Job2Vec: Job Title Benchmarking with Collective Multi-View Representation Learning

Denghui Zhang[1], Junming Liu[1], Hengshu Zhu[2*], Yanchi Liu[1]
Lichen Wang[3], Pengyang Wang[4], Hui Xiong[1,2*]
[1]Magagement Science and Information Technology Department, Rutgers University, USA
[2]Baidu Talent Intelligent Center, Baidu Inc, China
[3]Electrical and Computer Engineering Department, Northeastern University, USA
[4]Computer Science Department, University of Central Florida, USA

ABSTRACT

Job Title Benchmarking (JTB) aims at matching job titles with similar expertise levels across various companies. JTB could provide precise guidance and considerable convenience for both talent recruitment and job seekers for position and salary calibration/prediction. Traditional JTB approaches mainly rely on manual market surveys, which is expensive and labor intensive. Recently, the rapid development of Online Professional graph has accumulated a large number of talent career records, which provides a promising trend for data-driven solutions. However, it is still a challenging task since (1) the job title and job transition (job-hopping) data is messy which contains a lot of subjective and non-standard naming conventions for a same position (*e.g.*, *Programmer*, *Software Development Engineer*, *SDE*, *Implementation Engineer*), (2) there is a large amount of missing title/transition information, and (3) one talent only seeks limited numbers of jobs which brings the incompleteness and randomness for modeling job transition patterns. To overcome these challenges, we aggregate all the records to construct a large-scale Job Title Benchmarking Graph (Job-Graph), where nodes denote job titles affiliated with specific companies and links denote the correlations between jobs. We reformulate the JTB as the task of link prediction over the Job-Graph that matched job titles should have links. Along this line, we propose a collective multi-view representation learning method (Job2Vec) by examining the Job-Graph jointly in (1) graph topology view (the structure of relationships among job titles), (2) semantic view (semantic meaning of job descriptions), (3) job transition balance view (the numbers of bidirectional transitions between two similar-level jobs are close), and (4) job transition duration view (the shorter the average duration of transitions is, the more similar the job titles are). We fuse the multi-view representations in the encode-decode paradigm to obtain an unified optimal representations for the task of link prediction. Finally, we conduct extensive experiments to validate the effectiveness of our proposed method.

*Hui Xiong and Hengshu Zhu are corresponding authors. This work is supported by NSFC 91746301. The code is available at: https://github.com/zdh2292390/Job2Vec-Job-Title-Benchmarkingwith-Collective-Multi-View-Representation-Learning

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357825

CCS CONCEPTS

• **Information systems** → *Information systems applications*; *Web mining*.

KEYWORDS

Talent Intelligence, Job Title Benchmarking, Multi-view learning, Auto-encoder, Representation Learning

ACM Reference Format:
Denghui Zhang[1], Junming Liu[1], Hengshu Zhu[2*], Yanchi Liu[1] and Lichen Wang[3], Pengyang Wang[4], Hui Xiong[1,2]. 2019. Job2Vec: Job Title Benchmarking with Collective Multi-View Representation Learning. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357825

1 INTRODUCTION

Recent years have witnessed the increasing popularity of using data mining techniques for addressing human resource management (HRM) tasks (*e.g.*, intelligent job-person fit and intelligent interview assessment [14, 15]). However, few research efforts have been made on intelligent **J**ob **T**itle **B**enchmarking (JTB), which aims at matching job titles with similar expertise levels across various companies. For both job seekers and employers, JTB is important for talent recruitment and salary calibration. With appropriate JTB insights, employers can recruit relevant talent with the right title and salary. While for job seekers, JTB can provide guidance for their career development. In this paper, we study the problem of JTB from the data mining persective.

Traditional JTB relies heavily on manual market surveys, which is expensive and labor intensive. Recently, the emergence of Online Professional graph (e.g., Linkedin) helps to accumulate a large number of career records, which provides an unparalleled opportunity for a data-driven solution. However, JTB is still a challenging task due to the following three aspects. First, the job title and job transition (job-hopping) data is messy which contains a lot of subjective and non-standard naming conventions for the same position. For example, as shown in Figure 1, Software Engineer, SDE, Software Development Engineer, and Computer Programmer are the same-level jobs across different IT companies. Second, there is a large amount of missing title/transition information. Many users on Online Professional graph do not update their information in time. Too much missing information will hinder the applicability of data mining algorithms. Third, in individual career, one talent only seeks limited numbers of jobs compared to the total set of

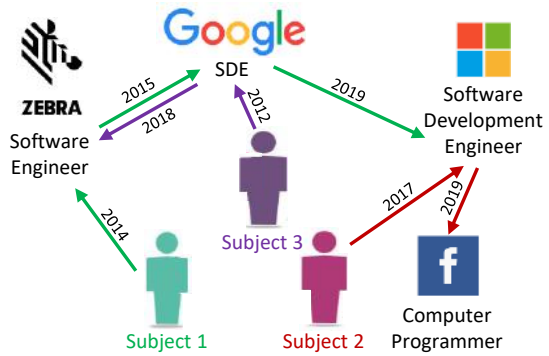


Figure 1: Job transitions of different subjects across companies and titles. Our approach aims to explore the multiple clues for high performance on job title benchmarking task.

job titles on the job market, which brings the incompleteness and randomness for modeling job transition patterns.

To tackle these challenges, we propose to construct a **J**ob **T**itle **B**enchmarking **G**raph (Job-Graph), where nodes denote job titles affiliated with the specific companies and links denote the numbers of transition between job titles. We hold the assumption that the bechmarked job title pairs should have strong correlations that there exists links between the job titles. Along this line, we reformulate JTB as the task of the link prediction over the Job-Graph.

Representation learning methods achieve outstanding performances for the link prediction task [24, 30]. However, due to the three unique properties of Job-Graph (the topology structure, rich semantic information of job titles, and job transition patterns), existing representation learning methods are unable to model these properties at the same time. Therefore, we propose a collective multi-view representation learning method to learn the representations of job titles for the task of link prediction.

