DuerQuiz: A Personalized Question Recommender System for Intelligent Job Interview

Chuan Qin[1,2], Hengshu Zhu[2], Chen Zhu[2], Tong Xu[1,2]
Fuzhen Zhuang[4,5,2], Chao Ma[2], Jingshuai Zhang[2], Hui Xiong[1,2,3]
[1]University of Science and Technology of China, [2]Baidu Talent Intelligence Center, Baidu Inc.
[3]Business Intelligence Lab, Baidu Research, [4] Key Lab of IIP of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, China, [5] University of Chinese Academy of Sciences, China
{chuanqin0426, xionghui}@gmail.com, tongxu@ustc.edu.cn, zhuangfuzhen@ict.ac.cn
{zhuhengshu, zhuchen02, machao13, zhangjingshuai}@baidu.com

ABSTRACT

In talent recruitment, the job interview aims at selecting the right candidates for the right jobs through assessing their skills and experiences in relation to the job positions. While tremendous efforts have been made in improving job interviews, a long-standing challenge is how to design appropriate interview questions for comprehensively assessing the competencies that may be deemed relevant and representative for person-job fit. To this end, in this research, we focus on the development of a personalized question recommender system, namely DuerQuiz, for enhancing the job interview assessment. DuerQuiz is a fully deployed system, in which a knowledge graph of job skills, *Skill-Graph*, has been built for comprehensively modeling the relevant competencies that should be assessed in the job interview. Specifically, we first develop a novel skill entity extraction approach based on a bidirectional Long Short-Term Memory (LSTM) with a Conditional Random Field (CRF) layer (LSTM-CRF) neural network enhanced with adapted gate mechanism. In particular, to improve the reliability of extracted skill entities, we design a label propagation method based on more than 10 billion click-through data from the large-scale Baidu query logs. Furthermore, we discover the hypernym-hyponym relations between skill entities and construct the *Skill-Graph* by leveraging the classifier trained with extensive contextual features. Finally, we design a personalized question recommendation algorithm based on the *Skill-Graph* for improving the efficiency and effectiveness of job interview assessment. Extensive experiments on real-world recruitment data clearly validate the effectiveness of DuerQuiz, which had been deployed for generating written exercises in the 2018 Baidu campus recruitment event and received remarkable performances in terms of efficiency and effectiveness for selecting outstanding talents compared with a traditional non-personalized human-only assessment approach.

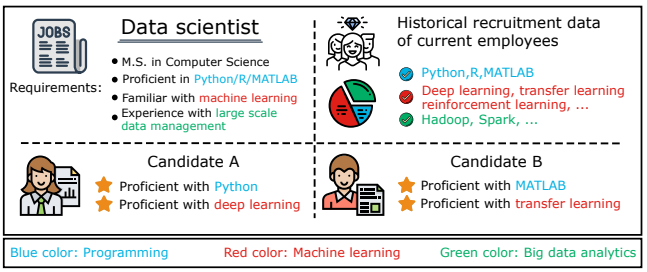

Figure 1: A motivating example.

ACM Reference Format:
Chuan Qin[1,2], Hengshu Zhu[2], Chen Zhu[2], Tong Xu[1,2] Fuzhen Zhuang[4,5,2], Chao Ma[2], Jingshuai Zhang[2], Hui Xiong[1,2,3]. 2019. DuerQuiz: A Personalized Question Recommender System for Intelligent Job Interview. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330706

1 INTRODUCTION

Talent recruitment is critically important for companies to keep their competitive edges, and always has direct impact on the success of a business. To recruit the right talent, the job interview helps to assess the candidates' skills and experiences in relation to the job positions. However, it is a non-trivial task to interview talents in a right way, and bad hiring decisions cost both time and money. For example, as reported in [19], companies nowadays have to pay $4,129 dollars on average for hiring a suitable talent, where the process of job interview usually takes about 24 days. Therefore, in the past decades, substantial efforts have been made on improving the job interview process [3, 20, 23, 31, 38], such as the person-job fit assessment, job skill analysis, interviewer's arrangement, and targeted assessment.

However, a critical challenge for job interview is how to design appropriate questions for comprehensively assessing the competencies that may be deemed relevant and representative for person-job fit. On one hand, if the questions are generally designed with a focus on the basic job requirements, just as the scenario of written test for candidate screening, the job interview will be not discriminative. For example, a candidate applying for *Machine Learning Engineer*, who has rich experiences in *Deep Learning* related applications, might not be properly distinguished from other candidates, if only evaluated with questions about basic algorithmic and coding skills

*Hui Xiong and Hengshu Zhu are corresponding authors.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330706

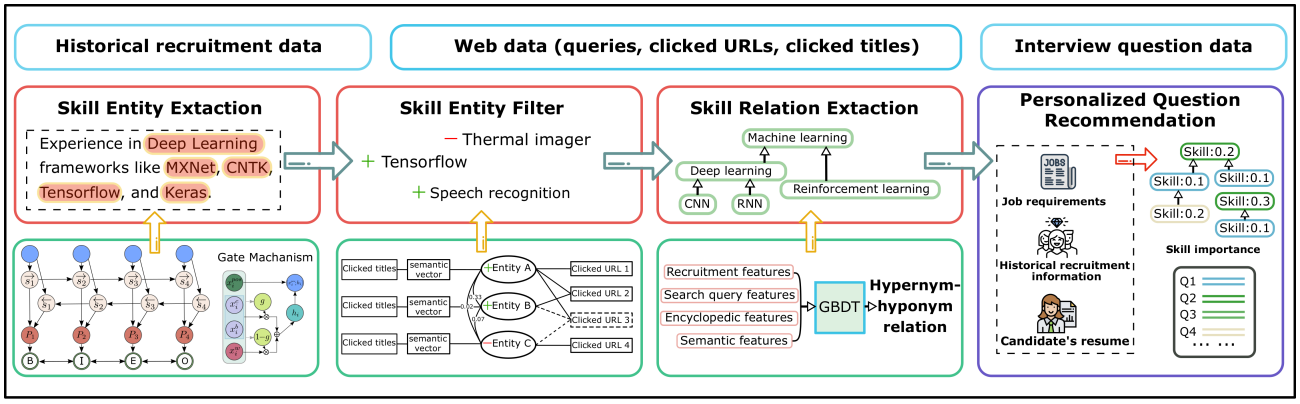

Figure 2: A framework overview of the DuerQuiz system.

instead of experience-specific skills (e.g., the experiences in using deep neural networks), which indeed are very relevant to the job. On the other hand, if too much attention is paid on the questions related to the personal background of candidates, the interview might neglect the essential requirements of jobs and fail to identify the talents who are well-qualified for the position. Therefore, for the design of interview questions, it should strike a balance between the job requirements and the candidates' experiences.

