Job and Employee Embeddings: A Joint Deep Learning Approach

Hao Liu, Yong Ge, *Senior Member, IEEE*

Abstract—The accumulated massive job and employee data at various platforms such as LinkedIn and Glassdoor are very valuable for many online applications such as job/employee search and recommendations. In order to exploit these data, an interesting and practical problem is how to learn effective job and employee representations, which could be further utilized for many computing tasks such as searching for similar jobs and employees. Yet this problem is very challenging because these user-generated job and employee data are semi-structured and created without standards, which makes them very messy, sparse, and difficult to model. Developing novel and advanced methods to learn job and employee representations has become an urgent need. To this end, in this paper, we develop a novel neural network model for job and employee embeddings. Our proposed approach consists of three components to model career data from three levels of granularity: job content, job context, and job sequence. We fine-tune a transformer model to learn the semantics of massive text in job content, build a shallow neural network to accumulate contextual information in job sequences, and develop an RNN encoder-decoder model to learn representations of employees' career paths. To evaluate the proposed method, we conduct two experimental tasks: job similarity and employee similarity searches. The experimental results with a real-world dataset demonstrate the superiority of the developed approach.

Index Terms—Career Path, Job Embedding, Employee Embedding, Job Similarity Search, Employee Similarity Search

1 Introduction

With the booming of online recruitment, millions of workers build their resumes and professional profiles and many firms and recruiters post jobs on various platforms such as LinkedIn[1], Glassdoor[2] and Indeed[3]. These online platforms have collected a lot of job and employee data. These information-rich data become valuable resources for many applications such as job recommendation [1], [2], [3] and career planning [4], [5]. In order to fully exploit these data, one important task is to learn effective representations of jobs and employees [6], which could benefit many analytical tasks on these online platforms. For example, platforms such as LinkedIn and Glassdoor could utilize the representations for job similarity search, personalized job recommendations, and even career path prediction [1], [4], [7], [8], [9]; online recruiters may leverage the representations to effectively locate similar employees and profiles [10].

To this end, in this paper, we aim to learn job and employee representations from massive user-generated career path data. The career path of an employee is a sequence of job positions. It consists of two informative parts. One is job content including job title, company, job description, and job duration; the other part is job transition information such as promotions. Both parts include important information about jobs and employees and should be incorporated simultaneously to learn job and employee representations. However, there are significant challenges in modeling the employees' career paths. First, the user-generated career path data is semi-structured and difficult to tackle. It has structured fields such as company and massive unstructured text such as job descriptions. Moreover, the current job position in one career path is different from previous positions because the current one is ongoing [10], and its duration is from a certain date to the "present". Second, employees describe their jobs without standards, which makes much noise in the data. They may add auxiliary information to enrich the job title such as *Microsoft Azure* in *Software Development Engineer, Microsoft Azure*. Abbreviated job titles may also exist such as *SDE* for *Software Development Engineer*. Third, there may be different yet synonymous job titles [11] in the user-generated career data. For example, *Software Engineer* and *Software Development Engineer*, which exist at different firms, almost represent the same job. Fourth, the data has long-tail distribution, i.e., some jobs have high-frequency occurrences while others have very low frequencies.

In the literature, there are some prior works on studying job or employee representations. The approach with handcrafted features to represent jobs and employees heavily relies on external domain knowledge [10]. It is costly to collect such domain knowledge, and the handcrafted features are usually not applicable for other tasks. Job and employee representations learned by supervised methods are often used for a specific task, and it is also difficult to obtain annotated data [7], [8]. Besides, a number of unsupervised embedding techniques have been proposed to learn job representations [1], [11] and employee representations [12]. The advantage of these unsupervised embedding methods is that the generated representations can be deployed in many tasks rather than a specific one. However, these unsupervised methods did not solve the problem well. For job representations, previous works treat a job title as one position but neglect the discrepancy among companies [1], [11]. Different companies may belong to different industries and

• *Hao Liu and Yong Ge are with The University of Arizona. E-mail: {liuhao16,yongge}@email.arizona.edu.*

1. https://www.linkedin.com/
2. https://www.glassdoor.com
3. https://www.indeed.com/

have different business functions, reputations and cultures that attract employees. Thus the job representations should consider both title and company within the context of the job market. Extant graph-based models, where nodes represent jobs or employees and edges denote relations between them, emphasize the job transitions, but ignore the semantic information of unstructured text data such as job descriptions [1], [11], [12]. The semantics of the text data can help depict the scope and functions of jobs. Furthermore, in order to alleviate long-tail distribution, prior studies normalize the job titles by using external systems [1], [11] that may not be reliable or available. For employee representations, Cai et al. developed DBGE [12], which constructed employee representations from a bipartite graph that only considers companies while ignoring job titles and descriptions.

To address the aforementioned research gap, in our study, we propose a novel neural network method to jointly learn representations of jobs and employees by modeling talent career data. In our method, we model each talent's career path from three levels of granularity: job content, job context and job sequence. Specifically, we first fine-tune a transformer model to learn a content embedding for each job by modeling its job title, company and job description, which is able to tackle noise in the job data. We then build a neural network model to accumulate the content embeddings of contextual job positions to learn the embedding of one target job. Finally, an encoder-decoder component is designed to learn employee embeddings from the job sequence. Our proposed method is different compared to previous works in multiple aspects. First, our model learns both job and employee representations simultaneously as the representation of an employee depends on job positions in her career path. Second, we utilize transitional and textual information in massive career data, both of which are important for learning embeddings of jobs and employees. Third, our job embeddings represent both job title and company (e.g., data scientist at Amazon) rather than job title (e.g., data scientist) alone. Fourth, our model does not require the normalization of job titles, which is often needed in previous related studies.

The main contributions of this study are summarized in the following:

- We study the problem of jointly learning job and employee embeddings by utilizing both transitional and textual information in massive career data, which has not been studied in the extant literature.
- We propose a novel neural network model to simultaneously learn job and employee embeddings, which models employees' career data from three levels of granularity, including job content, job context and job sequence.
- We conduct two empirical tests to evaluate the performance of our method with a large-scale real-world employee career dataset. Experimental results show that our proposed method is effective and outperforms baseline approaches.

The rest of the paper is organized as follows. We first review related literature in Section 2. Then, the problem formulation and method details are described in Section 3. Experiments and evaluations are introduced in Section 4. Finally, Section 5 concludes our work.

2 Related Work

In this section, we review the following related work in three aspects.

2.1 Job and Employee Embedding

Existing job and employee embedding models could be classified into two categories: supervised and unsupervised methods. Supervised methods model annotated data for a specific task, such as job-employee matching [7], [8]. Bian et al. developed a multi-view graph method to match job applicants with job openings [7]. A bipartite neural network based on Convolutional Neural Networks (CNN) [13], [14] was designed to match talent career paths with job postings [8]. The drawbacks of such supervised methods include: (1) it is often difficult to obtain sufficient labeled data, and (2) the supervised learning model is usually trained for a specific task and could not work well for other tasks. Unsupervised methods utilize career path data to learn general job and employee representations without label information [11], [12]. Previous studies have built various graph-based models to learn the transition dependencies among jobs. A multi-view encoder-decoder model was developed to learn representations of job titles, which created a large-scale graph based on job titles, where edges represent the job transitions among job titles [11]. A job and skill embedding model was designed to utilize three networks including a job transition network, a job-skill network and a skill co-occurrence network [1], [9]. DBGE was proposed to learn employee embeddings [12], where the bipartite graph consists of two types of nodes: employee and company, and then Horary Random Walk [12] was proposed to sample sequences from the graph, and Skip-gram algorithm [15] was applied to learn the vector representations of nodes. For job embedding, one challenge of graph-based methods is messy job titles. Therefore, they constructed the graph using only companies [12] or job titles [1], [11], but it was required to normalize the job titles using external systems [1], [11] before they built the graph. However, those normalization methods may not be reliable or available. Besides, graph-based methods usually have to aggregate job positions, where some important attributes of jobs are approximated with averages such as job duration or even ignored such as job descriptions [11]. As for employee embedding, Cai et al. [12] constructed employee representations from a bipartite graph, which only considered companies but ignored massive text data including job titles and descriptions.

