{\sum_{t \in \mathcal{T}} \frac{1}{|\mathcal{V}_v^t|} \sum_{v' \in \mathcal{V}_v^t} \boldsymbol{W}_t^{(l)} \boldsymbol{n}_{v'}^{(l)}}_{\text{propagation } \textit{w.r.t.} \text{ specific relations}} + \boldsymbol{W}_o^{(l)} \boldsymbol{n}_v^{(l)}\big), \quad (6)$$

where v is a node placeholder that can be a job node or a resume node, $\boldsymbol{n}_v^{(l)} \in \mathbb{R}^D$ is the representation of node v at the l-th layer, $\mathcal{V}_v^t$

denotes the set of v's neighbors under the relation type t, $\boldsymbol{W}_t^{(l)}$ is a parameter matrix specific to relation t, and $\boldsymbol{W}_o^{(l)}$ is the parameter matrix specific to original node representation.

In this equation, each relation type t is associated with a specific parameter matrix $\boldsymbol{W}_t^{(l)}$, so that node representations are able to incorporate relation-aware semantics and reduce the influence of irrelevant aspects.

Once we have derived the node representations, the matching score between a job and a resume can be defined in a similar way as Eq. 5:

$$\hat{y}_{j,r} = \sigma\left(\boldsymbol{W}_2\left[\boldsymbol{n}_j^{(L')}; \boldsymbol{n}_r^{(L')}\right] + b_2\right), \tag{7}$$

where is the predicted $\boldsymbol{n}_j^{(L')}$ and $\boldsymbol{n}_r^{(L')}$ are the representations at the final layer for job document j and resume document r respectively, and $\boldsymbol{W}_2$ and b_2 are the parameter matrix or vector.

4.3 Multi-View Co-Teaching Network

Previously, we have described two individual components for job-resume matching from different perspectives, namely text- and relation-based views. Each of the two models has its own merits on our task, and we further study how to integrate them into a unified approach.

We design two integration strategies and develop a multi-view co-teaching network. First, we share the learned information or parameters for enhancing the original representations of each components. Second, as motivated in Section 1, we focus on reducing the influence of noise from training data, especially for negative samples. We borrow the idea of co-teaching method [7] introduced in machine learning, and let the two components help each other by selecting more reliable training instances. Next, we present the details of the two strategies.

4.3.1 Representation Enhancement. Recall that for a job j we have learned two kinds of different representations for both jobs and resumes, namely the text-based representation $\boldsymbol{h}_j$ and relation-based representation $\boldsymbol{n}_j$. Similarly for a resume r, we have the representations of $\boldsymbol{h}_r$ and $\boldsymbol{n}_r$, correspondingly.

To enhance the semantic matching model, we concatenate the original text-based representation with the relation-based representation as:

$$\widetilde{\boldsymbol{h}}_j = \boldsymbol{h}_j \oplus \boldsymbol{n}_j, \tag{8}$$

$$\widetilde{\boldsymbol{h}}_r = \boldsymbol{h}_r \oplus \boldsymbol{n}_r, \tag{9}$$

where "$\oplus$" is the vector concatenation operation. Furthermore, we feed the enhanced representations into Eq. 5 for achieving a better prediction result. For the relation-based matching model, we do not directly adopt the above concatenation method. Instead, the initialization of the RGCN model is particularly important to the final performance. Hence, we utilize the learned representations from semantic matching model for initializing node states:

$$\boldsymbol{n}_j^{(0)} = \boldsymbol{h}_j, \tag{10}$$

$$\boldsymbol{n}_r^{(0)} = \boldsymbol{h}_r. \tag{11}$$

Algorithm 1: The proposed co-teaching algorithm.

```
   Input:
   The set of training data, $\mathcal{D}$;
   Model parameters $\theta_A$, $\theta_B$;
   Epoch $E$ , iteration $N$, learning rate $\eta$;
   Output: $\theta_A$, $\theta_B$;
1  for $b = 1, 2, \ldots, N$ do
2      Shuffle training set $\mathcal{D}$;
3      for $e = 1, 2, \ldots, E$ do
4          Fetch a batch of training data $\overline{D}$;
5          Learning $(\widetilde{\mathcal{D}}_B, L_B)$ from model $A$ and $\overline{\mathcal{D}}_B$;
6          Learning $(\widetilde{\mathcal{D}}_A, L_A)$ from model $B$ and $\overline{\mathcal{D}}_A$;
7          Update $\theta_A = \theta_A - \eta \nabla L_A(\widetilde{\mathcal{D}}_A)$;
8          Update $\theta_B = \theta_B - \eta \nabla L_B(\widetilde{\mathcal{D}}_B)$;
9      end
10 end
```

4.3.2 Co-Teaching for Data Enhancement. In this part, we study the second integration strategy for enhancing the quality of training data. Our basic assumption is that true samples usually receive similar predictions under different model views, while the noisy ones are not easy to cheat all the models. In the co-teaching framework, our two components can be considered as two peer learners. The samples for training one learner will be firstly examined by the other one, and only the instances labeled as "high quality" will be kept in the training phrase. Since the two learners have very different views to model the data characteristics, it is expected that they can help each other to select "high quality" training samples, and thus improve the final performance.

