JAMES: Job Title Mapping with Multi-Aspect Embeddings and Reasoning

Michiharu Yamashita  
The Pennsylvania State University  
University Park, PA, USA  
michiharu@psu.edu

Jia Tracy Shen  
The Pennsylvania State University  
University Park, PA, USA  
jqs5443@psu.edu

Hamoon Ekhtiari  
FutureFit AI  
Toronto, Canada  
hamoon@futurefit.ai

Thanh Tran  
Amazon  
Pittsburgh, PA, USA  
tdt@amazon.com

Dongwon Lee  
The Pennsylvania State University  
University Park, PA, USA  
dongwon@psu.edu

ABSTRACT

One of the most essential tasks needed for various downstream tasks in career analytics (e.g., career trajectory analysis, job mobility prediction, and job recommendation) is Job Title Mapping (JTM), where the goal is to map user-created (noisy and non-standard) job titles to predefined and standard job titles. However, solving JTM is domain-specific and non-trivial due to its inherent challenges: (1) user-created job titles are messy, (2) different job titles often overlap their job requirements, (3) job transition trajectories are inconsistent, and (4) the number of job titles in real-world applications is large-scale. Toward this JTM problem, in this work, we propose a novel solution, named as JAMES, that constructs three unique embeddings of a target job title: topological, semantic, and syntactic embeddings, together with multi-aspect co-attention. In addition, we employ logical reasoning representations to collaboratively estimate similarities between messy job titles and standard job titles in the reasoning space. We conduct comprehensive experiments against ten competing models on the large-scale real-world dataset with more than 350,000 job titles. Our results show that JAMES significantly outperforms the best baseline by 10.06% in Precision@10 and by 17.52% in NDCG@10, respectively.

1 INTRODUCTION

Background. The recent growth of technology has witnessed an increasing popularity of online professional platforms—e.g., LinkedIn has reportedly more than 675 million users from 200 countries [20]. With a wealth of career-related data available on such platforms (e.g., job seekers’ resumes or job advertisements), many companies have been exploiting AI techniques to improve career-domain applications (e.g., next career prediction [25], job recommendation [16], and career analysis [18]), helping job seekers to land on their ideal jobs and companies to recruit role-fitting talents. However, there is one critical step in the workflow of such applications as illustrated in Figure 1. Before building models for any downstream tasks, one has to sort out and consolidate various entities found in raw data, especially job titles. For instance, a position called “user interface engineer” in a company A and another position “front end developer” in a company B may refer to the identical job. In this work, we refer to this problem as the Job Title Mapping (JTM) (to be defined in Section 3.1).

Challenges. Despite the simple nature of the JTM problem, however, solving it in practice is non-trivial due to several challenges. (Challenge 1) User-written job titles (collected from resumes) are often messy due to non-standard naming convention. A job title “software developer” in one resume could be written as “SDE” in another. Similarly, creative job titles such as “data geek” or “strategic futurist” that people list in their resumes do not necessarily appear in the industry-wide job title taxonomy. (Challenge 2) Different job titles can be semantically related through overlapping skills. For example, “data scientist”, a frequently appearing job title today, possess a set of skills such as programming, mathematical modeling, and statistics. These skills may also overlap with the skills of other jobs—e.g., “software developer” and “business analyst” [9]. As such, in solving JTM, simply comparing the names of jobs is not necessarily helpful. (Challenge 3) Job transition trajectories in a job seeker’s resume are often inconsistent and incomplete. Many users on platforms do not update their historical jobs until they seek new ones, yielding incomplete job records. (Challenge 4) While the number of job titles in an industry job taxonomy is in the thousands, the number of job titles that one encounters in platforms is orders of magnitude larger. Yet, existing solutions used either small-scale or company-created (not user-written) job title datasets, in which JTM was solved by manual labeling/cleaning or text normalization processes. For example, [36] used a dataset with only 26 unique job titles for similar expertise job matching, while [8] used a dataset of 4,325 unique job titles for job and skill recommendation task. Recently [20] built a 30,000 job title taxonomy on LinkedIn for a job understanding task. However, little is known about how the JTM task with the aforementioned challenges can be solved.

Proposed Ideas. Toward these challenges, in this paper, we propose JAMES (Job Title Mapping with Multi-aspect Embeddings and reasoning) to solve the JTM task. We use a large-scale and real-world career dataset with more than 350,000 job titles that a
private internet company has collected and shared with us. Specifically, JAMES considers three unique multi-aspect (i.e., syntactic, semantic, and topological) embeddings of candidate job titles.

