Towards Job-Transition-Tag Graph for a Better Job Title Representation Learning

Jun Zhu, Céline Hudelot
Laboratoire Mathématiques et Informatique pour la Complexité et les Systèmes
CentraleSupélec, Université Paris-Saclay
Gif-sur-Yvette, France
{jun.zhu, celine.hudelot}@centralesupelec.fr

Abstract

Works on learning job title representation are mainly based on *Job-Transition Graph*, built from the working history of talents. However, since these records are usually messy, this graph is very sparse, which affects the quality of the learned representation and hinders further analysis. To address this specific issue, we propose to enrich the graph with additional nodes that improve the quality of job title representation. Specifically, we construct *Job-Transition-Tag Graph*, a heterogeneous graph containing two types of nodes, i.e., job titles and tags (i.e., words related to job responsibilities or functionalities). Along this line, we reformulate job title representation learning as the task of learning node embedding on the *Job-Transition-Tag Graph*. Experiments on two datasets show the interest of our approach.

1 Introduction

The learning of job title (referred to as job for short) [1] representation has received much attention in the recruitment field because the learned representation is beneficial to various tasks, such as job recommendation (Dave et al., 2018; Liu et al., 2019b), job title benchmarking (Zhang et al., 2019), and job mobility prediction (Zhang et al., 2021; Zhu et al., 2021). However, in practice, learning a good representation is challenging for the following reasons: (i) **Noisy data**: job title data is noisy due to personal subjective reasons (i.e., spelling errors) or objective reasons (i.e., resume parsers are not perfect). (ii) **Messy data**: job titles are messy because people have different ways of thinking, and naming conventions vary by company and industry. For example, there are many alternative job titles for the same position, e.g., "purchasing clerk" and "buyer". Another problem is that due to the ambiguity of certain terms, they can refer to different positions in different contexts, e.g., "registered nurses sandwich rehab" and "sandwich maker". For these reasons, standard semantic-based approaches that aggregate (e.g., mean or sum) word representations to get job title semantic representation may lead to mismatches. Moreover, these methods ignore hidden relationships between job titles, e.g., titles in the same resume may be similar. (Dave et al., 2018; Zhang et al., 2019) learn representations from graphs. They create graphs from career trajectories, where nodes represent job titles and edges represent job transitions. Then they design different loss functions to embed the nodes into a low-dimensional space. However, the generated graphs are usually sparse due to the above reasons, limiting the performance of graph-based methods. Standardizing job titles before generating graphs can alleviate the sparsity issue to a certain extent, but at the cost of losing some information. To tackle these challenges, we propose to enrich graphs with structured contextual information and learn job title representations through network embedding methods. More specifically, inspired by domain-specific Named Entity tags (i.e., *RES*ponsibility and *FUN*ction) proposed in (Liu et al., 2019a), we treat the job title as a combination of responsibilities, functionalities, and other additional information. Words related to *responsibility* and *functionality* are defined as tags. We assume that job titles with the same tag describe similar job responsibilities or functionalities, making them more likely to have similar representations. Along this line, we construct *Job-Transition-Tag Graph*, a heterogeneous graph containing two types of nodes, i.e., job titles and tags, which carries more information, thereby alleviating the sparsity problem.

2 Methodology

2.1 Preliminaries

A graph/network is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V}$ and edge set $\mathcal{E}$. Nodes and edges

[1] In this paper, we use *job title* and *job* interchangeably.

can optionally have a type, so a graph can be homogeneous or heterogeneous. In the recruitment field, the career trajectory of talents can be represented by graphs. Formally, consider a job seeker set $\mathcal{U}$ and their working history set $\mathcal{H} = \{H^u\}_{u \in \mathcal{U}}$, where the working history of each u is represented as a sequence of n work records ordered by time $H^u = \{J_1, \ldots, J_n\}$. The i-th record J_i is denoted by (j_i, p_i), indicating that u is engaged in a position (titled j_i) during the p_i period. The set of job titles j_i that occurred in $\mathcal{H}$ is denoted as $\mathcal{J}$. Based on $\mathcal{H}$, *Job-Transition Graph* (Figure 1a) can be constructed, which is formally defined as:

Definition 1 (Job-Transition Graph) *is defined as a directed homogeneous graph $\mathcal{G}^{jj} = (\mathcal{J}, \mathcal{E}^{jj})$ generated from $\mathcal{H}$, where $\mathcal{J}$ is a set of job titles, and the edge $e^{jj}_{xy} \in \mathcal{E}^{jj}$ represents the job transition from the former job j_x to the next job j_y.*

2.1.1 Learning from Job-Transition Graph: An Overview

$\mathcal{G}^{jj}$ is often used for job title representation learning tasks. The current procedure is first to build a $\mathcal{G}^{jj}$, and then learn job title representation from it. More specifically, (Dave et al., 2018) first builds $\mathcal{G}^{jj}$ and the other two graphs. Then, the Bayesian personalized ranking and margin-based loss functions are used to learn job title representations from graphs. Job2Vec (Zhang et al., 2019) constructs a $\mathcal{G}^{jj}$, where the node denotes a job title affiliated with the specific company, and a multi-view representation learning method is proposed. (Zhang et al., 2021) adds company nodes in $\mathcal{G}^{jj}$ to build a heterogeneous graph. Then they use a graph neural network to represent the nodes. As mentioned in Section 1, the job title and job transition data are messy. Therefore, $\mathcal{G}^{jj}$ may be sparse (Zhang et al., 2019), which we will further prove in Section 3.1. In order to alleviate this issue, a simple method is to standardize job titles and then construct a normalized and denser graph based on the standardized job titles. For example, (Dave et al., 2018) normalizes titles by using Carotene (Javed et al., 2015), (Zhang et al., 2019) aggregates titles by filtering out low frequency words, and (Zhang et al., 2021) unifies titles according to IPOD (Liu et al., 2019a). However, the standardization of job titles may lose some specific information. Furthermore, these methods either ignore the semantic information contained in job titles (Dave et al., 2018; Zhang et al., 2021) or separate the semantic information from the graph topology (Zhang et al., 2019).

2.1.2 Job Title Composition

Generally, a job title consists of three parts (Liu et al., 2019a; Zhang et al., 2019): (i) **Responsibility**: describes the role and responsibility of a position from different levels (e.g., director, assistant, and engineer). (ii) **Functionality**: describes the business function of a position from various dimensions (e.g., sales, national and security). (iii) **Additional Information**: contains personal-specific information. We denote the words related to *responsibility* and *functionality* as tags, and they form a tag set $\mathcal{T}$. These tags are the essence of the job title and provide important information about the position. However, few works directly include this information in the representation learning scheme. In this paper, we consider these tags when generating graphs. These tags can alleviate the graph sparsity problem of $\mathcal{G}^{jj}$ and provide additional information for the learning of job title representations.

2.2 Proposed Graphs

In order to address the sparsity issue of $\mathcal{G}^{jj}$, we consider adding more information when generating graphs, i.e., tags related to job responsibilities or functionalities, driven by the job title composition. Along this line, we define various types of graphs:

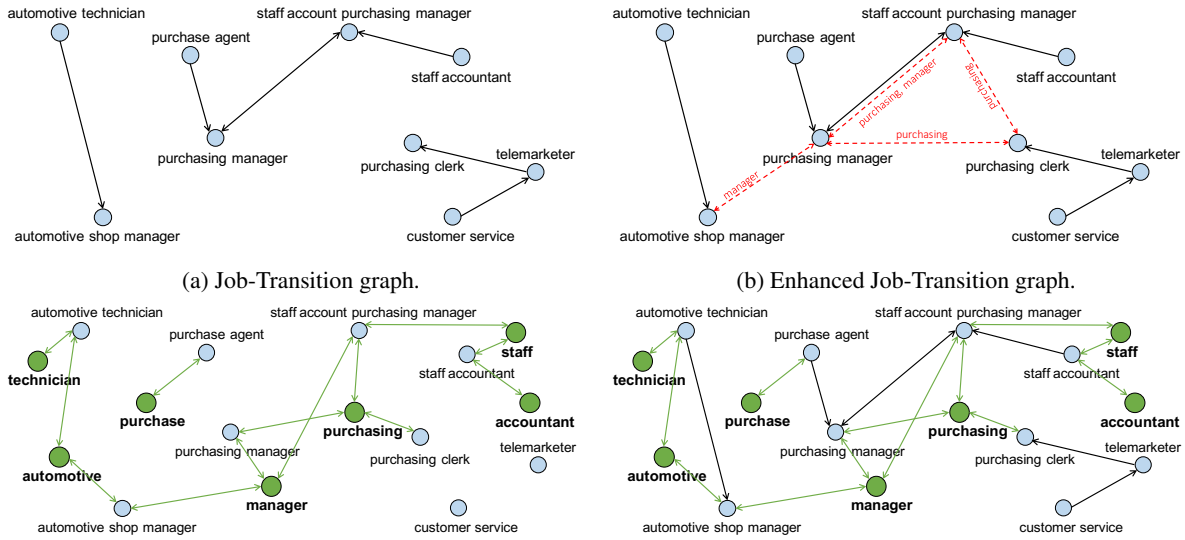
Definition 2 (Enhanced Job-Transition Graph) *is based on $\mathcal{G}^{jj}$ with additional enhanced edges. It is defined as $\mathcal{G}^{jj}_E = (\mathcal{J}, \mathcal{E}^{jj} \cup \mathcal{E}^{jj}_E)$, where $\mathcal{E}^{jj}_E$ is a set of enhanced edges. More specifically, if j_x and j_y share a tag w, then we add a bi-directional edge between them, i.e., e^{jj}_{xy} and e^{jj}_{yx}.*

As shown in Figure 1b, red dashed lines represent additional enhanced edges, e.g., "purchasing manager" shares the tag "purchasing" with "purchasing clerk", so we add edges between them.

Definition 3 (Job-Tag Graph) *is defined as a heterogeneous graph $\mathcal{G}^{jt} = (\mathcal{J} \cup \mathcal{T}, \mathcal{E}^{jt})$, with job titles and tags, two node types. $\mathcal{E}^{jt}$ is a set of bi-directional edges between a job title and a tag, representing the "has/in" relationship.*

An example of the "has/in" relationship is given in Figure 1c (i.e., the green line), where the job title "automotive technician" *has* a tag "automotive", and "automotive" is *in* "automotive technician". In order to aggregate more information, we further combine *Job-Transition Graph* and *Job-Tag Graph* to build *Job-Transition-Tag Graph*:

Definition 4 (Job-Transition-Tag Graph) *is defined as a heterogeneous graph $\mathcal{G}^{jtj} = (\mathcal{J} \cup$*



(a) Job-Transition graph.

(b) Enhanced Job-Transition graph.

(c) Job-Tag graph.

(d) Job-Transition-Tag graph.

Figure 1: Examples of four types of graphs with blue/green circles representing job titles/tags. Black lines represent job transitions, red dotted lines represent additional enhanced edges, and green lines represent "has/in" relationships.

$\mathcal{T}, \mathcal{E}^{jj} \cup \mathcal{E}^{jt}$) *with job titles and tags, two node types, and transition and "has/in", two edge types.*

Inspired by the achievements of network embedding models in the node representation learning problem (Hamilton et al., 2017), we apply different network embedding models to learn job title representation from the graphs defined above.

3 Experiments

We evaluate the proposed job title representation learning scheme through (i) a node classification task (i.e., *job title classification*) and (ii) a link prediction task (i.e., *next-job prediction*). This section will first describe the two datasets used and experimental settings, followed by discussing the results.

3.1 Datasets

CareerBuilder12 (CB12): an open dataset from a Kaggle competition. [2] It contains a collection of working experiences represented by sequences of job titles. For the node classification task, we use *AutoCoder* [3] to assign a SOC 2018 to each job title. The labeling details are given in Appendix A.1.

Randstad: a private French resume dataset provided by Randstad company, where each resume is parsed into multiple sections, e.g., *PersonalInformation*, *EducationHistory* and *EmploymentHistory*. An example is given in Figure 3 of Appendix. Graphs are built from *EmploymentHistory* section.