Overall Algorithm. The overall co-teaching process is presented in Algorithm 1. Formally, let A and B denote text-based and relation-based matching models, respectively. For each batch update, a batch dataset $\overline{\mathcal{D}}$ is randomly split into two equal-sized subsets namely $\overline{\mathcal{D}}_A$ and $\overline{\mathcal{D}}_B$ (*line* 4). Instead of directly feeding them into model A and B, $\overline{\mathcal{D}}_A$ and $\overline{\mathcal{D}}_B$ are firstly "examined" by their peer learner to filter noisy samples (*line* 5-6). Finally, based on the derived new datasets $\widetilde{\mathcal{D}}_A$ and $\widetilde{\mathcal{D}}_B$, the parameters of A and B will be updated using stochastic gradient decent (SGD) (*line* 7-8). As we can see, the key point of this algorithm lies in peer examination (*line* 5-6). Next, we give two implementations for peer examination.

Training with Instance Re-Weighting. The first training method is to re-weight the training instances. Given a model, its peer model aims to increase the weight of "high-quality" samples and decrease the weight of unreliable samples from its perspective. In order to train model B, we assume that K instances $\overline{\mathcal{D}}_B = \{\langle j_i, r_i, y_i\rangle\}_{i=1}^K$ are generated during a batch update in the training process. We let model A assign a weight to each instance in $\overline{\mathcal{D}}_B$, denoted by w_i. The key idea is to penalize the instance according to the disagreement degree between the given information and A's prediction:

$$w_i^A = 1 - s_A(j_i, r_i, y_i), \tag{12}$$

where $s_A(j_i, r_i, y_i)$ is the confidence score by model A to predict the label of y_i for the pair $\langle j_i, r_i\rangle$. After instance re-weighting, we can generate a new batch training dataset by augmenting the original instances with the weights, $\widetilde{\mathcal{D}}_B = \{\langle j_i, r_i, y_i, w_i^A \rangle\}_{i=1}^K$. Then, we can rewrite the loss function in a batch to train model B as:

$$L_B(\overline{\mathcal{D}}_B) = \sum_{i=1}^{K} w_i^A \cdot g\left(y_i, y_i^B\right), \tag{13}$$

where $g\left(y_i, y_i^B\right)$ is the loss for B's prediction y_i^B. As mentioned before, the noisy information is mainly from negative sampling. Hence, we fix the weights for positive instances and true negative instances as 1 in the learning process, and the instances obtained by sampling from the dataset are weighted according to Eq. 12.

Training with Instance Filtering. Besides re-weighting different samples, we can also directly drop the instances which are not "good enough". Intuitively, if an instance can lead to a small loss value by a model, then it is far from the decision boundary, and is more likely to be a reliable sample instead of noises. This idea is modeled by the following formula:

$$\widetilde{\mathcal{D}}_B \leftarrow \operatorname{argmin}_{\widetilde{\mathcal{D}}_B \subset \overline{\mathcal{D}}_B, \left|\widetilde{\mathcal{D}}_B\right| = \delta |\overline{\mathcal{D}}_B|} L_A\left(\widetilde{\mathcal{D}}_B\right), \tag{14}$$

where $L_A\left(\widetilde{\mathcal{D}}_B\right)$ denotes the accumulative loss of A on the given dataset, and δ is a hyper-parameter defining the held-out rate. Here, we select a subset from the original data $\overline{\mathcal{D}}_B$ that is able to minimize the loss from the peer model.

In the above, we only discuss the case of updating B with peer model A. It is similar to update model A with peer model B. Both instance re-weighting and filtering methods aim to select more reliable samples for model learning. They achieve this purpose with different approaches. Instance re-weighting is a "soft" method, where all the instances are remained, but with different training importances. As a comparison, instance filtering is a "hard" method, where some samples are directly dropped. One can also combine these two methods by filtering the samples before re-weighting them.