To this end, in this paper, we develop a novel personalized question recommender system, namely DuerQuiz, for enhancing the job interview assessment in talent recruitment. Figure 1 shows a motivating example of our recommender system. As can be seen, in the job description of *Data Scientist*, there are three general requirements, including *programming*, *machine learning* and *big data analytics*, respectively. According to the resumes of two candidates, they have different personal preferences of skills for satisfying corresponding requirements. In other words, candidate A is proficient in *Python* and *Deep Learning*, while candidate B is familiar with *Matlab* and *Transfer Learning*. Therefore, an ideal scenario of DuerQuiz is that it can distinguish the different skill preferences of candidates for the question recommendations. For example, for candidate 1, DuerQuiz will recommend questions about *Python* related programming skills and *Deep Leaning* related machine learning models. Meanwhile, through mining the historical recruitment data of employees who currently hold the position of *Data Scientist*, we realize that *Hadoop* and *Spark* are two important skills for big data analytics. In this case, DuerQuiz will also recommend relevant *Hadoop* and *Spark* questions for both candidates, even if the skills are not listed in their resumes.

Indeed, the key idea of DuerQuiz is to construct a knowledge graph of job skills, *Skill-Graph*, for comprehensively modeling the competencies that should be assessed in the job interview, through mining the abundant historical recruitment data and large-scale job skill data available from the Internet. Specifically, we first develop a novel skill entity extraction approach based on a bidirectional LSTM-CRF neural network with adapted gate mechanism. In particular, to improve the reliability of extracted skill entities, we design a label propagation method based on an *entity-url graph* constructed from more than 10 billion click-through data from the query logs of the Baidu Search Engine. Furthermore, we discover the hypernym-hyponym relations between skill entities and construct the *Skill-Graph* by leveraging the classifier trained with extensive contextual features, such as the recruitment features and search query features. Finally, we propose a personalized question recommendation algorithm based on the *Skill-Graph* for improving the efficiency and effectiveness of job interview assessment. Figure 2 shows the overview of the DuerQuiz system.

Finally, we perform extensive experiments on real-world recruitment data. The results clearly validate the effectiveness of DuerQuiz, which had been deployed for generating written exercises in the 2018 Baidu campus recruitment event and received remarkable performances in terms of efficiency and effectiveness for selecting outstanding talents compared with a traditional non-personalized human-only assessment approach.

2 THE DUERQUIZ FRAMEWORK

In this section, we will introduce the framework of our DuerQuiz system in detail. As shown in Figure 2, our framework mainly consists of three components for *Skill-Graph* construction (i.e., *Skill Entity Extraction*, *Skill Entity Filter and Skill Relation Extraction*), and one component for the *Personalized Question Recommendation*.

2.1 Skill Entity Extraction

To construct our *Skill-Graph*, we first extract skill entities from the recruitment data, i.e., job requirements in job postings and work/project experiences in candidates' resumes. For example, we want to extract skills *Deep Learning* and *PaddlePaddle* in the job requirement sentence "Experience in Deep Learning frameworks like PaddlePaddle". Here, we follow the state-of-the-art name entity recognition model, namely LSTM-CRF [11], to extract the skill entities. Moreover, character-based LSTM-CRF model with word information has shown better performance than word-based model [16, 35] on the languages without explicit word separators, such as Chinese and Japanese. Therefore, we use character-based LSTM-CRF as the main structure with considering word information. Additionally, following the findings in [33], we also involve the character bigram information as inputs for better character representation.

Formally, given an input sentence X, i.e., a job requirement in the job posting or a work/project experience in the candidate's resume, we consider all of the above three elements (i.e., character, word and character bigram) for skill entity extraction. Here, we denote the character sequence of X by $\{c_1, c_2, ..., c_n\}$, where c_i is

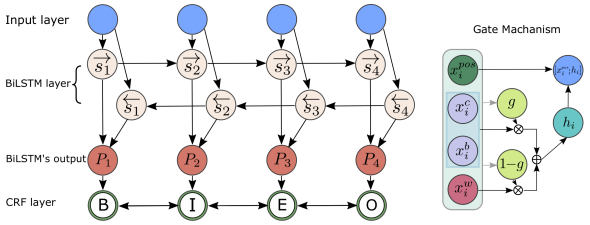

Figure 3: The skill entity extraction model.

the i-th character in X. And the character bigram sequence is denoted as $\{b_1, b_2, ..., b_n\}$, where $b_i = c_i c_{i+1}$, and c_{n+1} is the padding word that represents the *ending of the sentence*. For collecting word information, we first use a Chinese segmentor to split X into m words, i.e., $\{w_1, w_2, ..., w_m\}$. As the same time, in order to align the length of word sequence with the character sequence, for each word w_i we repeat it l_{w_i} times, where l_{w_i} denotes the character length of w_i. For simplicity, we denote the aligned word sequence as $\{w'_1, w'_2, ..., w'_n\}$.

The proposed character-based model is shown in Figure 3. Specifically, we first embed characters and words to vectors, respectively.

$$x_i^c = W_c c_i, \quad x_i^b = W_b b_i, \quad x_i^w = W_w w'_i, \quad i \in [1, n], \tag{1}$$

where W_c, W_b, W_w are initialized by pre-trained character, character bigram and word vector matrices based on Baidu Baike data. Here, we concatenate the representations of character and character bigram, and use a fully connected layer to reduce the dimension to be the same as x_i^w. Then, we use a gate mechanism to dynamically combine the semantic character-level and word-level representations, that is,

$$x_i^s = W_m[x_i^c; x_i^b] + b_m, \quad g_i = \sigma(W_g[x_i^s; x_i^w; x_i^{pos}] + b_g), \tag{2}$$

where x_i^{pos} denotes the part-of-speech (POS) tag of word w'_i, σ is the sigmoid function and W_m, W_g, b_m, b_g are the parameters to be learned during the training processing. Then, the combined semantic representation is calculated by

$$h_i = g_i \odot x_i^s + (1 - g_i) \odot x_i^w, \tag{3}$$

where $\odot$ denotes the element-wise product. Now, we use the concatenation of the hidden vector h_i and POS tag x_i^{pos} as the input of a bidirectional LSTM (BiLSTM) layer, which can be formulated as:

$$s_i = BiLSTM([x_i^{pos}; h_i]). \tag{4}$$

Finally, we take the hidden states $s = \{s_1, s_2, ..., s_n\}$ as the input to a standard CRF layer, and it outputs the final prediction label sequence $y = \{y_1, y_2, ..., y_n\}$, where $y_i \in \{I, O, B, E, S\}$ indicates current character is *Inside*, *Outside*, *Beginning*, *Ending* or *Singleton* of the skill entity. So the conditional probability for sequence y is:

$$\begin{aligned} p(y|s) &= \frac{exp(score(s, y))}{\sum_{y' \in \mathrm{Y}} exp(score(s, y'))}, \\ P_i &= W_o s_i + b_o, \quad score(s, y) = \sum_{i=1}^{n} P_{i, y_i} + \sum_{i=0}^{n} A_{y_i, y_{i+1}}, \end{aligned} \tag{5}$$

where Y denotes all the possible label sequences for s, P_{i,y_i} is the score of assigning the i-th word with label y_i, $A_{y_i, y_{i+1}}$ indicates the transition score from label y_i to y_{i+1}, and W_o, b_o are trainable parameters. During training, we maximize the log-probability of the correct tag sequence:

$$log(p(y|s)) = score(s, y) - log(\sum_{y' \in \mathrm{Y}} exp(score(s, y'))). \tag{6}$$

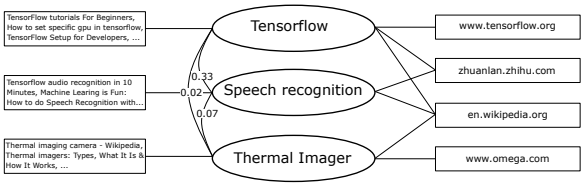

Figure 4: An intuitive example of filtering skill entities by using search query logs.