2.2 Transformer Models

As a revolutionary technique to learn sequential data, transformer models have achieved state-of-the-art results in a lot of natural language understanding (NLU) tasks [16], [17], [18]. They stack self-attention and fully-connected layers to capture contextual information. It is computationally expensive to train a transformer model. This has led to the development of transfer learning [19], [20] with transformers such as BERT [21], DistilBERT [22] and RoBERTa [23], which have been trained on large amounts of unannotated language corpora and could be fine-tuned for a specific NLU task [24], [25], [26], [27], [28], [29], [30]. For example, the

TABLE 1. Description of Notations

Symbol	Definition
$\mathcal{U}$	The set of all career paths, the size $M = \|\mathcal{U}\|$
$\mathcal{U}[u] = \{p_{u1}, p_{u2}, \ldots, p_{un}\}$	The career path of employee u
$p_{ui} = (l_{ui}, c_{ui}, \tau_{ui}, d_{ui})$	The i-th job position (title, company, duration, description) of employee u
$\mathcal{J}$	The set of all jobs (job title and company), the size $N = \|\mathcal{J}\|$
$\mathcal{X}$	The set of all job content embeddings
$\mathcal{Y}$	The set of all target job embeddings
$\mathcal{Z}$	The set of all employee embeddings
K	The dimension size of embeddings in $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$
$\mathcal{C}$	The set of all companies
$\mathcal{V}$	The set of all company embeddings
D	The dimension size of embeddings in $\mathcal{V}$
$NEG(p_{ui})$	The sampled negative position set for p_{ui}, the size is $Q = \|NEG(p_{ui})\|$

popular RoBERTa-base model with 110 million parameters is trained on five corpora with 160GB of uncompressed text [23], and it has achieved state-of-the-art results on tasks such as GLUE [31], RACE [32] and SQuAD [33]. Compared to previous pre-trained language models such as Word2vec [15], [34] and Glove [35], which produce a single vector representation for each word even if it is polysemous, transformer models could provide a context-dependent vector for each token and have outstanding performance in many sophisticated NLU tasks.

2.3 RNN Encoder-Decoder Models

The Encoder-Decoder architecture with Recurrent Neural Networks (RNN) consists of two RNNs as the encoder and decoder. The encoder maps an input sequence into a fixed-length internal vector, and the decoder converts this internal vector into the output sequence. RNN Encoder-Decoder models have been widely designed for many sequence-to-sequence problems. Machine language translation has benefited much from the encoder-decoder architecture with RNN [36], [37], [38], [39] due to its ability to map source and target sentences and to tackle variant lengths of input and output sequences. Other RNN encoder-decoder models have also been developed for sequence-to-sequence tasks such as automatic speech recognition [40], [41], text summation [42], [43]. Besides, another popular application of RNN encoder-decoder models is sequence encoding. The interval vector is a valuable representation of the input sequence [38], which could be used for tasks such as classification.

3 METHOD

In this section, we first formalize our research problem based on the notations listed in Table 1 and then introduce our proposed Job and Employee Embedding (JaEE) model for solving the problem.

3.1 Problem Statement

Let $\mathcal{U}$ denote the set of employees, and the size of $\mathcal{U}$ is denoted by $M = |\mathcal{U}|$. The career path of employee u is

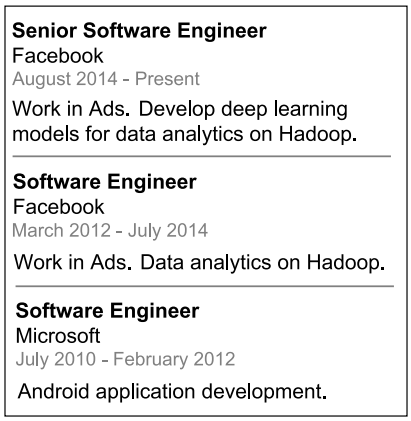


Fig. 1. An example of a talent career path.

represented as $\mathcal{U}[u] = \{p_{u1}, p_{u2}, \ldots, p_{un}\}$, where p_{ui} is the i-th job position of employee u. Each position has four components, denoted by a tuple $p_{ui} = (l_{ui}, c_{ui}, \tau_{ui}, d_{ui})$, which means employee u worked at company c_{ui} with title l_{ui} for duration τ_{ui}, and d_{ui} represents the job description. An example of a talent career path is demonstrated in Fig. 1, where the employee has worked in three job positions from July 2010 to the present. We collect all jobs (i.e., job title and company), denoted as $\mathcal{J}$. Let $N = |\mathcal{J}|$ denote the total number of jobs. All companies form a set, denoted as $\mathcal{C}$. Our goal in this paper is to learn job and employee embeddings jointly, denoted as $\mathcal{Y}$ and $\mathcal{Z}$, respectively. Both $\mathbf{y} \in \mathcal{Y}$ and $\mathbf{z} \in \mathcal{Z}$ have K dimensions. As an ancillary product, we also learn a vector representation for company $c \in \mathcal{C}$, and the set of company embeddings is denoted as $\mathcal{V}$. The dimension of company embeddings is denoted by D. In this paper, we use bold lower case letters for vectors and bold capital letters for matrices and calligraphic capital letters for sets. The i-th element of set $\mathcal{S}$ is denoted by $\mathcal{S}[i]$ and the i-th column of matrix $\mathbf{H}$ is denoted by $\mathbf{H}[i]$. Euclidean norm is denoted by $||\cdot||$. A hat sign is used to label a predicted value.

3.2 An Overview of the Model

An overview of the study in this paper is demonstrated in Fig. 2. Our model iteratively takes the career path of each employee as input and learns job embeddings $\mathcal{Y}$ and employee embeddings $\mathcal{Z}$. The career paths contain two parts of information for jobs: job content (title, company, description, duration) and job transition sequence. In order to learn our target embeddings with these two parts of information, we construct the model at three levels of granularity: job content, job context and job sequence. Accordingly, the model consists of three components, namely *Job Content Embedding*, *Job Context Learning* and *Employee Embedding*. Given an employee, *Job Content Embedding* processes the job content of each job position to learn a content embedding. Unlike many previous methods for job representation learning, no job title normalization is required to merge messy titles in our model. Then *Job Context Learning* takes into account job context dependency to learn target job embeddings by accumulating the content embeddings of prior job positions in individual career paths. Furthermore, *Employee Embedding* deploys RNN encoder-decoder technique to learn the employee representation from the job transition sequence. As

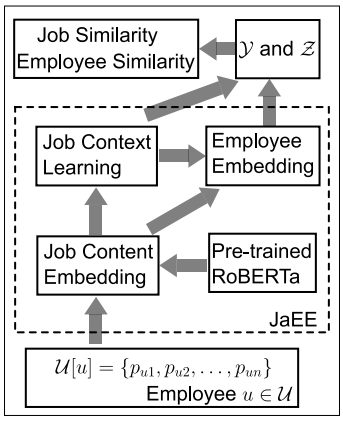


Fig. 2. An overview of our study.

the final outputs of our model, the target job embeddings $\mathcal{Y}$ and employee embeddings $\mathcal{Z}$ are further applied for the job and employee similarity searches. The architecture of our proposed JaEE is shown in Fig. 3. We introduce the details of the three components in the following sections.