Complexity Analysis. Compared with traditional supervised job-resume matching methods, the increased training complexity for instance re-weighting depends on the loss computation in Eq. 13. Suppose the complexities for computing the loss (or weight) of a sample by model A and B are C_A and C_B, respectively. Then the total additional complexity is $O(NM(C_A+C_B))$ for instance re-weighting, where N and M are the numbers of training epochs and instances, respectively. For the method of instance filtering, we need to compute the sampling loss and rank them for selecting a candidate set. Since the ranking operation takes a cost of $O(NM \log M)$, the total additional complexity for this method is $O(NM(C_A + C_B + \log M))$.

5 EXPERIMENTS

In this section, we conduct extensive experiments to verify effectiveness of our model. We mainly address the following research questions:

• **RQ1.** Whether our method can outperform the state-of-the-art job-resume matching models?

Table 1: Statistics of the datasets. #Accept/#Reject denote the number of explicit interaction that an employer sends notifications of acceptance or rejection; #Clicking denotes the number of interaction that an employer clicks the link to homepage of a job seeker and chats online after reviewing a resume, but does not send notifications; #Browsing denotes the number of interaction that an employer only reviews a resume without subsequent behavior.

Statistics	Technology	Sales	Design
#Jobs	21695	7964	7332
#Resumes	35902	6731	10347
#Reject	2875	567	1124
#Accept	109125	16417	31293
Density	0.0144%	0.0317%	0.0427%
#Clicking	7468577	2523337	2615530
#Browsing	14110159	2484599	3863614

• **RQ2.** What are the effects of different components in our model?

• **RQ3.** How does the performance the performance of our model vary with different parameters or settings?

• **RQ4.** What is the effect of our co-teaching mechanism from the qualitative perspective?

In what follows, we first setup the experiments, and then present and analyze the evaluation results to answer these questions.

5.1 Experimental Setup

5.1.1 Datasets: We evaluate our model on a real-world dataset provided by a popular online recruiting platform named "BOSS Zhipin" (the BOSS Recruiting)[2] in China. To protect the privacy of candidates, all the records have been anonymized by removing identity information. The original dataset is split into three categories to test the robustness of our model for different domains. The statistics of the processed data are summarized in Table 1. We can see: (1) All the datasets are extremely sparse, where the density ranges from 0.0144% to 0.0427%; (2) Different categories correspond to varying data characteristics, *e.g., Technology* is a large and sparse dataset, while *Sales* is much smaller but denser. (3) For each category, the number of reject samples (*i.e.,* negative instances) is much smaller than that of the accept ones (*i.e.,* positive instances). Since such an imbalanced dataset may bias the model learning process [8], a commonly adopted method to balance the data distribution is to randomly sample job-resume pairs without explicit status [20]. We consider two types of samples as negative instances: (1) an employer clicks the link to homepage of a job seeker and then chats online after reviewing her/his resume, but does not send notifications (*clicking*), and (2) an employer reviews the resume without any further behavior (*browsing*). As discussed before, sampled negative samples are likely to be "*false negative*": although without explicit online acceptance notification, they may have actually reached the agreement on the job offer. To incorporate such negative samples in training data, we would like to examine

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

the capability of our model that learns from noisy data. By equally sampling from these two resources, the final ratio between the numbers of positive and negative samples is set to 1:1. It should be noted that we only use these negative samples in the training stage, while for the validation and testing sets, we use the samples with explicit acceptance or rejection status to make sure our evaluation is accurate and reliable.

5.1.2 Baselines: We compare our model with the following representative methods:

• **DSSM** [10] leverages convolutional layers to extract semantic information for making the final prediction.

• **NFM** [9] extends factorization machine (FM) with neural architectures to learn nonlinear and high-order interaction signals, where we use the textual and ID features as input.

• **BPJFNN** [20] leverages BiLSTM to derive the job and resume representations.

• **PJFNN** [26] is a CNN based method, and the matching degree is computed by the cosine similarity.

• **APJFNN** [20] proposes to use hierarchical recurrent neural networks to process the job and resume contents, and the final prediction is regarded as a classification problem.

• **JRMPM** [30] captures the preference information of the jobs and resumes by introducing a profiling memory module.

• **DGMN** [2] is a deep model focusing on the global sentence interactions when matching the jobs and resumes.

• **UBD** [19] is a method designed for learning from noisy labels, and updates parameters based on their disagreement between the two classifiers.

Our baselines have covered most of the recently proposed job-resume matching models with different model architectures.

5.1.3 Evaluation and implementation details. Following previous work [20, 26], we adopt the evaluation metrics including **AUC**, **Accuracy**, **Precision**, **Recall** and **F1** to evaluate our models.