Both datasets use a tree-like taxonomy, as described in Appendix A.1 and A.2. For example, from root class to leaf class, the *CB12* taxonomy is organized as *MajorGroup* $\rightarrow$ *MinorGroup* $\rightarrow$ *BroadGroup* $\rightarrow$ *DetailedOccupation*. Consistent with the reality of the recruitment market, some occupations rarely appear in both datasets. To balance the data, we filter out occupations with fewer than 200 occurrences, i.e., *MinorGroup* for *CB12* and *JobGroup* for *Randstad*. Also, for graph construction, we remove working histories with less than two work records, i.e., $|H^u| < 2$.

	#C	#W	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{E}^{jj}\rvert$	$\lvert\mathcal{E}^{jj}_E\rvert$	$\lvert\mathcal{E}^{jt}\rvert$
CB12	16	1,682	9,216	20,640	6,477,819	22,149
Randstad	18	2,303	12,864	36,722	6,663,267	22,897

Table 1: Statistics of datasets and corresponding graphs, #C represents the number of categories, and #W represents the vocabulary size for node one-hot encoding.

In order to generate tags, we first tokenize titles into tokens and remove stopwords, numbers, and punctuation. Then, we use the Top-200 tokens that appear most frequently in job titles and belong to IPOD (Liu et al., 2019a) as tags. The details of tag generation are given in Appendix A.3. We assign the one-hot encoding of the corresponding title for each title node as the node feature. The vocabulary set is obtained by filtering words with a frequency of 1 from the tokenized job titles. Statistics for

[2] https://www.kaggle.com/c/job-recommendation

[3] http://www.onetsocautocoder.com/plus/onetmatch

		N2V	GCN	GAT	M2V	RGCN	HAN	W2V	BERT
CB12	$\mathcal{G}^{jj}$	0.206/0.360	0.576/0.688	0.568/0.664	0.154/0.334	0.524/0.637	0.670/0.747	0.713/0.767	0.688/0.719
	$\mathcal{G}_E^{jj}$	0.599/0.714	0.628/0.720	0.692/0.759	0.571/0.688	0.591/0.701	0.698/0.781		
	$\mathcal{G}^{jt}$	-	-	-	0.588/0.692	0.687/0.752	0.703/0.766		
	$\mathcal{G}^{jtj}$	-	-	-	0.588/0.692	0.703/0.766	**0.742/0.797**		
Randstad	$\mathcal{G}^{jj}$	0.201/0.304	0.520/0.616	0.529/0.593	0.166/0.282	0.388/0.536	0.592/0.665	0.595/0.671	0.580/0.609
	$\mathcal{G}_E^{jj}$	0.523/0.623	0.484/0.621	0.607/0.677	0.469/0.585	0.452/0.580	0.607/0.689		
	$\mathcal{G}^{jt}$	-	-	-	0.590/0.665	0.552/0.643	0.572/0.663		
	$\mathcal{G}^{jtj}$	-	-	-	0.590/0.665	0.600/0.678	**0.641/0.708**		

Table 2: Job title classification results (Macro-F1/Micro-F1). The score in bold is the best among all methods applied to all graphs, and the scores underlined are the best in all graphs of each method. For *M2V*, we report the best results obtained by the meta-path *Job-Tag-Job*.

datasets and graphs are summarized in Table 1. We can observe that $\mathcal{G}^{jj}$ (i.e., $|\mathcal{J}|$ and $|\mathcal{E}^{jj}|$) are sparse.

3.2 Experimental Settings

For *job title classification* task: we classify job titles into root categories in this work, i.e., *Major-Group* for *CB12* and *JobClass* for *Randstad*. We randomly split the data into training/validation/test sets with a ratio of 60%/20%/20%.

For *next-job prediction* task: we treat it as a link prediction task on *Job-Transition Graph* to predict whether there exists an edge (transition) between two nodes (job titles). We keep the same split ratio on positive/negative edges, where negative edges are randomly picked from unconnected node pairs (i.e., the same size as positive edges).

We evaluate the performance of our proposed learning scheme against the following baselines. A detailed description of these baselines is provided in Appendix A.4.

- **Homogeneous**: *Node2Vec (N2V)* (Grover and Leskovec, 2016), *GCN* (Kipf and Welling, 2017) and *GAT* (Velickovic et al., 2018).
- **Heterogeneous**: *Metapath2Vec (M2V)* (Dong et al., 2017), *RGCN* (Schlichtkrull et al., 2018) and *HAN* (Wang et al., 2019b).
- **Semantic-based**: *Word2Vec (W2V)* (Le and Mikolov, 2014) and *BERT* (Devlin et al., 2019).