2.2 Skill Entity Filter

After extracting the skill entities from the recruitment data, we get a set of skills, which are denoted as $\mathcal{V}_e = \{v_1^e, v_2^e, .., v_{p_e}^e\}$. Although the proposed skill extraction model performs well, it is also inevitable to contain some non-skill words in $\mathcal{V}_e$. To handle this problem, we propose to leverage the web search data, i.e., the click-through logs (queries, clicked URLs and titles), as the extra knowledge data for filtering the entities, since web search engines have been providing ever-richer experiences around entities. Inspired by *Li et al.* [15], we propose to label a small set of $\mathcal{V}_e$, i.e., whether each entity is a real skill or not, and design a novel label propagation (LP) based algorithm to iteratively propagate information in a special entity-url graph.

Specifically, we first create an entity-url graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of nodes $\mathcal{V}$ contains two kinds of nodes, i.e., entities $\mathcal{V}_e$ and clicked URLs (or clustered URLs) $\mathcal{V}_u = \{v_1^u, v_2^u, .., v_{p_u}^u\}$ recorded in the query log containing those entities. While the set of edges $\mathcal{E}$ also includes two parts, i.e., $\mathcal{E}_{eu}$ and $\mathcal{E}_{ee}$. Specifically, $\mathcal{E}_{eu} \subset \mathcal{V}_e \times \mathcal{V}_u$ is the set of links between the nodes in $\mathcal{V}_e$ and their corresponding clicked URLs in $\mathcal{V}_u$. In particular, we remove the edges of the URLs that are connected both skill entities and non-skill entities in labeled data, in order to reduce the noise of propagating information between entity nodes and URL nodes. Let $\mathcal{W}_{eu} \in \mathbb{R}^{p_e \times p_u}$ denotes the weight matrix, where each element $w_{i,j}^{eu}$ equals the frequency associated with v_i^e and v_j^u. And, since there are some nodes which do not connect any node in $\mathcal{V}_u$, we define a set of edges $\mathcal{E}_{ee} \subset \mathcal{V}_e \times \mathcal{V}_e$ between the entity nodes $\mathcal{V}_e$ for propagate the information of those nodes. Here, we first generate the topic vector t_i for each entity node v_i^e by training a topic model on the clicked URLs' titles. Then we can get a Gaussian kernel matrix S, where each element $s_{i,j} = exp\{-||t_i - t_j||^2 / 2\sigma^2\}$. Finally, we only create the edges between each entity and its k^e closest nodes. The corresponding edge set is $\mathcal{E}_{ee}$. We denote weight matrix $\mathcal{W}_{ee} \in \mathbb{R}^{p_e \times p_e}$, where each element $w_{i,j}^{ee} = s_{i,j}$ if there exists a edge between v_i^e and v_j^e. The Algorithm 1 shows the detail of constructing $\mathcal{E}_{ee}$.

Afterwards, we learn the probability of whether an entity is a skill through a LP method, which is shown in Algorithm 2. Here we denote the $\mathcal{Y} \in \mathbb{R}^{p_e \times 2}$ as the entity label. Specifically, we set $y_{i,1}^0 = 1$ when the entity v_i^e is labeled as a skill word and $y_{i,0}^0 = 1$ if it is labeled as a non-skill word. Then, we calculate two normalized weight matrices as follows.

$$\mathcal{N}_{eu} = \mathcal{D}_{eu}^{-1/2} \mathcal{W}_{eu}, \quad \mathcal{N}_{ee} = \mathcal{D}_{ee}^{-1/2} \mathcal{W}_{ee} \mathcal{D}_{ee}^{-1/2}, \tag{7}$$

where $\mathcal{D}_{eu}$ and $\mathcal{D}_{ee}$ are two diagonal matrices, in which each element $d_{i,i}^{eu} = \sum_{j=1}^{p_e} (\mathcal{W}_{eu} \mathcal{W}_{eu}^{\top})_{i,j}$, and $d_{i,i}^{ee} = \sum_{j=1}^{p_e} w_{i,j}^{eu}$. Then, we use LP iteratively to update the $\mathcal{Y}_{eu}, \mathcal{Y}_{ee} \in \mathbb{R}^{p_e \times 2}$, which denote

Algorithm 1 $\mathcal{E}_{ee}$ Construction.

Input: The set of entity nodes $\mathcal{V}_e$, the corresponding clicked URLs' titles $X_i^t = \{x_1, x_2, ...\}$ for each entity v_i^e, hyperparameters k^t, k^e, k^g, n^t, σ;
Output: The set of edges $\mathcal{E}_{ee}$ and corresponding weight $\mathcal{W}_{ee}$;
1: Train a Latent Dirichlet Allocation (LDA) topic model $\mathcal{M}^t$ on the clicked URLs' titles, where topic number equals n^t;
2: Select k^t clicked URLs' titles with most clicks for each entity v_i^e and use $\mathcal{M}^t$ to generate its topic vector t_i;
3: Calculate the Gaussian kernel matrix $S \in \mathbb{R}^{p_e \times p_e}$;
4: For each entity v_i^e, find other k^e nodes closest to it by S and satisfy their cosine similarity is greater than k^g. We establish edges between v_i^e and those nodes, and their weights equal corresponding cosine similarity values.

Algorithm 2 The LP Algorithm.

Input: Search query log L, the set of candidate entities $\mathcal{V}_e$, a set of labeled entities $\overline{\mathcal{V}_e}$, hyperparameters α_{eu}, α_{ee}, β;
Output: The probability $p(v_i^e)$ of each entity is a skill;
1: Build graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, compute the weight matrices $\mathcal{W}_{eu}$, $\mathcal{W}_{ee}$ and initialize $\mathcal{Y}^0$ according to $\overline{\mathcal{V}_e}$;
2: Compute $\mathcal{N}_{eu}$ and $\mathcal{N}_{ee}$ by Equation 7;
3: **while** Not converged **do**
4: Compute $\mathcal{Y}_{eu}^t$, $\mathcal{Y}_{ee}^t$ by Equation 8;
5: **end while**
6: For each $v_i^e \in \mathcal{V}_e - \overline{\mathcal{V}_e}$, compute $p(v_i^e)$ by Equation 9.

the score of entity label learned by $\mathcal{W}_{eu}$ and $\mathcal{W}_{ee}$, respectively. And for the t-iteration, we have:

$$\begin{aligned} \mathcal{Y}_{eu}^t &= \alpha_{eu} \mathcal{N}_{eu} \mathcal{N}_{eu}^\top \mathcal{Y}_{eu}^{t-1} + (1 - \alpha_{eu}) \mathcal{Y}^0, \\ \mathcal{Y}_{ee}^t &= \alpha_{ee} \mathcal{N}_{ee} \mathcal{Y}_{ee}^{t-1} + (1 - \alpha_{ee}) \mathcal{Y}^0, \end{aligned} \tag{8}$$

where α_{eu}, α_{ee} are the parameters. Finally, after $\mathcal{Y}_{eu}^t$, $\mathcal{Y}_{ee}^t$ converging, we can calculate the probability of whether entity v_i^e is a skill by calculating the weight sum of their normalized values, that is

$$p(v_i^e) = \beta \frac{y_{i,0}^{eu}}{y_{i,0}^{eu} + y_{i,1}^{eu}} + (1 - \beta) \frac{y_{i,0}^{ee}}{y_{i,0}^{ee} + y_{i,1}^{ee}}, \tag{9}$$

where $y_{i,j}^{eu}$ and $y_{i,j}^{ee}$, $j \in \{0, 1\}$ denote i-th element of $\mathcal{Y}_{eu}$ and $\mathcal{Y}_{ee}$.