Specifically, first, we model four views of representations: (1) **Graph Structure View**, which refers to the topology structure of the Job-Graph that encodes the graph structure and neighborhood information; (2) **Semantic View**, which refers to the semantic meaning of job titles; (3) **Job Transition Balance View**, which is compliant with the observation that the numbers of bidirectional transitions between two similar-level jobs are close; (4) **Job Transition Duration View**, which reveals the fact that the shorter the average duration of transitions is, the more similar the job titles are. Then, to obtain an unified representation, we design a representation fusion process based on the encode-decode paradigm. The multi-view representations are fed into the associated multi-layer perceptrons which are attached behind with a representation ensemble layer to work as an encoder. The ensembled representation is dispatched to the corresponding decoder by a representation dispatching layer to reconstruct the multi-view representations. The loss between the original and reconstructed multi-view representations will be minimized to guarantee the optimal unified representation. Moreover, we train the multi-view representation learning procedure and the representation fusion procedure in an alternative way. The loss from four views and the representation fusion procedure will be minimized simultaneously to generate high quality representation for job titles for the task of link prediction.

Table 1: An example of job transitions.

Job Title	Company	Duration
Production Engineer	Square Inc	2011/7-2016/10
Senior Site Reliability Engineer	Google	2010/10-2011/7
Architect	Yahoo!	2009/7-2010/6
Systems Engineer	Yahoo!	2006/6-2009/7
Systems Engineer	IBM	2006/2-2006/6

Table 2: Statistics of Job-Graphs in IT and Finance.

Dataset	IT	Finance
# Edges	39927	77118
# Nodes	44030	89851
Average out degree	0.91	0.86

In summary, in this paper, we propose a data-driven solution for the problem of JTB. Specifically, we first construct the Job Title Benchmarking Graph based on the job transition records. Then, we reformulate the problem of JTB as the task of link prediction. We propose a collective multi-view representation learning method, by learning an unified job title representation from the graph structure view, semantic view, job transition balance view, and job transition duration view. Finally, we conduct extensive experiment to evaluate our proposed method over the real-world dataset. The promising results validate the effecitveness of our proposed method.

2 PRELIMINARIES

In this section, we first briefly introduce the real-world dataset we collected for JTB task. Then, we introduce some essential definitions. Followed the definitions, we propose our problem statement. Finally, we present an overview of the propposed framework.

2.1 Data Description

In this study, we analyze real-world talent job title transition data, collected from a major commercial Online Professional Network. The data includes two main categories, IT-related and Finance-related job titles. Table 2 shows Job-Graphs constructed from these two datasets are very sparse.

Table 1 presents an example of job transition records from an individual talent. Each line consists of a job title, company name and the duration holding this position.

2.2 Definition

Here, we introduce some essential definitions, which will be used throughout this paper.

Definition 2.1. **Job Title Bechmarking (JTB)** JTB is a process that matches job titles with similar expertise levels across various companies. Formally, given two job title-company pair, $(Title_i, Company_i)$ and $(Title_j, Company_j)$, the objective is determine whether the given paired job titles are on the same level. JTB could provide precise guidance and considerable convenience for both talent recruitment and job seekers for position and salary calibration/prediction.

Definition 2.2. **Job Title Transition Graph (Job-Graph).** Job-Graph is defined as a directed graph $G = (V, E)$, where each node $v_i \in |V|$ represents a job title affiliated a company(i.e., $v_i = (Title_i, Company_i)$), and each link $e_{ij} \in |E|$ between two nodes v_i

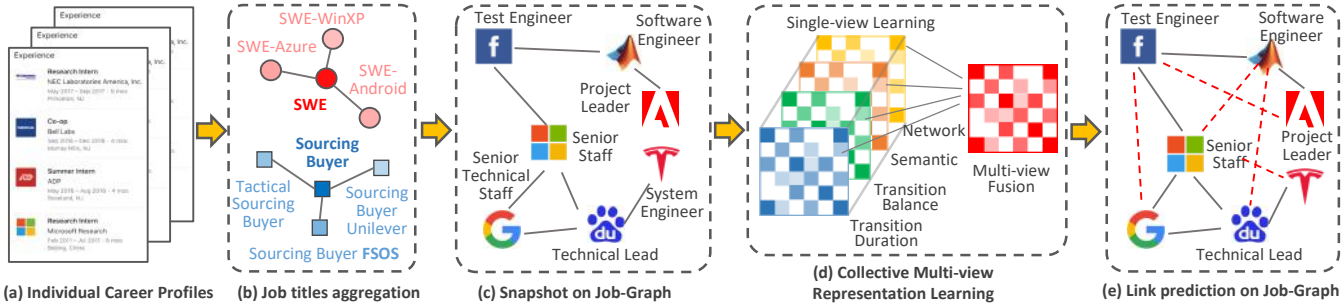


Figure 2: Overview of our job title benchmarking framework.

and v_j indicates that there exist job transitions from the $(Title_i, Company_i)$ to $(Title_j, Company_j)$, and the weight of edge e_{ij} represents the number of transitions observed from $(Title_i, Company_i)$ to $(Title_j, Company_j)$

2.3 Problem Statement

In this paper, we study the problem of Job Title Benchmarking (JTB). We first construct job title transition graph to depict the job transition patterns. We formulate the JTB as the the task of link prediction over the Job-graph, based on the assumption that similar-level job titles should have strong correlations to enable a link between them. To enable the link prediction task, we push forward the problem formulation to the representation learning over the Job-Graph to learn unified and optimal representations for job titles.

Formally, given the Job-Graph $G = (V, E)$, we aim to find a mapping function $f : \mathbf{v} \rightarrow \mathbf{z}$ that takes node (job title) $\mathbf{v}$ as the input, and outputs the vectorized representation $\mathbf{z}$ of the job titles, while preserving properties of the Job-Graph and job transition patterns. The generated node (job title) representation $\mathbf{v}$ is then utilized to solve the problem of link prediction.

2.4 Framework Overview

Figure 2 shows an overview of our proposed framework that includes the following essential tasks: (i) constructing the job title transition graph; (ii) developing a collective multi-view representation learning method for learning job title representations; (iii) applying learned job title representations for link prediction on Job-Graph. In the first task, given job transition records of talents, we construct a job title transition graph. In the second task, a collective multi-view representation learning method is developed for jointly modeling the graph structure, semantic meaning of job titles and job transition patterns. In the last task, we apply our proposed method to learn job title representations for link prediction on Job-Graph to benchmark job titles.