Our implementation is based on the DGL package (Wang et al., 2019a). [4] In both tasks, for unsupervised methods, node representations are learned from the entire dataset. Then train the logistic regression classifier on both the training and validation sets. Each semi-supervised model is trained on the training set, and the parameters are optimized on the validation set. The final performance is evaluated on the test set. To ensure fairness, we keep the same data split for both methods, repeat each prediction experiment ten times, and report the average performance scores (i.e., Macro-F1 and Micro-F1 for *job title classification* and AUC for *next-job prediction*). For details of other parameter settings, see Appendix A.5.

[4] Source code will be available at `https://github.com/zhujun81/Job_title_representation`.

3.3 Results

3.3.1 Job Title Classification

Table 2 summarizes the best results of all methods on different graphs. We have the following observations: (i) Among all graphs, all models generally have the lowest scores on $\mathcal{G}^{jj}$ because this graph is often sparse and can only provide limited information. (ii) All models perform better on $\mathcal{G}_E^{jj}$ (except Macro-F1 of *GCN*) than $\mathcal{G}^{jj}$, which shows that the enhanced edges provide additional information. One interpretation for enhanced edges is adding semantic information, i.e., two job titles are more likely to be similar if they share the same word, represented by edges from the graph perspective. (iii) The heterogeneous models perform well on our proposed $\mathcal{G}^{jtj}$, which indicates that the added tag nodes can effectively improve the quality of representation. Note that we did not apply homogeneous methods to $\mathcal{G}^{jtj}$, but the results on $\mathcal{G}_E^{jj}$ prove that the information given by tags is useful. (iv) The models with attention mechanisms outperform the models without attention, demonstrating that the attention mechanism is good at capturing important information from noisy graphs.

3.3.2 Next-Job Prediction

We further evaluate the learning scheme using *next-job prediction*, which can be viewed as a link prediction task to predict whether a position will be recommended as the next-job. For unsupervised methods, edge features are represented by applying binary operators (Grover and Leskovec, 2016)

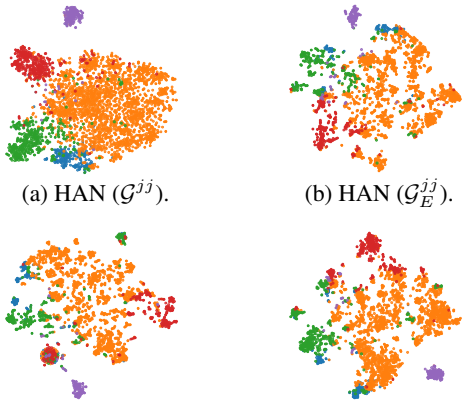
	N2V	GAN	M2V	HAN (Dot)	W2V (Dot)	BERT (Dot)	W2V (Hadamard)	BERT (Hadamard)
$\mathcal{G}^{jj}$	0.564	0.704	0.548	0.685				
$\mathcal{G}_E^{jj}$	0.692	0.789	0.593	0.792	0.763	0.477	0.777	**0.840**
$\mathcal{G}^{jt}$	-	-	0.604	0.768				
$\mathcal{G}^{jtj}$	-	-	0.604	<u>0.833</u>				

Table 3: Next-job prediction results (AUC) on *CB12*. The bold score is the best among all methods, and the underlined score is the second-best.

on node pairs, and then the best binary operator is selected based on the validation set, while the dot product is used for semi-supervised methods. The results on *CB12* given in Table 3 show the promising results of our proposed graphs. Like *job title classification*, the scores of all network embedding methods, i.e., *N2V*, *GAN*, *M2V* and *HAN* better on $\mathcal{G}_E^{jj}$ compared to $\mathcal{G}^{jj}$, and the heterogeneous models perform best on $\mathcal{G}^{jtj}$. Such results further demonstrate the effectiveness of our proposed method for constructing graphs, whether adding additional information based on tags (i.e., $\mathcal{G}_E^{jj}$) or directly adding tags to the graph (i.e., $\mathcal{G}^{jt}$ and $\mathcal{G}^{jtj}$). *BERT* using Hadamard operator performs best, followed by *HAN* on $\mathcal{G}^{jtj}$ with a slight difference of 0.007. However, when we use the dot product used in *HAN* to obtain edge features for *BERT*, the AUC of *BERT* drops sharply to 0.477, while *W2V* only drops a little to 0.763. We will discuss such results in future work. Overall, the results of link prediction also demonstrate the effectiveness of our proposed graphs.

