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) 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., RESPONSibility and FUNCTIONtion) 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 $G = (V, E)$, with node set $V$ and edge set $E$. Nodes and edges
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 \( G_{jj} = (\mathcal{J}, \mathcal{E}_{jj}) \) generated from \( \mathcal{H} \), where \( \mathcal{J} \) is a set of job titles, and the edge \( e_{xy}^{jj} \in \mathcal{E}_{jj} \) 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

\( G_{jj} \) is often used for job title representation learning tasks. The current procedure is first to build a \( G_{jj} \), and then learn job title representation from it. More specifically, (Dave et al., 2018) first builds \( G_{jj} \) 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 \( G_{jj} \), 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 \( G_{jj} \) 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, \( G_{jj} \) 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 \( T \). 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 \( G_{jj} \) and provide additional information for the learning of job title representations.

### 2.2 Proposed Graphs

In order to address the sparsity issue of \( G_{jj} \), 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 \( G_{jj} \) with additional enhanced edges. It is defined as \( G_{jj}^E = (\mathcal{J}, \mathcal{E}_{jj}^E \cup \mathcal{E}_{jj}^E) \), 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., \( e_{xy}^{jj} \) and \( e_{yx}^{jj} \).

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 \( G_{jt} = (\mathcal{J} \cup \mathcal{T}, \mathcal{E}_{jt}) \), 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 \( G_{jjt} = (\mathcal{J} \cup \mathcal{T}, \mathcal{E}_{jjt}) \),
The vocabulary set is obtained by filtering words with a frequency of 1 from the tokenized job titles. Statistics for

\[ (b) \text{Enhanced Job-Transition graph.} \]

\[ (d) \text{Job-Transition-Tag graph.} \]

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.

\[ \mathcal{T}, \mathcal{E}^{ij} \cup \mathcal{E}^{ij} \] 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. 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., \(|H^u|<2\).

|        | #C | #W | \(|J|\) | \(|E^{ij}|\) | \(|E^{ij}|\) | \(|E^{ij}|\) |
|--------|----|----|--------|--------|--------|--------|
| CB12   | 16 | 1,682 | 9,216 | 9,216 | 6,477,819 | 22,149 |
| Randstad | 18 | 2,303 | 12,864 | 6,722 | 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
datasets and graphs are summarized in Table 1. We can observe that \( G^{jj} \) (i.e., \(|J|\) and \(|E^{jj}|\) 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). 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.

### 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 \( G^{jj} \) because this graph is often sparse and can only provide limited information. (ii) All models perform better on \( G^{jj} \) (except Macro-F1 of GCN) than \( G^{jj} \), 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 \( G^{jj} \), which indicates that the added tag nodes can effectively improve the quality of representation. Note that we did not apply homogeneous methods to \( G^{jj} \), but the results on \( G^{jj} \) 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).
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.

<table>
<thead>
<tr>
<th>Graph</th>
<th>N2V</th>
<th>GAN</th>
<th>M2V</th>
<th>HAN (Dot)</th>
<th>W2V (Dot)</th>
<th>BERT (Dot)</th>
<th>W2V (Hadamard)</th>
<th>BERT (Hadamard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{G}^{ij}_j$</td>
<td>0.564</td>
<td>0.704</td>
<td>0.548</td>
<td>0.685</td>
<td>0.763</td>
<td>0.477</td>
<td>0.777</td>
<td>0.840</td>
</tr>
<tr>
<td>$\mathcal{G}^{ij}_E$</td>
<td>0.692</td>
<td>0.789</td>
<td>0.593</td>
<td>0.792</td>
<td>-</td>
<td>-</td>
<td>0.768</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{G}^{jt}_j$</td>
<td>-</td>
<td>-</td>
<td>0.604</td>
<td>0.768</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{G}^{jtj}_j$</td>
<td>-</td>
<td>-</td>
<td>0.604</td>
<td>0.833</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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).

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.

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, Centrale-Supélec, Université Paris-Saclay. We would like to thank the MesoCentre computing center of CentraleSupélec and École Normale Supérieure Paris-Saclay for providing computing resources.

Acknowledgment

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

http://mesocentre.centralesupelec.fr/
References


Jean-Philippe Fauconnier. 2015. French word embeddings.


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. 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., RESponsibility, FUNction) 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.

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

\[\text{Figure 3: An example of parsed resume in Randstad.}\]

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

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 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.

### 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 un-connected 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).