3 JOB-GRAPH CONSTRUCTION AND REFINEMENT

In this section, we show how to construct the Job-Graph. Intuitively, the Job-Graph can be directly constructed from the raw job tranistion record data. However, messy, noisy and non-standard name convention of job titles makes the Job-Graph extremely sparse and redundant, which hinders the further analysis. Therefore, we refine the Job-Graph into an applicable fashion in the following steps: (1) Extract job transitions from the raw career records; (2) Map and aggregate all the transitions into the Job-Graph, where each node represents a job title, each edge represents the number of transitions between the nodes.

3.1 Job Transition Extraction

As shown in Table 1, each transition consists of a source job and destination job, i.e., (job_{src}, job_{des}). We first set all the job titles existed in the raw data as nodes. Then we sum the transition frequencies as the link from job_{src} to job_{des}. The constructed graph is set as the base graph for the further refinement.

3.2 Job Title Aggregation

Generally speaking, job titles usually consist of three parts:

(1) **Title level**, such as *Senior*, *Principle* and *Director*.
(2) **Title core function**, such as Software Engineer, Product Manager.
(3) **Unique additional information**, such as Software Engineer *in Bid Data*, Sales Rep *on Small and Medium businesses*.

After we study the word frequencies of job titles, there is an interesting observation that the word frequency distribution subjects to *power law distribution*, as shown in Figure 3. It also can be observed that noise words or those additional user-related words usually appear in the long tail. We also show the top 10 frequent words and bottom 10 frequent words. It is obvious that the most frequent words like *manager* and *engineer* usually describe the core function of job titles, while the less frequent words looks more like user's unique information. With this observation, we aggregate job titles by filtering out low frequency words in job titles and thus get a normalized and denser Job-Graph. Specifically, in this work, we filtered out the words that have a frequency lower than 30.

Table 3 shows three real examples of aggregated job titles by filtering out low frequency words. Words in bold indicate the low-frequency words. For example, *"Tactical Sourcing Buyer (Unilever)"* and *"Sourcing Buyer, MARCOM & FSOS"* are originally thought to be different, after filtering low-frequency words, they are aggregated to the same title *"Sourcing Buyer"*. To be noted, the reason we use filtering low-frequency words instead of using cluster algorithm to cluster the job titles is that there are no standard target classes for job titles, and it is hard to decide the number of clusters for job titles if cluster algorithm is applied.

With this aggregated Job-Graph, we can obtain some concise job title matching insights like: A Senior Software Engineer of

Table 3: Examples of aggregating job titles.

Original Job Titles	Aggregated Job Title
Tactical Sourcing Buyer **(Unilever)**	Sourcing Buyer
Sourcing Buyer, **MARCOM & FSOS**	
Software Design Engineer-**(Azure)**	Software Design Engineer
Software Design Engineer-**WindowsXP**	
Software Design Engineer-**(Contracting) Encarta**	
Cyber Security Architect	Security Architect
Security Architect	

LinkedIn can match a Software Engineer of Google since most Senior Software Engineers of LinkedIn obtained the title of Software Engineer when they just made a career transition to Google. However, the sparsity issue of Job-Graph still limits the performance of traditional representation learning method. Therefore, in next section, we introduce our collective multi-view representation learning method, Job2Vec, and show how to perform link prediction to enrich Job-Graph based on the proposed method.

4 JOB2VEC: COLLECTIVE MULTI-VIEW REPRESENTATION LEARNING FOR JOB TITLES

In this section, we introduce our collective multi-view representation learning methods fore job titles.

4.1 Model Intuition

We learn representations of job titles with the following intuitions.

Intuition 1: Topology Structure Preservation. Job-Graph is built to depict job transitions on the job market. The topology structure of Job-Graph reveal the connectivity and neighbor information of job titles which can help to describe the latent structures among job titles. We should preserve the topology structure of job-graph in the representation learning.

Intuition 2: Semantics Preservation. Job titles contain rich semantic descriptions which can further enhance the quality of job title representation. Therefore, we should preserve the semantic meanings of job titles.

Intuition 3: Job Transition Patterns Preservation. Job transitions have unique latent patterns. Transitions among different job title pairs are also different. Consequently, we should preserve the job transition patterns.

Therefore, we model the job title representations in multi views. Specifically, we introduce *graph topology view* for Intuition 1, *semantic view* for Intuition 2, *job transition balance view* and *job transition duration view* for Intuition 3. In addition, we propose a collective method to fuse the multi-view representations into an unified representation. We introduce the details as follows.

4.2 Graph Topology View

Job graph structure explicitly illustrates the similarity and correlations between different titles. It is the most crucial and effective information which provides comprehensive and accurate title connections. However, the topology information is hidden in the graph

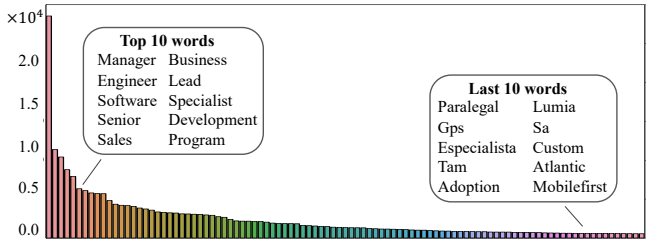


Figure 3: Words frequency distribution.

structure. Motivated by the success of graph representation learning methods in link prediction on social graph and knowledge graph [1, 4, 7, 19, 29], the first view we use in Job2vec is the **Graph Topology** view, which can encode the graph structure and neighborhood information into the representations. In Graph Topology view, we aim to learn a low-dimensional representation for each job title which can keep the neighborhood structure information, *i.e.*, job titles that share similar neighbors in Job-Graph should be close to each other in the graph view representation space.