3.3.3 Visualization

For a more intuitive comparison, we select five occupations and then visualize the job title representations learned by *HAN* in Figure 2, with each color corresponding to an occupation category. Overall, the representations learned by *HAN* on all graphs are clustered into groups. However, when considering tags, representations are easier to be subdivided further in each category. For example, in Figure 2d, the orange occupation can be further divided into three sub-clusters, which proves that adding tag nodes can help capture more detailed information and make the learned representation more informative. This detailed information helps further categorize occupations, as we only classify job titles into the root category (i.e., *MajorGroup*) in this work.



(a) HAN ($\mathcal{G}^{jj}$). (b) HAN ($\mathcal{G}_E^{jj}$). (c) HAN ($\mathcal{G}^{jt}$). (d) HAN ($\mathcal{G}^{jtj}$).

Figure 2: Visualization of representations (CB12). *Healthcare support* (green), *Healthcare practitioners and technical* (blue), *Architecture and engineering* (purple), *Office and administrative support* (orange) and *Transportation and material handling* (red).

4 Conclusion

This paper first proposes to enrich *Job-Transition Graph* commonly used in job title representation learning tasks by adding tag-related information or directly adding tag nodes, and then learn job title representations through network embedding methods. This enhancing method can alleviate the sparsity problem in *Job-Transition Graph*, thereby improving the quality of learned representations, as demonstrated in the experimental results of *job title classification* and *job title classification*. In future work, we would like to explore why the Hadamard operator and dot product lead to such different link prediction results for *BERT*. Furthermore, other research lines are (i) considering edge weights when learning from graphs, (ii) classifying job titles into different occupational levels, and (iii) improving the tag generation approach.

Acknowledgment

This work is supported by the Randstad research chair in collaboration with MICS Lab, CentraleSupélec, Université Paris-Saclay. We would like to thank the *Mésocentre* [5] computing center of CentraleSupélec and École Normale Supérieure Paris-Saclay for providing computing resources.

[5] http://mesocentre.centralesupelec.fr/

References

Vachik S. Dave, Baichuan Zhang, Mohammad Al Hasan, Khalifeh AlJadda, and Mohammed Korayem. 2018. A combined representation learning approach for better job and skill recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1997–2005. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 135–144. ACM.

Jean-Philippe Fauconnier. 2015. French word embeddings.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *ArXiv preprint*, abs/1709.05584.

Faizan Javed, Qinlong Luo, Matt McNair, Ferosh Jacob, Meng Zhao, and Tae Seung Kang. 2015. Carotene: A job title classification system for the online recruitment domain. In *2015 IEEE First International Conference on Big Data Computing Service and Applications*, pages 286–293. IEEE.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Quoc V. Le and Tomás Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.

Junhua Liu, Yung Chuen Ng, Kristin L Wood, and Kwan Hui Lim. 2019a. Ipod: An industrial and professional occupations dataset and its applications to occupational data mining and analysis. *ArXiv preprint*, abs/1910.10495.

Mengshu Liu, Jingya Wang, Kareem Abdelfatah, and Mohammed Korayem. 2019b. Tripartite vector representations for better job recommendation. *ArXiv preprint*, abs/1907.12379.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019a. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *ArXiv preprint*, abs/1909.01315.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019b. Heterogeneous graph attention network. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2022–2032. ACM.

Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

Denghui Zhang, Junming Liu, Hengshu Zhu, Yanchi Liu, Lichen Wang, Pengyang Wang, and Hui Xiong. 2019. Job2vec: Job title benchmarking with collective multi-view representation learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 2763–2771. ACM.

Le Zhang, Ding Zhou, Hengshu Zhu, Tong Xu, Rui Zha, Enhong Chen, and Hui Xiong. 2021. Attentive heterogeneous graph embedding for job mobility prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2192–2201.