For each category, we first split the augmented dataset into train, validation, and test sets with an approximate ratio of 8:1:1. Note that all the instances in validation and tests are guaranteed to be with ground-truth labels. While, the negative instances in training set are likely to contain noisy labels, as they are sampled from the interaction pairs without explicit status in the recruitment platform (see Section 5.1.1).

When tuning our model parameters, the sentence embeddings are initialized via the BERT-Base-Chinese[3], and the document representation is derived as the output of the Transformer encoder. The filter parameter δ and the number of Transformer layers are varied in the ranges of $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $\{1, 2, 3, 4\}$, respectively. The batch size is empirically set to 32. The Adam optimizer is used to learn our model, and the learning rate is tuned in $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$. Early stopping is used with a patience of 5 epochs. For the baselines, the parameters are set as their default values or tuned on the validation set. Our dataset and code are available at this link: https://github.com/RUCAIBox/Multi-View-Co-Teaching.

[3]https://github.com/huggingface/transformers

5.2 RQ1: The Overall Comparison

Table 2 presents the comparison between our model and the baselines. First, the feature interaction method NFM is difficult to achieve good results on our task. The main reason is that the sparsity level of the interaction between job and resume is much smaller than that of traditional recommendation systems. Meanwhile, DSSM does not perform well in most cases because it fails to capture the sequential properties in textual information. The performance difference between BPJFNN, PJFNN, APJFNN, JRMPM and DGMN is small, and the winner varies on different metrics or datasets. Furthermore, UBD is the only baseline that specially learns from noisy labels, which updates parameters using the instances with different predictions from multiple classifiers. As we can see that, it substantially improves over the other baseline methods on all the datasets, indicating the necessarity of handling noisy labels.

As a comparison, our model achieves the best performance on all the metrics across different datasets. In specific, our model can on average improve the best baseline by 3.1%, 2.9% and 3.5% on the datasets of Technology, Sales and Design, respectively. This result positively answers our first research question (**RQ1**), and verifies the effectiveness of our designed multi-view co-teaching network. Compared with the baselines, the co-teaching mechanism in our model is able to identify more informative and reliable samples for learning the parameters. Our model is potentially more resistible to the negative impact brought by noisy data, and thus performs better than the other comparison methods. Comparing the two strategies of instance filtering and re-weighting, we find that the latter is better in most cases. A possible reason is that re-weighting strategy adopts a "soft" denoising approach that is more robust in dealing with noisy labels.

5.3 RQ2: Ablation Study

To effectively learn from sparse, noisy interaction data for job-resume matching, our model has incorporated three technical components including text-based matching model (denoted by $\underline{\text{T}}$), relation-based matching model (denoted by $\underline{\text{R}}$) and co-teaching mechanism (denoted by $\underline{\text{C}}$). Here, we examine how each of them affects the final performance.

We consider the following five variants of our approach for comparison: **(A)** $\underline{\text{R}}$ is the single relation-based matching model, **(B)** $\underline{\text{T}}$ is the single text-based matching model, **(C)** $\underline{\text{TR}}$ is the simple fusion model that directly averages the output from the text- and relation-based components, **(D)** $\underline{\text{TTC}}$ is a co-teaching method that only utilizes text-based matching model, and **(E)** $\underline{\text{RRC}}$ is a co-teaching method that only utilizes relation-based matching model. Here, $\underline{\text{TRC}}$ denotes our proposed model. In the experiments, the parameters are remained as the default settings, taking re-weighting strategy for the co-teaching mechanism. Due to the space limit, we only report the results on the Technology dataset, and similar conclusions can be obtained on the other datasets.

In Table 3, we can see that the performance order can be summarized as: $\underline{\text{R}} < \underline{\text{TR}} < \underline{\text{RRC}} < \underline{\text{T}} < \underline{\text{TTC}} < \underline{\text{TRC}}$. These results indicate that all the three components are indeed useful to improve the performance of job-resume matching. Especially, the text-based matching model and co-teaching mechanism bring more improvement to our approach. Besides, an interesting observation

Table 2: Performance comparison between the baselines and our model. For each metric on different datasets, we use bold fonts and "*" to mark the best performance and the best baseline performance, respectively. MV-CoN(F) and MV-CoN(R) are our models with instance filtering and re-weighting as the co-teaching strategies, respectively.

