Large-Scale Talent Flow Embedding for Company Competitive Analysis*

Le Zhang[1], Tong Xu[1], Hengshu Zhu[2], Chuan Qin[1], Qingxin Meng[4], Hui Xiong[1,2,3], Enhong Chen[1]
[1]Anhui Province Key Lab of Big Data Analysis and Application, University of Science and Technology of China
[2]Baidu Talent Intelligence Center, Baidu Inc, [3]Business Intelligence Lab, Baidu Research
[4]Rutgers-the State University of New Jersey
{zhangle0202, zhuhengshu, chuanqin0426, xinmeng320, xionghui}@gmail.com, {tongxu, cheneh}@ustc.edu.cn

ABSTRACT

Recent years have witnessed the growing interests in investigating the competition among companies. Existing studies for company competitive analysis generally rely on subjective survey data and inferential analysis. Instead, in this paper, we aim to develop a new paradigm for studying the competition among companies through the analysis of talent flows. The rationale behind this is that the competition among companies usually leads to talent movement. Along this line, we first build a Talent Flow Network based on the large-scale job transition records of talents, and formulate the concept of "competitiveness" for companies with consideration of their bi-directional talent flows in the network. Then, we propose a Talent Flow Embedding (TFE) model to learn the bi-directional talent attractions of each company, which can be leveraged for measuring the pairwise competitive relationships between companies. Specifically, we employ the random-walk based model in original and transpose networks respectively to learn representations of companies by preserving their competitiveness. Furthermore, we design a multi-task strategy to refine the learning results from a fine-grained perspective, which can jointly embed multiple talent flow networks by assuming the features of company keep stable but take different roles in networks of different job positions. Finally, extensive experiments on a large-scale real-world dataset clearly validate the effectiveness of our TFE model in terms of company competitive analysis and reveal some interesting rules of competition based on the derived insights on talent flows.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Talent Flow, Competitive Analysis, Network Embedding

ACM Reference Format:
Le Zhang[1], Tong Xu[1], Hengshu Zhu[2], Chuan Qin[1], Qingxin Meng[4], Hui Xiong[1,2,3], Enhong Chen[1]. 2020. Large-Scale Talent Flow Embedding for Company Competitive Analysis. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380299

*Hui Xiong, Hengshu Zhu and Tong Xu are corresponding authors.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380299

1 INTRODUCTION

In fast-evolving business environments, the competition among companies becomes intensified, which results in a critical need for company competitive analysis. Indeed, with the help of competitive analysis, companies can conduct prospective strategies and talent planning, and establish business advantages over competitors. However, existing studies for company competitive analysis generally rely on subjective survey data and inferential analysis, such as the study [9] based on three factors, namely static status, potential development and influence factors, for competitiveness measurement. Due to the lack of large-scale market data for quantitative analysis, there are limits to deliver the objective analysis of competition.

Recently, the prevalence of online professional platforms (e.g., LinkedIn) enables the accumulation of a large number of digital resumes, which contain rich information about the career paths of talents [23]. These large-scale data comprehensively describe the phenomenon of talent flow which represent the job transitions among different companies. Nowadays, the frequent job-hopping has become a new norm for employees. For example, as reported by [2], employees who graduated between 2006 and 2010 have attempted 2.85 jobs on average during their first five years after graduation, almost twice the frequency of the seniors between 1986 and 1990. As a result, the massive accumulative job transition records provide unprecedented opportunities of talent flow analysis. As an alternative solution, in this paper, we aim to develop a new paradigm for studying the competition among companies through the analysis of talent flows. The rationale behind this is that the competition among companies usually leads to talent movement, while talent flows can be regarded as one of the significant signs of competitions among companies.

Along this line, we first build a Talent Flow Network based on the large-scale job transition records of talents, and formulate the concept of "competitiveness" for companies with the Personalized PageRank (PPR) proximity by considering their bi-directional talent flows in both the original and transpose networks. Then, following the idea of *Network Embedding* techniques, we propose a **T**alent **F**low **E**mbedding (TFE) model to learn the bi-directional talent attractions of each company. Specifically, we define two low-dimensional vectors for each company to represent its attraction to talent *from* other companies, as well as the attraction of its talents *to* other companies. With the bi-directional talent attractions of each company, the pairwise competitive relationships between companies can be effectively measured. To learn these vectors, we employ the random-walk based model with Noise Contrastive Estimation (NCE) in both original and transpose networks to efficiently learn representations of companies by preserving their competitiveness.

Furthermore, we design a multi-task strategy to refine the learning results from a fine-grained perspective, which can jointly embed multiple talent flow networks by assuming the features of company keep stable but take different roles in networks of different job positions. Finally, extensive experiments on a large-scale real-world dataset clearly validate the effectiveness of our TFE model in terms of company competitive analysis and reveal some interesting rules of competition based on the derived insights on talent flows.

Specifically, the major contributions of this paper can be summarized as follows:

- We develop a new paradigm for company competitive analysis through the analysis of large-scale talent flow data.
- We propose a TFE model to obtain representations of companies to learn the bi-directional talent attractions of each company, which can preserve their competitiveness.
- We design a multi-task strategy to refine the TFE model from a fine-grained perspective, which can integrate the information of multiple talent flow networks.
- We conduct extensive experiments on a large-scale real-world data set, which clearly validate the effectiveness of our TFE model in terms of company competitive analysis and reveal some interesting rules of competition based on the derived insights on talent flows.

2 RELATED WORK

In this section, we will summarize the related works as following two categories, namely company competitive analysis and network embedding techniques.

Company Competitive Analysis. As a crucial issue to guide the strategy planning of companies, competitive analysis has attracted wide attention. Existing techniques, especially those designed by researchers with business background, mainly depend on statistical analysis or heuristic solutions. For instance, [9] establishes the competitiveness measurement based on three factors, i.e., static status, potential development and influence factors. Also, [36] reveals the role of information sharing, and [16] discusses more comprehensive factors, like company size or location, to further refine the competitiveness measurement. Recently, data mining techniques are adopted to deal with the competitive analysis task. For example, [46] introduces a sequential latent variable model for measuring the competitiveness of companies in terms of talent recruitment. [17] studies the problem of company profiling based on collaborative topic regression, which can reveal the competitive edge of different companies. [20] uses a tensor factorization framework to predict patent litigation, which further indicates the competitive relationships. [11, 27] identify comparative sentences (i.e., A vs B) in text document to analyze the competitive relationships directly. [21, 38] define the edge between companies to construct network for competitive analysis.

Network Embedding. Traditionally, the learning of low dimensional representation of network structure is achieved through matrix factorization techniques, like LLE [29], GraRep [3], HOPE [25] and AROPE [43]. Thanks to the recent advances in natural language processing techniques, current network embedding approaches are largely based on the random-walk-based sampling, which is firstly practiced by DeepWalk [28]. Along this line, Node2Vec [5] further improves the random walk to capture the property of homogeneity and structural equivalence. APP [45] and VERSE [32] capture both asymmetric and high-order similarities between node pairs via random walk strategies. Meanwhile, node representation can also be learned directly from the structure of graph by using deep neural networks. For example, DNGR [4] and GAE [13] employ the auto-encoder with a target proximity matrix to learn corresponding embedding. DRNE [33] and NetRA [41] feed node sequences to a long short-term memory (LSTM) model to get node embedding. In addition, node attributes [34] and network dynamics [39] are also considered. Recently, some models are designed for multiplex network embedding. For instance, MTNE [37] enforces an extra information-sharing embedding to learn embedding vectors for each layer in a multiplex network. MELL [22] embeds each layer into a lower dimensional embedding space and enforces these embeddings close to others for sharing each layer's connectivity among embeddings. PMNE [19] proposes three different approaches to learn one overall embedding, i.e., network aggregation, result aggregation and Co-analysis. MNE [42] builds a bridge among different layers by sharing a common embedding across each layer of the multiplex networks. MCNE [35] represents multiple aspects of similarity between nodes via designing a binary mask layer. Different from them, our model is more suitable for the scenario of competitiveness analysis.