To achieve this, we first assign each job title v_i into two representation vectors, "self representation", $e_i \in \mathbb{R}^{N_g}$, and "neighbor representation", $e'_i \in \mathbb{R}^{N_g}$, where N_g is dimension of the representation vector of Graph Topology View. Both e_i and e'_i are randomly initialized. We utilize its "self representation" while "neighbor representation" will be used if v_i is just the neighbor of the one we focus on. Then, in order to enforce embeddings to be close to each other if they share similar graph neighbors, we define the loss function O_N as followed:

$$O_N = -\sum_{(i,j)\in E} w_{ij} log(p(v_j|v_i)), \tag{1}$$

where w_{ij} is the weight between v_i and v_j, E is the set of all edges in Job-Graph, to incorporate high-order proximity, we extend E by adding edges of k-steps paths into the set. When $k = 1$, O_N is the same as the second-order proximity in LINE. $p(v_j|v_i)$ is the probability of v_j occured as neighbor given v_i, defined as a softmax function followed:

$$p(v_j|v_i) = \frac{\exp(\vec{e}'^T_j \cdot \vec{e}_i)}{\sum_{k=1}^{|V|} \exp(\vec{e}'^T_k \cdot, \vec{e}_i)}, \tag{2}$$

where v_i is the current job title we focus on, v_j is the neighbors of v_i, e_i is the "self representation" of v_i, e'_j is the "neighbor representation" of v_j, V is the set of all job titles, *i.e.*, all nodes in Job-Graph, $|V|$ is the cardinal number of V. Minimizing O_N equals to maximizing the conditional probability of v_j given v_i. Since the conditional probability $p(v_j|v_i)$ is parameterized by e'_j and e_i, as a result, the "self representation" of those job titles that share similar neighbors will be similar, *i.e.*, close in the graph view space. To be noted, in testing stage, we utilizes the "self representation" of each job title to calculate the similarity score.

4.3 Semantic View

Normally, each job title is consisted of several key words which describe the basic function and duty of this job (*e.g.*, Project Manager and Computer Engineer). Therefore, the semantic information

contained in these keywords is crucial to be explored in the representation learning process for two reasons: (1) Talents tend to make transition between functionally similar jobs, and semantic information guides the model to learn a better representation to tackle the complex job transition pattern. (2) The shared key words in job titles could further connect them even though the Job Transition Graph is extremely sparse, thus can alleviate the sparsity issue and improve the prediction capability of the learned representation.

We consider this view as the semantic view of Job Transition Graph. In semantic view, we aim to learn a low-dimensional representation of each job title which can keep the semantic information, *i.e.*, job titles that have similar key words should be close to each other in the semantic view representation space. To achieve this, we first assign each job title v_i a vector $s_i \in \mathbb{R}^{N_s}$, and each word w_j in the Job-Graph vocabulary a vector $s'_j \in \mathbb{R}^{N_s}$ which are randomly initialized. N_s is dimension of the representation vector of Semantic View. Then, we enforce s_i to be close to each other if they share similar words, based on the loss function O_S defined as followed:

$$O_S = -\sum_{w_j \in v_i} f_{ij} log(p(w_j|v_i)), \tag{3}$$

where f_{ij} is the frequency of the word w_j occurred in v_i, v_i is a job title, w_j is the words in v_i, $p(w_j|v_i)$ is the probability of w_j occurred in v_i, defined as a softmax function:

$$p(w_j|v_i) = \frac{\exp(\vec{s}'^T_j \cdot \vec{s}_i)}{\sum_{k=1}^{|W|} \exp(\vec{s}'^T_k \cdot \vec{s}_i)}, \tag{4}$$

where W is the vocabulary set of Job-Graph, s_i is the semantic representation of job title v_i, s'_j is the semantic representation of word w_j. Minimizing O_S equals to maximizing the conditional probability of w_j given v_i, as a result, job titles v_i and v_j will have similar representations s_i and s_j if they are similar.

4.4 Job Transition Balance View

The numbers of bidirectional transitions between two similar-level jobs are close. For example, software engineer in Apple is on the same level as the SDE in Facebook. The transition number from software engineer (Apple) to the SDE (Facebook), and the transition number from the SDE (Facebook) to software engineer (Apple) should be close. However, for two different-level jobs, like junior software engineer and senior software engineer, the transition usually is in one direction, from the junior to the senior one. In other words, the transition number for two-directions will be very different. To this end, we further consider Job Transition Balance as an important factors for JTB, which effectively indicates the matches of each pair of titles. The intuition of Transition Balance is that if comparable amounts of talent transitions can be found in both direction between two job titles, then these two job titles are highly likely to be in the same level. To model Transition Balance, we first assign each job title v_i a vector $b_i \in \mathbb{R}^{N_b}$ which is randomly initialized. N_b is the dimension of the representation vector of Transition Balance View. Then given two job titles v_i and v_j, we define the Transition Balance (TB) between them as:

$$TB(v_i, v_j) = \exp(-\frac{|w_{ij} - w_{ji}|}{w_{ij} w_{ji}}), \tag{5}$$

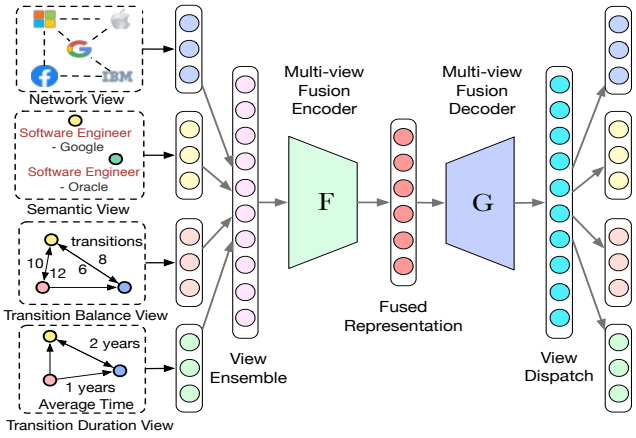


Figure 4: Collective Multi-View Representation Learning

where w_{ij} is the weight of the edge from v_i to v_j. Then, based on the loss function O_B we enforce b_i to be similar to each other if they have balanced transitions between each other. O_B is defined as followed:

$$O_B = -\sum_{(i,j) \in v_i} TB(v_i, v_j) log(p(v_i, v_j)). \tag{6}$$

where $p(v_i, v_j)$ is the joint probability of v_i and v_j defined as:

$$p(v_i, v_j) = \frac{1}{1 + \exp(-\vec{b}_i^T \cdot \vec{b}_j)}. \tag{7}$$

Minimizing O_B will "drag" the representation vectors of those "balanced" job title pair to be close in the representation space.