Dataset	Technology					Sales					Design				
Metric	AUC	ACC	P	R	F_1	AUC	ACC	P	R	F_1	AUC	ACC	P	R	F_1
DSSM	0.690	0.636	0.645	0.636	0.640	0.682	0.629	0.640	0.631	0.635	0.696	0.641	0.649	0.638	0.644
NFM	0.652	0.608	0.613	0.602	0.607	0.654	0.601	0.612	0.606	0.609	0.668	0.618	0.627	0.620	0.624
BPJFNN	0.704	0.650	0.651	0.638	0.644	0.696	0.644	0.648	0.632	0.640	0.708*	0.655	0.655	0.639	0.647
PJFNN	0.686	0.639	0.639	0.630	0.634	0.677	0.633	0.635	0.625	0.630	0.690	0.643	0.645	0.631	0.638
APJFNN	0.700	0.648	0.650	0.639	0.644	0.694	0.641	0.645	0.634	0.639	0.705	0.652	0.656	0.640	0.647
JRMPM	0.694	0.645	0.643	0.633	0.638	0.688	0.636	0.639	0.628	0.634	0.697	0.648	0.649	0.632	0.640
DGMN	0.699	0.651	0.652*	0.639	0.645	0.693	0.643	0.648	0.635	0.642	0.704	0.655	0.656	0.641	0.648
UBD	0.706*	0.652*	0.652*	0.642*	0.647*	0.698*	0.648*	0.654*	0.643*	0.648*	0.708*	0.658*	0.661*	0.643*	0.651*
MV-CoN(F)	0.728	0.672	0.679	0.646	0.664	0.725	**0.671**	0.677	0.640	0.659	0.736	0.679	0.682	0.646	0.663
MV-CoN(R)	**0.737**	**0.678**	**0.683**	**0.649**	**0.668**	**0.727**	0.669	**0.679**	**0.644**	**0.661**	**0.743**	**0.683**	**0.688**	**0.651**	**0.669**

Table 3: Effects of different model components. Here, T denotes text-based component, R denotes relation-based component and C denotes co-teaching component.

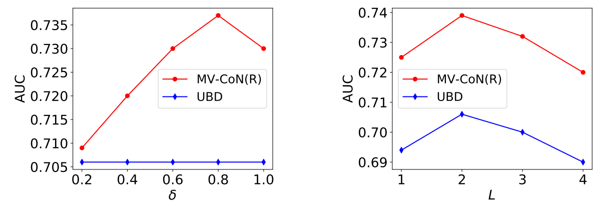
Metric	AUC	ACC	P	R	F_1
R	0.706	0.651	0.653	0.641	0.647
T	0.717	0.663	0.665	0.653	0.659
TR	0.712	0.654	0.659	0.648	0.653
TTC	0.720	0.666	0.668	**0.657**	0.662
RRC	0.714	0.655	0.662	0.647	0.654
TRC	**0.737**	**0.678**	**0.683**	0.649	**0.668**

is that simply fusing the multi-view data may not lead to a good performance (*i.e.,* TR < T). It indicates that we need to carefully design the fusion strategy that utilizes multi-view data for our task.

5.4 RQ3: Performance Tuning

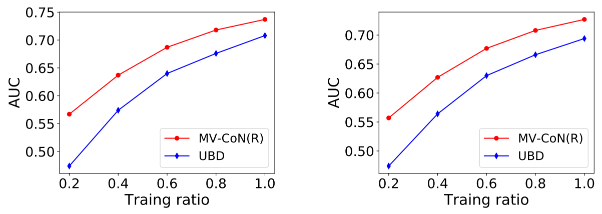
In this part, we examine the robustness of our model, and analyze the influence of parameters (or hyper-parameters) and training data on model performance. For simplicity, we only incorporate the best baseline UBD from Table 2 as a comparison.

In our model, we have two important parameters, namely the selection ratio δ of instance filtering strategy and the number of Transformer layers L. First, we vary the selection ratio δ in the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. As shown in Eq. 14, δ controls the data amount from a model for the peer model. In Figure 3(a), we can see that a selection ratio of 0.8 achieves the best performance for our model. As δ decreases below 0.8, the performance continues to drop. It is because that when filtering more training instances, we also tend to exclude true instances from training data. When the training set becomes small, it is not sufficient to train a good peer model. Then, we vary the number of Transformer layers in the set $\{1, 2, 3, 4\}$. It can be observed from Figure 3(b) that two-layer Transformer achieves the best performance for our model. While, the overall performance is relatively stable when we use more or fewer Transformer layers.



(a) The selection ratio of instance filtering. (b) The number of Transformer layers.

Figure 3: Performance tuning with the selection ratio of instance filtering strategy (δ) and the number of Transformer layers (L).

