Learning Job Representation Using Directed Graph Embedding

Haiyan Luo
Indeed, Inc.
hluo@indeed.com

Shichuan Ma
Indeed, Inc.
shichuanm@indeed.com

Anand Joseph Bernard Selvaraj
Indeed, Inc.
aselvaraj@indeed.com

Yu Sun
Indeed, Inc.
sunyu@indeed.com

ABSTRACT

In recent years, embedding technologies have gained popularity in many areas of machine learning, such as NLP, computer vision, information retrieval, etc.. In this paper, we propose a latent representation of job positions consisting of job title and company pairs, which can capture not only similarity relations but also ordering relations among job positions. We first construct a directed graph of job positions from the user's job transition history in the resume data, then we train the job position embedding using an asymmetric relation preserving graph embedding algorithm. Experimental results on a career move prediction task using real-world data set demonstrated that the proposed embedding solution can outperform state-of-the-art embedding methods.

KEYWORDS

directed graph embedding, job recommendation, representation learning

ACM Reference Format:
Haiyan Luo, Shichuan Ma, Anand Joseph Bernard Selvaraj, and Yu Sun. 2019. Learning Job Representation Using Directed Graph Embedding. In *1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data (DLP-KDD'), August 5, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

In online job platforms such as Indeed.com, recommender systems play an important role in ensuring an efficient job marketplace by providing job seekers with the best jobs from the right employers. To achieve this, good representations of jobs and users' preferences for different jobs are essential.

In recommender systems, latent representations are often used to explain the interactions between users and items. For example, many Collaborative Filtering (CF) algorithms are item-based [17] in the sense that they analyze item-item relations in order to compute item similarities. With the remarkable success of deep learning in computer vision and natural language processing, more and more works have been trying to leverage deep or shallow neural networks to learn latent representations for users and items in recommender systems [2, 3, 6]. These embeddings are either pre-trained in an unsupervised fashion then plugged into supervised models as features, or computed by back-propagation from user action labels (e.g. clicks) in an end-to-end pipeline.

In most existing unsupervised embedding works of recommender systems, item embedding focuses more on preserving the similarity between items, i.e. the goal is to make embedding vectors close to each other in the latent space when the corresponding items are similar to each other (e.g. frequently occurred in similar contexts). This works to some extent for job recommendations because for most job seekers, recommending a job similar to his current job or recently applied jobs is usually reasonable. However, for most job seekers, other things such as career development and employer brand are also important factors when considering a career opportunity. Thus, it would be ideal if the job embedding can also capture such preferences for similar jobs.

Specifically, when considering new job opportunities, the job seeker most likely will evaluate multiple factors such as compensation, location, employer brand, career potential and so on before deciding to make a move. The preference of one job opportunity over another in general can be reflected in the transition moves of hundreds of thousands of other job seekers who have already made similar career moves. This pair-wise ordering, when aggregated over many job seekers, can be valuable information for many applications like job recommendations, salary estimation, next career move suggestions and so on.

For each job seeker, the historic employment information and held positions at each company can be extracted from his/her resume. Using embedding technologies, a conventional approach in this case is to treat each resume as a context and each job position as a word, then following Word2Vec [12] model, Skip-Gram with Negative Sampling (SGNS) can be used to train the embedding for job positions, given a resume database. However, this approach has its shortcomings: 1) By ignoring the sequential order among job positions in the same resume, it does not preserve the pair-wise preference ordering information resulting from career choices. One typical example is that nowadays more engineers would switch jobs from hardware companies to software companies than the other way around. 2) For a job seeker, the job position of his/her early year experience is likely to be quite different from his/her current job position. By treating all job positions in the same resume equally, embedding vectors that should be apart from each other are brought closer together, degrading the quality of the representation. For instance, in the computer industry, there was a time when fiber companies were very lucrative which made them very easy to

DLP-KDD' , August 5, 2019, Anchorage, AK, USA

attract top talent, but now Internet companies has become one of the new crazes.

To address this issue, we propose to treat user resumes as sampled trajectories on a directed graph, the nodes on which are (job title, company) pairs, and the edges represents job transition intensity aggregated over all job seekers. Figure 1 shows an example of the career path of one job seeker modeled as a directed graph using the job title and company pairs extracted from the employment history of the resume. We use a variation of asymmetric proximity preserving (APP) [21] graph embedding algorithm to generate job embeddings, which can preserve the pair-wise ordering relations within the resumes. Further, to alleviate the data sparsity problem, We introduce truncated re-sampling to generate virtual resumes to supplement training data.