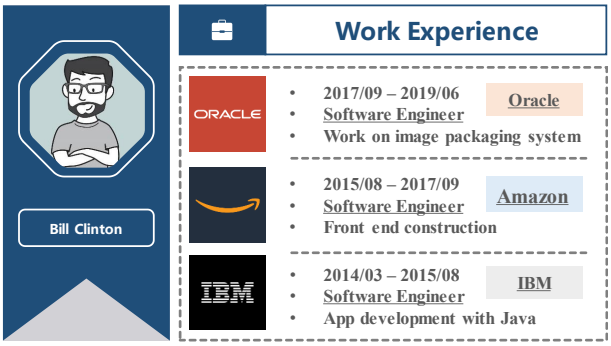


Figure 1: An example of the resume in our dataset.

3 PRELIMINARIES

In this section, we first introduce the real-world dataset used in our study, and then formulate the concept of company competitiveness. At last, we present the problem of Talent Flow Embedding for Competitive Analysis. For facilitating illustration, Table 1 lists some important mathematical notations used throughout this paper.

3.1 Data Description

The data used in this paper were collected from LinkedIn [1], one of the largest online professional social platform, where users can build professional profiles as resumes to introduce their work experiences. Specifically, as shown in Figure 1, each resume contains a list of job experience records, where each record consists of company name, job title with brief job description and the working

[1] http://api.linkedin.com/v2/people

Table 1: Mathematical notations.

Symbol	Description
$G(V, E)$	The talent flow network with company set V and transition set E;
G^T	The transpose network of G;
n, m	The number of companies and the number of job positions;
d	The dimension size of embedding;
c	The size of negative samples;
$\varrho_o(u, v), \varrho_i(u, v)$	The PPR proximity of u to v at outflow and inflow networks;
$\varrho_o^*(u, v), \varrho_i^*(u, v)$	The estimated PPR proximity of u to v at outflow and inflow networks;
S_u	The source vector of company u;
T_u	The target vector of company u;
R_k	The role embedding for a specific job position network k;
S_u^o, T_u^o	The source and target vectors of company u at outflow network;
S_u^i, T_u^i	The source and target vectors of company u at inflow network.

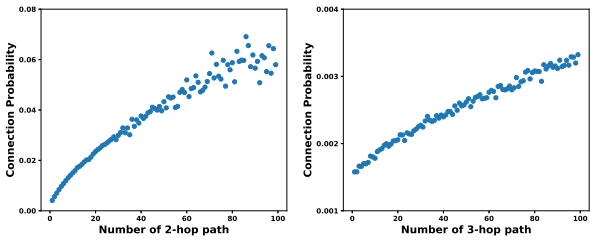


Figure 2: The transitivity of talent flow. Given a pair of nodes (u, v), the horizontal axis is the number of 2-hop (or 3-hop) paths from u to v. For the blue circle, the vertical axis represents the connection probability from u to v. As the two curves increases monotonically, we can claim that the more paths from u to v there are, the more probability that there exists an edge from u to v, which reflects the transitivity.

duration recorded in months. More details of our dataset can be found in Section 5.

The talent flows are formulated as the job transition frequencies among companies, which can be extracted from the resumes in our dataset. Obviously, the talent flow is directional and asymmetric, which means that there exists directed job transitions from company u to v, but the reverse transitions from v to u may not exist. Besides, Figure 2 illustrates the transitivity of talent flow, which means if there exists job transitions between company u and w, as well as company w and v, then it is likely that the job transitions between u and v also exist.

3.2 Formulation of Company Competitiveness

With the job transition records, we can construct a network structure to formally describe talent flows, which is defined as follows.

Definition 1 (Talent Flow Network). *The talent flow network is defined as $G = (V, E)$, where V presents the set of nodes (companies), and E presents the set of edges (talent flows). Each edge E_{ij} indicates the number of talents hopping from company i to j.*

Talent flows can be regarded as the explicit phenomenon that reflects the potential competition among companies. In turn, the competition among companies should be able to explain the talent flows. Thus we can formulate the concept of competitiveness depending on talent flows. The talent flow between two companies is directed and asymmetric, so does the competitiveness. In addition, it is intuitive to compare the talent flow to the company competitiveness, when more talents trend to move from company u to v, the competitiveness of v to u increases. For instance, 100 employees move from u to v while only 20 employees move to w, then we think v is more competitive than w with respect to u. However, this idea just considers the aspect from the source company of talent flow when judging the competition. As for the target aspect, if these 100 employees account for only 10% of the talent source of v, and 20 employees account for 100% of the talent source of w, we think w is probably more competitive than v with respect to u, which leads to the opposite conclusion. To balance the dual situations, we can formulate the competitiveness of v to u bilaterally by considering both the outflow situation of source company u and inflow situation of target company v.

As mentioned before, talent flow is asymmetric and transitive, which leads to the assumption that the more and the shorter career transition paths from company u to v, the more probability that talents move from u to v, and the more probability that the talents of company v mainly come from u. Actually, the assumption coincides with the high-order proximity of nodes in graph, i.e., Personalized PageRank (PPR) [8, 30]. The PPR proximity of u to v reflects the probability that a random walk path starting from u and ending at v. Therefore, we can use the PPR proximity of u to other companies to represent the corresponding degree that the talents of u move to others. However, the calculation of PPR proximity is based on the outgoing edge of each node, we cannot use it to find the main talent source for a company directly, which depends on the incoming edges of nodes. Alternatively, by transposing the origin network G, the nodes pointed from u could represent the talent sources of u, so that the PPR proximity of u to others in transpose network G^T can be adapted to represent the degree that the talents of u mainly come from the others. According to the meaning of edge, we call the original network G as the outflow network, and the transpose network G^T as the inflow network. Here we formulate the competitiveness of company v to u as follows:

$$comp(u, v) = \varrho_o(u, v) \cdot \varrho_i(v, u), \tag{1}$$

where $\varrho(u, v)$ denotes the PPR proximity of u to v in network, the subscript of "o" and "i" represent the outflow network G and inflow network G^T respectively. We denote $\varrho(u, \cdot)$ as the PPR proximity of u to other nodes, which satisfies the following recursive equation:

$$\varrho(u, \cdot) = (1 - \alpha) \cdot \varrho(u, \cdot)A + \alpha \cdot r, \tag{2}$$

where A denotes the transition matrix of the network with normalized rows, α is the jump rate and r is a one-hot vector which is all zero except the position of u to be 1. Directly calculating $\varrho(u, \cdot)$ will take $O(n^2)$ time complexity to compute the PPR matrix. However, the stationary distribution of a random walk with restarting probability α converges to PPR [26]. Thus, a sample from $\varrho(u, \cdot)$ is the end node in a random walk path that starts from node u. According to the idea of Monte Carlo approach, if we get enough samples, $\varrho(u, \cdot)$ can be estimated efficiently.

3.3 Formulation of Talent Flow Embedding

In this paper, we target at revealing the competition among companies from the perspective of talent flow. Intuitively, the talent flows

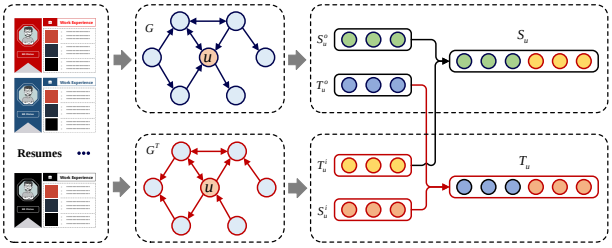


Figure 3: The diagrammatic sketch of TFE.

represent the potential "*attractions*" for talents of different companies, which further indicates their competitiveness. Therefore, the objective of this study is to learn two attraction vectors S_u and T_u from talent flow network G, which are defined as follows.