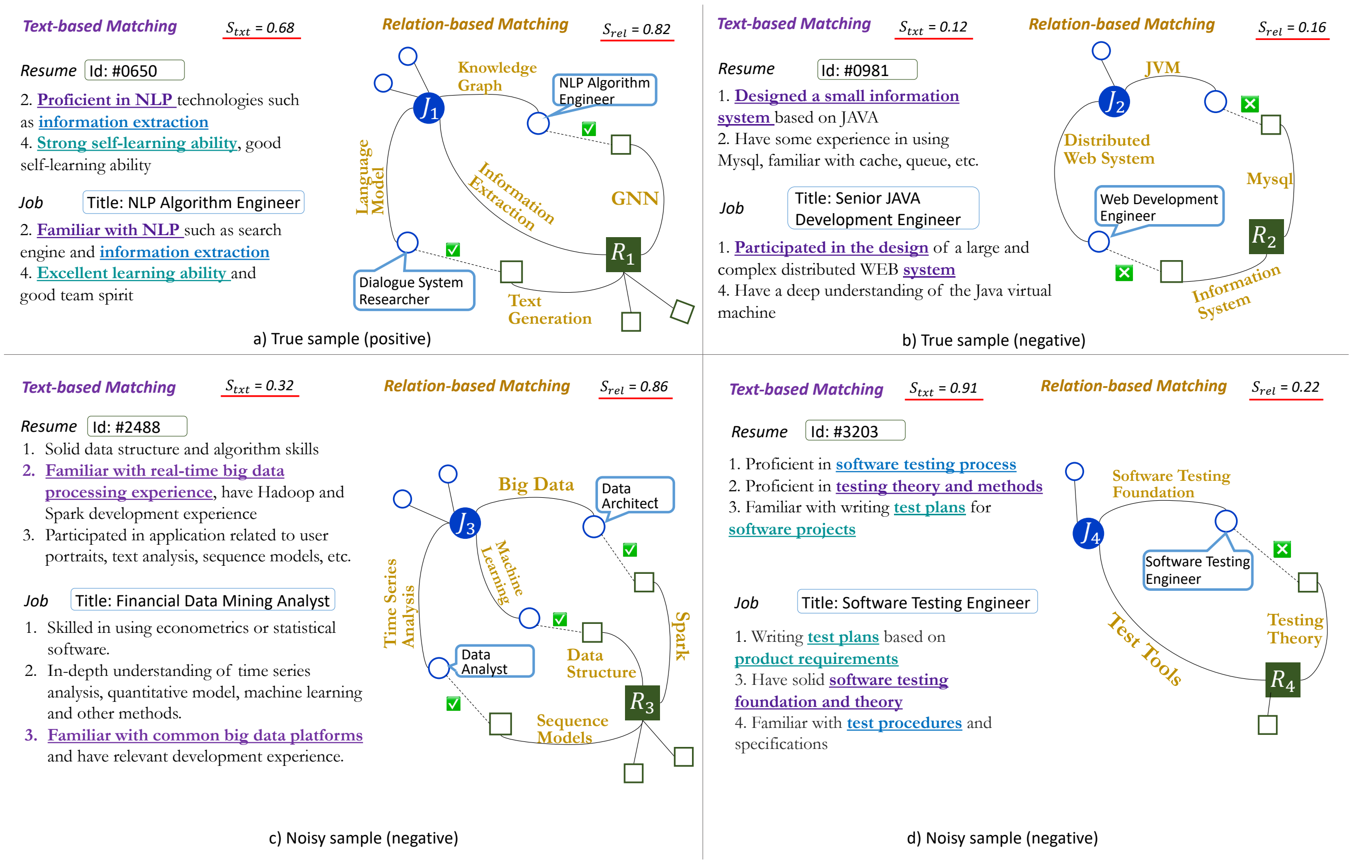


(a) Technology dataset. (b) Sales dataset.

Figure 4: Performance comparison by varying the amount of training data.

Next, we study the influence of training data size on model performance. To examine this, we take 20%, 40%, 60% and 80% from the complete training data to generate four smaller training sets. We fix the test set as original, then learn the model with new training sets, and finally report the corresponding results on the test set. Figure 4 presents the results of our model with different ratios of training data.

As we can see, our model consistently outperforms the best baseline in Table 2. Especially, when training data is extremely


Text-based Matching S_{txt} = 0.68

Resume Id: #0650

2. Proficient in NLP technologies such as information extraction
4. Strong self-learning ability, good self-learning ability

Job Title: NLP Algorithm Engineer

2. Familiar with NLP such as search engine and information extraction
4. Excellent learning ability and good team spirit

Relation-based Matching S_{rel} = 0.82

a) True sample (positive)

Text-based Matching S_{txt} = 0.12

Resume Id: #0981

1. Designed a small information system based on JAVA
2. Have some experience in using Mysql, familiar with cache, queue, etc.