There are various alternative ways to computing job embeddings. For example, job embeddings can be computed from co-click or co-apply data, which assumes that if two jobs are applied by the same person, they should be similar to each other. Job embeddings can also be obtained from pooling or concatenating embeddings of skills, companies and job titles together if these embeddings are available. Comparing with these embedding approaches, job embeddings from resume data proposed in this paper carries additional information that comes from the underlying graph structure, which is missing in the co-click or co-apply data. However, it may be more difficult to learn the embedding because the data set is sparser.

It is also worth mentioning that in reality locations can be another major factor contributing to the decision of job changes. However, due to data sparsity issue and simplicity of modeling, we only model the transition pairs of (title, company) instead of the tuple of (title, company, location). Further, in most situations, the company signal already covers the location information, since given a company its number of working locations is very limited, if not fixed.

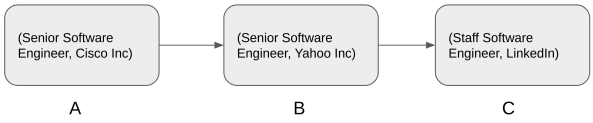

Figure 1: An example of the career path of one job seeker modeled as a directed graph, where each node is represented by the job title and company/employer pair.

The rest of this paper is organized as follows. In Section 2, we introduce the related work in the research community. In Section 3, we describe in detail our proposed algorithm. In Section 4, we provide our experimental results.

2 RELATED WORK

Graph embedding seeks to represent vertices of a graph in a low-dimensional vector space in which meaningful relations and structural information of the graph can be captured. With graph embedding, vector-based machine learning algorithms can be applied to graph data. Graph embedding approaches can be broadly categorized into three classes [4]: Factorization based, Random Walk based and Deep Learning based. Overall, a common underlying assumption of the graph embedding methods is that the nodes sharing the similar set of neighborhood are embedded into vectors close to each other in the latent space [9].

Motivated by the success of Word2Vec [12] and GloVe [15] in NLP task, various neural network based embedding algorithms have been developed to model neighborhood relations on graphs [9, 20]. For example, DeepWalk [16] uses truncated random walks on the graph to produce context, and applies a sliding window to sample node-context pairs for Skip-Gram model [13] training. Node2vec [5] uses a biased random walk that balances breadth-first search (BFS) and depth-first search (DFS) methods to sample neighboring nodes. LINE [18] optimizes objectives, preserving both first-order proximity (pairwise proximity among the nodes) and second-order proximity (among nodes sharing many neighbors). These methods are proven equivalent to factorizing a high-order proximity matrix [19].

These graph embedding methods cannot preserve the asymmetric transitivity well, which is very important for directed graphs. Asymmetric transitivity means that if there is a directed edge from node s_i to node s_j and a directed edge from node s_j to node s_k, there is likely a directed edge from s_i to s_k, but not from s_k to s_i. One example of such asymmetric relation could be the global node importance ranking induced from directed edges. The experiments in [10] demonstrate that being able to preserve the global ranking in node embedding can not only boost the performance of a learning-to-ranking task, but also the performance of classification task trained by treating node embedding as features. In our job transition graph each node represents a job position, this asymmetric transitivity relation indicates a preference order associated with each pair of jobs.

Most of the existing asymmetric transitivity preserving graph embedding algorithms generate two embedding vectors for each node, one corresponding to the outgoing direction and one for the incoming direction. HOPE [14] preserves the asymmetric role information of the nodes by approximating high order proximity metrics like Katz[8], Rooted PageRank [11], Common Neighbors [11], and Adamic-Adar [1]. It essentially decomposes the induced similarity matrices and use the decompositions to represent nodes. APP [21] uses a directed random walk to sample node from the graph and to generate training data, with each node also having two embeddings as its representation. It can preserve rooted PageRank proximity. ATP [7] takes a factorization-based approach and applies the embedding to a few Community Question Answering (CQA) tasks.

3 ASYMMETRIC JOB POSITION EMBEDDING

3.1 Job position transition graph

The *Experience* sections in user resumes contain the user's work experiences, with information such as company, position, start time and end time, etc. We can construct a directed graph $G = (V, E)$ from user resumes. In this graph, each node $s \in V$ represents the job position (i.e., a job title and company pair) and each edge $e \in E$ represents a transition (i.e. a job change). Thus, each resume can be regarded as a path on graph G from the job seeker's very first job position s_{start} to the most up-to-date one $s_{current}$. Associated with each directed edge e_{ij} there is a weight w_{ij} representing

the frequency of this transition. We normalize the weights to get transition probabilities at each node s_i:

$$p_{ij} = \frac{w_{ij}}{\sum_{j \in U_i} w_{ij}}$$

where U_i is the set of nodes reachable from s_i in 1 transition step.

We consider two job positions to be similar when job seekers are very likely to move from one to the other and vice versa (first-order proximity). Higher order similarities are also considered, e.g. if many job seekers moved from (to) a similar set of jobs to (from) these two job, then these two jobs should be similar to each other.