3.3 Job Content Embedding

A job title can usually reveal the job level and responsibility. For instance, in the job title *Senior Project Manager*, *Senior* reflects the job level, and *Manager* reflects management responsibility. However, job titles are usually under-exploited in talent analytics due to two challenges. First, it is difficult for machines to understand title synonyms. For instance, *Software Engineer*, *Software Development Engineer* and *Software Programmer* represent the same job position in labor markets. Second, employees may use various abbreviations of job titles or add auxiliary information to titles, which causes titles to have a very sparse distribution. Besides, a job description can provide additional details such as skills and knowledge about the job position, but it varies much among employees and is even missing in many employee-generated career data. In order to model both job title and description and overcome these challenges, inspired by the success of transformer models in Natural Language Understanding (NLU), we fine-tune RoBERTa [23] model to learn a semantic vector from the job title and description. Let $p_{ui} = (l_{ui}, c_{ui}, \tau_{ui}, d_{ui})$ denote the i-th position of employee u. As illustrated in Fig. 3, job title l_{ui} and description d_{ui} are tokenized into two sequences of tokens and fed into the RoBERTa model, where *CLS* and *SEP* are reserved tokens in RoBERTa. Then the outputs of *CLS* and title are converted into vectors with $(K - D)$ dimensions through a fully-connected (FC) layer. We use function $f_R(\cdot)$ to denote RoBERTa and the FC layer. Let $\mathbf{b}_{ui} \in \mathbb{R}^{K-D}$ denote the output vector of *CLS*, $\mathbf{H}_{ui}$ denote the output of title tokens. We have:

$$[\mathbf{b}_{ui}, \mathbf{H}_{ui}] = f_R(l_{ui}, d_{ui}), \tag{1}$$

where $\mathbf{H}_{ui} \in \mathbb{R}^{(K-D)\times g}$, g is the number of tokens in the job title l_{ui}. Key tokens and ancillary tokens in those job titles should contribute differently to the title representations. Therefore, an attention mechanism [37] is applied after FC layer. The attention mechanism is defined as follows:

$$\begin{aligned} w_j &= \frac{\exp\left(\mathbf{b}_{ui}^\top \mathbf{H}_{ui}[j]\right)}{\sum_{k=1}^{g} \exp\left(\mathbf{b}_{ui}^\top \mathbf{H}_{ui}[k]\right)} \\ \mathbf{a}_{ui} &= \sum_{j=1}^{g} w_j \mathbf{H}_{ui}[j], \end{aligned} \tag{2}$$

where superscript $\top$ means transpose. $\mathbf{a}_{ui} \in \mathbb{R}^{K-D}$ is the job title embedding of position p_{ui}. $\mathbf{H}_{ui}[j]$ is the j-th column of $\mathbf{H}_{ui}$, and it is the representation of j-th token in job title l_{ui}.

Company is also significant information for jobs. We concatenate the job title embedding $\mathbf{a}_{ui}$ with the company vector $\mathbf{v}_{ui} \in \mathbb{R}^D$ as the final content representation for job position (l_{ui}, c_{ui}), and the concatenation is defined as follows:

$$\mathbf{x}_{ui} = concat(\mathbf{a}_{ui}, \mathbf{v}_{ui}), \tag{3}$$

where $\mathbf{x}_{ui} \in \mathbb{R}^K$ is the job content embedding and $\mathbf{v}_{ui} \in \mathcal{V}$ is the embedding for company c_{ui}.

3.4 Job Context Learning

With the job content embeddings, we aim to learn the final vector representation for each job in this component, which is named target job embedding, by considering the contextual information. The talent career path is a sequence of job positions. The acquisition of each job position usually relies on the accumulation of knowledge and skills from previous job positions. Therefore, the target job embedding should consider two types of contextual information from prior positions: job content and job duration. We adapt the core idea of Continuous Bag-of-Words (CBOW) from Word2vec [34] to build a shallow neural network to learn our target job embedding set $\mathcal{Y}$. Our method is different from the CBOW of Word2vec in two aspects. First, we consider one-side context (prior job positions) rather than both-side context. Second, we incorporate job duration into the context learning. Besides, our design can address the missing duration of the current position as an employee's current position is ongoing, and it is unknown when the current job position will terminate. Specifically, we treat the content embeddings of prior job positions as contextual representations, weight them by using job durations, and accumulate them together, which is defined as:

$$\mathbf{e}_{ui} = \sum_{k=1}^{i-1} \alpha \tanh(\tau_{uk}) \mathbf{x}_{uk}, \tag{4}$$

where $\mathbf{e}_{ui}$ is the cumulative contextual embedding for position p_{ui}, and α is a learnable parameter. Similar to CBOW, given the observed context of position p_{ui}, denoted as $context(p_{ui})$, the conditional probability for position p_{ui} is $P(p_{ui}|context(p_{ui}))$. We use Negative Sampling [15] to sample a set of negative positions for position p_{ui}, denoted as $NEG(p_{ui})$. The conditional probability of a negative position $p_j \in NEG(p_{ui})$ is $P(p_j|context(p_{ui}))$. They are defined as:

$$P(p_{ui}|context(p_{ui})) = \sigma(\mathbf{y}_{ui}^\top \mathbf{e}_{ui}), \tag{5}$$

$$P(p_j|context(p_{ui})) = \sigma(\mathbf{y}_j^\top \mathbf{e}_{ui}), \tag{6}$$

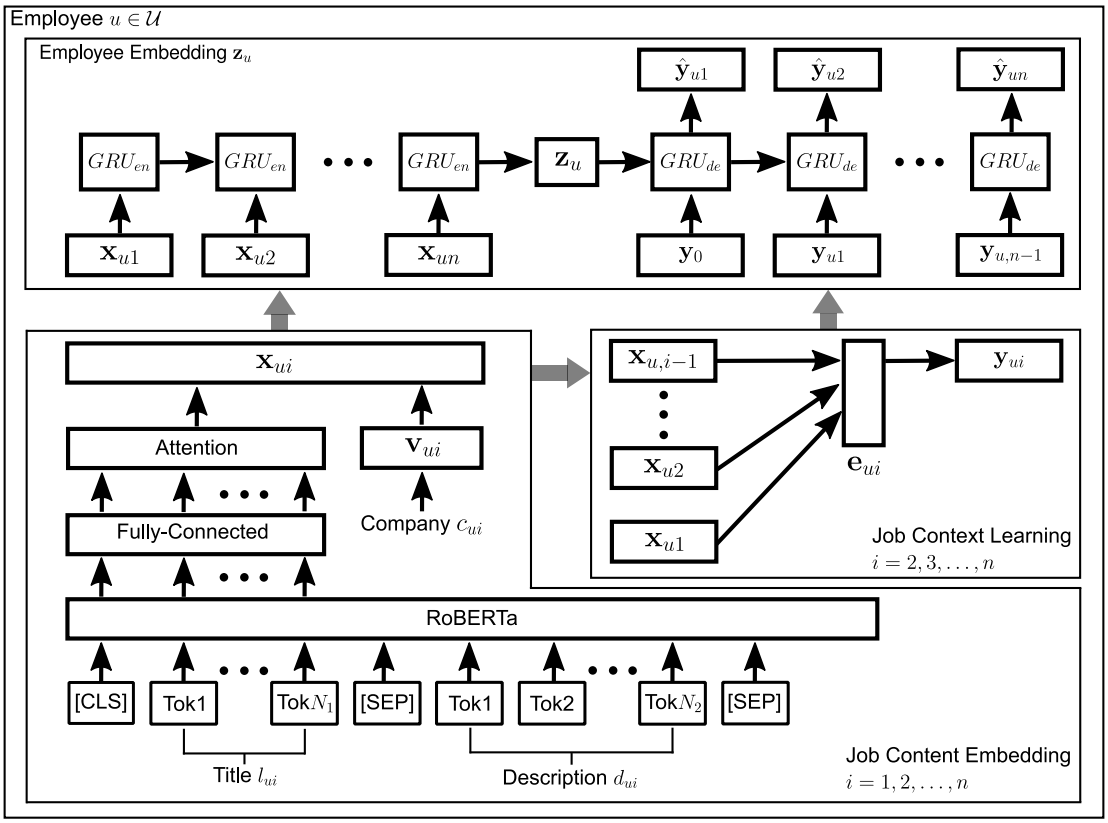


Fig. 3. The architecture of JaEE.

where $\mathbf{y}_{ui} \in \mathcal{Y}$ and $\mathbf{y}_j \in \mathcal{Y}$ are the target job embeddings for position p_{ui} and the negative position p_j, respectively. σ is the logistic sigmoid function. Let Q denote the size of $NEG(p_{ui})$. Given the context of p_{ui}, we want to maximize $P(p_{ui}|context(p_{ui}))$ while minimizing $P(p_j|context(p_{ui}))$ since we anticipate position p_{ui} fit its context well while sampled negative positions do not. Thus, we define the loss function in *Job Context Learning* for employee u as the following:

$$\begin{aligned} loss1 = &- \log\{\prod_{i=2}^{n} P(p_{ui}|context(p_{ui})) \\ &(\prod_{p_j \in NEG(p_{ui})} (1 - P(p_j|context(p_{ui}))))^{\frac{1}{Q}}\} \\ = &- \sum_{i=2}^{n}\{\log \sigma(\mathbf{y}_{ui}^{\top}\mathbf{e}_{ui}) + \frac{1}{Q}\sum_{p_j \in NEG(p_{ui})} \log(1 - \sigma(\mathbf{y}_j^{\top}\mathbf{e}_{ui}))\}. \end{aligned} \tag{7}$$