2.3 Skill Relation Extraction

After the process of the skill entity extraction and filtration, we turn to extract the relations, specifically hypernym-hyponym ("is-a") relations, between the skills. For example, the skills *Machine Learning* and *Reinforcement Learning* have a hypernym-hyponym relation. This problem can be formulated as a classification problem, which is to determine whether a skill pair (v_i^e, v_j^e) is a hypernym-hyponym relation. Here we leverage historical recruitment data, click-through log of the web search data, and encyclopedia data to cook several effective features.

To generate the training data, we follow the idea from *Fu et al.* [7], to collect candidate hypernyms for each skill entity. Specifically, we select the co-occurrence skills appearing in historical successful applications (i.e., one skill in a given job posting and another in the corresponding resume of a candidate) and click-through data (i.e., one skill in the search query and another in a clicked URL's title) as the candidate hypernyms words. And, we propose to manually label hypernym-hyponym relation of a subset of the skill entities with their candidate hypernyms as the training data.

The features for training the classification model can be divided into four categories, which are illustrated in the following.

• **Recruitment Features:** Normally, the skills in the job postings are often as hypernyms of some skills in the work/project experiences in the resumes. For example, the skill *Machine Learning* appearing in the job requirement is the hypernym of the *SVM* and *LDA*, which appear in the resumes of a corresponding successfully-enrolled candidate. Meanwhile, the skill pairs appear in the same job requirement or work/project experience will also reflect the relation between them.

• **Search Query Features:** The click-through log can also help us understand the relationships between skills. A skill that appears in the clicked URL's title will have a strong relationship with the retrieved skill word. Moreover, lots of search queries and URLs' titles contain multiple skills, which reflect some of the collaborative relationships between them. Therefore we also involve the frequency of the co-occurrence in queries or titles as features.

• **Encyclopedic Features:** The encyclopedic data contain a large amount of knowledge about entity relations. The skills that appear in the *summary* of another skill's Baidu Baike page might have the hypernym-hyponym relation with that skill. So, we use the features to indicate such co-occurrence of skills, which are extracted from the Baidu Baike.

• **Semantic Features:** Recently, the word semantic representation like word embedding has shown the effectiveness of detecting hypernym-hyponym relation [6, 32]. Therefore, we also involve this kind of features. Specifically, we first get the semantic vectors of skills by using a pre-trained word2vec model. Please note that if a skill does not appear in the word2vec vocabulary, we use the average of semantic vectors of the corresponding segment words. Then for each skill pair, we concatenate the semantic vectors of the corresponding two skills in it and the difference between them together as its semantic features.

Here we use Gradient Boosting Decision Tree (GBDT) as the classifier. After we predict all the hypernyms of the skill words, we can construct a skill-graph by using all the hypernym-hyponym relations as the directed edges in the graph. And we further remove the weakened edges to avoid forming the directed rings. It is similar to the construction method of *Fu et al.* [6], while the difference is that we use the probability of each hypernym-hyponym relation learned by classifier as the corresponding edge weight. Finally, we remove all indirect edges (i.e., the edge from A to B, if there exists a direct path from A to B), and denote the graph as $\mathcal{G}_r = \{\mathcal{V}_r, \mathcal{E}_r\}$.

2.4 Personalized Question Recommendation

Now we introduce how to leverage the *Skill-Graph* $\mathcal{G}_r$ for interview question recommendation. We first collect a set of interview questions and manually link them with skills in $\mathcal{G}_r$. Then for an application (i.e. given a pair of candidate's resume and job posting), we leverage all of the candidate's resume, the job posting, and historical recruitment data to find the suitable skills, which should be evaluated in the interview, as well as their weights. Finally, we use these skills as a bridge for generating a set of interview questions for this job application. We hope the interview questions generated by our recommendation algorithm can not only cover

the skill requirements of the job, but also take into account the candidate's personal skill preference. Thus the key point here is how to automatically weight the importance of skills, which are extracted from different sources (i.e., the candidate's resume, the job posting, or historical recruitment data). Next, we will introduce our recommender in detail.

Specifically, given a job posting J, we denote the current employees' resumes as $\mathcal{R} = \{R_1, R_2, ..., R_p\}$. In addition, we also propose to use their work performances, since those employees with relevant practical work-related skills are usually able to adapt to work more quickly and get good job performance. Here, we denote the job performance as $\mathcal{P} = \{P_1, P_2, ..., P_p\}$, where the performance of each employee P_i. The candidate's resume is donated as S. We first get the skills contained in the above textual data according to the skill graph $\mathcal{G}_r$, and denote $\mathcal{V}_J$, $\mathcal{V}_{R_i}$, $\mathcal{V}_R$ and $\mathcal{V}_S$ as the skill sets of J, R_i, $\mathcal{R}$ and S respectively. And we count the number of skill v_k appearing in R_i and S as $C_k^{R_i}$ and C_k^S, respectively. We also denote $descendant(v_k)$ as the set of all the descendant nodes of skill v_k in $\mathcal{G}_r$ and $ancestor(v_k)$ as the set of its all ancestor nodes. Afterwards, we can measure the weight of each skill v_k in $\mathcal{G}_r$ from historical recruitment information, i.e., W_k^h, and the candidate's resume, i.e., W_k^p, by:

$$\begin{aligned} W_k^h &= \frac{\sum_{m\in[1,p]} P_m(\sum_{v_n\in descendant(v_k)} C_n^{R_m} + C_k^{R_m})}{\sum_{m\in[1,p]} P_m \sum_{v_n\in\mathcal{V}_{R_m}} C_n^{R_m}}, \\ W_k^p &= \frac{\sum_{v_n\in descendant(v_k)} C_n^S + C_k^S}{\sum_{v_n\in\mathcal{V}_S} C_n^S}. \end{aligned} \tag{10}$$

Meanwhile, we define the major skill set $\mathcal{V}'_J$ as the set of all root skills of skills in $\mathcal{V}_J$, namely $\mathcal{V}'_J = \{v_n | v_n \in \bigcup^{v_k\in\mathcal{V}_J} ancestor(v_k), ancestor(v_n) = \phi\}$. Similarly, we also define the major skill set $\mathcal{V}'_{R_i}$, $\mathcal{V}'_R$ and $\mathcal{V}'_S$ for $\mathcal{V}_{R_i}$, $\mathcal{V}_R$ and $\mathcal{V}_S$ respectively.