Job Title: Senior JAVA Development Engineer

1. Participated in the design of a large and complex distributed WEB system
4. Have a deep understanding of the Java virtual machine

Relation-based Matching S_{rel} = 0.16

b) True sample (negative)

Text-based Matching S_{txt} = 0.32

Resume Id: #2488

1. Solid data structure and algorithm skills
2. Familiar with real-time big data processing experience, have Hadoop and Spark development experience
3. Participated in application related to user portraits, text analysis, sequence models, etc.

Job Title: Financial Data Mining Analyst

1. Skilled in using econometrics or statistical software.
2. In-depth understanding of time series analysis, quantitative model, machine learning and other methods.
3. Familiar with common big data platforms and have relevant development experience.

Relation-based Matching S_{rel} = 0.86

c) Noisy sample (negative)

Text-based Matching S_{txt} = 0.91

Resume Id: #3203

1. Proficient in software testing process
2. Proficient in testing theory and methods
3. Familiar with writing test plans for software projects

Job Title: Software Testing Engineer

1. Writing test plans based on product requirements
3. Have solid software testing foundation and theory
4. Familiar with test procedures and specifications

Relation-based Matching S_{rel} = 0.22

d) Noisy sample (negative)

Figure 5: An illustrative example for showing the effect of different views in our co-teaching network. Here, we use circles and squares to denote jobs and resumes, respectively. For the links between two nodes, we attach the relation types or interaction labels. We mark the matched or similar text across jobs and resumes in underlined fonts.

sparse (*i.e.,* 20%), the improvement becomes more significant. These results show that our model is able to alleviate the influence of data sparsity to some extent, since we have utilized multi-view data in a principled way.

5.5 RQ4: Qualitative Analysis

Above, we mainly verify the effectiveness of our model in a quantitative manner. It would be also helpful to present some case studies for understanding the working mechanism from a qualitative perspective. In this part, we present four representative examples from the Technology dataset, and illustrate how our model works on them.

In Figure 5, the upper part corresponds to two true samples, where the acceptance or rejection status has been confirmed in the recruitment platform. While, the bottom part corresponds to two noisy samples. They have been randomly sampled via via clicking or browsing interaction, originally labeled as negative samples in training data. However, after the investigation by platform staff members, the two pairs turn out to be *false negative*, *i.e.,* employment agreements have been reached offline for the two cases. Thus, the latter two samples are noisy instances, which are likely to affect the performance of the matching model.

From cases (a) and (b), we can see, both of the text- and relation-based models produce consistent matching scores for the true samples, while their working mechanisms are quite different. The text-based model assigns a high score (*i.e.*, 0.68) to case (a), because many skills in the resume, such as *"familiar with NLP"* and *"excellent learning ability"*, have exactly satisfied the job requirements. As a comparison, in the relation-based model, the matching result is mainly based on the relation-based connections (*e.g.*, *knowledge graph*, *GNN* and *information extraction*) and matching signals of neighbor nodes. Similar results can be also observed from the negative instance of case (b). These examples show that, in our approach, different model views can usually obtain consistent matching results for the true samples, though their mechanisms are different.

From cases (c) and (d), we can see that the two models have made inconsistent predictions on the noisy negative samples. For example, in case (c), since this position is a relatively new position (*i.e., Financial Data Mining Analyst*), the text-based model assigns a low score to the candidate job-resume pair, while the relation-based model considers it as a positive instance due to the rich connections through the constructed relation graph. Similar to case (c), although the relation-based model in case (d) negates the instance, the text-based model assigns a high matching score since the job-resume documents are similar in contents. Once a negative sample receives inconsistent prediction scores, our model will consider them as low-quality negative samples and penalize them with re-weighting (Eq. 12) or filtering (Eq. 14) strategy.

These examples have shown the capability of our model for noisy information filtering from a qualitative perspective.

6 CONCLUSION

This paper presented a multi-view co-teaching network that is able to learn from sparse, noisy interaction data for job-resume matching. We considered two views for developing the matching algorithm, namely text- and relation-based models. Furthermore, the two models were integrated into a unified approach that was able to combine their merits for this task. We designed two strategies for model integration, namely representation enhancement and data enhancement. Representation enhancement referred to the sharing of the learned parameters or representations across the two models; data enhancement referred to the process of filtering or re-weighting training instances according to their quality, which was implemented by the co-teaching algorithms. Extensive experiments showed that the proposed approach is able to achieve a better matching performance from sparse and noisy interaction data by comparing with several competitive baselines.

In this paper, we only focus on the macro interaction behaviors, *i.e.,* the acceptation of interview or rejection. While, it is intuitive that other kinds of micro interactive actions should be also useful to the matching task, such as *click* or *dwell time*. We will investigate into this topic and develop a more comprehensive interaction model. Besides, we will also consider applying our approach to more categories and study the domain adaptation problem across different categories.