4.5 Job Transition Duration View

Most people require a relatively long time (*e.g.*, one or several years) to get a promotion. In contrast, if a person can change his/her jobs quickly and frequently, then there are high possibilities that these jobs are similar titles requires similar expertise and working experience. Make a long story short, the shorter the average duration of transitions is, the more similar the job titles are. To this end, we define Job Transition Duration which is the average duration time between two job titles. To include Transition Duration property into our model, we first assign each job title v_i a vector $d_i \in \mathbb{R}^{N_d}$ which are randomly initialized. N_d is dimension of the representation vector of Transition Duration View. Given two job titles v_i and v_j, we define the Transition Duration (TD) between them as:

$$TD(v_i, v_j) = \exp(-t_{ij}), \tag{8}$$

where t_{ij} is the average duration time from v_i to v_j. Then we designed a loss function O_D, which enforces d_i and d_j to be closer to each other if the average transition time between them is short, *i.e.*, it is easy to transit from v_i to v_j. O_D is defined as:

$$O_D = -\sum_{(i,j) \in v_i} TD(v_i, v_j) log(p(v_i, v_j)), \tag{9}$$

where $p(v_i, v_j)$ is the joint probability of v_i and v_j defined as:

$$p(v_i, v_j) = \frac{1}{1 + \exp(-\vec{d}_i^T \cdot \vec{d}_j)}. \tag{10}$$

Minimizing O_D will "drag" the representation vectors of those "easily transited" job title pair to be close in the representation space, which hopefully further improve the learning performance.

4.6 Multi-view Representation Fusion

Each views provides a comprehensive and unique aspect across different titles, and there are more informative and sophisticated correlation knowledge residing across different views. While, naively combining all the views together cannot efficiently utilize the information, or even hurt the performance due to the dramatic different scales and formats of the views.

To this end, we propose a Collective Multi-View Auto-Encoder (CMVAE) framework to compress the multiple representations into a single denser representation. As shown in Figure 4, the four different representations obtained by learning from the above mentioned objectives are feed into the CMVAE. To avoid losing information from all the representations, we directly concatenate the representations and feed them into the Fusion Encoder $\mathcal{F}(\cdot)$. $\mathcal{F}(\cdot)$ consists of two fully-connected layers and outputs a single and denser representation. Then this intermediate representation is feed to the Fusion Decoder $\mathcal{G}(\cdot)$ which is also a two fully-connected layer neural network and outputs the restored representation. The objective function of CMVAE is shown below:

$$L = \frac{1}{N}\sum_{i=1}^{N} \|X_i - \mathcal{G}(\mathcal{F}(X_i))\|_2^2, \tag{11}$$

where N is the number of the training samples. $X_i = [e_i; s_i; b_i; d_i]$ is the ensembled multi-view representation for a job title v_i, $\mathcal{G}(\mathcal{F}(X_i))$ is the restored representation. Minimizing the difference between the raw representations X_i and restored representations $\mathcal{G}(\mathcal{F}(X_i))$ will enforce the model to learn a denser and unified representation $\mathcal{F}(X_i)$. CMVAE hopefully captures the distinctive aspects from different views and further reveals the latent correlations across the views. We jointly optimize CMVAE associate with the other individual-view representation graphs. This jointly training strategy could let each graph assistant other graphs and further enhance the learning performance. Finally, we use $\mathcal{F}(X_i)$ as the fused multi-view representation for subsequent link prediction task.

5 EXPERIMENT RESULTS

This section details our empirical evaluation of the proposed method on real-world data.

5.1 Experimental Data

Table 4 presents the statistics of our data sets from Information Technology industry and Finance industry. Here we provide more details about our real-world data as follows:

IT Data. To construct IT data, we randomly sampled one million user career records who have been working at several well-known IT companies in US. For ease of analysis, we chose 15 most famous and leading companies in IT and only keep transition records of these companies. The 15 companies include *Facebook, Google, Amazon, Microsoft, Apple, IBM, LinkedIn, Cisco, Oracle, Airbnb, Uber, Yahoo, Nokia, Apple, Intel, HP.* Then we used the methods mentioned and Section 3 to construct the Job-Graph and aggregated it. Finally we got the IT Job-Graph shown in Table 4. The time span of the data is from 06/18/1998 to 12/01/2018.

Table 4: Statistic Details of the Dataset

	IT		Finance	
	train	test	train	test
# Nodes	44,030	2,838	89,851	5,512
# Edges	38,133	1,794	72,140	4,978
# Transitions	45,095		92,085	
# Job titles	44,030		89,851	
# Companies	15		7,669	
# Time span	1998/06-2018/12		2004/03-2018/12	
Job title examples	Software Engineer Google		Asset Manager Goldman Sachs	

Finance Data. For finance data, we randomly sampled one million records according to the rule: whose records contains finance related keywords such as *Finance, Asset Manager, Financial Research Analyst, Investment Banking Analyst, Equity Research Analyst, Trust Officer, Commercial Banker, etc.* Then again we used the methods mentioned in Section 3 to construct the Job-Graph and aggregated it. Finally we got the Finance Job-Graph shown in Table 4. The time span of the data is from 03/20/2004 to 12/01/2018.

5.2 Baselines & Evaluation Metrics

We compare our proposed method with the following representative baselines of representation learning:

Deepwalk [13]: DeepWalk adopted a truncated random walk on a graph to generate a set of walk sequences and train Skip-Gram on these sequences. It only considers graph topology.

Node2Vec [2]: Node2Vec generalizes DeepWalk by defining a more flexible notion of a node's graph neighborhood. It only considers graph topology.

LINE(1st order) [17]: In LINE, first-order and second-order proximity are modeled by the joint probability distribution between two nodes and the similarity between their neighborhood respectively. LINE(1st order) only keeps first-order proximity. It only considers graph topology.

LINE(1st+2nd order): This is the full model of LINE. It keeps both first-order and second-order proximity. It only considers graph topology.

Word2Vec[9]: Word2Vec only applies semantic view. Specifically, we treat each job title as a sentence, then train Word2Vec on all the job titles on Job-Graph. Finally we get the embedding vector of a job title by averaging the vectors of the words in it.

Job2Vec: The model proposed in this paper which considers four crucial aspects (graph topology, semantic, transition balance, transition duration) of the Job-Graph.

Metrics: We use two metrics, namely, *MRR* and *MP@K*, to evaluate the link prediction performance.