Then we divide all of skills in $\mathcal{V}'_J$, $\mathcal{V}'_R$, $\mathcal{V}'_S$ into three parts, namely matching skills, personalized skills and unmatched skills, and calculate their weights respectively. In addition, to handle the cold-start problem, we also consider the skills that just appear in the job posting but do not appear in historical recruitment data or candidate's resume. The mathematical definition of skill $v_k \in \mathcal{V}'_J \cup \mathcal{V}'_R \cup \mathcal{V}'_S$ of weight for skill is as follows.

$$\begin{aligned} W_k^f &= \alpha_f W_k'^h \mathbf{1}_{\mathcal{V}'_R\cap\mathcal{V}'_S}(v_k) + (1-\alpha_f) W_k'^p \mathbf{1}_{(\mathcal{V}'_R\cup\mathcal{V}'_J)\cap\mathcal{V}'_S}(v_k) \\ &+ \alpha_f W_k'^h g(v_k) \mathbf{1}_{\mathcal{V}'_R - \mathcal{V}'_S}(v_k) + \beta_f \mathbf{1}_{\mathcal{V}'_J - (\mathcal{V}'_R\cup\mathcal{V}'_S)}(v_k), \end{aligned} \tag{11}$$

where $g(v_k) = 1$ if $min\{p(v_k|v_m), v_m \in \mathcal{V}'_R \cap \mathcal{V}'_S\} > \gamma_f$ else 0, $W_k'^h = \frac{W_k^h}{\sum_{v_m\in\mathcal{V}'_R} W_m^h}$, $W_k'^p = \frac{W_k^p}{\sum_{v_m\in\mathcal{V}'_S} W_m^p}$, $\mathbf{1}_A(v_k)$ denotes the indicator function of an element v_k in a subset A and $p(v_k|v_m) = \frac{\sum_i \mathbf{1}_{\mathcal{V}'_{R_i}}(v_m)\mathbf{1}_{\mathcal{V}'_{R_i}}(v_k)}{\sum_i \mathbf{1}_{\mathcal{V}'_{R_i}}(v_m)}$. Here we use function $g(\cdot)$ to remove the skills that the candidate does not have and can be replaced by other acquired skills. For example, assume that the historical recruitment data have shown that the conditional probabilities of skill *C++* and *PHP* are low. If the candidate is expertise in *C++*, we would ignore the weight of *PHP* in the historical data.

Algorithm 3 Personalized Interview Question Recommendation.

Input: Skill graph $\mathcal{G}_r$, job posting J, current employee resumes $\mathcal{R}$, current employees' job performance $\mathcal{P}$, candidate's resume S, interview questions $\mathcal{Q}$, hyperparameters α_f, α_c;
Output: Recommend interview questions O;

1: Generate skills set $\mathcal{V}_J$, $\mathcal{V}_{R_i}$ and $\mathcal{V}_S$, calculate the corresponding frequency $C_k^{R_i}$ and C_k^S for each skill v_k, $i \in [1, p]$;
2: Calculate W_k^h, W_k^p by Equation 10. And collect major skills sets $\mathcal{V}'_J$,$\mathcal{V}'_R$ and $\mathcal{V}'_S$;
3: Calculate W_k^f for each skills $v_k \in \mathcal{V}'_J \cup \mathcal{V}'_R \cup \mathcal{V}'_S$ by Equation 11, and normalize it as $W_k'^f$;
4: Calculate all children of each major skill v_k by Equation 12;
5: Update v_k weight by $W_k'^f \leftarrow W_k'^f - \sum_{parent(v_{k'})=v_k} W_k'^f$;
6: Iterate 4-5 to collect the weight of all descendants of v_k;
7: Count the number of question from all the nodes in $\mathcal{V}_J$, $\mathcal{V}_R$ and $\mathcal{V}_S$, if skill v_k has no candidate questions, then we add its weight into its parent node. Recursively calculate the number of questions recommended for all the skills and generate the recommend interview questions O.

Then, we normalize W_k^f as $W_k'^f$ by $\frac{W_k^f}{\sum_{v_m\in\mathcal{V}'_J\cup\mathcal{V}'_R\cup\mathcal{V}'_S} W_m^f}$. And, for each skill $v_k \in \mathcal{V}'_R \cup \mathcal{V}'_S$, the weight of its child $v_{k'}$ is defined as:

$$W_{k'}'^f = W_k'^f(\alpha_c g'(v_k)\frac{W_{k'}^h}{W_k^h} + (1-\alpha_c)\frac{W_{k'}^p}{W_k^p}), \tag{12}$$

where $g'(v_k) = 1$ if $min\{p(v_k|v_m), v_m \in Sibling(v_k)\} > \gamma_c$ else 0 and $Sibling(v_k)$ is the set of sibling nodes of v_k whose $W_k^p = 0$. We set $p(v_k|v_m) = \frac{\sum_i \mathbf{1}_Q(v_m)\mathbf{1}_Q(v_k)}{\sum_i \mathbf{1}_Q(v_m)}$, where $Q = \bigcup^{v_n\in\mathcal{V}_{R_i}} ancestor(v_n)$. Then, we update $W_k'^f$ by $W_k'^f - \sum_{parent(v_{k'})=v_k} W_k'^f$. Therefore, we can iterate through the weights of all leaf nodes under each root node v_k. On the other side, for $v_k \in \mathcal{V}'_J - (\mathcal{V}'_S \cup \mathcal{V}'_R)$, we set weights of all the skills in $descendant(v_k) \cap \mathcal{V}'_J$ as $\frac{\beta_f}{|descendant(v_k)\cap\mathcal{V}'_J|}$.

Now given a new job application, we have got the skills, which are needed to be tested, and their corresponding weights. Next, we try to recommend a set of suitable interview questions by these skills. For facilitating explanation, we set the size of recommended question set as N. Please note that, in order to expand the question data of the skills, given an interview question q_i and its corresponding skill v_j in $\mathcal{G}_r$, we also label q_i as v_k, if v_j is the ancestor node of v_k and v_k appears in the question text of q_i.

When generating the interview question set, for each skill v_k in $\mathcal{V}_J$, $\mathcal{V}_R$, or $\mathcal{V}_S$, we generate $W_k'^f \cdot N$ pieces of questions for this skill. And if there is no question linked to v_k, we add its weight into its parent node and further generate questions for the parent.

Recursively, we can recommend a set of interview questions covering all the skills needed to be tested. The overall recommendation algorithm is shown in Algorithm 3.

3 EXPERIMENTAL RESULT

In this section, we will introduce the experimental results, which clearly demonstrate the effectiveness of each of our technical components. The real-world dataset is provided by one of the largest

Table 1: The statistics of the job application dataset.

Sentence Category	All	Labeled	Training	Validation	Testing
job requirement	25K	2K	1.6K	0.2K	0.2K
candidate experience	104.6K	3.7K	2.9K	0.4K	0.4K

high-tech company in China, which contains historical recruitment data, interview questions data and search query log data.

3.1 The Performance of Skill Entity Extraction

Data Description. Here, we used a historical recruitment dataset for skill entity extraction, which consists of 3,605 job postings (25,034 requirement sentences) and 17,931 resumes of successful job applications (104,589 experience sentences). For training our model, we manually labeled the skill entities in 2,000 requirement sentences and 3,700 experience sentences. The statistics of the historical recruitment dataset are summarized in Table 1.