3.2 Advantage score

When considering directional transitions in graph G between job positions, a job position may act as a transition source or destination. Like in most asymmetric transitivity preserving graph embedding algorithms, we compute two embeddings for each job, one (input embedding) when it is the source and one (output embedding) when it is the destination of a transition. For example, given job positions i and j, their embeddings are u_i, u_j, v_i, v_j, where u is the embedding for the source node and v the embedding for the destination node.

The advantage score of node s_i over s_j can be computed as

$$u_i \cdot v_j - u_j \cdot v_i, \tag{1}$$

A positive advantage score means that more job seekers tends to move from job i to job j rather than the other way round. Advantage scores between job position pairs summarizes the job seekers' preference as reflected in the job transition graph G. In Section 4.4 we use advantage score to refine the results of node prediction.

Note that there may be cycles in graph G, in this case advantage score can still be used to compare each pair of nodes.

3.3 Algorithm

3.3.1 Training data generation. For a given path on G, the positive training samples are generated by collecting all pairs of nodes following the transition order. For example a path $s_i -> s_j -> s_k$ will lead to positive pairs (s_i, s_j), (s_i, s_k) and (s_j, s_k), as illustrated in Figure 1. According to the normalized weights on outgoing edges, we conduct random walk with stopping probability γ starting from randomly sampled nodes $s \in V$. Following the same protocol, positive training samples can be generated from these paths.

Further, for each positive training sample (s_i, s_j), we randomly draw K nodes not reachable from s_i in any path as negative samples. Additionally, a truncated back-ward random walk is conducted starting from s_i, following the reverse direction of edges. Any node on the ensuing path observing the reverse transition order is added as a negative sample. The same stopping probability γ is used in back-ward random walk to control how many negative samples from random walk we would like to introduce into the training data.

3.3.2 Cost function. For each positive training sample (s_i, s_j), let e_i and e_j be the embeddings for node s_i and s_j, respectively. Let $U_{neg,i}$ be the set of nodes in graph G sampled to make negative training pairs with node s_i. We optimize the following objective

$$\log \sigma(u_i \cdot v_j) + \sum_{s_k \in U_{neg,i}} \log \sigma(-u_i \cdot v_k) \tag{2}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function. Note that optimizing this cost function brings output embedding of j close to input embedding of i if s_j can be reach from s_i in the sampled paths. On the other hand, output embedding of k will be moved away from input embedding of i if s_i can be reached from s_k in the sample paths.

The cost function can be obtained by summing up Eqn. 2 over all positive pairs from sample paths. It is then optimized using stochastic gradient descent (SGD). The job position embedding algorithm is summarized in Algorithm 3.3.2.

Input : Graph $\mathcal{G} = (V, E)$ constructed from user resume data, stopping factor γ for random walk, and learning rate λ. **Output** : embedding e_u and e_v for each node $s_i \in V$
Initialize u_i, v_i, $\forall s_i \in V$ as random vectors;
each $s \in V$ initialize training data set U_p, U_n as empty set $U_p \leftarrow RandomWalk(s, \gamma) \bigcup PairsFromResume(s)$
$U_n \leftarrow ReverseRandomWalk(s, \gamma) \bigcup NegativeSample(s)$
$U_s \leftarrow U_p \bigcup U_n$
training sample $x \in U_s$ Stochastic Gradient Descent(x)

4 EXPERIMENT

4.1 Data Set

The data we use is a small subset of recently uploaded or updated user resumes from our job platform. It contains a total of $6.79M$ resumes, $1.06M$ distinct job title-company pairs, and $11.5M$ job transitions. The data set is pre-processed to remove low-frequency job title-company pairs.

The data set covers a variety of jobs from different industries and different requirements, ranging from *"psychologist"*, *"receptionist"* to *"certified welder"* and *vice president of finance*. We built a directed graph from the data set to characterize the job transition. There are $9.73M$ directed edges in the graph, corresponding to the number of distinct job transitions pairs observed in the resumes. We can observe the graph is very sparse. Somewhat surprisingly, if we ignore the direction on edges, all nodes in this graph form a single connected component.

4.2 Baseline Algorithms

We compare the performance of the proposed algorithm with 2 baseline algorithms: *Skipgram* and *APP*. *Skipgram* applies SGNS directly to the resume data, treating each resume as the context and each job position as the word. It models only the co-occurrence relation between jobs, ignoring the transition ordering and directions. *APP* uses random walk with restart to sample paths on the underlying directed graph. The starting and ending nodes of the paths are collected to form the training data set. The embedding is trained using SGNS. *APP* uses two embedding spaces to represent a job, which can preserve asymmetric relations between node pairs.