- S_u, the source vector of u, which indicates the attraction of talents in company u **to** other companies.
- T_u, the target vector of u, which indicates the attraction of company u to the talents **from** other companies.

Formally, the studied problem in this paper can be defined as the task of **Talent Flow Embedding** (TFE) as follows.

Definition 2 (Talent Flow Embedding). *Given a talent flow network $G = (V, E)$. For each node $u \in V$, we aim to learn two low-dimensional vector representations $S_u \in \mathbb{R}^d$ and $T_u \in \mathbb{R}^d (d << |V|)$ to indicate the bi-directional competitiveness.*

In the following section, we will introduce the technical details of our solution for TFE and explain the relationship between the learned embedding results with the competitiveness.

4 TECHNICAL DETAILS

In this section, we first introduce the formulation of our TFE model. Then the property of the TFE model will be discussed. Finally, we design a multi-task strategy to refine the TFE model from a fine-grained perspective.

4.1 TFE Model Formulation

In the embedding space, we try to learn the representations of each company for preserving the competitiveness among companies. According to Equation 1, the competitiveness depends on two independent components, i.e., the PPR proximities at the inflow and outflow networks. Therefore, here we can calculate the two parts separately. Specifically, we divide the source vector and target vector into two parts:

$$S_u = [S_u^o, T_u^i], \quad T_u = [T_u^o, S_u^i], \tag{3}$$

where $[\cdot, \cdot]$ is the operation of vector concatenating. S_u^o, T_u^o, S_u^i and T_u^i represent the source and target vectors at the outflow and inflow networks respectively, the dimension of them is $\frac{d}{2}$. Furthermore, we use S_u^o and T_v^o to preserve $\varrho_o(u, v)$, S_v^i and T_u^i to preserve $\varrho_i(v, u)$. The graphical representation of the TFE is shown in Figure 3.

Without loss of generality, here we take the embedding learning at the outflow network as an example. As the PPR proximity of company u to other companies can be interpreted as a distribution with $\sum_{v \in V} \varrho_o(u, v) = 1$. Meanwhile, we can also generate an estimated distribution $\varrho_o^*(u, \cdot)$ for company u in the embedding space. We would like to fit the distribution $\varrho_o^*(u, \cdot)$ to $\varrho_o(u, \cdot)$. As an optimization objective, our target is to minimize the Kullback-Leibler (KL) divergence from the given probability distribution $\varrho_o(u, \cdot)$ to that of $\varrho_o^*(u, \cdot)$ in the embedding space:

$$\sum_{u \in V} KL\left(\varrho_o(u, \cdot) \,||\, \varrho_o^*(u, \cdot)\right). \tag{4}$$

To define the closeness between two nodes u, v in the embedding space, we choose the inner product of the u's source vector and v's target vector $S_u^o \cdot T_v^o$ as the distance from u to v. Then $S_v^o \cdot T_u^o$ denotes the distance from v to u, so that the asymmetry of competitiveness can be captured. Moreover, the proximity distribution in the embedding space is normalized by softmax function:

$$\varrho_o^*(u, v) = \frac{\exp(S_u^o \cdot T_v^o)}{\sum_{w \in V} \exp(S_u^o \cdot T_w^o)}. \tag{5}$$

By Equation 4, we aim to minimize the KL-divergence from ϱ_o to ϱ_o^*, which is equivalent to minimize the cross-entropy loss function:

$$\mathcal{L}_o = -\sum_{u \in V} \varrho_o(u, \cdot) \log(\varrho_o^*(u, \cdot)). \tag{6}$$

To solve Equation 6, the Stochastic Gradient Descent (SGD) method can be used directly. Nevertheless, the direct optimization of this task is computationally expensive, since the denominator of $\varrho_o^*(u, v)$ requires the summation over all nodes in the network. Thus, to improve the training efficiency, we adopt the Noise Contrastive Estimation (NCE) method [6, 24], which is used to estimate unnormalized continuous distributions. The idea of NCE is to train a binary classifier to distinguish node samples coming from the empirical distribution $\varrho_o(u, \cdot)$ or generated by a noise distribution Φ_n. Specifically, suppose the random variable D represents node classification, such that $D = 1$ represents a node drawn from the empirical distribution and $D = 0$ represents a sample drawn from the noise distribution. According to the NCE method, we draw a node u from a default distribution Φ_e at first, and then draw a node v from the empirical distribution of $\varrho_o(u, \cdot)$ and c nodes from the noise distribution Φ_n. Finally, the origin objective function is transformed to maximize the following function:

$$\mathcal{L}_n^o = \sum_{\substack{u \sim \Phi_e \\ v \sim \varrho_o(u, \cdot)}} [\log P_u^o(D = 1|v, \theta^o) + c\mathbb{E}_{x \sim \Phi_n} \log P_u^o(D = 0|x, \theta^o)], \tag{7}$$

where $\theta^o = \{S^o, T^o\}$ denotes the parameters of model, c denotes the number of noise samples and the conditional probability $P_u^o(D|\cdot)$ is calculated as follows:

$$\begin{aligned} P_u^o(D = 1|v, \theta^o) &= \sigma(S_u^o \cdot T_v^o - \log c \cdot \Phi_n), \\ P_u^o(D = 0|v, \theta^o) &= 1 - \sigma(S_u^o \cdot T_v^o - \log c \cdot \Phi_n), \end{aligned} \tag{8}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. As the number of noise samples c increases, the negative NCE gradient approaches to the cross-entropy gradient. Besides, the convergence of NCE does not depend on the choice of distribution Φ_e and Φ_n [7], but the noise distribution Φ_n influences the performance of NCE [14]. In this paper, inspired by the idea of [32], we set both distributions Φ_e

and Φ_n as $\frac{1}{n}$, where n denotes the number of nodes. Similarly, we can obtain the NCE loss for the inflow network G^T as below:

$$\mathcal{L}_n^i = \sum_{\substack{u\sim\Phi_e \\ v\sim\varrho_i(u,\cdot)}} [\log P_u^i(D=1|v,\theta^i) + c\mathbb{E}_{x\sim\Phi_n} \log P_u^i(D=0|x,\theta^i)], \tag{9}$$

where θ^i denotes the parameters $\{S^i, T^i\}$. Combining the loss of inflow and outflow network representation, we can get the final objective:

$$\mathcal{L}_n = \mathcal{L}_n^o + \mathcal{L}_n^i. \tag{10}$$

The objective function can be solved by the SGD method. After that, we can get the corresponding embeddings S_u^o, S_u^i, T_u^o and T_u^i. The final embeddings of company u can be calculated by the Equation 3.