- For each test i, the correct answer job title is identified at position rank$[i]$ for closest job titles. The *Mean Reciprocal Rank (MRR)* is

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}[i]}. \tag{12}$$

Table 5: Link Prediction Performance Comparison.

	IT Dataset					Finance Dataset				
	MRR	*MP*@5	*MP*@10	*MP*@15	*MP*@20	*MRR*	*MP*@5	*MP*@10	*MP*@15	*MP*@20
DeepWalk	0.0688	0.0858	0.1070	0.1198	0.1293	0.1044	0.1164	0.1444	0.1600	0.1675
Node2Vec	0.0645	0.0785	0.0925	0.1042	0.1153	0.0979	0.1065	0.1235	0.1335	0.1460
Line(1st order)	0.0644	0.0752	0.0947	0.1081	0.1198	0.0983	0.1071	0.1245	0.1341	0.1428
Line(1st+2nd order)	0.0651	0.0791	0.0958	0.1064	0.1125	0.0943	0.0994	0.1150	0.1274	0.1381
Word2Vec	0.1295	0.2135	0.2792	0.3194	0.3334	0.1110	0.1239	0.1452	0.1643	0.1775
Job2Vec	**0.1734**	**0.2391**	**0.2976**	**0.3288**	**0.3511**	**0.1311**	**0.1378**	**0.1635**	**0.1815**	**0.1978**

Higher MRR means that correct answers appear more closely with the query job title.

- Additionally, for test i consisting of a query job title and target job title pair, consider the closest K job titles to the query embedding. If the correct target job title to the query job title is among these K titles, then the *Precision@K* for test i (denoted P@K[i]) is 1; else, it is 0. Then the *Mean Precision@K* is defined as

$$MP@K = \frac{1}{N}\sum_{i=1}^{N}(P@K[i]). \tag{13}$$

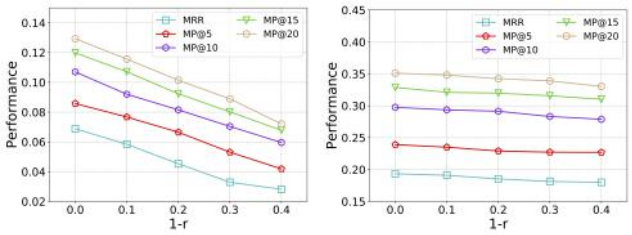
Higher precision indicates a better ability to acquire correct answers using close embeddings.

5.3 Link Prediction Performance Comparison

We performed link prediction on both IT and Finance datasets, i.e., predicting missing links on Job-Graph. Since edges with a larger weight in Job-Graph indicates a better match between job titles, we kept edges that have weight larger than a threshold (here is 5) for training and also try to predict links that have weight larger than 5 (predicted links if have weights lower than 5 will be considered as wrong). Then we randomly split the Job-Graph links into 10 equal parts, 8 of them as training set, 1 as validation set and 1 as testing set, no data in validation set and testing set can be used for training the embeddings. To avoid "cold start", we only kept job titles that have occurred in training data. We trained all the baseline models and our Job2Vec on the training set to get the embeddings, to avoid overfitting we tuned parameters on the validation set, finally we predict links on the testing set using the learned embeddings. Given a job title job_i, to predict which job may have links with it, we calculated the cosine similarity score between the embeddings of job_i and the rest of other jobs, then ranked them based on the similarity score. Higher ranked jobs have a higher probability to be matched with job_i.

We obtained the best hyper parameters of our model on the validation set. The dimensions of the four views' representations are $N_g, N_s, N_b, N_d = 128$. In the Collective Multi-View Auto-Encoder (CMVAE), for the fusion encoder, we use a two-layer fully-connected network (512*512*248) with LeakyReLU activation (negative-slope=0.7) in the first layer and Tanh activation before output. For the fusion decoder, we use a two-layer fully-connected network (248*512*512) with LeakyReLU activation (negative-slope=0.7) in the first layer and no activation in the second layer.

From the results on the IT dataset, as shown in Table 5, we can tell that graph embedding models that only keep graph topology information get similar but poor performances, where DeepWalk achieves the best *MRR* and *MP@K*. Word2Vec(averaging word vectors in job title) achieves nearly 100% improvements on *MRR* and on *MP@K* compared with DeepWalk, thus proves that semantic view is tremendously helpful for link prediction on Job-Graph. Job2Vec achieves further improvements on *MRR* compared with Word2Vec, which shows the effectiveness of preserving more views than only semantic view. Job2Vec achieves the best performance on *MRR* and *MP*@5, *MP*@10, *MP*@15, *MP*@20 among all the models. Specifically, it improves 200% over DeepWalk and 50% over Word2Vec. This confirms the superiority of the multi-view representation and the effectiveness of the encode-decode paradigm for fusing multiple views. From the results on the Finance dataset, similar conclusions can be drawn.



(a) DeepWalk (b) Job2Vec

Figure 5: Robustness Comparison

5.4 Robustness Analysis

In this subsection, we explore the robustness of different models against the sparsity of job transition graph. Specifically, we make the original graph sparser by subsampling the training edges at different rates $r = \{0.9, 0.8, 0.7, 0.6\}$(only keep 90%, 80%, 70%, 60% edges). Then we retrain our model and baseline models on the subsampled graph and compare the link prediction performance degradation. Here we use IT dataset as an example. We compare our model Job2Vec with DeepWalk for conciseness. Job2Vec take four views (graph, semantic, transition balance and transition duration view) into account while DeepWalk only considers graph view. From Figure 5, we can observe that the performance of DeepWalk degrades sharply as r decreases while Job2Vec seems to hold steady. This again confirms the effectiveness of incorporating more views especially semantic view over the sparsity issue of job transition graph. This can also be well explained: when the graph is sparse, many nodes in graph have poor connectivity, existing graph embedding models can not learn sufficient representations from graph view. But in semantic view, shared key words can connect different job titles in Job-Graph even though when it is sparse. Learning feasible representations from semantic view does not rely on graph connectivity and thus can perform excellent performance over sparse graph.

Table 6: Job Title Benchmarking Cases.