ACKNOWLEDGEMENT

This work was partially supported by the National Key R&D Program of China (2019AAA0105200), the National Natural Science Foundation of China under Grant No. 61872369 and 61832017, Beijing Academy of Artificial Intelligence (BAAI) under Grant No. BAAI2020ZJ0301, and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China under Grant No.18XNLG22 and 19XNQ047. Xin Zhao is the corresponding author.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML 2009*. 41–48.

[2] Shuqing Bian, Wayne Xin Zhao, Yang Song, Tao Zhang, and Ji-Rong Wen. 2019. Domain Adaptation for Person-Job Fit with Transferable Deep Global Match Network. In *EMNLP-IJCNLP 2019*. 4809–4819.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019*. 4171–4186.

[4] Mamadou Diaby, Emmanuel Viennet, and Tristan Launay. 2013. Toward the next generation of recruitment tools: an online social network-based job recommender system. In *ASONAM 2013*. IEEE, 821–828.

[5] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2018. Learning to Teach. In *ICLR 2018*. 350–357.

[6] Jiazhan Feng, Chongyang Tao, Wei Wu, Yansong Feng, Dongyan Zhao, and Rui Yan. 2019. Learning a Matching Model with Co-teaching for Multi-turn Response Selection in Retrieval-based Dialogue Systems. In *ACL 2019*. 3805–3815.

[7] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS 2018*. 8536–8546.

[8] Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009), 1263–1284.

[9] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR 2017*. 355–364.

[10] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM 2013*. 2333–2338.

[11] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *ICML 2018*. 2309–2318.

[12] Miao Jiang, Yi Fang, Huangming Xie, Jike Chong, and Meng Meng. 2019. User click prediction for personalized job recommendation. *WWW 2019* (2019), 325–345.

[13] Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned. In *RecSys 2017*. 346–347.

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP 2014*. 1746–1751.

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR 2017*. 436–444.

[16] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *NeurIPS 2010*. 1189–1197.

[17] Yao Lu, Sandy El Helou, and Denis Gillet. 2013. A recommender system for job seeking and recruiting website. In *WWW 2013*. 963–966.

[18] Yong Luo, Huaizheng Zhang, Yonggang Wen, and Xinwen Zhang. 2019. ResumeGAN: An Optimized Deep Representation Learning Framework for Talent-Job Fit via Adversarial Learning. In *CIKM 2019*. 1101–1110.

[19] Eran Malach and Shai Shalev-Shwartz. 2017. Decoupling "when to update" from "how to update". In *NeurIPS 2017*. 960–970.

[20] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing Person-Job Fit for Talent Recruitment: An Ability-aware Neural Network Approach. In *SIGIR 2018*.

[21] 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 2018*. 2253–2261.

[22] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2010. Item-based collaborative filtering recommendation algorithms. In *WWW 2010*.

[23] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC 2018*. 593–607.

[24] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC 2018*. 593–607.

[25] Walid Shalaby, BahaaEddin AlAila, Mohammed Korayem, Layla Pournajaf, Khalifeh AlJadda, Shannon Quinn, and Wlodek Zadrozny. 2017. Help me find a job: A graph-based approach for job recommendation at scale. In *IEEE BigData 2017*. 1544–1553.

[26] Dazhong Shen, Hengshu Zhu, Chen Zhu, Tong Xu, Chao Ma, and Hui Xiong. 2018. A Joint Learning Approach to Intelligent Job Interview Assessment. In *IJCAI 2018*. 3542–3548.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS 2017*. 5998–6008.

[28] Lijun Wu, Fei Tian, Yingce Xia, Yang Fan, Tao Qin, Jian-Huang Lai, and Tie-Yan Liu. 2018. Learning to Teach with Dynamic Loss Functions. In *NeurIPS 2018*. 6467–6478.

[29] Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling. In *EMNLP 2015*. 536–540.

[30] Rui Yan, Ran Le, Yang Song, Tao Zhang, Xiangliang Zhang, and Dongyan Zhao. 2019. Interview Choice Reveals Your Preference on the Market: To Improve Job-Resume Matching through Profiling Memories. In *KDD 2019*. 914–922.

[31] Fengxiang Yang, Ke Li, Zhun Zhong, Zhiming Luo, Xing Sun, Hao Cheng, Xiaowei Guo, Feiyue Huang, Rongrong Ji, and Shaozi Li. 2020. Asymmetric Co-Teaching for Unsupervised Cross-Domain Person Re-Identification. In *AAAI 2020*. 12597–12604.

[32] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling. In *COLING 2016*. 3485–3495.