3.5 Employee Embedding

The *Employee Embedding* component learns a vector representation from the career path of each employee by employing an RNN encoder-decoder model. Specifically, for employee u, with the learned job content embeddings $\{\mathbf{x}_{u1}, \mathbf{x}_{u2}, \ldots, \mathbf{x}_{un}\}$, we feed them into a Gated Recurrent Units Network (GRU) [38] encoder to learn the employee embedding $\mathbf{z}_u$. Then the learned target job embeddings $\{\mathbf{y}_0, \mathbf{y}_{u1}, \ldots, \mathbf{y}_{u,n-1}\}$ and $\mathbf{z}_u$ are input into another GRU decoder to learn predicted job embeddings $\{\hat{\mathbf{y}}_{u1}, \hat{\mathbf{y}}_{u2}, \ldots, \hat{\mathbf{y}}_{un}\}$. We do not integrate attentions from encoder to decoder in this component, and the single vector $\mathbf{z}_u$ is the only bridge connecting the encoder and decoder, because we want to compress as much information as possible into $\mathbf{z}_u$. The mathematical formula of this component is expressed as:

$$\mathbf{z}_u = GRU_{en}(\{\mathbf{x}_{u1}, \mathbf{x}_{u2}, \ldots, \mathbf{x}_{un}\}), \tag{8}$$

$$\{\hat{\mathbf{y}}_{u1}, \hat{\mathbf{y}}_{u2}, \ldots, \hat{\mathbf{y}}_{un}\} = GRU_{de}(\{\mathbf{y}_0, \mathbf{y}_{u1}, \ldots, \mathbf{y}_{u,n-1}\}, \mathbf{z}_u), \tag{9}$$

where $\mathbf{y}_0$ is a reserved vector, and its elements are zeros. There are too many unique jobs because of messy job titles. Thus, it is impracticable to predict a N-dimensional vector as the predicted job probabilities via Softmax function. Instead, we output $\hat{\mathbf{y}}_{ui} \in \mathbb{R}^K (i = 1, 2, \ldots, n)$ to approximate $\mathbf{y}_{ui} (i = 1, 2, \ldots, n)$. The loss function of the encoder-decoder part for employee u is defined as $loss2$:

$$loss2 = \sum_{i=1}^{n} \sqrt{|\mathbf{y}_{ui} - \hat{\mathbf{y}}_{ui}|^2}. \tag{10}$$

Note that both *Job Context Learning* and *Employee Embedding* incorporate transition information in the career path yet in different ways. *Job Context Learning* models the relationship between the accumulation of prior job content embeddings and the target job embeddings. On the contrary, within *Employee Embedding*, the encoder learns the sequence representation by modeling the transition information among the job content embeddings, and the decoder learns transition dependency among target job embeddings.

Algorithm 1 The learning algorithm of JaEE.

Input: M career paths $\mathcal{U}$; Embedding dimension K; Maximum epochs T; Batch size m.
Output: Job Embeddings $\mathcal{Y}$; Employee Embeddings $\mathcal{Z}$.

```
Randomly initialize Y, Z, V. Load pre-trained RoBERTa layers.
for t ← 1 to T do
    Sample a mini batch B from U.
    for u ∈ B do
        Get the career path of u: B[u] = {p_u1, p_u2, ..., p_un}.
        for p_ui ∈ B[u] do
            Compute job content embedding x_ui (Eq.(3)).
            Sample negative positions NEG(p_ui).
        end for;
        Compute loss1 (Eq.(7))
        Compute loss2 (Eq.(10))
        Compute the loss L^u(θ^{t-1}) for employee u (Eq.(11))
    end for;
    Update θ with loss as (1/m) Σ_{u∈B} L^u
end for;
```

3.6 Learning Objective and Algorithm

Let θ denote all learnable parameters in JaEE model. We combine $loss1$ and $loss2$ with a hyper-parameter λ to define our final loss function as:

$$\mathcal{L}(\theta) = \sum_{u=1}^{M} \lambda loss1 + (1-\lambda) loss2 + \beta||\theta||^2, \qquad (11)$$

where $||\theta||^2$ is the L_2 regularization penalty controlled by the hyper-parameter β. Our task is to learn $\hat{\theta}$ by minimizing the loss function:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta). \qquad (12)$$

We train our JaEE model with mini-batch gradient descent optimizer [44]. The detailed algorithm is shown in Algorithm 1.

4 Experiments

In this section, we demonstrate the performance of our proposed JaEE model by comparing it with several baseline methods.

4.1 Data Description

We evaluate our method on a real-world data set. We collected employee career paths in IT field from a major employment website in March 2017. The samples have been anonymized by deleting identity information such as names and user ID to protect their privacy. We filtered out career paths with only one or two positions. In our dataset, there are 69620 career paths, who worked in 860 companies with 190060 unique jobs (i.e., the combination of title and company). The data set is very sparse as 88% of those jobs only appear once. In Fig. 4, we plot the histogram to illustrate the distribution of the number of positions in career paths. We see that most career paths have less than ten job positions in our data. The histogram of job position duration is also shown in Fig. 5. The duration of most positions ranges from 0.5 to 6 years.

4.2 Benchmark Collection and Evaluation Metric

After we obtained the job and employee embeddings, we conducted two experiments to evaluate the effectiveness of embeddings: job similarity comparison and employee similarity comparison. In order to obtain the similarity benchmarks, we conducted two human-labeling tasks on Amazon Mechanical Turk (MTurk)[4], which is a popular crowdsourcing platform that uses human intelligence to annotate the data. We only hire MTurk Master workers[5] from US with at least 99% approval rate since those MTurk workers are more conscientious and responsible. The details of the two tasks are as follows.

In the first task, we aim to obtain the job similarity benchmark. We randomly sample different 6300 tuples, and each tuple consists of three different jobs (title and company) A, B and C, and we ask five MTurk workers to compare the similarity between jobs A and B with the similarity between jobs A and C based on their experience about occupation type, title level, company reputation. They have three options as:

1) The similarity between jobs A and B is higher than the similarity between jobs A and C.
2) The similarity between jobs A and B is lower than the similarity between jobs A and C.
3) The similarity between jobs A and B is almost the same as the similarity between jobs A and C.

We add the third option for workers in case they are not sure about the relationship among A, B and C. Therefore, we only keep the results choosing option 1 or 2. Besides, in order to ensure high-quality labels, we only keep tuples that all five workers agree on the same option. We obtained 1289 effective tuples, which are randomly split into 258 tuples as the validation set for hyper-parameter tuning and 1031 tuples as a testing set for final job similarity evaluation.

For the second task, our goal is to obtain the employee similarity benchmark. We randomly sample 1000 tuples, and each tuple has three different employees' career paths A, B and C. Following the same settings as the first task, we ask MTurk workers to compare the similarity between career paths A and B with the similarity between career paths A and C, and we obtained 419 effective employee tuples. We use these tuples as a testing set for employee similarity evaluation.

We use Accuracy as our metric to evaluate our job and employee embeddings on the testing sets. Given a tuple (A, B, C), where A, B and C are three different jobs or career paths, let $r(A, B, C)$ denote the similarity relationship among A, B and C. $r(A, B, C)$ is defined as:

$$r(A,B,C) = \begin{cases} 1 & \text{if similarity(A,B) > similarity(A,C),} \\ 0 & \text{if similarity(A,B) < similarity(A,C),} \end{cases} \qquad (13)$$

where $similarity(A, B)$ means the similarity between A and B. $r(A, B, C)$ is annotated by MTurk workers. Given a set of learned job embeddings $\mathcal{Y}$ and a set of employee embeddings $\mathcal{Z}$, we calculate Euclidean distance to measure the dissimilarity between any pair of jobs or career paths.