IT	
Project Manager -IBM	Product Manager PC Accessories - Microsoft
IT Support Lead and System Trainer - HP	IT Support Lead - IBM
SWE - Google	Machine Learning Engineer - Airbnb
Software Engineer - Facebook	Data Scientist - Microsoft
Finance	
Investment Banking Analyst - Citi	Investment Banking Analyst - J.P. Morgan
Equity Research Analyst - Nomura	Equity Research Analyst - Goldman Sachs
Financial Analyst - Goldman Sachs	Financial Analyst - Rushmark Properties
Portfolio Manager - WellsFargo	Portfolio Manager - ReMark Capital Group

5.5 Visualization

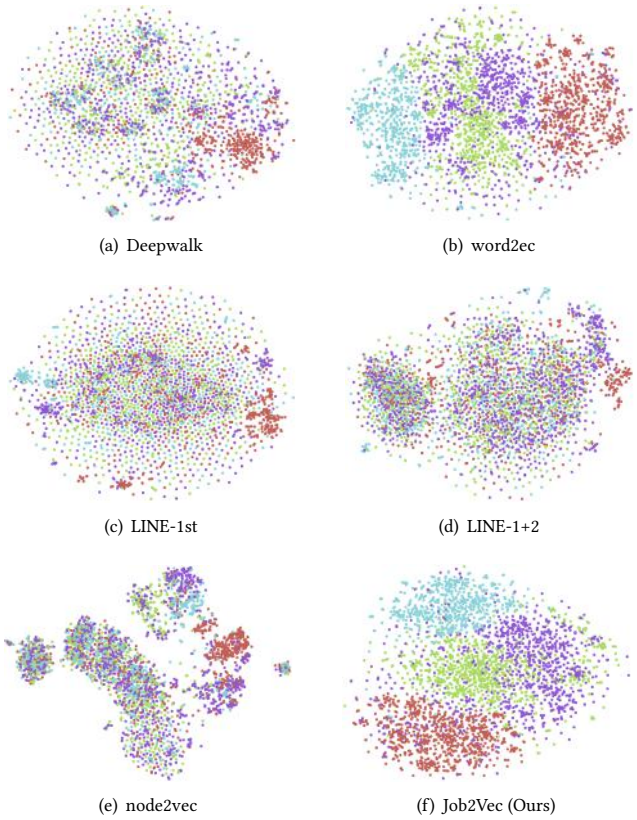
We visualize the learned representations in Figure 6 to show the promising benchmarking results of our proposed model. For convenience, we select four categories of job title representations, including engineer, sales, consultant and manager. We randomly sampled 1000 job titles for each categories. We utilize t-SNE [18] to reduce the representation dimensions to do the visualization. Each color corresponds to one category of job titles.

Figure 6 shows that our proposed Job2Vec achieves the best results. Each category of representations learned by our model can be clustered into four groups very well. In another word, job titles are benchmarked by our proposed model effectively. However, the representations learned by baselines are distributed randomly in the space which cannot reveal the becnmarking relations among job titles. The potential explanation is that our proposed method jointly model four views to preserve the topology structure, semantics and job transition patterns, which plays an essential role in the task of job title benchmarking.

5.6 Job Title Benchmarking Case Studies

In this subsection, we show some Job Title Benchmarking (JTB) results extracted from existing Job Transition Graph as well as some link prediction JTB results generated by our model and baseline models.

Table 6 shows eight JTB cases extracted from the aggregated Job-Graph, apparently job titles that have similar responsibilities and expertise level while also from companies of the same volume are matched. It is interesting that JTB can find some matching pairs that are not similar literally, such as (Software Engineer, Data Scientist), (SWE, Machine Learning Engineer). Table 7 shows the Top 3 link prediction results of different models for "Software Engineer-Facebook". Titles in bold font are wrong predictions. It can be observed that DeepWalk tends to make predictions that have similar neighborhood structure in Job-Graph, while Word2Vec inclines to job titles that are semantically similar. With only graph topology view, DeepWalk may make anomalous predictions such as "Product Manager-Microsoft". With only semantic view, Word2Vec are likely to predict repetitive job titles and miss some interesting matching pairs that are not very similar literally, such as (Software Engineer-Facebook, SDE-Microsoft). Our model, Job2Vec, incorporating graph topology, semantic, transition balance and transition duration views, makes more reasonable predictions.



(a) Deepwalk (b) word2ec (c) LINE-1st (d) LINE-1+2 (e) node2vec (f) Job2Vec (Ours)

Figure 6: Visualization of the learned representations

6 RELATED WORK

In this section, we review two categories of literatures that are related to this paper, namely research on data mining for career trajectory analysis, and research on graph embedding.

Data Mining for Career Trajectory Analysis. With the rise of Online Professional graphs (OPNs), Career Trajectory Analysis have been proved useful in many human resource management (HRM) problems[5, 28]. For example, Xu et al. build a organizational level job transition graph from OPN data and proposed a talent circle detection method to identify the right talent sources for recruitment[27]. For better assessing the expertise level or rank of a job title, Xu et al. proposed a Gaussian Bayesian graph to extract the job title hierarchy of an organization from employees' career trajectory data[26] . In [6], a contextual LSTM model is proposed to accurately predict an employee's next career move. However, few existing works pay attention to the problem of Job Title Benchmarking(JTB), which has broad application prospect in human resource management. To the best of our knowledge, we are the first to extract JTB insights from job transition graph.

Graph Representation learning graph representation learning assigns nodes in a graph to low-dimensional representations and effectively preserves the graph structure. Recently, a significant

Table 7: Top 3 Results of Link Prediction Comparison.

