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

Anonymous ACL submission

Abstract

Works on learning job title representation are mainly based on Job-Transition Graph, built from the working history of talents. However, since the 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 functions). Along this line, we reformulate job title representation learning as the task of learning node embedding on the Job-Transition-Tag Graph. Experiments on a public CareerBuilder12 dataset and a private Randstad dataset show interest of our approach.

1 Introduction

The learning of job title 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). 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., the resume parser is 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 approaches that aggregate (e.g., mean or sum) word representations to get job title semantic representation may lead to mismatches. Moreover, semantic-based 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, 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. Specifically, inspired by domain-specific Named Entity tags (i.e., RESPONSIBILITY and FUNCTION) 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 functions or responsibilities, so they are 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. The experimental results verify that the added nodes can improve the quality of job title representation.

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 $U$ and their working history set $H = \{H^u\}_{u \in U}$, 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 $(j_i, p_i, o_i)$, indicating that $u$ is engaged in a position (titled $j_i$) during the $p_i$ period. $o_i$ represents other information related to this work record, like company name and job content. The set of job titles $j_i$ that occurred in $H$ is denoted as $J$. Based on $H$, Job-Transition Graph (Figure 1a) can be constructed, which is formally defined as follows:

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

### 2.1.1 Learning from Job-Transition Graph: An Overview

Job-Transition-Graph is often used for job title representation learning tasks. The current procedure is to first build a Job-Transition-Graph, and then learn job title representation from it. More specifically, (Dave et al., 2018) first build $G^{jj}$ and 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 job title affiliated with the specific company, and a multi-view representation learning method is proposed. (Zhang et al., 2021) add company nodes in $G^{jj}$ to build a heterogeneous graph. Then they use an attentive graph neural network to represent the company and job titles. As mentioned above, the job title and job transition data are messy. Therefore, $G^{jj}$ may be sparse (Zhang et al., 2019). To alleviate this issue, a simple method is to standardize 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). Another method is to consider semantic information in addition to the graph topology, e.g., (Zhang et al., 2019) enforces job title representations to be close to each other if they share similar words. However, these methods either ignore the semantic information contained in job titles or separate the semantic information from the topology structure.

### 2.1.2 Job Title Composition

Generally speaking, a job title usually 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. For example, job titles with the same tag are more likely to describe the same level of ability/authority or belong to the same industry. However, few works directly include this information in the representation learning scheme. In this paper, we consider these tags when generating graphs from working history. These tags can alleviate the graph sparsity problem of Job-Transition-Graph and provide additional information for the task of job title representation learning.

### 2.2 Proposed Graphs

In order to address the sparsity issue of Job-Transition Graph mentioned above, we consider adding more information when generating graphs, i.e., tags related to the job responsibility or function, driven by the composition of job titles. Along this line, we define various types of graphs as follows:

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

As shown in Figure 1b, red dashed line 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 $G^{jt} = (J \cup T, E^{jt})$, with job titles and tags, two node types. $E^{jt}$ is a set of bi-directional edges between a job title and a tag, representing the “has/in” relationship.

An example is given in Figure 1c, the job title “automotive technician” has a tag “automotive”,
so the bi-directional edge $e^j_i$ (i.e., the green line) means that “automotive technician” has the 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 $G^{jtj} = (\mathcal{J} \cup \mathcal{T}, \mathcal{E}^{ij} \cup \mathcal{E}^{j})$, with job titles and tags, two node types, and 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

#### 3.1 Datasets

Here, we provide details about our two datasets: 

- **CareerBuilder12 (CB12):** an open dataset from a Kaggle competition\(^1\). It contains a collection of working experiences represented by sequences of job titles. For the node classification task, we use AutoCoder \(^2\) 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, an example is given in Figure 3 of Appendix. Graphs are built from EmploymentHistory section.

For both datasets, we use the Top200 tokens that appear most frequently in job titles 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. The statistics of datasets and graphs are summarized in Table 1. We can observe that the generated Job-Transition Graphs (i.e., $|\mathcal{J}|$ and $|\mathcal{E}^{j}|$) are sparse.

|                | #C | #W | $|\mathcal{J}|$ | $|\mathcal{E}^{j}|$ | $|\mathcal{E}^{ij}|$ | $|\mathcal{E}^{j}|$ |
|----------------|----|----|----------------|----------------|----------------|----------------|
| CB12           | 16 | 1,682 | 9,216 | 20,640 | 6,475,850 | 22,108 |
| Randstad       | 18 | 2,303 | 12,864 | 36,722 | 6,663,267 | 22,230 |

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.