4.2 Property of TFE

Now we analyze the property of the proposed TFE model. We prove the learned embeddings implicitly preserve the competitiveness of any pair of nodes. Since $S_u \cdot T_v = S_u^o \cdot T_v^o + S_v^i \cdot T_u^i$, here we take the first part as an example, i.e., $S_u^o \cdot T_v^o$, whose optimization objective function is shown in Equation 7. Following the idea of [15], when the dimension of the vector is sufficiently large, the objective can be treated as a function of each independent $S_u^o \cdot T_v^o$ term. The Equation 7 can be rewritten as:

$$\begin{aligned}
\mathcal{L}' &= \sum_u \sum_v \mathcal{Z} \cdot \varrho_o(u,v) \cdot \{\log\sigma(S_u^o \cdot T_v^o - \log\frac{c}{n}) \\
&\quad + \frac{c}{n}\sum_x \log\sigma(-(S_u^o \cdot T_x^o - \log\frac{c}{n}))\} \\
&= \mathcal{Z}\{\sum_u \sum_v \varrho_o(u,v) \cdot \log\sigma(S_u^o \cdot T_v^o - \log\frac{c}{n}) \\
&\quad + \sum_u \sum_x \frac{c}{n} \log\sigma(-(S_u^o \cdot T_x^o - \log\frac{c}{n}))\} \\
&= \mathcal{Z}\{\sum_u \sum_v [\varrho_o(u,v) \cdot \log\sigma(S_u^o \cdot T_v^o - \log\frac{c}{n}) \\
&\quad + \frac{c}{n} \log\sigma(-(S_u^o \cdot T_v^o - \log\frac{c}{n}))]\},
\end{aligned} \tag{11}$$

where $\mathcal{Z}$ denotes the sample size. To maximize the objective, we set the partial derivative of each independent $Y = S_u^o \cdot T_v^o$ be zero,

$$\frac{\partial\mathcal{L}'}{\partial Y} = \mathcal{Z}\{-(\varrho_o(u,v) + \frac{c}{n})\sigma(Y - \log\frac{c}{n}) + \varrho_o(u,v)\}. \tag{12}$$

Afterwards, we can obtain:

$$Y = S_u^o \cdot T_v^o = log(\varrho_o(u,v)). \tag{13}$$

Similarity, $S_u^i \cdot T_v^i$ has the similar property. Hence, the inner product of S_u and T_v can be calculated as follows:

$$S_u \cdot T_v = \log(\varrho_o(u,v)) + \log(\varrho_i(u,v)) = \log(comp(u,v)). \tag{14}$$

Obviously, $S_u \cdot T_v$ preserves the competitiveness of v to u.

4.3 Multi-Task Strategy for TFE

As our basic TFE model has been introduced, in this subsection, we discuss how to refine the basic model to learn more comprehensive embedding for companies based on multiple talent flow networks. We realize that even for the same company, the talent flow in different job positions could be different. The concerns may be different for talents with different job positions when deciding a company to hop. e.g., engineers are more likely to be attracted by the companies with high-innovation and high-welfare, while salesperson are more likely to choose companies with big brands and good products. We assume that the features of company keep stable in different job position networks, while these features play different roles for attracting talents.

Indeed, each dimension of the attraction vectors, i.e., S_u and T_u, can be treated as a kind of feature. In the formulation of Multiple Talent Flow Embedding (MTFE), we set S_u and T_u as the overall embedding for company u and keep them the same in all job position networks. As mentioned before, we use $S_u \cdot T_v = \sum_{j=1}^d S_{uj}T_{vj}$ to indicate the competitiveness of v to u, where d is the dimension size. Directly calculating the inner product presumes a strong assumption that each dimension in the embedding space is equally important. To show the importance of different features, we expand the two-way inner product to the three-way tensor product by introducing a role vector R_k for each job position k, of which each dimension R_{kj} is used to represent the importance of the j-th feature in job position k. Then the j-th feature's influence to the competitiveness of v to u at the job position k is calculated as $(R_{kj} \cdot S_{uj} \cdot T_{vj})$. As a result, the competitiveness of company v to u at the job position k can be calculated as follows:

$$(S_u \odot T_v) \cdot R_k = \sum_{j=1}^d S_{uj} \cdot T_{vj} \cdot R_{kj}, \tag{15}$$

where the $\odot$ denotes the cross product operation. Moreover, we limit each dimension of the role vector between 0 and 1 to prevent gradient explosion.

In MTFE, we also consider the inflow and outflow network separately following the idea of TFE. Thus we define the corresponding role vectors as R_k^o and R_k^i, so that $R_k = [R_k^o, R_k^i]$. We aim to minimize the expectation sum of cross-entropy loss over multiple networks, which leads to the following objective function:

$$\mathcal{L} = -\sum_k \sum_{u\in V} \{\varrho_o^k(u,\cdot)\log(\varrho_o^{*k}(u,\cdot)) + \varrho_i^k(u,\cdot)\log(\varrho_i^{*k}(u,\cdot))\}, \tag{16}$$

where the superscript k denotes the variable of a specific job position network k, the superscript $*$ denotes the estimated result in embedding space. Besides, the NCE method is also used to estimate the original loss, hence we turn to maximize the following:

$$\begin{aligned}
\mathcal{L}^* &= \sum_k \sum_{\substack{u\sim\Phi_e \\ v\sim\varrho_i^k(u,\cdot)}} [\log P_u^{ik}(D=1|v,\theta) + c\mathbb{E}_{x\sim\Phi_n} \log P_u^{ik}(D=0|x,\theta)] \\
&\quad + \sum_k \sum_{\substack{u\sim\Phi_e \\ v\sim\varrho_o^k(u,\cdot)}} [\log P_u^{ok}(D=1|v,\theta) + c\mathbb{E}_{x\sim\Phi_n} \log P_u^{ok}(D=0|x,\theta)],
\end{aligned} \tag{17}$$

where $\theta = \{S, T, R\}$ denotes the parameters of MTFE, the conditional probability $P_u^{ok}(D|\cdot)$ can be calculated as follows:

$$\begin{aligned}
P_u^{ok}(D=1|v,\theta) &= \sigma((S_u^o \odot T_v^o) \cdot R_k^o - \log c \cdot \Phi_n), \\
P_u^{ok}(D=0|v,\theta) &= 1 - \sigma((S_u^o \odot T_v^o) \cdot R_k^o - \log c \cdot \Phi_n), \\
&s.t.\ \forall j \in [1,d],\ 0 \le R_{kj}^o \le 1.
\end{aligned} \tag{18}$$

The calculation of $P_u^{ik}(D|\cdot)$ is similar to $P_u^{ok}(D|\cdot)$. After learning all the parameters, we can concatenate the corresponding vectors to get the overall embeddings S_u and T_u for each company, as well as the role vector R_k for each job position.

4.4 MTFE Model Learning

Here we introduce how to learn the parameters of MTFE model. As for the source and target vectors of each company at the outflow and inflow networks, i.e., S_u^o, T_u^o, S_u^i and T_u^i, we use the SGD method to learn them directly. As for the role vectors of each job position k, i.e., R_k^o and R_k^i, they must be bound between 0 and 1. So we employ the Projected Gradient (PG) [18] to learn them. Given a sample pair (u, v) at the outflow or inflow network of job position k, the update rule for the j-th dimension of R_k^o or R_k^i is shown as follows:

$$R_{kj}^o \leftarrow f(R_{kj}^o - \eta(D - \sigma((S_u \odot T_v) \cdot R_k^o - \log\frac{c}{n}) \cdot S_{uj} \cdot T_{vj}))),$$
$$R_{kj}^i \leftarrow f(R_{kj}^i - \eta(D - \sigma((S_u \odot T_v) \cdot R_k^i - \log\frac{c}{n}) \cdot S_{uj} \cdot T_{vj}))), \tag{19}$$

where η is the learning rate, D is an indicator number that equals 0 if the node pair (u, v) is sampled from the noise distribution Φ_n, and 1 otherwise. And the function $f(x)$ is defined as follows:

$$f(x) = \begin{cases} 0, & \text{if } x < 0, \\ x, & \text{elif } 0 \le x \le 1, \\ 1, & \text{otherwise.} \end{cases} \tag{20}$$

Algorithm 1 describes the optimization process of the MTFE model.