First, we build a syntactic embedding to account for the string-2-string similarity between two input job titles (Challenge 1). Second, we use a pretrained BERT embedding to account for the semantic similarity between two candidate job titles, which can identify contextually-related job titles with overlapping skills (Challenge 2). Third, we use a hyperbolic graph embedding to capture the latent topological knowledge dependencies in a job title hierarchy (Challenge 3). Later, with improved accuracies in the link prediction task, we demonstrate that the hyperbolic graph embedding helps mitigate the problem of incomplete and inconsistent job transition patterns. After building multi-aspect embeddings using our large-scale resume dataset (Challenge 4), then we develop multi-aspect co-attention mechanism which considers all three embeddings. Finally, we design a neural collaborative reasoning [7] that takes the multi-aspect embeddings as an input and produces reasoning-based multi-aspect embeddings.

In short, we make the following contributions in this paper:

- We introduce a career-domain specific preprocessing task, Job Title Mapping (JTM) using our large-scale, real-world, and user-generated dataset, consisting of 350,000 unique job titles. Our JTM task is 30 orders of magnitude larger than the largest previous JTM task.
- To solve the JTM task, we propose a novel model, JAMES, that employs multi-aspect embeddings and reasoning representations accounting for syntactic, semantic, and topological embeddings.
- We conduct extensive experiments and demonstrate the effectiveness of JAMES against ten competing baseline models. For instance, JAMES significantly outperforms the best baseline by 10.06% in Precision@10 and by 17.52% in NDCG@10, respectively.
- We apply JAMES to downstream career-domain tasks and report the findings and implications.

2 RELATED WORK

Job Title Classification. The task of classifying job titles has been studied mainly in industry. [32] proposed CNN-based job title classification where they made three types of text vectors using job description dataset. [38] built a KNN model for this task using Word2Vec. [24] made a job transition graph and Random Walk-based vector. However, they did not account for job transition trajectories. [35] proposed Job Title Benchmarking from job transition records, which is a reasonable embedding in the current society. However, they used just IT and Finance domains. In the IT dataset, they only chose 15 famous companies such as Facebook, Google, and Microsoft, which might be easy to analyze but is not realistic. Hence, our study focuses on more realistic and large scale dataset from industry.

Representation Learning in Career Domain. As rising representation learning approach in multiple domains such as graph and NLP, a career domain has recently been explored. [22] tried career path prediction with multiple social media. Their features are from multiple social network features such as demography, LIWC, and user discussion topics. For a job skill representation, [31] deployed market-aware skill extraction system "Job2Skills" that is able to consider the salient level of a skill and to extract important skill entities from job postings and target members using multi-resolution. [28] developed a person-job fit model applying a word-level semantic representation for both job requirements and job seekers’ experiences based on RNN. Those studies used rich text dataset such as job description and user profiles, but in a real world, we do not necessarily have all companies’ job description dataset, and especially for talent recruiting, we often only have career trajectory dataset (i.e., job seekers’ resumes). Hence, we aim to develop practical and useful embedding using only resume dataset.

Hyperbolic Machine Learning. Hyperbolic geometry is a non-Euclidean geometry that focuses spaces of constant negative Gaussian curvature. Hyperbolic space is recently used to develop embedding and machine learning models on hierarchical and/or graph structures because of multiple benefits such as embedding on small dimensions [11]. [26] proposed Poincare embedding wherein hierarchical data can be represented more than the Euclidean. [6] developed hyperbolic technique for graph convolutional networks. In our case, career trajectory dataset can be a graph as used in [36]. However, to the best of our knowledge, there has been no prior work on applying hyperbolic geometry to the career transition data. Hence, we use hyperbolic embedding in our model and compare the baselines.

Multi-Aspect Embeddings. Using multi-aspect embeddings such as multi-view and multi-modal is becoming common in interdisciplinary domains [33]. [21] is a review paper about multi-view biological data integration, in which they surveyed omics and clinical data integration techniques. The multi-view data integration is beneficial for genomic and clinical studies. [15] developed the contrastive multi-view representation learning to improve contrastive representation for graph tasks by contrasting node and graph encodings from first-order neighbors and a general graph diffusion. In our scenario, our target is job title mapping and we think of the topological representation, semantic representation, and syntactic representation as our multi-aspect embeddings.

3 PRELIMINARY

Major notations of the paper are listed in Table 1.