Jun Zhu, Gautier Viaud, and Céline Hudelot. 2021. Improving next-application prediction with deep personalized-attention neural network. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1615–1622. IEEE.

A Appendix

A.1 Job Title Label Assignment

Job titles are not pre-labeled in the original working experience dataset provided by CareerBuilder12. Therefore, for the job title classification task, we use an online third-party API O*Net-SOC AutoCoder [6] to assign a Standard Occupation Classification code (SOC) 2018 to each job title, as well as a match score (i.e., scores above 70 means that the correct code is accurately predicted at least 70% of the time). SOC 2018 is a four-level taxonomy structure, including *MajorGroup* (23), *MinorGroup* (98), *BroadGroup* (459) and *DetailedOccupation* (867). For example, *O*Net-SOC AutoCoder* assigns the code 11-2022 (Sales Managers) for the title "sales director", which belongs to the level of *DetailedOccupation*. 11-2020 (Marketing and Sales Managers) is *BroadGroup* level, 11-2000 (Advertising, Marketing, Promotions, Public Relations, and Sales Managers) is *MinorGroup* level, and 11-0000 (Management Occupations) is *MajorGroup* level. In this work, we categorize job titles into *MajorGroup*. We have annotated a total of 30,000 job titles. The developer guarantees that the code assigned to the title plus description has an accuracy rate of 85%. However, only the job title is provided in our experiments, so the SOC 2018 code may be incorrectly assigned. For this reason, we filtered out job titles with scores below 70. Therefore, 12,908 unique job titles remain.

A.2 Randstad Data Description

Figure 3 shows an example of parsed resume in *Randstad* dataset. We build graphs from *EmploymentHistory*, which contains a *JobTitle*, and its corresponding occupation labels (i.e., *JobCode*, *JobGroup* and *JobClass*). The hierarchical taxonomy structure used in the Randstad dataset has a three-level hierarchy, where *JobCode*s are leaf classes, and each internal class (i.e., *JobGroup*) or root class (i.e., *JobClass*) is the aggregation of all its descendant classes. There are 25 *JobClass*s, 295 *JobGroup*s and 4,443 *JobCode*s, respectively. In this work, we categorize job titles into *JobClass*.

A.3 Tag Generation

For both datasets, we first tokenize titles into tokens and remove stopwords, numbers, and punctuation. The word frequency distribution of words in two datasets are shown respectively in Figure 4, which are subject to the long-tail distribution, similar to the observation in (Zhang et al., 2019). Most words appear only once, i.e., 53.55% of words only appear one time in *CB12* dataset, and this ratio is 56.55% in *Randstad* dataset. Figure 4 further shows the top ten and last ten frequent words in each dataset. Obviously, high-frequency words like "manager" and "sales" describe the responsibility or functionality of the job title, while low-frequency words are usually noise or person-specific words. Based on the domain-specific NE tags (i.e., *RES*ponsibility, *FUN*ction) proposed in IPOD (Liu et al., 2019a), we then select the Top-200 tokens that appear most frequently and appear in the IPOD NE tag set as tags for each dataset.

[6] http://www.onetsocautocoder.com/plus/onetmatch

PersonalInformation
- **Name**
- **Address**

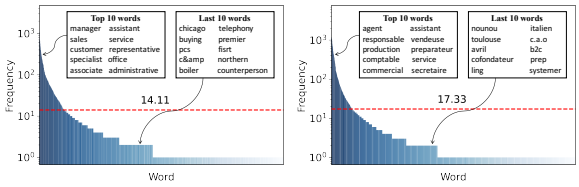
EducationHistory

EducationItem
- EducationLevelCode: *BAC2*
- DegreeDirection: *Technicien en maintenance industrielle*
- StartDate: *2017-09-01*
- EndDate: *2018-06-30*
- InstituteName: *AFPA MEUDON 92*

EmploymentHistory

EmploymentItem
- Description: *Contrôle des cartes électroniques et changes des composants électroniques ...*
- StartDate: *2014-01-01*
- EndDate: *2015-12-31*
- JobTitle: *Technicien électronique*
- EmployerName: *EBO (Courneuve) 93*
- JobCode: *Technicien Électronique (h/f)*
- JobGroup: *Ingénieurs, Projeteurs et Techniciens Électricité*
- JobClass: *Ingénierie*

Figure 3: An example of parsed resume in Randstad.