ALGORITHM 1: Multiple Talent Flow Embedding (MTFE)

Require: Set of talent flow networks $\{G^k\}$, sample size $\mathcal{Z}$, learning rate η, noise distribution Φ_n, negative samples size c, dimension size d.
Ensure: Embedding vectors for $v \in V$ and role vectors for G^k .

$\{G^{ko}\} \leftarrow \{G^k\}$, $\{G^{ki}\} \leftarrow \{\text{transpose of } G^k\}$
$S^o, T^o, R^o \leftarrow TFME(\{G^{ko}\})$
$S^i, T^i, R^i \leftarrow TFME(\{G^{ki}\})$
$S, T, R \leftarrow$ concatenate the corresponding vectors
Function *TFME* $(\{G^k\})$
  $S \leftarrow \mathcal{N}(0, d^{-1})$, $T \leftarrow \mathcal{N}(0, d^{-1})$
  $R \leftarrow [d^{-1}, d^{-1}, \ldots, d^{-1}]$
  for $i = 1$ to $\mathcal{Z}$ **do**
    $k \sim$ uniform(1, m)
    $u \sim$ uniform(1, n)
    $v \sim \varrho(u, \cdot)$ in G^k
    $S_u, T_v, R_k \leftarrow UpdateByPair(u, v, k, 1)$
    for $j = 1$ to c **do**
      $x \sim \Phi_n$
      $S_u, T_x, R_k \leftarrow UpdateByPair(u, x, k, 0)$
  Return S, T, R
Function *UpdateByPair* (u, v, k, D)
  $g \leftarrow (D - \sigma((S_u \odot T_v) \cdot R_k - \log c \cdot \Phi_n)) \cdot \eta$
  $S_u \leftarrow g \cdot T_v \cdot R_k$, $T_v \leftarrow g \cdot S_u \cdot R_k$
  $R_k \leftarrow$ Update according to Equation 19

4.5 Complexity Analysis

The main execution time of both TFE and MTFE are determined by the optimization process. Given d as the embedding dimensionality, n as the number of nodes, m as the number of job position network and $\mathcal{Z}$ as the number of samples. The overall time complexity of TFE is $O(\mathcal{Z}dn)$, and that of MTFE is $O(\mathcal{Z}mdn)$ which is close to m times of the previous one. Apart from this, the space complexity of MTFE processes is $O((n+m)d)$, which is efficient for other practical applications due to that $m \ll n$.

Table 2: Statistics of each job position network.

Dataset	#Nodes	#Aligned Nodes	#Edges	#Resumes
Engineer	15,237	15,244	311,567	596,058
Consultant	15,236	15,244	310,613	486,682
Salesperson	15,243	15,244	263,725	533,549
Operator	15,241	15,244	281,860	407,143

5 EXPERIMENTS AND DISCUSSIONS

In this section, we evaluate the proposed model with a number of baselines on a large-scale dataset. Meanwhile, many discussions and case studies on company competition analysis will be presented.

5.1 Experimental Setting

Data Pre-processing. In our real-world dataset, there are more than 400 million professional resumes. We set the observational time window as 3 years from January 1st, 2015 to December 31, 2017. At first, we employed the methods mentioned in [44] to extract the job transition records from the resumes. Then we utilized the API called MonkeyLearn [2] to normalize the job titles, which categorized the original job titles according to the job description written in the resumes, then we got 26 job positions. Afterwards, we removed the resumes of employees who only stayed in the same company and filtered the companies which appeared less than 800 times to avoid noise. Finally, we obtained the job transitions among the filtered companies in the remained resumes. Totally, 15,244 companies and 7,066,978 job transitions were extracted.

Dataset Splitting. We selected 4 common job positions to analyze the competition, they are *Salesperson*, *Consultant*, *Operator* and *Engineer*. In detail, for each job position, only job transitions fitting the position would be kept. After that, four job positions led to four talent flow networks. The number of nodes in each network is a little different, and to better compare the single and the multiple network embedding methods, here we aligned the nodes set by adding nodes with degree equals to 0. Some statistics about these datasets are summarized in Table 2.

5.2 Baseline

To evaluate the performance of the proposed models, we compared our model with following network embedding algorithms:

- **DeepWalk** [28]: which takes the truncated random walks for each node to generate the training corpus of node sequences, and then learns the node embedding via maximizing the likelihood of context node prediction from the center nodes. We used the default parameters mentioned in the original paper, i.e., walk length $l = 80$, number of walks $\gamma = 80$ and window size $w = 10$.

[2] https://app.monkeylearn.com/main/classifiers

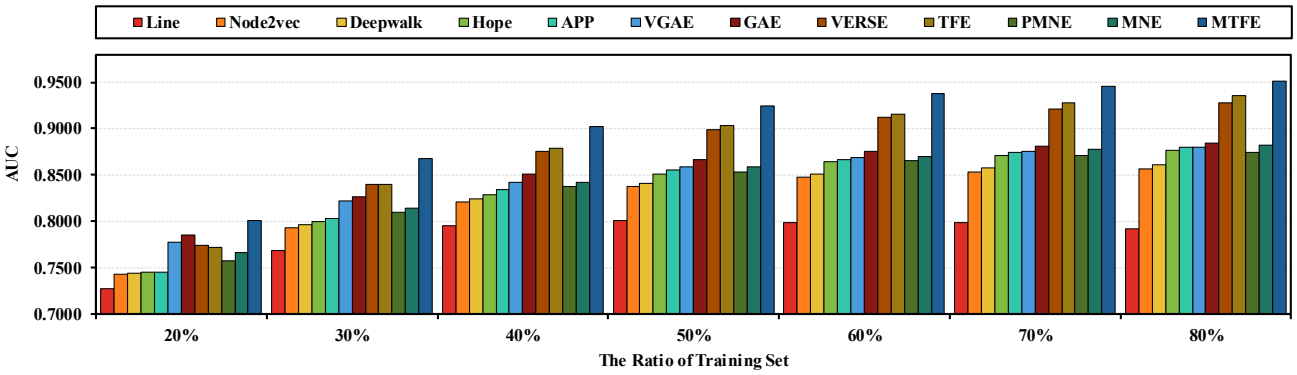


Figure 4: The average performance of each model with different proportion of training edges.

- **Node2Vec** [5]: which designs a biased truncated random walks to efficiently explore diverse neighborhoods by adding two parameters p and q. Here we set $p = 2$, $q = 0.5$ and the other parameters were the same as DeepWalk.
- **LINE** [31]: which learns the network embedding by preserving the first-order proximity and second-order proximity of network structure separately. It learns $d/2$ dimensions using first-order proximity and learns another $d/2$ dimensions using second-order proximity. The final embeddings are obtained by concatenating two parts. We set the negative samples s to 5.
- **HOPE** [25]: which aims to preserve high-order similarity of nodes and capture asymmetric transitivity. HOPE learns embeddings using a generalized SVD framework. Here we selected the RPR similarity, and set the decay probability α as 0.15.
- **APP** [45]: which estimates the PPR proximity between each pair of nodes to capture both asymmetric and high-order similarities among them. We set the number of samples per node to 5000 and the decay factor α to 0.15.
- **VERSE** [32]: which uses the PPR, Adjacency and SimRank as the proximities and learns embedding by the skip-gram model. Here we chose the PPR proximity for best performance. Besides, we set the damping factor α to the default value 0.85, and the size of noise samples to 3.
- **VGAE** [13]: which is a model for unsupervised learning on graph-structured data under the Variational Auto-Encoder (VAE) framework. It applies Graph Convolutional Network (GCN) [12] as the encoder and the inner product of embeddings as decoder. Here we replaced the usual symmetric normalization of A by the out-degree normalization $D_{out}^{-1}(A + I)$ for directed graphs.
- **GAE** [13]: which is a variant model of VGAE by using the auto-encoder(AE) framework. We used the same setting as VGAE.