3.1 Problem Definition

We formally define the JTM task as follows:

\[
\text{Job Title Mapping (JTM): Given a set of job titles } \mathcal{X} \text{ and a set of predefined and standardized job titles in a job taxonomy } \mathcal{Y}, \text{ the job Title Mapping task aims to build a function } f(\cdot) \text{ that produces matching probabilities from each job title } j_i \in \mathcal{X} \text{ to each standardized job title } v_k \in \mathcal{Y}.
\]
Table 1: Definition of our notations used in the paper

| Notation | Definition |
|----------|------------|
| 𝐖         | Job title from resume |
| 𝑨         | A set of job titles from all resumes |
| 𝑣         | Standardized job title from the ground truth |
| 𝑦         | A set of standardized job titles from the ground truth |
| 𝐓         | Hyperbolic graph embeddings |
| 𝑊         | BERT embeddings |
| 𝑋         | Syntactic embeddings |
| 𝐴(𝑏𝑏)     | Affinity matrices between 𝑋𝑏 and 𝑋𝑏 |
| 𝐴(ℎ𝑠)     | Affinity matrices between 𝑋ℎ and 𝑋𝑠 |
| 𝐴(𝑏𝑠)     | Affinity matrices between 𝑋𝑏 and 𝑋𝑠 |
| 𝑊′(𝑏𝑏)    | Learnable weights between 𝑋𝑏 and 𝑋𝑏 |
| 𝑊′(ℎ𝑠)    | Learnable weights between 𝑋ℎ and 𝑋𝑠 |
| 𝑊′(𝑏𝑠)    | Learnable weights between 𝑋𝑏 and 𝑋𝑠 |
| 𝑘𝑏        | Attention graph for 𝑋𝑏 , acknowledging supports from 𝑋𝑏 and 𝑋𝑡 through 𝐴(𝑏𝑏) and 𝐴(ℎ𝑠) |
| 𝑘𝑝        | Attention graph for 𝑋𝑝 , acknowledging supports from 𝑋𝑝 and 𝑋𝑡 through 𝐴(ℎ𝑝) and 𝐴(ℎ𝑠) |
| 𝑘𝑖        | Attention graph for 𝑋𝑖 , acknowledging supports from 𝑋𝑖 and 𝑋𝑡 through 𝐴(𝑏𝑖) and 𝐴(ℎ𝑖) |
| 𝑋𝑟        | Co-attentive multi-aspect embeddings from 𝑋𝑟 |
| 𝑋𝑡        | Co-attentive multi-aspect embeddings from 𝑋𝑡 |
| 𝑋′        | Reasoning-based representation from 𝑋′ |
| 𝑋′′        | Reasoning-based representation from 𝑋′ |

Table 2: Our large-scale resume dataset

|                  | Count   |
|------------------|---------|
| # of Resumes     | 401,253 |
| # of Job Titles  | 354,168 |
| # of Companies   | 165,086 |
| # of Transitions | 2,738,403 |
| Average length of words | 4.15 |
| Average length of characters | 28.9 |

Table 3: Scale comparison of different JTM datasets.

| Data              | # of unique titles |
|-------------------|-------------------|
| Zhang et al. [36] | 26                |
| Dave et al. [8]   | 4,325             |
| Li et al. [19]    | >10,000           |
| Our dataset       | 354,168           |

3.2 Dataset

In this section, we describe our job title dataset that we use to evaluate our JAMES model. Our dataset comes from a career platform FutureFit AI1 which globally partners with other companies and governments. From this platform, we randomly selected 400K resumes that have at least five valid working experiences in the United States (i.e., a path with five nodes in a job transition graph) so that the job-transition graph becomes meaningful while the graph is reasonably large. Table 2 summarizes our dataset.

1https://www.futurefit.ai/

Figure 2: Toy example of JAMES in Job Title Mapping.

In our dataset, there are 354K of unique job titles coming from 165K unique companies, and a total of more than 2.7M job transition trajectories. To solve the JTM task, we only extract the job seeker’s basic info from his/her resume such as company id, job title, start working date and end working date. All employees’ private information is anonymized. The average number of words of all job titles is 4.15, and the average number of characters of all job titles is 28.9. Top 5 most frequent job titles include sales associate (0.33%), research assistant (0.23%), administrative assistant (0.23%), project manager (0.19%) and CEO (0.18%).

Table 3 shows the scalability comparison of the number of job titles in different JTM datasets. Compared to previous works [8, 19, 36], ours have a much larger number of job titles than others: the large number of job titles, >11,500 times larger than [36], >69 times larger than [8], and >30 times larger than the LinkedIn job title taxonomy dataset [19]. To create the ground truth dataset, we perform an exact match of job titles in our dataset with ESCO (European Skills/Competences, qualifications and Occupations\(^2\)), which includes job titles and standardized job titles as written as job groups, in which we use NLTK [1] to remove proper nouns. ESCO is a job title dictionary of European skills/competences, qualifications and occupations, and provides a hierarchical structure of occupation and its group. For example, “Software developers” consists of “application developer”, “software engineer”, “software architect”, etc. Then, we match the job titles in our dataset which also exist in the ESCO taxonomy, and use the matched job titles for the experiment. While there is another publicly available job title taxonomy ISCO\(^3\), ESCO is an extended version of ISCO and has a much larger volume. Thus, we adopted ESCO and created the ground truth label using the job title defined by ESCO.