Experimental Setup. Here we introduce the detailed settings of our experiments. We pre-trained the character, character bigram and word embeddings from Baidu Baike textual data by using Skip-gram Model. Specifically, the dimension of the embedding vector was set to 300. And, in our skill entity extraction model, the dimension of the hidden state in BiLSTM was set to 300. Then, we also followed the idea in [8] to initialize all matrices and vector parameters. All the models were optimized by using Adam algorithm. Finally, to validate the performance of our model, we selected several model as the baseline methods, including (1) character baseline, namely character-based BiLSTM CRF, which directly uses the character as the input of BiLSTM (2) character baseline with word information which was introduced in [16] and (3) character baseline with word and character bigram information.

Results. The overall performances on both job postings and resumes dataset are shown in Table 2. According to the result, clearly, we observe that our model outperforms all the baselines. Especially, adding character bigram and word representation increases the performance. And, as our model performs better than the character baseline that both consider *word* and *character bigram* features, it demonstrates the effectiveness of our gate mechanism. After we trained two skill entity extraction model on the labeled data, we used them to predict the skill entities on the unlabeled data. With removing the low frequency words, we finally got 5,976 skill entities from the historical recruitment data.

3.2 The Performance of Skill Entity Filter

Data Description. Here, we totally collected about 10 billion click-through data ranging from January to June 2018, where each search query contains the above-mentioned candidate entities. After removing the noise data, we segmented the query and matched the n-gram terms to the candidate entities. Finally, we got 374 million entity-url-title triples. For evaluating our LP algorithm, we manually labeled 1,416 skill entities and 502 non-skill entities. And, we randomly selected 60% of the labeled data as the training set, another 10% for tuning the parameters, and the last 30% as test data to validate the performance.

Experimental Setup. In our experiment, we set the k^t as 20 to select the clicked URLs' titles with most clicks for each entity. And we set the topic number n^t as 100 to train the LDA model $\mathcal{M}^t$ and set α as 0.5 to calculate the Gaussian kernel matrix S. Then we set k^e as 20, k^g as 0.7 to construct the edge in $\mathcal{E}_{ee}$ between each two entity nodes. The hyperparameters α_{eu}, α_{ee} and β in our LP algorithm were set to 0.2, 0.05 and 0.3 respectively.

Table 2: The performance of Skill Entity Extraction.

Dataset	Methods	Precision	Recall	F1
Job posting	character baseline	0.8371	0.7543	0.7935
	+ word	0.8443	0.7976	0.8203
	+ word+character bigram	0.8628	0.7996	0.8300
	Our model	**0.8759**	**0.8016**	**0.8371**
Resume	character baseline	0.6370	0.5883	0.6117
	+ word	0.7099	0.6536	0.6806
	+ word+character bigram	0.7039	0.6993	0.7016
	Our model	**0.7181**	**0.7183**	**0.7182**

Table 3: The performance of Skill Entity Filter.

Methods	Precision	Recall	F1
Decision Tree	0.8354	0.9326	0.8813
Random Forests	0.8045	**0.9953**	0.8898
Gradient Boosting Decision Tree	0.8653	0.9558	0.9082
Our model (w/o information in $\mathcal{E}_{ee}$)	**0.9701**	0.7535	0.8481
Our model	0.9406	0.9581	**0.9493**

To demonstrate the effectiveness of our LP algorithm, we constructed several baseline models by using classic supervised models including *Decision Tree*, *Random Forests* and *Gradient Boosting Decision Tree*. Specifically, we used the number of clicks for different URLs as the input features for classifiers. And, we also used a variant of our model as a baseline, where we did not use the information in $\mathcal{E}_{ee}$, that is we changed Equation 9 as $p(v_i^e) = \frac{y_{i,0}^{eu}}{y_{i,0}^{eu}+y_{i,1}^{eu}}$.

Results. The performances of our model and baselines are shown in Table 3. Here, we find that our model without the information in $\mathcal{E}_{ee}$ has achieved the highest precision rate of 0.97, which demonstrates the effectiveness of using the LP algorithm to filter the skill entities. Indeed, in the variant of our model, the lack of $\mathcal{E}_{ee}$ leads to 20% entity nodes in the test set do not connect any URL and cannot be predicted by it. Thus although it has high precision but the recall is just 0.75. And, with leveraging $\mathcal{E}_{ee}$, our LP algorithm can predict all the entities label and have achieved the best performance compared with all the baseline models. Although the random forest can achieve the best recall rate, its precision value is not competitive. Finally, by our LP algorithm, we totally got 4,836 skill entities.

3.3 The Performance of Skill Relation Extraction

Experimental Setup. We first selected 100 skills and manually labeled the hypernym-hyponym relation between them and all of 4,836 skills. And, as described in Section 2.3, we collected the candidate hypernyms words for those 100 skills. Specifically, we only selected the top 20 co-occurrence skills appeared in historical successful job applications and click-through data, respectively. And, we found that our candidate hypernyms can cover 86.04% hypernyms of labeled 100 skills. Then, we further manually labeled the hypernym-hyponym relation of another 264 skills with their candidate hypernyms. Finally, we collected a data set of 364 skills with 24.01 candidate words and 1.23 hypernyms on average, that is

Table 4: The performance of Skill Relation Extraction.

Methods	Precision	Recall	F1
Our model	**0.8228**	**0.6250**	**0.7104**
- w/o recruitment features	0.7564	0.5673	0.6484
- w/o search query features	0.7949	0.5962	0.6813
- w/o baike features	0.8133	0.5865	0.6816
- w/o semantic features	0.7805	0.6154	0.6882

447 positive instances and 8,292 negative instances. We randomly selected 170 skills of the labeled data as the training set, 44 for tuning the parameters, and 100 skills for evaluating the performance.

Results. The results are shown in Table 4. Clearly, when we remove any one of the features, the performance is degraded more or less, which demonstrates the effectiveness of each kind of features. And the most effective features are recruitment features. Moreover, we also used the pattern-based method of *Hearst et al.* [9] on our skills. And, we used the Chinese Hearst-style lexical patterns that proposed by *Fu et al.* [7]. However, only 10.73% hypernyms can be extracted. It also indicates the value of our solution.

3.4 The Performance of Question Recommendation

Here we evaluate our recommender system on a real-world job interview scenario. Specifically, we deployed our system to the 2018 Baidu campus recruitment[1]. For four selected major categories of job posting (i.e., Machine Learning/NLP Engineer, C++/PHP R&D Engineer, Java R&D Engineer, and Mobile Software R&D Engineer) in the campus recruitment, we invited the candidates to attend our online written exercise, which was automatically generated by our DuerQuiz system. To validate the performance of our system, we collected their final recruitment results and their performances in the interview, including our intelligent written exercise, traditional written exercise, and on-site interview. In particular, the traditional written exercise is a standard non-personalized method to assess the basic knowledge background related to the job requirements. The detailed statistics are shown in Table 5.