4. https://www.mturk.com/
5. https://www.mturk.com/help

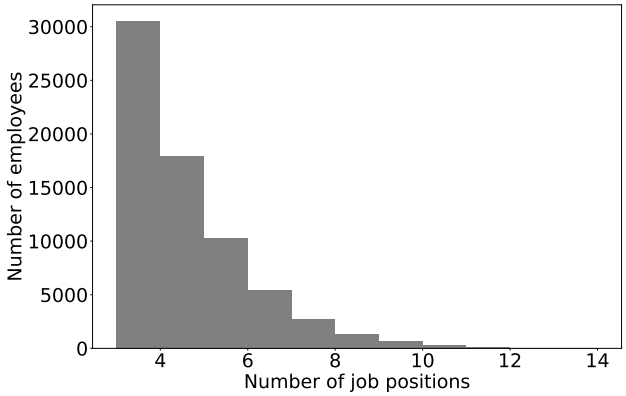


Fig. 4. The histogram of the number of job positions.

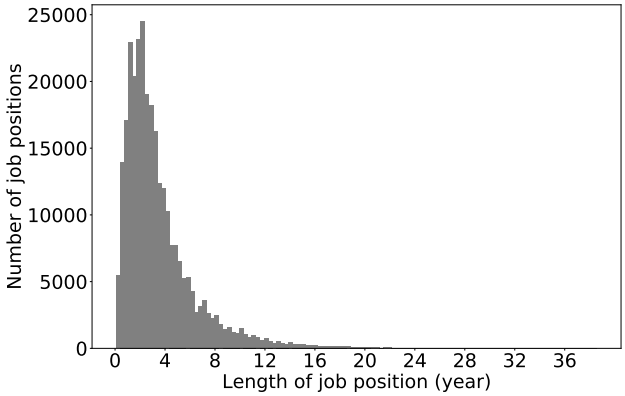


Fig. 5. The histogram of job position duration.

Let $\hat{r}(A, B, C)$ represent the estimated similarity relationship among three different jobs or career paths, and it is defined as:

$$\hat{r}(A,B,C) = \begin{cases} 1 & \text{if euclidean(A,B) < euclidean(A,C),} \\ 0 & \text{if euclidean(A,B) > euclidean(A,C),} \end{cases} \tag{14}$$

where $euclidean(A, B)$ means the Euclidean distance between embeddings of A and B. Thus, Accuracy is the fraction of correctly predicted tuples. It is defined as:

$$Accuracy = \frac{\sum_{(A,B,C)\in\mathcal{S}} I(\hat{r}(A,B,C) = r(A,B,C))}{|\mathcal{S}|}, \tag{15}$$

where $\mathcal{S}$ is the testing set for jobs or career paths, and $|\mathcal{S}|$ is the total number of samples in the testing set. $I(x)$ is an indicator function which equals 1 if x is true or equals 0 otherwise.

4.3 Baseline Methods

We compare our proposed JaEE model with the following baseline methods for both position similarity and employee similarity comparisons:

- SimCareers: It extracts manual-defined features of each pair of jobs and trains a logistic regression model to learn job level similarity. Then a sequence alignment algorithm is developed to measure the similarity between a pair of career paths [10]. We mimic SimCareers model with our data set except for logistic regression in job similarity learning since our setting lies on unsupervised learning. Instead, given a pair of jobs, we represent the pair-wise features with a vector, and the length of this vector is used as the job similarity score.
- Deepwalk: It is a famous graph embedding method that treats truncated random walks as sequences and borrows ideas of Skip-gram from Word2vec [15] to learn node embeddings of a graph [45]. We build a bipartite graph, where an employee node and a job node are connected if he or she worked with this job. Then Skip-gram is applied on random walk sequences to learn job node embeddings and employee node embeddings.
- Doc2vec: Originally, it extends Word2vec to learn word embeddings as well as document embeddings [46]. We deploy Doc2vec to learn job and employee embeddings, where career paths are analogous to a sequence of tokens.
- DBGE: By extending DBGE [12], we build a bipartite graph between employee nodes and job nodes, where an employee node is connected to a job node if he or she worked with this job. Then Horary Random Walk is used to generate node sequences [12], and Skip-gram is applied on those node sequences to learn job node embeddings and employee node embeddings.
- JaEE-RoBERTa: We replace fine-tuning RoBERTa with the pre-trained Word2vec from Google to measure the attribution of RoBERTa. An aggregated vector of job title tokens represents the job title.
- JaEE-JCL: We only keep *Job Content Embedding* and *Employee Embedding* to measure the attribution of *Job Context Learning*.

4.4 Parameter Setting

In the *Job Content Embedding*, we set the dimension of company embeddings D as ten, which is big enough to encode 860 companies. Because of our GPU capacity limitation, we fine-tune the RoBERTa base version rather than the large one. In the *Job Context Learning*, we sample $Q = 20$ jobs as negative positions every time. The embedding dimension is an important parameter of our model. We evaluate our JaEE model with four different dimensions $K = 50, 100, 200, 300$, respectively. With each dimension, we perform a random search with the validation set to find the best parameters. The search space consists of the following five hyper-parameters. The number of hidden layers num_layers in GRU is in the range $[1, 2, 3]$. The L_2 regularization β is selected from the range $[0.1, 0.2, 0.3, 0.4, 0.5]$. The weight λ is chosen in $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$. Because our training procedure includes fine-tuning the RoBERTa, we set up two learning rates for RoBERTa and other trainable parameters, respectively. The $bert\_rate$ for RoBERTa is chosen from the range $[10^{-7}, 10^{-6}, 5 \times 10^{-6}, 10^{-5}]$, and $learning\_rate$ for other trainable parameters is set in the range $[10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}, 0.01]$. We use mini-batch gradient descent to optimize our model, where the batch size is set as 16. The baselines also have their own search spaces for hyper-parameters tuning. All models are tuned to have the best performance. For example, the best parameters for JaEE with $K = 100$ are

$num\_layers = 1$, $beta = 0.2$, $\lambda = 0.5$, $bert\_rate = 10^{-6}$, $learning\_rate = 5 \times 10^{-5}$.

4.5 Results

The accuracy results are demonstrated in Table 2 for job similarity comparison and Table 3 for employee similarity comparison with varying sizes of embedding dimension K. There is no dimension parameter for SimCareers because it is a method with handcrafted features rather than embedding. As shown in the tables, our proposed JaEE model always outperforms other baselines. SimCareers performs poorly since it relies heavily on external knowledge systems such as similar companies and similar job titles. Thus, the accuracy results of SimCareers are 0.613 for job similarity and 0.570 for employee similarity comparisons without external knowledge. The performance of Deepwalk, Doc2vec and DBGE are limited, and the reason could be that they can not overcome messy job title problems without job title normalization. We find that fine-tuning RoBERTa has better performance than pre-trained word2vec in *Job Content Embedding* by comparing JaEE with JaEE-RoBERTa. JaEE-RoBERTa has the best performance with $K = 300$ because the vector size of pre-trained word2vec is 300, and we need to transform the pre-trained vectors for other sizes. Besides, *Job Context Learning* has significant attribution, and the accuracy results of JaEE-JCL reduces 7.6% for job similarity comparison and 14.4% for employee similarity comparison with $K = 100$.

TABLE 2. Accuracy Results for Job Similarity Comparison.

K	50	100	200	300
SimCareers	0.613			
Deepwalk	0.554	0.614	0.606	0.602
Doc2vec	0.628	0.678	0.641	0.617
DBGE	0.589	0.601	0.610	0.637
JaEE-RoBERTa	0.659	0.740	0.767	0.807
JaEE-JCL	0.735	0.774	0.762	0.769
JaEE	**0.778**	**0.838**	**0.813**	**0.826**

TABLE 3. Accuracy Results for Employee Similarity Comparison.