The approaches introduced above are designed for single network, we also chose several multiple network embedding approaches.

- **PMNE** [19]: which proposes three different models to multiplex networks, such as network aggregation, result aggregation and Co-analysis. We compared all three models and chose the best one PMNE(r), which applied the result aggregation based on Node2vec. The parameters were the same as Node2vec.

Table 3: The AUC scores for different models on the link prediction task.

Method	Consultant	Salesperson	Engineer	Operator
Common Nbrs	0.7358	0.6927	0.7739	0.7521
Jaccard	0.7667	0.7237	0.8022	0.7791
Adamic Adar	0.7360	0.6934	0.7739	0.7533
DeepWalk	0.8312	0.8312	0.8568	0.8458
Node2Vec	0.8276	0.8276	0.8537	0.8425
LINE	0.8053	0.7314	0.8552	0.8110
APP	0.8538	0.8473	0.8582	0.8599
HOPE	0.8373	0.8395	0.8532	0.8735
GAE	0.8624	0.8470	0.8742	0.8813
GVAE	0.8561	0.8418	0.8664	0.8698
VERSE	0.9044	0.8925	0.9011	0.8950
PMNE	0.8534	0.8450	0.8618	0.8534
MNE	0.8384	0.8402	0.8667	0.8598
TFE	0.9059	0.8954	0.9029	0.9075
MTFE	**0.9298**	**0.9212**	**0.9214**	**0.9267**

- **MNE** [42]: which represents information of multi-type relations into a unified embedding space by using a high-dimensional common embedding and a lower-dimensional additional embedding for each type of relation to represent each node. The dimension of the additional vectors was set to be 10.

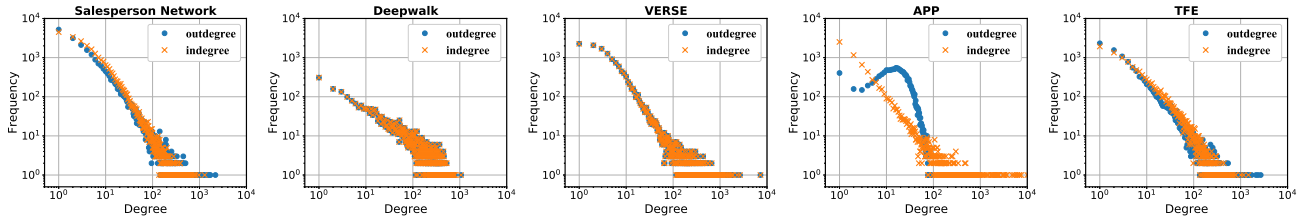
For the task of link prediction, we also selected three heuristic algorithms, which are commonly used for link prediction, including:

- **Common Neighbors**: i.e., $|N(u) \cap N(v)|$,
- **Jaccard Coefficient**: i.e., $\frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$,
- **Adamic Adar**: i.e., $\sum_{t \in N(u) \cap N(v)} \frac{1}{log|N(t)|}$,

where $N(u)$ denotes the set of neighbors of u.

5.3 Validations on Link Prediction

Similar as other network embedding tasks, here we first validate the effectiveness of our TFE models via the link prediction task. Since the competition of companies leads to talent flows, so that link prediction is a good task to evaluate the attraction representations.



(a) Salesperson Network (b) DeepWalk (c) VERSE (d) APP (e) TFE

Figure 5: The degree distributions of Engineer network and predicted distributions of DeepWalk, VERSE, APP and TFE.

Specifically, we conducted the validations on all the four data sets. In each dataset, we randomly removed 50% of edges as the positive samples in test sets, while the rest were treated as training samples. At the same time, we also expanded the test set by randomly sampling an equal number of node pairs as negative samples, which have no edge connecting them.

For all embedding based models, the dimension of embedding vector was set to 128. As for the multiple network embedding models, we learned jointly on four networks and evaluated on each one respectively. Furthermore, as for the TFE and MTFE, we drew 3 negative samples for each positive one and set the jump rate α as 0.85. Given a node pair (u, v), we adopted the inner product of the corresponding embedding vector of u and v to estimate the similarity score, i.e., $S_u \cdot T_v$ for asymmetric models, e.g., Hope and APP, and $S_u \cdot S_v$ for symmetric models, e.g., Deepwalk and VERSE. Along this line, we selected **AUC** as the valuation index.

The results are summarized in Table 3. Obviously, our TFE methods, even the simplified TFE without the multi-task strategy, achieve the relatively better performance than all of the baselines. Moreover, we have the following observations. First, the heuristic methods always get the worst performance, since they just consider the direct neighbor information and fail to describe the structure in full view. Second, we realize that techniques which preserve high-order similarity for nodes can get good performance, such as APP and VERSE. However, VERSE always gets much better performance than APP though both of them can preserve the PPR proximity. This is because the asymmetry of talent flow is not very obvious, which weakens the performance of asymmetric preserved models to some extent. Third, jointly modeling multiple networks is helpful to get better performance. For instance, MTFE gets better performance than TFE in all datasets. Last, PMNE and MNE are much better than their underlying model designed for a single network, i.e., Node2vec. However, they are not as good as our MTFE model, because they are limited by the underlying algorithm, so that the results further demonstrate the effectiveness of our TFE model.

In addition, we also discuss how the performance of each model be influenced by the different training ratios. Figure 4 shows the average performance of each model on all networks. Clearly, the performances of most models improve with the training ratio increasing. Meanwhile, the results of different training ratios are consistent with that mentioned before.

To summarize, our models outperform most of the other baselines, and the stable results suggest their robustness.

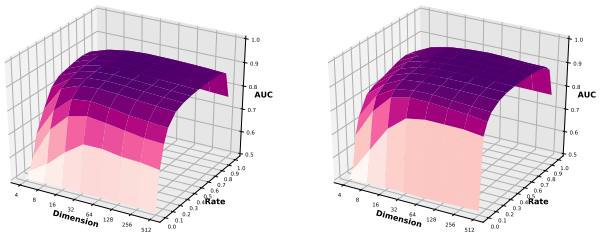
Table 4: The NDCG@K Scores for link weight prediction. All the numbers are the averaged value at four job networks.

Method	NDCG@5	NDCG@10	NDCG@20	NDCG@50
DeepWalk	0.0454	0.0511	0.0590	0.0758
Node2Vec	0.0388	0.0434	0.0506	0.0657
LINE	0.0624	0.0648	0.0688	0.0784
APP	0.1496	0.1650	0.1836	0.2125
HOPE	0.0629	0.0617	0.0642	0.0724
GAE	0.0104	0.0108	0.0127	0.0189
GVAE	0.0035	0.0047	0.0059	0.0105
VERSE	0.2021	0.2168	0.2357	0.2638
PMNE	0.0692	0.0769	0.0877	0.1078
MNE	0.0412	0.0474	0.0560	0.0737
TFE	0.2183	0.2350	0.2536	0.2802
MTFE	**0.2318**	**0.2513**	**0.2730**	**0.3021**

5.4 Validations on Link Weight Prediction

The task of link prediction targets at judging the existence of job transitions among companies, so we further predict the degree of talent flows. Specifically, we split job transitions by year, and chose the job transitions between 2015 and 2016 as the training set, and that in 2017 as the test set. Then we trained all models with parameters as same as link prediction. Moreover, the inner product of nodes indicated the prediction score. Given a node u, we ranked the others according to the prediction scores. Finally, we evaluated the performance of the predicted rank list by using the Normalized Discounted Cumulative Gain (NDCG@K, K = 5, 10, 20, 50) index. The higher value of NDCG@K is, the better result is obtained.