(a) CB12. (b) Randstad.

Figure 4: Word frequency distribution, where the red line represents the average value. *Top-10* words are sorted by frequency, and *Last-10* are randomly selected from the words with a frequency of 2.

A.4 Baseline Description

We explore the following network embedding methods on our proposed graphs to learn job title representation. According to the type of graph, the network embedding methods are naturally divided into *Homogeneous* and *Heterogeneous*. Then, we further categorize each category into *Unsupervised* and *Semi-Supervised* according to whether node

labels are provided for learning.

Homogeneous&Unsupervised

- *Node2Vec (N2V)* (Grover and Leskovec, 2016): is an extension of DeepWalk with a biased random walk process for neighborhood exploration.

Homogeneous&Semi-supervised:

- *GCN* (Kipf and Welling, 2017): is a semi-supervised Graph Neural Network (GNN) that generalizes the convolutional operation to homogeneous graphs.
- *GAT* (Velickovic et al., 2018): uses a self-attention strategy to learn the importance between a node and its neighbors.

Heterogeneous&Unsupervised:

- *Metapath2Vec (M2V)* (Dong et al., 2017): performs meta-path-guided walks and utilizes Skip-Gram to embed heterogeneous graphs.

Heterogeneous & Semi-supervised:

- *RGCN* (Schlichtkrull et al., 2018): is an extension of *GCN* on heterogeneous graphs, introducing relation-specific transformations based on the type of edges.
- *HAN* (Wang et al., 2019b): proposes a hierarchical attention mechanism, i.e., node-level and semantic-level for heterogeneous graphs.

In addition to the comparison between network embedding methods, we will also compare the representation learned through graphs with the representation obtained by semantic-based methods.

Semantic-based:

- *Word2Vec (W2V)* (Le and Mikolov, 2014): the representation of a job title is obtained by averaging word vectors in it. We use word vectors trained on Google News [7] for *CB12*, and a pre-trained French embedding model (Fauconnier, 2015) for *Randstad*.
- *BERT* (Devlin et al., 2019): the job title representations are obtained by using the bert-as-service package (Xiao, 2018), a sentence encoding service for mapping variable-length sentences to fixed-length vectors. We default to using the pre-trained BERT models provided by the package, i.e., BERT-Base-Uncased is used for *CB12*, and BERT-Base-Multilingual-Cased (New) for *Randstad*.

[7] https://code.google.com/archive/p/word2vec/

A.5 Parameter Settings

Our implementation is based on the PyTorch version of the DGL package (Wang et al., 2019a). For *job title classification*, we randomly split the data into training/validation/test sets with a ratio of 60%/20%/20%. We keep the same split ratio on positive/negative edges for *next-job prediction*, where negative edges are randomly picked from unconnected node pairs (i.e., the same size as positive edges). To ensure fairness, we keep the same data split for all methods, and we set the node embedding to 128 for all methods, except for *W2V*.

In the *job title classification* and *next-job prediction* tasks, for unsupervised methods, node representations are learned from the entire dataset. The logistic regression classifier is then trained on both the training and validation sets. Each semi-supervised model is trained on the training set, and the parameters are optimized on the validation set. The final performance is evaluated on the test set. Models are optimized with the Adam (Kingma and Ba, 2015) with a learning rate of 1e-3, and we apply L_2 regularization with value 5e-4. We use an early stop with a patience of 100, i.e., if the validation loss does not decrease in 100 consecutive epochs, we stop training. For models applying the attention mechanism, the dropout rate of attention is set to 0.2. For random-walk-based methods, including *N2V* and *M2V*, we set the window size to 5, walk length to 10, walks per node to 50, the number of negative samples to 5. For *M2V*, we test all meta-paths and report the best performance. For *next-job prediction* task, edge features are represented by applying binary operators (Grover and Leskovec, 2016) on pairs of nodes, and then the best operator is chosen based on the validation set, while the dot product is used for all semi-supervised methods.

We repeat each prediction experiment ten times and report the average performance scores (i.e., Macro-F1 and Micro-F1 for *job title classification* and AUC for *next-job prediction*).