4.3 Link prediction

The link prediction task can be regarded as a binary classification problem. Given a set of node pairs, we need to predict if there is a link between them in the graph. We randomly picked 10% edges from the job transition graph as positive samples. Same amount of negative samples are generated by randomly sampling node pairs

that are not in the graph. To manifest the direction of the links in positive samples, we add their reverse links to the negative samples if they are not in the graph.

Unlike the settings of link prediction tasks in social networks, our objective here is to examine how well the embedding learned from the directed graph can preserve neighborhood information. Instead of splitting graph edges into training and testing set, we use the full data set for training and test its performance on testing set.

For *Skipgram*, the similarity score between given node pairs computed from their input embeddings are used to predict the link. For *APP* and our proposed algorithm, we used similarity between input embedding and output embedding of the node pairs to predict the link.

Algorithm	AUC
Skipgram	0.655
APP	0.792
Proposed	0.801

Table 1: Comparison of link prediction performance of Skipgram, APP and the proposed algorithm, evaluated by AUC.

We use ROC-AUC (Area Under the ROC-Curve) to evaluate the link prediction performance. As shown in Table 1, our proposed algorithm outperforms *Skipgram* by a large margin and slightly better than *APP*. This is not surprising since *Skipgram* cannot capture the local structure of transition graph well, especially the direction of links. *APP* does not have this problem, but in the job transition graph there are many low frequency job positions with low in-degree, which are not well represented in *APP*'s training data.

4.4 Node prediction: next career move recommendation

While link prediction can be used to choose most probable job transitions between random job position pairs, it makes more sense to calculate the top-k possible next job positions based on the job seeker's current job for recommendation purpose.

We assume that in the absence of any other information, for a given job position, the set of other job positions to which job seekers have transitioned from this job position could serve as a good signal for tasks such as next career move recommendations. In a simplified scenario we only use learned graph embedding to score next possible jobs. This won't deliver the best performance but can serve as an indicator on how well a certain embedding can model the job transition propensity.

We randomly pick 1% of the node in the job transition graph, for each such node s the set of nodes that s can transit to are ranked by their difference in advantage score. We truncate the list of nodes to keep only the top 10 nodes (when the out-degree of s is less than 10 we keep all nodes). From embeddings we find top-10 most similar jobs and compute the overlap with the top-10 list computed from the graph. This is essentially a graph reconstruction task. We want to recover top outgoing edges from a particular node, and the corresponding order in the weights. The precision result for all 3 embedding algorithms are shown in Table 2.

Algorithm	P@10
Skipgram	0.036
APP	0.049
Proposed	0.051

Table 2: Comparison of next job recommendation performance of Skipgram, APP and the proposed algorithm, evaluated by Precision@10.

It can be observed that all 3 algorithms achieved low precision in this task. Indeed such prediction based only on historical transition data is challenging. We can observe that both *APP* and the proposed algorithm outperforms *Skipgram*. This could be because they can better capture the local structure of directed graphs. The proposed algorithms is slightly better than *APP*.

5 CONCLUSION

In this work, we have proposed to derive job embeddings from resume data using asymmetric transitivity graph embedding algorithm. The embeddings generated with our proposed algorithm can preserve transition preference information in the directed job transition graph, making it a good feature in supervised machine learning models for job recommendations. We have compared our embedding method with two baseline algorithms. It outperforms both *Skipgram* and *APP* in link prediction and node prediction tasks.

REFERENCES

[1] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *SOCIAL NETWORKS*, 25:211–230, 2001.

[2] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 947–956. PMLR, 29–31 Oct 2018.

[3] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA, 2016.

[4] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv: 1705.02801*, 2017.

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 1725–1731. AAAI Press, 2017.

[7] Armin Bashizade Jiongqian Liang P. Sadayappan Jiankai Sun, Bortik Bandyopadhyay and Srinivasan Parthasarathy. Atp: Directed graph embedding with asymmetric transitivity preservation. In *AAAI Conference on Artificial Intelligence*, 2019.

[8] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.

[9] Megha Khosla, Avishek Anand, and Vinay Setty. A comprehensive comparison of unsupervised network representation learning methods. *CoRR*, abs/1903.07902, 2019.

[10] Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh, and Shou-De Lin. Prune: Preserving proximity and global ranking for network embedding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5257–5266. Curran Associates, Inc., 2017.

[11] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, May 2007.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[14] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1105–1114, New York, NY, USA, 2016. ACM.

[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[16] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014.

[17] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: large-scale information network embedding. *CoRR*, abs/1503.03578, 2015.

[19] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. Fast network embedding enhancement via high order proximity approximation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3894–3900, 2017.

[20] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[21] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *AAAI*, 2017.