The results are summarized in Table 4. Actually, this task is challenging because of the dynamic nature of talent flow. Despite of this, our TFE models consistently achieve the best performances compared with baselines. Some models cannot effectively handle this task even if they are effective on link prediction, such as the GCN based model. This is due to they cannot make full use of information from the weight of edges. Besides, although LINE performs poorly on link prediction, it is more effective than the DeepWalk and Node2vec. Because Line uses the weight of edges to construct model's objective. Finally, we also observe that the multiple network embedding approaches are more effective than the underlying one, which further indicates the effectiveness of integrating multiple network information when representing a node.



(a) TFE (b) MTFE

Figure 6: The performances of TFE and MTFE with different dimension size and jump rate.

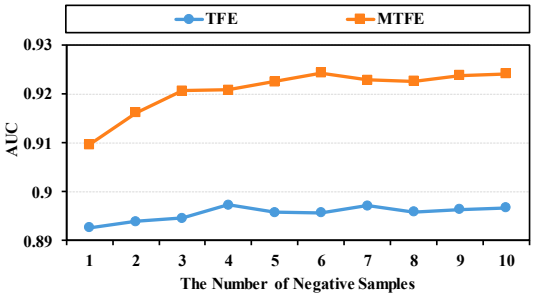


Figure 7: The performances of TFE and MTFE with different number of noise samples.

5.5 Validations on Graph Reconstruction

Furthermore, we performed the graph reconstruction task by judging if the representation could capture the in/out-degree distribution of the talent flow network. Specifically, here we take the Salesperson network as an example. We firstly trained the TFE model on the Salesperson network. Then we ranked node pairs (u, v) according to the inner product of the corresponding vectors, i.e., $S_u \cdot T_v$ for asymmetric models, such as TFE and APP, and $S_u \cdot S_v$ for symmetric models, such as DeepWalk and VERSE. Finally, we took the top-$|E|$ pairs of nodes without self-loop to reconstruct the graph, where $|E|$ is the number of edges in the original graph.

Here we show the performances of two symmetric models, i.e., DeepWalk and VERSE, and two asymmetric models, i.e., APP and our basic model TFE. Figure 5(a) shows the original degree distribution of the Salesperson network, and the other parts of Figure 5 show the result of different models. Obviously, the generated in/out-degree distributions of the symmetric models are identical, because they just learn one vector for a node. In addition, the in-degree distributions of VERSE and APP are similar to the original distribution. As mentioned in [1, 40], the in-degree distribution and PPR proximity follow the same distribution so that the PPR similarity preserved models can capture the in-degree distribution of talent flow network. Furthermore, the in-degree and out-degree distributions of original network are very similar so that the out-degree distribution generated by VERSE is similar to original network. To some extent, it can explain why VERSE always gets the comparable performance in the above tasks. And we find that our TFE model is the only one that can generate both in/out-degree distributions similar to that of the original graph, because the TFE model preserve the PPR similarity in both inflow and outflow network, i.e, G and G^T, and the in-degree of G^T is equivalent to the out-degree of G. As a result, the out-degree distribution of G can be captured.

5.6 Parameter Sensitivity

Afterwards, we turn to discuss the sensitivity of the parameters of the TFE and MTFE. We evaluated the performances on link prediction task with 50% training edges. Specifically, three parameters will be discussed, namely the dimension of embedding vector d, the size of negative samples c and the jump rate α.

Firstly, we discuss the sensitivity of the dimension size, which are summarized in Figure 6. Clearly, we find that generally the increasing d improves the performance, which could be reasonable as more dimensions may probably keep more useful information. However, there is no significant improvement when d is more than 128, which explains why we set d as 128 in our overall experiments to keep the balance between effectiveness and efficiency.

Secondly, we discuss the sensitivity of jump rate, which is also shown in Figure 6. We realize that a relatively larger α may lead to better results, as the longer paths are useful to capture the high-order relationship. However, when α is larger than 0.85, the performance drops significantly. This phenomenon illustrates that the overlong paths may disturb the effectiveness as outliers.

Finally, we verify the impact of the number of negative samples, as shown in Figure 7. Obviously, the performances fluctuate in a small range. According to [24, 32], the theoretical guarantees of NCE depend on the number of negative samples, yet small values work well in practice, and our result further proves it.

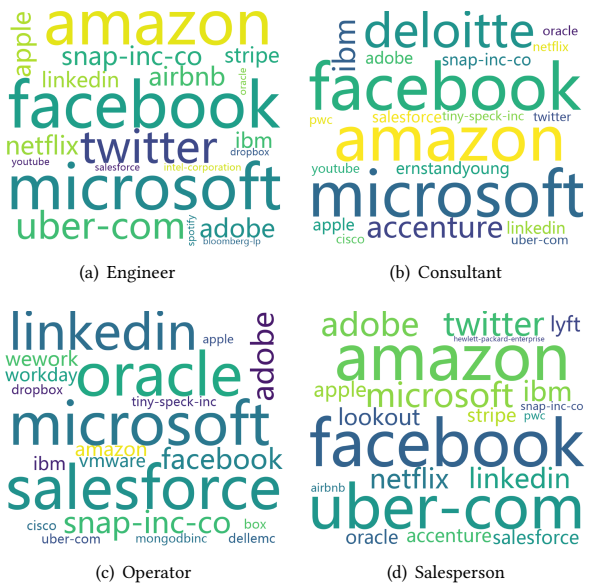
5.7 Case Study

Finally, in this part, some case studies will be presented to illustrate competitive analysis, and reveal some interesting rules of competition based on the derived insights on talent flows.

5.7.1 Competitor Analysis. In this case study, we would like to reveal the biggest competitors of the given company. We selected Google as an example, for each job transition network, we adopted $((S_u \odot T_v) \cdot R_k)$ to calculate the competitiveness of company v to u (i.e., Google). Then we sorted the companies according to the competitiveness scores to get the top-n competitor.

The results are shown in Figure 8, where the size of name indicates the degree of competition. Obviously, we realize that the distribution of the competitors is quite different, which proves that the effectiveness of the role embedding R_k. For different job position, the features of the company take the different roles. As for the Engineer network, we find the top competitors are most high-tech companies (e.g., Facebook and Microsoft). For the Consultant network, the famous law firms (e.g., Ernstandyoung and Accenture) and management consulting firms (e.g., Accenture and IBM) become more attractive. One interesting finding is that Facebook in most cases is very competitive with Google, but in the network of Operator, it is not so competitive.

Also, apart from the competitors detection in a specific network, as S_u and T_u represent the essential features of companies. To detect



(a) Engineer (b) Consultant

(c) Operator (d) Salesperson

Figure 8: The top-20 competitors for Google in four different job positions.

(a) Google (b) Facebook

Figure 9: The top-20 overall competitors for Google and Facebook respectively.

the direct competitors in a full view, we use the $(S_u \cdot T_v)$ directly. Figure 9 shows the top-20 competitors of Google and Facebook. Comparing the overall competition results of Google with competitions of a specific job position, the overall competition can be treated as the combination of the specific competitions. The result of comparison shows the effectiveness of the multi-task strategy. Besides, comparing the result of Google and Facebook, we discover that they are the biggest competitor for each other.

5.7.2 Clustering on Bi-directional Attraction. We conducted the MTFE model on the entire data to get the embedding for each company u, S and T. Then we used the K-means [10] to cluster companies, according to the embedding to get 20 clusters and further projected these nodes into a two-dimension space.