K	50	100	200	300
SimCareers	0.570			
Deepwalk	0.628	0.625	0.597	0.609
Doc2vec	0.542	0.549	0.539	0.568
DBGE	0.613	0.611	0.578	0.590
JaEE-RoBERTa	0.697	0.704	0.745	0.730
JaEE-JCL	0.632	0.668	0.666	0.663
JaEE	**0.747**	**0.780**	**0.797**	**0.761**

For job similarity comparison, we also compare the job similarity performance between job content embeddings $\mathcal{X}$ and target job embeddings $\mathcal{Y}$ in TABLE 4. The results of target job embeddings are much better than job content embeddings, which indicates that the combination of job content and transition information has more contribution than only job content information.

For employee similarity comparison, we also compare the employee similarity performance among employee embeddings $\mathcal{Z}$, job content embeddings $\mathcal{X}$, target job embeddings $\mathcal{Y}$ in TABLE 5. For each employee, beside the employee embedding, we also aggregate the embeddings of observed jobs in each career path using job content embeddings and target job embeddings to represent this employee, respectively. As we can see from the table, both the sum of job content embeddings and the sum of target job embeddings perform poorly for employee similarity comparison. It suggests that a simple summation of job embeddings could not represent an employee well, and the *Employee Embedding* component has significant attribution for employee representations.

TABLE 4. Accuracy Results for Job Similarity with $\mathcal{X}$, $\mathcal{Y}$.

K	50	100	200	300
$\mathcal{X}$	0.517	0.537	0.535	0.522
$\mathcal{Y}$	0.778	0.838	0.813	0.826

TABLE 5. Accuracy Results for Employee Similarity with $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$.

K	50	100	200	300
$\mathcal{X}$	0.649	0.609	0.623	0.592
$\mathcal{Y}$	0.644	0.613	0.597	0.623
$\mathcal{Z}$	0.747	0.780	0.797	0.761

In addition, we visualize the learned job representations $\mathcal{Y}$ in Fig. 6. There is no category information in our data. Thus we select three groups of job embeddings including engineer, sales and consultant by searching the keywords "engineer", "sale/sales" and "consulting/consultancy/consultant" in job titles, respectively. We randomly sample 1000 jobs for each group and use t-SNE [47] to project selected job embeddings ($K = 100$) to two-dimensional vectors. From Fig. 6, we see that engineer jobs and sales jobs are well separated. The overlap between consultant jobs and engineer jobs is considerable because many consultant jobs provide technical consultancy services and also require several years of engineering experience.

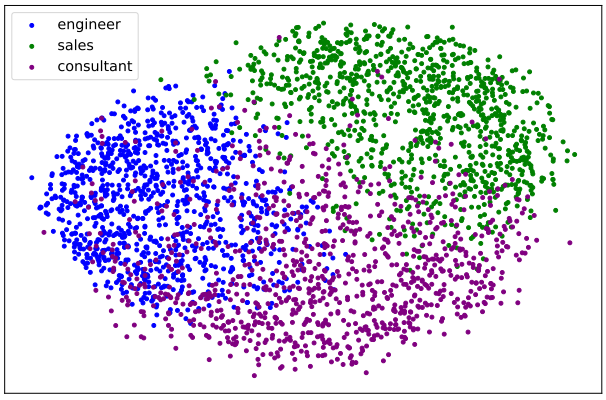


Fig. 6. Visualization of learned embeddings for engineer, sales and consultant jobs.

4.6 Case Studies

We find some interesting cases based on our embeddings $\mathcal{Y}$ and $\mathcal{Z}$ with $K = 100$. In this section, we demonstrate multiple cases for top similar jobs in Table 6, top dissimilar jobs in Table 7, cases for messy job titles in Table 8 and top similar career paths in Table 9.

People usually compare job similarity/dissimilarity from perspectives such as company, job function, job seniority. With our embeddings, we find seven jobs and their top three similar jobs shown in Table 6. The top similar jobs are ranked by similarity, which is calculated via Euclidean distance. The first four jobs are examples for software engineers. For the first job *Software Engineer at Google*, itself and its top similar jobs *Software Engineer at Facebook* and *Software Development Engineer at Amazon* are among the most competitive technical jobs in IT field. The second similar job *Senior Associate at Cognizant* usually requires several years of software development experience though the company *Cognizant* is not as famous as *Google*. For the second job *Software Design Engineer at Microsoft*, *Software Design Engineer* is a specific case of *Software Engineer*, but it is very difficult for machines to merge them. The job titles of the top three similar jobs are *Software Engineer* though the companies *Accenture* and *Tech Mahindra* are not well-known as *Microsoft*. For the third job *Senior Engineer at Intel*, *Senior Engineer* has higher seniority than entry level *Engineer*. The top three similar jobs have seniority as *Senior* or *Staff* level, and the companies *Google* and *IBM* have a great reputation as *Intel*. Note that the career trajectory of *Software Engineer* in *IBM* includes *Software Engineer*, *Staff Software Engineer*, *Advisory Software Engineer*, *Senior Software Engineer* and so on. Thus, both *Staff*, *Advisory* and *Senior* levels in IBM are usually equivalent to *Senior* level in other companies. The fourth job *Software Engineer at Wipro* is an example for small IT companies while the first three example jobs are from big IT companies. The fifth job *Senior System Engineer at Infosys* is an interesting example of *System Engineer*. Our model can find out similar *System Engineer* jobs at similar employers. The third similar job *Senior Systems Engineer at Infosys Limited* has a typo with *Systems*. It indicates potential capacity of our model for correcting job titles. The sixth job *Consultant at Accenture* is a case for consulting positions. It usually requires several years of experience in software/system engineering. Thus, it is reasonable that the second and third similar jobs are *Software Engineer* positions. The seventh job *Retail Sales Representative at T-mobile* is an example of sales related positions. The top three similar jobs are from *Verizon*. Both *T-mobile* and *Verizon* are top telecommunication companies in US. And the most similar job also matches the job title *Retail Sales Representative* exactly.

For the seven example jobs demonstrated in Table 6, their top three dissimilar jobs are shown in Table 7. Generally, most top dissimilar jobs are much different from the example jobs from job functions.

As discussed in Section 1, messy job titles are very challenging to process in job embedding. We demonstrate three interesting cases in Table 8. The top similar jobs are also ranked by similarity via Euclidean distance. A tough challenge is employees may describe jobs with abbreviated job titles and how to match an abbreviated job title with the full job title. For example, *SDE* stands for *Software Development Engineer*, *SDET* represents *Software Development Engineer in Test* and *SE* is the abbreviation of *Software Engineer*. For the first and second jobs in Table 8, we see that the exactly matched jobs (full job title and company) are ranked in the top ten by our model. For the third job *SE at Amazon*, we find *Software Development Engineer II at Amazon* in the twentieth place and *Software Development Engineer at Amazon* in the twenty-third place. Another big challenge is that employees may add auxiliary information to job titles. The second job *SDE, Microsoft Azure. at Microsoft* is a typical example. We see that *Microsoft Azure.* does not influence our model much to rank *Software Development Engineer II at Microsoft* and *Software Design Engineer at Microsoft* in the top five.

TABLE 6. Example Jobs for Most Similar Jobs.

	Jobs (job title at company)
1	Software Engineer at Google
	Software Engineer at Facebook
	Senior Associate at Cognizant
	Software Development Engineer at Amazon
2	Software Design Engineer at Microsoft
	Software Engineer at Accenture
	Software Engineer at Tech Mahindra
	Software Engineer at Intel
3	Senior Engineer at Intel
	Senior Software Engineer at Google
	Senior Software Engineer at IBM
	Staff Software Engineer at IBM
4	Software Engineer at Wipro
	Project Engineer at Wipro
	Software Engineer at Infosys
	Programmer Analyst at Cognizant
5	Senior System Engineer at Infosys
	System Engineer at Infosys
	Senior Software Engineer at Wipro
	Senior Systems Engineer at Infosys
6	Consultant at Accenture
	Consultant at Capgemini
	Senior Software Engineer at Infosys
	Software Engineer at IBM
7	Retail Sales Representative at T-mobile
	Retail Sales Representative at Verizon
	General Manager at Verizon
	Solutions Manager at Verizon

We demonstrate three example career paths of employees and their top three similar career paths in Table 9. In some career paths, there are two consecutive job positions that are the same except for the job duration. The possible reason is that the employee may switch teams or departments internally within the same company. The first example employee is a case for career advancement within one company. The employee worked at *IBM* in four job positions. The top three similar employees also worked at *IBM* with four jobs. They have the same job title *Senior Software Engineer* for current jobs. The second example employee is an instance of switching jobs among different companies. Her top three similar employees also switched among different employers and currently are working as *Software Engineer* at *Google*. The third example employee is a case for consulting job. She switched from a development-related job to a consulting job. The top similar employees also have similar career advancements. We find that the top similar employees and the example employee are working at the same job position, and the most recent job position has more impact on the employee embedding. Because the recent job position is closer to the GRU final output in the encoder of *Employee Embedding*, and it contributes more to the employee vector.