Here we used Spearman's rank correlation coefficient to measure the correlation between candidates' different written exercise results and their recruitment results. Specifically, we first mapped the final interview score into $[0, 8]$. And, the final recruitment results are of three types, *failed*, *normal offer* and *special offer* are corresponding to score 0, 1 and 2. The results of the correlation coefficient analysis are shown in Table 6. According to the result, clearly, we find that both our DuerQuiz framework and traditional written exercise are significantly correlated with the interview results and the recruitment results for most of the jobs. And our intelligent written exercise has a larger correlation than the traditional exam, which indicates that DuerQuiz is more discriminative for selecting candidates during interview assessment. Especially, to further validate the effectiveness of DuerQuiz, we also conducted a user survey after the written exercise, regarding the relevance and coverage of the recommended questions. According to the user feedbacks, DuerQuiz can achieve satisfied user experiences, with average 3.89/5 and 3.80/5 ratings in relevance and coverage, respectively. Therefore, we believe that our framework can assess candidates abilities effectively, and help recruiters to select candidates to participate in the follow-up on-site interview.

[1] https://talent.baidu.com/external/baidu/campus.html

Table 5: The statistics of collected data.

Job category	Candidates	Normal offer	Special offer
Machine Learning / NLP Engineer (ML/NLP)	115	27	3
C++/PHP R&D Engineer (C++/PHP)	163	34	4
Java R&D Engineer (JAVA)	90	15	1
Mobile Software R&D Engineer (Mobile)	37	6	4

Table 6: The Spearman correlation result.

Job category	Metric	Interview score		Recruitment result	
		DuerQuiz	Traditional	DuerQuiz	Traditional
ML/NLP	Spearman correlation	0.2831	0.2477	0.1877	0.1952
	p-value	2.169e-3	7.599e-3	0.04451	0.03651
C++/PHP	Spearman correlation	0.2305	0.2477	0.2551	0.1879
	p-value	1.666e-4	3.074e-3	1.012e-3	0.01628
JAVA	Spearman correlation	0.1974	0.2807	0.2073	0.1430
	p-value	6.207e-2	7.358e-3	0.0499	0.1787
Mobile	Spearman correlation	0.5591	0.4844	0.5261	0.4577
	p-value	3.208e-4	0.02376	8.242e-4	4.386e-3
Total	Spearman correlation	0.2951	0.2691	0.2478	0.1868
	p-value	1.386e-9	3.761e-8	4.369e-6	1.561e-4

4 CASE STUDY AND DISCUSSION

To further illustrate the effectiveness of our DuerQuiz system, Table 7 shows an example of top 4 questions recommended by our system and a start-of-the-art approach JLMIA [23] which recommend the interview question by using a pre-trained jointly latent variable model on job postings, candidate resumes and interview assessments from the successful job interview records.

Specifically, we find that the questions recommended by DuerQuiz involve the skills which are both mentioned in job posting and candidate's resume (Q1 is about *Recommendation System*, Q2 is about the *Clustering->Machine Learning*, Q3 is about the *Python* and Q4 is about *RNN*, where "->" denotes the hyponym-hypernym relation) . Specifically, *Recommendation System* and *Machine Learning* are directly mentioned in job posting and candidate resume. Meanwhile *Tensorflow->Python*,*Tensorflow->Machine Learning*, *Seq2Seq->RNN*, are related to the skills in the resume and also be mentioned many times among the successful candidates' resume in the historical recruitment data.

Meanwhile, since JLMIA model cannot generate questions by jointly considering both job posting and candidate's resume together, here we generate the top 4 questions for them, respectively. Obviously, their recommendation results cannot take into account the historical information from the successful job interview records and the candidate's personalized skills at the same time. For example, JLMIA recommended two questions about *C++* and *Python* for matching the job posting requirement. However, since it could not capture the fact that the candidate has experiences in *Python*, it recommends *C++* and *Python* at the same time, which is not appropriate for this candidate. Similarly, when using resumes as input, JLMIA did not assess the skill *Recommender System* which not only occurs in the resume but also highlighted in the job.

In addition, although both DuerQuiz and JLMIA aim to leverage texture information to find suitable interview questions for a given candidate's resume and a job posting. JLMIA tries to bridge the gap by latent representations got by co-occurrence. Our framework explicitly extracts a set of skills by a well-defined skill-graph, where the meaning of each skill is intelligible, and further provides better

Table 7: The case study of question recommendation.

The given job posting	Familiar with **data mining**, **machine learning**, **natural language processing**. Familiar with **C++/python/Java** programming, have a deep understanding of **algorithm design**. Experience with **recommender system** is preferred.
The given candidate's resume	Machine translation system: Implementing a crawler program for bilingual data source collection, extracting features, and training the **Seq2Seq** translation model using **Tensorflow**... English article **recommender system**: Extracting features for massive users, using the **clustering** algorithms to do user group positioning, using **machine learning** related algorithms to recommend English articles to users ...
Questions recommended by DuerQuiz	Q1. Try to introduce the **collaborative filtering recommendation algorithm**.
	Q2. Please introduce several common feature selection algorithms for **text clustering**.
	Q3. Please introduce the characteristics of **Python's recursive function**.
	Q4. Describe a method of applying the attention mechanism to the **RNN** model.
Questions recommended by JLMIA on the job posting	Q1. Please introduce the role of the **SVM kernel function**.
	Q2. Please introduce how to calculate **mean average precision**.
	Q3. Briefly describe the memory allocation of **C** and **C++** program compilation.
	Q4. Please introduce the characteristics of **Python's recursive function**.
Questions recommended by JLMIA on the resume	Q1. How does **LSTM** avoid gradient dispersion and gradient explosion?
	Q2. Please introduce **CNN's convolution kernel**.
	Q3. Please introduce the **PageRank algorithm**.
	Q4. When **C++** creates an object, where is the memory of the object and the pointer to the object allocated?

interpretability. Thus, it will greatly help interviewers to subsequent in-depth analysis of candidates' various abilities.

5 RELATED WORK

Generally, the related works of our study can be grouped into two aspects, namely *intelligent interview systems*, as well as prior arts on *entity extraction and relation extraction*.

Intelligent Interview System. As the basic tool for recruitment to indicate the future performance of job candidate [10, 22], the employment interview task has attracted wide attentions on various topics, including qualitative interview design [28], interview performance analysis [3] and interview simulation [1]. Recently, with the prevalence of AI technologies, some efforts have been made in novel perspectives. For instance, *Naim et al.* analyzed the interview videos for automatically quantifying verbal and nonverbal behaviors in the context of job interviews [17], and *Shen et al.* developed a latent variable model with comprehensively learning the large-scale real-world interview data to support job interview assessment [23]. Meanwhile, some other researchers attempted to leverage data analytical techniques to support intelligent interview with capturing trend of recruitment. For example, *Zhu et al.* proposed a novel sequential latent variable model to predict recruitment market trends [37], and *Xu et al.* proposed a data driven approach for modeling the popularity of job skills [31], while *Lee et al.* even designed a comprehensive job recommender system with considering the preferences of both employers and candidates [12].

Different from prior arts, in this paper, we study a novel task, i.e., personalized question recommendation task. Along this line, our research could be implemented in real-world job interview scenarios, which improves the efficiency of job interview assessment.

Entity Extraction and Relation Extraction. Another related topic is entity graph (skill-graph in this paper) construction, which includes two subtasks, entity extraction and relation extraction.