#### 3.2 Experimental Settings

We evaluate job title representations obtained through the node classification task. The baselines used are listed below, and the detailed description is given in Appendix A.4:

- **Homogeneous:** Node2Vec (Grover and Leskovec, 2016), GCN (Kipf and Welling, 2016) and GAT (Veličković et al., 2017).

- **Heterogeneous:** metapath2vec (Dong et al., 2017), RGCN (Schlichtkrull et al., 2018) and HAN (Wang et al., 2019b).
### 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>CB12</th>
<th>Rankstaid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Node2Vec</td>
<td>GCN</td>
</tr>
<tr>
<td>$G_{jj}^{13}$</td>
<td>0.206/0.360</td>
<td>0.576/0.688</td>
</tr>
<tr>
<td>$G_{jj}^{1j}$</td>
<td>0.599/0.714</td>
<td>0.628/0.720</td>
</tr>
<tr>
<td>$G_{jtj}^{1j}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$G_{jtj}^{1j}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$G_{jtj}^{1j}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$G_{jtj}^{1j}$</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Semantic-based**: Word2Vec (Le and Mikolov, 2014). The representation of a job title is obtained by averaging word vectors in it.

Our implementation is based on the DGL package (Wang et al., 2019a). The details of parameter settings are given in 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 usually have the lowest scores on $G_{jj}^{13}$ because this graph is often sparse and can only provide limited information. (ii) All models perform better on $G_{jj}^{1j}$ (except Macro-F1 of GCN) than $G_{jtj}^{1j}$, which shows that the enhanced edges provide additional information. One interpretation of enhanced edges is to add semantic information, i.e., if two job titles share the same word, they are more likely to be similar, which is represented by edges from the graph perspective. (iii) The heterogeneous models perform well on our proposed $G_{jtj}^{1j}$, which indicates that the added tag nodes can effectively improve the quality of representation. Note that we did not apply homogeneous methods to $G_{jtj}^{1j}$, but the results on $G_{jtj}^{1j}$ prove that the information given by tags is useful. For Metapath2Vec, we report the best results obtained by the meta-path Job-Tag-Job. (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 Visualization

For a more intuitive comparison, we visualize the learned representations in Figure 2, and each color corresponds 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, as shown in Figure 2d, the administrative occupation (orange) can obviously 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 to classify the positions further because we only classify job titles into the root category (i.e., MajorGroup) in this work.

![Figure 2: Visualization of representations (CB12).](image)

(a) HAN ($G_{jj}^{13}$). (b) HAN ($G_{jj}^{1j}$). (c) HAN ($G_{jtj}^{1j}$). (d) HAN ($G_{jtj}^{1j}$).
References


Jean-Philippe Fauconnier. 2015. French word embeddings.


Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In International conference on machine learning, pages 1188–1196. PMLR.


A Appendix

A.1 Job Title Label Assignment

In the original working experience dataset provided by CareerBuilder12, job titles are not pre-labeled. Therefore, for the job title classification task, we use an online third-party API O*Net-SOC AutoCoder 3 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, 22,590 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 JobCodes are leaf classes, and each internal (JobGroup)/root class (JobClass) is the aggregation of all its descendant classes. There are 25 JobClasses, 295 JobGroups and 4,443 JobCodes, 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. Then, we select the Top-200 tokens with the highest frequency as TOP tags for each dataset.

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 (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, 2016): is a semi-supervised GNN that generalizes the convolutional operation to homogeneous graphs.
- **GAT** (Veličković et al., 2017): uses a self-attention strategy to learn the importance between a node and its neighbors.

Heterogeneous & Unsupervised:

- **Metapath2Vec** (Dong et al., 2017): performs meta-path-guided random walk 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 \(^4\) for *CB12*, and a pre-trained French embedding model (Fauconnier, 2015) 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 split the data into training/validation/test sets with a ratio of 60%/20%/20%. To ensure fairness, we keep the same data split for both methods. Each semi-supervised model was trained on the training set, and the parameters were optimized on the validation set. The final performance was evaluated on the test set. Models are optimized with the Adam (Kingma and Ba, 2014) 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 a fair comparison, we set the dimension of node embedding to 128 for all the above methods, except for *Word2Vec*. For unsupervised methods, node representations are learned from the entire dataset. Then train the logistic regression classifier simultaneously on the training set and validation set. For random-walk-based methods include *Node2Vec* and *metapath2vec*, we set the window size to 5, walk length to 10, walks per node to 50, the number of negative samples to 5. For *metapath2vec*, we test all meta-paths and report the best performance. We repeat each prediction experiment ten times and report the average performance in terms of Macro-F1 and Micro-F1 scores.

\(^4\)https://code.google.com/archive/p/word2vec/