	Top 3 results	
Software Engineer-Facebook	Job2Vec	SDE-Microsoft, Software Development Engineer-Amazon, Software Developer-Google
	DeepWalk	Software Developer-Google, **Product Manager-Microsoft**, Software Test Engineer-Google
	Word2Vec	**Senior Software Engineer-Facebook**, Software Engineer-IBM, Software Engineer-Apple
Project Manager-IBM	Job2Vec	Product Manager-Microsoft, Project Manager-Microsoft, Program Manager-IBM
	DeepWalk	Product Manager-Microsoft, **Senior Consultant-IBM**, **Storage Administrator-IBM**
	Word2Vec	**Project Manager Lead-IBM**, Advisory Project Manager-IBM, Project Manager-Microsoft

amount of progresses have been made toward this emerging graph analysis paradigm[2, 4, 12, 16, 17, 20, 21]. Inspired by the success of representation learning in natural language processing [3, 8, 9], DeepWalk[13] is the first extension of Word2Vec to graph analysis. It uses random walks to sample paths from a graph and treat paths as "sentences" to train a SkipGram model to keep graph Proximity into learned embeddings. Node2Vec [2] generalizes DeepWalk by designing a more flexible random walk strategy. LINE[17] proposes first-order and second-order proximity to keep graph properties and combines both by concatenating first-order and second-order vectors. To better model the asymmetric property of graphs, Ou et al. proposes asymmetric proximity preserving (APP) graph embedding [12]. However, in our problem setting, we need to jointly model multi-view representations, and obtain an unified representation fused from multi-view. Current graph representation learning cannot be directly applied into the JTB scenario. To the best of our knowledge, our work is the first attempt to solve the JTB problem via multi-view graph representation learning.

Multi-View Representation learning has become attractive and urgent as the increasing multi-modal sensors are widely deployed in a great number of real-world applications [25]. It explores the complementary information among different views, where the views refer to various feature representations, modalities or sensors. Most of the approaches focus on the multi-view data including features [11], images [10], and videos [22, 23], while our approach reveal the information from multiple graph structure data which is challenging.

7 CONCLUSIONS

In this paper, we propose a data-driven solution for the problem of job title benchmarking (JTB). We construct the job title transition graphs (Job-Graph) to represent job transitions, and reformulate the JTB problem as the task of link prediction over the Job-Graph. Specifically, we propose a collective multi-view representation learning method by jointly learning the four views of representations, including graph topology view, semantic view, job transition balance view, and job transition duration view. Besides, we also propose a encode-decode based fusion method to obtain an unified representation from the multi-view representations. We present intensive experimental results with IT and finance related job transition data to demonstrate the effectiveness of our method.

REFERENCES

[1] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *Proc. ACM SIGKDD*. 119–128.

[2] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proc. ACM SIGKDD*. ACM, 855–864.

[3] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. ICML*. 1188–1196.

[4] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Signed networks in social media. In *Proc. ACM SIGCHI*. ACM, 1361–1370.

[5] Huayu Li, Yong Ge, Hengshu Zhu, Hui Xiong, and Hongke Zhao. 2017. Prospecting the career development of talents: A survival analysis perspective. In *Proc. ACM SIGKDD*. 917–925.

[6] Liangyue Li, How Jing, Hanghang Tong, Jaewon Yang, Qi He, and Bee-Chung Chen. 2017. Nemo: Next career move prediction with contextual embedding. In *Proc. International Conference on World Wide Web Companion*. 505–513.

[7] Manling Li, Denghui Zhang, Yantao Jia, Yuanzhuo none Wang, and Xueqi Cheng. 2018. Link Prediction in Knowledge Graphs: A Hierarchy-Constrained Approach. *IEEE Trans. on Big Data* (2018).

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*. 3111–3119.

[10] Feiping Nie, Guohao Cai, and Xuelong Li. 2017. Multi-view clustering and semi-supervised classification with adaptive neighbours. In *Proc. AAAI*.

[11] Feiping Nie, Jing Li, Xuelong Li, et al. 2016. Parameter-Free Auto-Weighted Multiple Graph Learning: A Framework for Multiview Clustering and Semi-Supervised Classification.. In *Proc. IJCAI*. 1881–1887.

[12] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proc. ACM SIGKDD*. 1105–1114.

[13] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proc. ACM SIGKDD*. 701–710.

[14] 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 *Proc. ACM SIGIR*. 25–34.

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

[16] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. 2009. Scalable proximity estimation and link prediction in online social networks. In *Proc. ACM SIGCOMM*. 322–335.

[17] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proc. WWW*. International World Wide Web Conferences Steering Committee, 1067–1077.

[18] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *JMLR* 15, 1 (2014), 3221–3245.

[19] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proc. ACM SIGKDD*. 1225–1234.

[20] Lichen Wang, Zhengming Ding, and Yun Fu. 2018. Learning Transferable Subspace for Human Motion Segmentation.

[21] Lichen Wang, Zhengming Ding, and Yun Fu. 2019. Low-Rank Transfer Human Motion Segmentation. *IEEE TIP* 28, 2 (2019), 1023–1034.

[22] Lichen Wang, Zhengming Ding, Zhiqiang Tao, Yunyu Liu, and Yun Fu. 2019. Generative Multi-View Human Action Recognition. In *Proc. IEEE ICCV*.

[23] Lichen Wang, Bin Sun, Joseph Robinson, Taotao Jing, and Yun Fu. 2019. EV-Action: Electromyography-Vision Multi-Modal Action Dataset. *arXiv preprint arXiv:1904.12602* (2019).

[24] Zhitao Wang, Chengyao Chen, and Wenjie Li. 2017. Predictive network representation learning for link prediction. In *Proc. ACM SIGIR*. 969–972.

[25] Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634* (2013).

[26] Huang Xu, Zhiwen Yu, Bin Guo, Mingfei Teng, and Hui Xiong. 2018. Extracting Job Title Hierarchy from Career Trajectories: A Bayesian Perspective.. In *Proc. IJCAI*. 3599–3605.

[27] Huang Xu, Zhiwen Yu, Jingyuan Yang, Hui Xiong, and Hengshu Zhu. 2016. Talent circle detection in job transition networks. In *Proc. ACM SIGKDD*. 655–664.

[28] Huang Xu, Zhiwen Yu, Jingyuan Yang, Hui Xiong, and Hengshu Zhu. 2018. Dynamic Talent Flow Analysis with Deep Sequence Prediction Modeling. *IEEE TKDE* (2018).

[29] Denghui Zhang, Manling Li, Yantao Jia, Yuanzhuo Wang, and Xueqi Cheng. 2017. Efficient parallel translating embedding for knowledge graphs. *arXiv preprint arXiv:1703.10316* (2017).

[30] Jiawei Zhang, Congying Xia, Chenwei Zhang, Limeng Cui, Yanjie Fu, and S Yu Philip. 2017. BL-MNE: emerging heterogeneous social network embedding through broad learning with aligned autoencoder. In *Proc. ICDM*. 605–614.