The entity extraction task, which targets at locating entities in textual content, has been widely studied in recent years. Traditionally, prior arts usually relied on high-quality hand-crafted features and well-designed models, e.g., Hidden Markov Model [36], Conditional Random Fields (CRF) [18] and Perceptron Models [14, 21]. Recently, with the development of deep learning techniques, which requires less labor-intensive features and achieved better results based on large-scale unlabeled textual data, several neural-based techniques have been proposed. For instance, *Lample et al.* combined bidirectional Long Short-Term Memory networks (BiLSTMs) and CRF in a supervised learning setting [11], and *Chiu et al.* used a hybrid BiLSTM and Convolutional Neural Network (CNN) to detect both word and character-level features [2]. Besides, some effective tools have been designed, such as Gate [4] and Stanford parser [5] for extracting entities of location, person and organization.

Correspondingly, the relation extraction task targets at revealing semantic relations between entities. Many efforts have been made on this task with several classic approaches [26, 34]. Among them, Hypernym-hyponym ("is-a") relation has been treated as one of the most important relations between the entities. To reveal this relation, traditional approaches mainly constructed entities hierarchies based on dictionaries and encyclopedias. For instance, *Suchanek et al.* linked the categories in Wikipedia to WordNet [25], and *Li* designed a set of language-specific features to construct a large-scale Chinese taxonomy from Wikipedia [13]. Though these approaches performed well, their coverage could be limited due to the constraint of the data source. Some other methods based on pattern matching, e.g., [24, 30], employed lexical patterns to extract "is-a" relations. Recently, *Fu et al.* employed uniform linear projection and piecewise linear projection to map the embedding of hyponym word to its hypernym [6], which have handled the context sparsity issue in Chinese stemmed to a certain extent, and further extended by several works like [27, 29, 32].

Different from prior arts, in this paper, we design a bidirectional LSTM-CRF neural network with adapted gate mechanism to extract skill entities, and then improve the reliability of extracted skill entities based on label propagation. Along this line, hypernym-hyponym relations between entities have been revealed to construct the Skill-Graph, which performed well based on the validation.

6 CONCLUSION

In this paper, we introduced a deployed personalized question recommender system, DuerQuiz, for intelligent job interview assessment in talent recruitment. The key idea of DuerQuiz is to construct a knowledge graph of job skills through mining the abundant historical recruitment data and large-scale job skill data available from the Internet. Specifically, we first developed a novel skill entities extraction approach based on a bidirectional LSTM-CRF neural network with adapted gate mechanism, and then designed a label propagation method based on more than 10 billion click-through data for improving the reliability of extracted skill entities. Furthermore, we discovered the hypernym-hyponym relations between skill entities for constructing the *Skill-Graph* and proposed a personalized question recommendation algorithm for improving job interview assessment. Finally, extensive experiments on real-world recruitment data showed the effectiveness of DuerQuiz. In particular, DuerQuiz had been deployed for generating written exercises in the 2018 Baidu Campus Recruitment event, and achieved remarkable performances in terms of efficiency and effectiveness for selecting the right talents compared with a traditional non-personalized human-only assessment method.

Acknowledgements. This work was partially supported by grants from the National Natural Science Foundation of China (No.91746301, 61836013, 61703386, 61773361) and the National Key Research and Development Program of China (No. 2018YFB1004300).

REFERENCES

[1] Tobias Baur, Ionut Damian, Patrick Gebhard, Kaska Porayska-Pomsta, and Elisabeth André. 2013. A job interview simulation: Social cue-based interaction with a virtual character. In *Social Computing (SocialCom), 2013 International Conference on*. IEEE, 220–227.

[2] Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4 (2016), 357–370.

[3] Amy JC Cuddy, Caroline A Wilmuth, Andy J Yap, and Dana R Carney. 2015. Preparatory power posing affects nonverbal presence and job interview performance. *Journal of Applied Psychology* 100, 4 (2015), 1286.

[4] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. A framework and graphical development environment for robust NLP tools and applications.. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 168–175.

[5] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 363–370.

[6] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

[7] Ruiji Fu, Bing Qin, and Ting Liu. 2013. Exploiting multiple sources for open-domain hypernym discovery. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1224–1234.

[8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[9] Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational Linguistics*.

[10] Allen I Huffcutt and David J Woehr. 1999. Further analysis of employment interview validity: a quantitative evaluation of interviewer-related structuring methods. *Journal of Organizational Behavior: The International Journal of Industrial, Occupational and Organizational Psychology and Behavior* (1999).

[11] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).

[12] D. H. Lee and Peter Brusilovsky. 2007. Fighting Information Overflow with Personalized Comprehensive Information Access: A Proactive Job Recommender. In *Third International Conference on Autonomic and Autonomous Systems*.

[13] Jinyang Li, Chengyu Wang, Xiaofeng He, Rong Zhang, and Ming Gao. 2015. User generated content oriented chinese taxonomy construction. In *Proceedings of the 17th Asia-Pacific Web Conference*. Springer, 623–634.

[14] Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 402–412.

[15] Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 339–346.

[16] Shotaro Misawa, Motoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. 2017. Character-based Bidirectional LSTM-CRF with words and characters for Japanese Named Entity Recognition. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*. 97–102.

[17] Iftekhar Naim, M Iftekhar Tanveer, Daniel Gildea, and Mohammed Ehsan Hoque. 2015. Automated prediction and analysis of job interview performance: The role of what you say and how you say it. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, Vol. 1. IEEE, 1–6.

[18] Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv:1404.5367* (2014).

[19] Karolina Piwiec. 2018. 65+ recruitment stats HR pros must know in 2018. https://devskiller.com/65-recruitment-stats-hr-pros-must-know-2018/. (2018).

[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 *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 25–34.

[21] Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 147–155.

[22] Frank L Schmidt and Ryan D Zimmerman. 2004. A counterintuitive hypothesis about employment interview validity and some supporting evidence. *Journal of Applied Psychology* 89, 3 (2004), 553.

[23] Dazhong Shen, Hengshu Zhu, Chen Zhu, Tong Xu, Chao Ma, and Hui Xiong. 2018. A Joint Learning Approach to Intelligent Job Interview Assessment.. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.

[24] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*. 1297–1304.

[25] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 697–706.

[26] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. 455–465.

[27] Liling Tan, Rohit Gupta, and Josef van Genabith. 2015. Usaar-wlv: Hypernym generation with deep neural nets. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. 932–937.

[28] Daniel W Turner III. 2010. Qualitative interview design: A practical guide for novice investigators. *The qualitative report* 15, 3 (2010), 754–760.

[29] Chengyu Wang and Xiaofeng He. 2016. Chinese hypernym-hyponym extraction from user generated categories. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 1350–1361.

[30] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 481–492.

[31] 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 *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 2572–2579.

[32] Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa, and Yutaka Sasaki. 2016. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 1871–1879.

[33] Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. *arXiv preprint arXiv:1704.08960* (2017).

[34] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

[35] Yue Zhang and Jie Yang. 2018. Chinese NER Using Lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

[36] GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 473–480.

[37] Chen Zhu, Hengshu Zhu, Hui Xiong, Pengliang Ding, and Fang Xie. 2016. Recruitment market trend analysis with sequential latent variable models. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 383–392.

[38] 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.