The career path of the first example employee in Table 9 is classic career advancement for *Software Engineer* at *IBM*.

TABLE 7. Job Cases for Most Dissimilar Jobs.

	Jobs (job title at company)	Top 3 dissimilar jobs (job title at company)
1	Software Engineer at Google	Lead Project Manager at Verizon Senior Analyst / Tier 1& 2 (Team Lead) – Switch Control / Provisioning Support at AT&T Lead Product Manager - Multiphysics and Aim at Ansys
2	Software Design Engineer at Microsoft	Vice President, Institutional Sales at FactSet Research Systems Lead Project Manager at Verizon Service Delivery Manager, Hosted/Cloud Services at Telefonaktiebolaget LM Ericsson
3	Senior Engineer at Intel	Field Action Project Lead at Medtronic SAP HR Payroll Administrator at Accenture Quality Assurance Senior Analyst – Inventory Control at Dell
4	Software Engineer at Wipro	System Administrator at ManTech International Vice President, Institutional Sales at FactSet Research Systems Application Analyst/Programmer at Hitachi Data Systems
5	Senior System Engineer at Infosys	Senior Analyst / Tier 1& 2 (Team Lead) – Switch Control / Provisioning Support at AT&T Market Analyst at Groupon Sales & Market Development Representative at Oracle
6	Consultant at Accenture	Lead Project Manager at Verizon Vice President, Human Resources at Comcast Area Manager - Operations at Verizon
7	Retail Sales Representative at T-mobile	Senior Software Engineer at Square Senior-Systems Administrator at AT&T Infrastructure Services Manager at Liberty Mutual

We draw the 100-dimensional vectors of those four jobs in Fig. 7. Each row represents a job. We see that the colors of many dimensions are deeper and deeper from *Software Engineer* to *Senior Software Engineer*. It indicates the embeddings could illustrate the process of knowledge accumulation in career advancement.

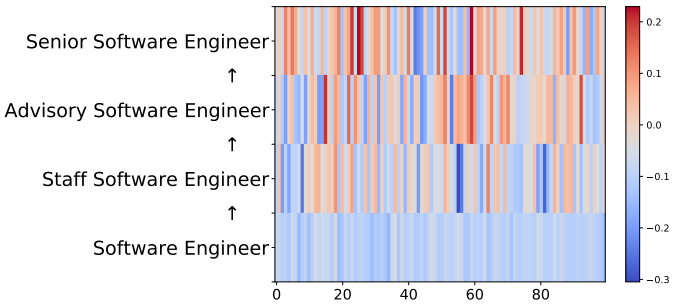


Fig. 7. Visualization of job embeddings of an example employee.

5 Conclusion

In this paper, we proposed a novel deep learning model named Job and Employee Embedding (JaEE) for jointly learning representations of jobs and employees. The learned job embeddings and employee embeddings are expected to effectively represent the characteristics of jobs and career paths in the talent market and benefit many talent analytics tasks such as job and employee similarity searches and recommendations. In order to exploit job content and transition information of massive career data that is prevalently available on many job search websites (e.g., LinkedIn and Glassdoor), we designed novel neural networks with three components *Job Content Embedding*, *Job Context Learning* and *Employee Embedding*, which naturally model career path data from three levels of granularity including job content, job context and job sequence. *Job Content Embedding* learns the content representation of a job position by modeling its company, job title and description. *Job Context Learning* learns the representation of a job by accumulating its prior job positions in individual career paths. *Employee Embedding* learns the representation of a career path using an RNN encoder-decoder model. Extensive experimental results with a real-world talent career dataset demonstrate the superiority of our proposed method against other state-of-the-art methods.

Acknowledgments

This research was partially supported by the National Science Foundation (NSF) via grant numbers 2007175, 1844983 and 2007437.

References

[1] V. S. Dave, B. Zhang, M. Al Hasan, K. AlJadda, and M. Korayem, "A Combined Representation Learning Approach for Better Job and Skill Recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, (New York, NY, USA), pp. 1997–2005, ACM, 2018.

[2] K. Kenthapadi, B. Le, and G. Venkataraman, "Personalized job recommendation system at linkedin: Practical challenges and lessons learned," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 346–347, 2017.

[3] S. T. Al-Otaibi, "A survey of job recommender systems," *International Journal of the Physical Sciences*, vol. 7, July 2012.

[4] L. Li, H. Jing, H. Tong, J. Yang, Q. He, and B.-C. Chen, "NEMO: Next Career Move Prediction with Contextual Embedding," in *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, (Perth, Australia), pp. 505–513, International World Wide Web Conferences Steering Committee, 2017.

[5] R. W. Lent, "Career-Life Preparedness: Revisiting Career Planning and Adjustment in the New Workplace," *The Career Development Quarterly*, vol. 61, pp. 2–14, Mar. 2013.

[6] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *arXiv:1206.5538 [cs]*, Apr. 2014.

[7] S. Bian, X. Chen, W. X. Zhao, K. Zhou, Y. Hou, Y. Song, T. Zhang, and J.-R. Wen, "Learning to Match Jobs with Resumes from Sparse Interaction Data using Multi-View Co-Teaching Network," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, (New York, NY, USA), pp. 65–74, Association for Computing Machinery, Oct. 2020.

TABLE 8. Example Jobs for Messy Job Titles.

	Jobs (job title at company)
1	SDET at Microsoft
	Project Manager at Infosys Senior System Engineer at Infosys System engineer at Infosys Senior Software Engineer at Wipro Manager at Accenture Software Development Engineer at Microsoft **Software Development Engineer in Test at Microsoft** Software Engineer II at Microsoft Consultant at Capgemini Software Engineer at Wipro
2	SDE, Microsoft Azure. at Microsoft
	Consultant at Infosys Analyst at Accenture **Software Development Engineer II at Microsoft** **Software Design Engineer at Microsoft** Staff Software Engineer at IBM Component Design Engineer at Intel Project Manager at Infosys Software Development Engineer II at Amazon Senior Program Manager at Microsoft System Engineer at Tata Consultancy Services
3	SE at Amazon
	Senior Software Engineer at Cisco Systems Senior Software Engineer at Google Project Manager at Wipro Assistant Consultant at Tata Consultancy Services Technology Lead at Infosys IT Analyst at Tata Consultancy Services Senior Software Engineer at Microsoft Programmer Analyst at Infosys Senior Software Engineer at Accenture Senior Associate at Cognizant Software Engineer at Google Program Manager at Microsoft Component Design Engineer at Intel Technical Lead at Wipro Software Engineer at Facebook Software Engineer at Cisco Systems Software Engineer II at Microsoft Senior Consultant at Capgemini Technology Analyst at Infosys **Software Development Engineer II at Amazon** Consultant at Accenture Software Engineer at Microsoft **Software Development Engineer at Amazon** Senior Systems Engineer at Infosys Software Development Engineer in Test at Microsoft

[8] C. Zhu, H. Zhu, H. Xiong, C. Ma, F. Xie, P. Ding, and P. Li, "Person-Job Fit: Adapting the Right Talent for the Right Job with Joint Representation Learning," *ACM Transactions on Management Information Systems*, vol. 9, pp. 12:1–12:17, Sept. 2018.

[9] M. Liu, J. Wang, K. Abdelfatah, and M. Korayem, "Tripartite Vector Representations for Better Job Recommendation," *arXiv:1907.12379 [cs]*, July 2019.