Figure 10 shows the clustering result of the source vector S_u, which summarizes the similar companies for the attraction of their talents. Obviously, we find that companies in similar industry are likely to cluster together. For example, the high-tech internet companies, such as Google and Baidu, are clustered together. And the

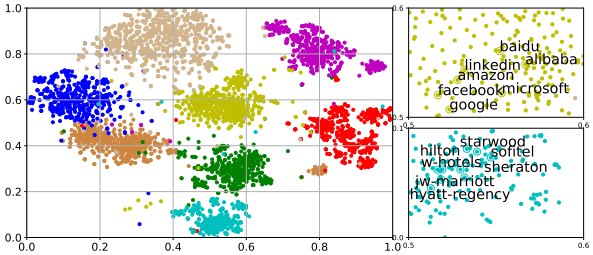


Figure 10: Clustering for attraction to talents.

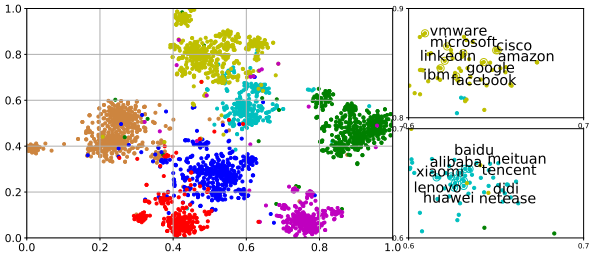


Figure 11: Clustering for attraction from talents.

hotel service companies, such as Sheraton and Hyatt-Regency are clustered together. Besides, K-means based on T_u is also conducted to summarize the similar companies in attracting talents, which is shown in Figure 11. Interestingly, we find that the clusters in Figure 10 are broken up, as companies now are clustered mainly based on their countries.

Considering that S_u reflects the attraction of talents in u, talents of the companies with the similar business are attractive for the others in the same industry, so that clustering on S_u reflects companies in similar industry. On the contrary, T_u denotes how company u attracts talents from other companies. Since job transitions usually happen in the same country, it is reasonable for partition companies via country and business.

6 CONCLUSION

In this paper, we studied the problem of company competitive analysis from the perspective of talent flows. Specifically, we formulated the concept of "competitiveness" for companies with the PPR proximity. Then we proposed a TFE model to learn representations of companies to learn the bi-directional talent attractions of each company, which could preserve their competitiveness. Furthermore, we designed a multi-task strategy to refine the TFE model by integrating the information of multiple talent flow networks. Finally, extensive experiments conducted on a large-scale real-world dataset clearly validated the effectiveness of the proposed models and revealed some interesting rules of competition based on the derived insights on talent flows.

7 ACKNOWLEDGMENTS

This research was partially supported by grants from the National Key Research and Development Program of China (Grant No. 2018YFB1402600), and the National Natural Science Foundation of China (Grant No. 61836013, 91746301, 61703386, U1605251).

REFERENCES

[1] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast incremental and personalized pagerank. *Proceedings of the VLDB Endowment*, 4(3):173–184, 2010.

[2] Guy Berger. Millennials job-hop more than previous generations, they aren't slowing down. http://bit.ly/2DEKAgJ, 2016.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, pages 1145–1152, 2016.

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[6] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[7] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

[8] Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM, 2002.

[9] Zhu-hui Huang and Yu Zhang. Indexes and models: Measurement of enterprise competitiveness [j]. *Journal of Zhejiang University (Humanities and Social Sciences)*, 4:025, 2002.

[10] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[11] Nitin Jindal and Bing Liu. Identifying comparative sentences in text documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 244–251. ACM, 2006.

[12] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[14] Matthieu Labeau and Alexandre Allauzen. An experimental analysis of noise-contrastive estimation: the noise distribution matters. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 15–20, 2017.

[15] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[16] Panagiotis Liargovas and Konstantinos Skandalis. Factors affecting firm competitiveness: The case of greek industry. *European institute Journal*, 2(2):184–197, 2010.

[17] Hao Lin, Hengshu Zhu, Yuan Zuo, Chen Zhu, Junjie Wu, and Hui Xiong. Collaborative company profiling: Insights from an employee's perspective. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1417–1423, 2017.

[18] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[19] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. Principled multilayer network embedding. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 134–141. IEEE, 2017.

[20] Qi Liu, Han Wu, Yuyang Ye, Hongke Zhao, Chuanren Liu, and Dongfang Du. Patent litigation prediction: A convolutional tensor factorization approach. In *IJCAI*, pages 5052–5059, 2018.

[21] Zhongming Ma, Gautam Pant, and Olivia RL Sheng. Mining competitor relationships from online news: A network-based approach. *Electronic Commerce Research and Applications*, 10(4):418–427, 2011.

[22] Ryuta Matsuno and Tsuyoshi Murata. Mell: effective embedding method for multiplex networks. In *Companion Proceedings of the The Web Conference 2018*, pages 1261–1268. International World Wide Web Conferences Steering Committee, 2018.

[23] Qingxin Meng, Hengshu Zhu, Keli Xiao, Le Zhang, and Hui Xiong. A hierarchical career-path-aware neural network for job mobility prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 14–24, 2019.

[24] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[25] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *KDD*, pages 1105–1114, 2016.

[26] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[27] Gautam Pant and Olivia RL Sheng. Avoiding the blind spots: Competitor identification using web text and linkage structure. *ICIS 2009 Proceedings*, page 57, 2009.

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[29] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[30] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 322–335. ACM, 2009.

[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. ACM, 2015.

[32] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference*, pages 539–548. International World Wide Web Conferences Steering Committee, 2018.

[33] Ke Tu, Peng Cui, Xiao Wang, Philip S Yu, and Wenwu Zhu. Deep recursive network embedding with regular equivalence. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2357–2366. ACM, 2018.

[34] Hao Wang, Enhong Chen, Qi Liu, Tong Xu, Dongfang Du, Wen Su, and Xiaopeng Zhang. A united approach to learning sparse attributed network embedding. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 557–566. IEEE, 2018.

[35] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. Mcne: An end-to-end framework for learning multiple conditional network representations of social network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1064–1072, 2019.

[36] Wei-ping Wu. Dimensions of social capital and firm competitiveness improvement: The mediating role of information sharing. *Journal of management studies*, 45(1):122–146, 2008.

[37] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and S Yu Philip. Multi-task network embedding. *International Journal of Data Science and Analytics*, 8(2):183–198, 2019.

[38] Yang Yang, Jie Tang, Jacklyne Keomany, Yanting Zhao, Juanzi Li, Ying Ding, Tian Li, and Liangwei Wang. Mining competitive relationships by learning across heterogeneous networks. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1432–1441. ACM, 2012.

[39] Yuyang Ye, Hengshu Zhu, Tong Xu, Fuzhen Zhuang, Runlong Yu, and Hui Xiong. Identifying high potential talent: A neural network based dynamic social profiling approach.

[40] Yuan Yin and Zhewei Wei. Scalable graph embeddings via sparse transpose proximities. *arXiv preprint arXiv:1905.07245*, 2019.

[41] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. Learning deep network representations with adversarially regularized autoencoders. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2663–2671. ACM, 2018.

[42] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. Scalable multiplex network embedding. In *IJCAI*, volume 18, pages 3082–3088, 2018.

[43] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2778–2786. ACM, 2018.

[44] Le Zhang, Hengshu Zhu, Tong Xu, Chen Zhu, Chuan Qin, Hui Xiong, and Enhong Chen. Large-scale talent flow forecast with dynamic latent factor model? In *The World Wide Web Conference*, pages 2312–2322. ACM, 2019.

[45] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *AAAI*, pages 2942–2948, 2017.

[46] Chen Zhu, Hengshu Zhu, Hui Xiong, Pengliang Ding, and Fang Xie. Recruitment market trend analysis with sequential latent variable models. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 383–392. ACM, 2016.