[10] Y. Xu, Z. Li, A. Gupta, A. Bugdayci, and A. Bhasin, "Modeling professional similarity by mining professional career trajectories," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1945–1954, ACM, 2014.

[11] D. Zhang, J. Liu, H. Zhu, Y. Liu, L. Wang, P. Wang, and H. Xiong, "Job2Vec: Job Title Benchmarking with Collective Multi-View Representation Learning," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, (New York, NY, USA), pp. 2763–2771, Association for Computing Machinery, Nov. 2019.

[12] X. Cai, J. Shang, Z. Jin, F. Liu, B. Qiang, W. Xie, and L. Zhao, "DBGE: Employee Turnover Prediction Based on Dynamic Bipartite Graph Embedding," *IEEE Access*, vol. 8, pp. 10390–10402, 2020.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, pp. 1097–1105, Curran Associates, Inc., 2012.

[14] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *arXiv:1408.5882 [cs]*, Aug. 2014.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, pp. 3111–3119, 2013.

[16] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, and C. Zhang, "BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (Virtual Event CA USA), pp. 1054–1064, ACM, Aug. 2020.

[17] H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and T. Zhao, "SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 2177–2190, Association for Computational Linguistics, 2020.

[18] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-Art Natural Language Processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 38–45, Association for Computational Linguistics, Oct. 2020.

[19] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[20] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, Oct. 2018.

[22] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," *arXiv:1910.01108 [cs]*, 2019.

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv:1907.11692 [cs]*, July 2019.

[24] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[25] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-Efficient Transfer Learning for NLP," Feb. 2019.

[26] S. Golovanov, R. Kurbanov, S. Nikolenko, K. Truskovskyi, A. Tselousov, and T. Wolf, "Large-Scale Transfer Learning for Natural Language Generation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 6053–6058, Association for Computational Linguistics, 2019.

[27] T. Wolf, V. Sanh, J. Chaumond, and C. Delangue, "TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents," Jan. 2019.

[28] A. Raganato and J. Tiedemann, "An Analysis of Encoder Representations in Transformer-Based Machine Translation," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (Brussels, Belgium), pp. 287–297, Association for Computational Linguistics, 2018.

[29] Y. J. Choe, J. Ham, K. Park, and Y. Yoon, "A Neural Grammatical Error Correction System Built On Better Pre-training and Sequential Transfer Learning," *arXiv:1907.01256 [cs]*, July 2019.

[30] S. M. Lakew, M. Cettolo, and M. Federico, "A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation," *arXiv:1806.06957 [cs]*, June 2018.

[31] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[32] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, "Race: Large-scale reading comprehension dataset from examinations," *arXiv preprint arXiv:1704.04683*, 2017.

[33] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

TABLE 9. Example Employees and Most Similar Employees.

	Career paths
1	Software Engineer at IBM April 2007 - July 2009 ⇒ Staff Software Engineer at IBM July 2009 - June 2013 ⇒ Advisory Software Engineer at IBM July 2013 - April 2016 ⇒ Senior Software Engineer at IBM April 2016 - Present
	Software Engineer at IBM June 1999 - April 2001 ⇒ Staff Software Engineer at IBM May 2001 - June 2007 ⇒ Advisory Software Engineer at IBM July 2007 - October 2013 ⇒ Senior Software Engineer at IBM October 2013 - Present
	Staff Software Engineer at IBM July 1999 - July 2003 ⇒ Advisory Software Engineer at IBM July 2003 - July 2010 ⇒ Senior Software Engineer at IBM July 2010 - November 2011 ⇒ Senior Software Engineer at IBM November 2011 - Present
	Software Engineer at IBM June 1999 - June 2003 ⇒ Staff Software Engineer at IBM June 2003 - June 2006 ⇒ Advisory Software Engineer at IBM June 2006 - October 2011 ⇒ Senior Software Engineer at IBM October 2011 - Present
2	Software Development Engineer in Test at Microsoft October 2006 - October 2008 ⇒ Software Development Engineer at Microsoft November 2008 - January 2012 ⇒ Software Engineer at Facebook January 2012 - July 2016 ⇒ Software Engineer at Google September 2016 - Present
	Software Development Engineer at Microsoft June 2011 - July 2012 ⇒ Software Engineer at Facebook July 2012 - October 2012 ⇒ Software Engineer at Facebook October 2012 - November 2013 ⇒ Software Engineer at Google December 2013 - Present
	Software Development Engineer in Test at Microsoft October 2011 - August 2014 ⇒ Software Development Engineer at Microsoft August 2014 - January 2015 ⇒ Software Engineer at Bloomberg January 2015 - October 2015 ⇒ Software Engineer at Google October 2015 - Present
	Engineer at Qualcomm August 2008 - March 2011 ⇒ Senior Engineer at Qualcomm April 2011 - August 2011 ⇒ Software Development Engineer at Amazon August 2011 - March 2016 ⇒ Software Engineer at Google March 2016 - Present
3	Programmer Analyst at Cognizant December 2006 - October 2010 ⇒ Associate at Cognizant October 2010 - November 2012 ⇒ Consultant at Infosys December 2012 - March 2014 ⇒ Senior Consultant at Infosys April 2014 - Present
	Technology Analyst at Infosys June 2006 - April 2010 ⇒ Senior Associate Consultant at Infosys June 2011 - March 2014 ⇒ Consultant at Infosys April 2014 - March 2016 ⇒ Senior Consultant at Infosys April 2016 - Present
	Programmer Analyst at Cognizant August 2005 - April 2008 ⇒ Programmer Analyst at Infosys May 2008 - October 2010 ⇒ Technology Lead at Infosys October 2010 - December 2012 ⇒ Senior Consultant at Infosys January 2013 - Present
	Software Engineer at Infosys June 2006 - January 2008 ⇒ Technology Analyst at Infosys June 2008 - December 2012 ⇒ Consultant at Infosys January 2013 - June 2014 ⇒ Senior consultant at Infosys June 2014 - Present

[34] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[35] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215 [cs]*, Sept. 2014.

[37] M.-T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," *arXiv:1508.04025 [cs]*, Aug. 2015.

[38] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv:1406.1078 [cs, stat]*, June 2014.

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv:1409.0473 [cs, stat]*, Sept. 2014.

[40] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A Comparison of Sequence-to-Sequence Models for Speech Recognition," in *Interspeech 2017*, pp. 939–943, ISCA, Aug. 2017.

[41] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold Fusion: Training Seq2Seq Models Together with Language Models," *arXiv:1708.06426 [cs]*, Aug. 2017.

[42] Y. Zhang, Y. Wang, J. Liao, and W. Xiao, "A Hierarchical Attention Seq2seq Model with CopyNet for Text Summarization," in *2018 International Conference on Robots & Intelligent System (ICRIS)*, (Changsha, China), pp. 316–320, IEEE, May 2018.

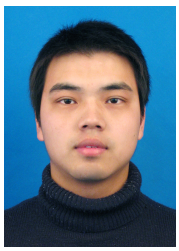
[43] Y. Zhang, D. Li, Y. Wang, Y. Fang, and W. Xiao, "Abstract Text Summarization with a Convolutional Seq2seq Model," *Applied Sciences*, vol. 9, p. 1665, Apr. 2019.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[45] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14*, (New York, New York, USA), pp. 701–710, ACM Press, 2014.

[46] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196, 2014.

[47] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE.," *Journal of machine learning research*, vol. 9, no. 11, 2008.



Hao Liu received his B.E. degree in Computer Science and Technology from the University of Science and Technology of China (USTC) in 2013, the M.S. degree in Computer Science from the University of North Carolina at Charlotte in 2017. He is currently a Ph.D. student in Management Information Systems at University of Arizona. His research interests include data mining, machine learning and recommender systems.

Yong Ge (SM'20) received his Ph.D. in Information Technology from Rutgers, The State University of New Jersey in 2013, the M.S. degree in Signal and Information Processing from the University of Science and Technology of China (USTC) in 2008, and the B.E. degree in Information Engineering from Xi'an Jiao Tong University in 2005. He is currently an Assistant Professor at University of Arizona. His research interests include data mining, machine learning and recommender systems.