4 OUR PROPOSED MODEL: JAMES

In this section, we describe our job title mapping model JAMES. The main idea of JAMES is to learn multi-aspect representations of an input job title, then produce its top-\(n\) mapping standard job titles that are predefined in a standard job title taxonomy. Figure 2 shows a toy example of how JAMES works. Job titles of an input resume are first extracted. Next, for each job title, JAMES first learns

\(^2\)https://ec.europa.eu/esco/portal/escopedia/ESCO
\(^3\)https://ec.europa.eu/esco/portal/escopedia/ISCO
its multi-aspect embeddings: topological embeddings, semantic embeddings, and syntactic embeddings. In this example, "mobile app specialist" is the input job title. Then, JAMES utilizes these multi-aspect embeddings to predict the matching standard job title, resulting in the "applications programmer".

A more pinpoint overview of JAMES can be seen in Figure 3. JAMES learns topological embeddings of the input job title using the state-of-the-art hyperbolic graph representations learning. To learn the input job title’s semantic embeddings, JAMES adopts the well-known pretrained BERT. To obtain the syntactic embeddings of the input job title, JAMES encodes a dense embedding vector with the size of the number of standard job titles, and each element in the vector is the string-based similarity score between the input job title and links as talent transitions between each job title. The hyperbolic graph embedding. Our career trajectory dataset includes how we build a graph from our dataset and how we develop the graph embedding which reflects topological feature. Figure 4 shows the graph from the career trajectory dataset to build a hyperbolic graph.

**4.1.1 Hyperbolic Graph Embedding.** We firstly create a job transition graph from the career trajectory dataset to build a hyperbolic graph. Then, to build hyperbolic graph, we consider the ending node (the most recent job) as the parent and the starting node (the least recent job) as the child, assuming the most recent job contains all the previous job requirements and skill sets. Afterwards, we embed the nodes on hyperbolic space adopting Poincare embedding [26] as a hyperbolic embedding, and train Poincare ball model from the relations of nodes.

Since the Poincare ball is a Riemannian manifold, the Riemannian metric tensor is represented in the $d$-dimensional ball $B^d = \{x \in \mathbb{R}^d ||x|| < 1\}$, where $||x||$ is the Euclidean norm. Then, the Riemannian metric tensor $r_x$ is defined as:

$$ r_x = \left( \frac{2}{1 - ||x||^2} \right)^{2r^E} $$

where $x \in B^d$ and $r^E$ is the Euclidean metric tensor. Then, the distance of two points $a, b \in B^d$ is defined as:

$$ d(a, b) = \arccosh \left( 1 + 2 \left( \frac{||a - b||^2}{(1 - ||a||^2)(1 - ||b||^2)} \right) \right) $$

Based on these metrics, we construct the Poincare embedding on the Poincare ball using [26]. Here, we input the parent-child pair dataset and then obtain the $m$-dimensional embedding. We output $X_h$ as the hyperbolic graph embedding of the each input job title.

**4.1.2 BERT Embedding.** Different job titles can refer to the same position. For example "Data Analyst" vs "Data Scientist". To alleviate this challenge, we learn the semantic embeddings of the input job titles via the well-known pretrained BERT [10]. Specifically, we use the pretrained DistilRoBERTa on Sentence-BERT [29] due to its efficiency and effectiveness.

Figure 5 shows the architecture for BERT embedding. For each input job title, we use the BERT-base uncased tokenizer to first tokenize the input job titles into wordpieces. Then, we use the pretrained BERT-base uncased embeddings to obtain the embeddings $X_h$ of the [CLS] token as final representations of the input job title.

**4.1.3 Syntactic Embedding.** We build the syntactic representation by scoring the string similarity between input job title and all of the standard job titles. Figure 6 shows the architecture for this embedding. Here, we use cosine similarity for the measurement. For instance, "software developers" is a parent of "user interface.
We define $X$ in such a way that each embedding view acknowledges the existence of $J$ job titles. Our three view embeddings guide the attention weights for the left-over view in parallel. Given $k$ views, our multi-aspect co-attention mechanism uses $p$ which takes only two input sources, and propose a multi-aspect attention is performed sequentially, leading to the model's latency. The multi-aspect embeddings of the input job title $X$ are non-redundant across the three views. Thus, our next step is to learn the multi-aspect embeddings of the input job title in a way that each embedding view acknowledges the existence of other embedding views. For this purpose, a typical method is to weigh each embedding view by a hierarchical attention [34], where the $X_h$ can be used as query, $X_b$ can be used as key/value. Then the $X_h$ and the attentive $X_b$ can be combined as a query, and $X_s$ can be used as key/value. This is not desired, as the hierarchical attention is performed sequentially, leading to the model's latency increment and is not ideal for real-world use cases with millions of job titles. Hence, we extend the traditional co-attention mechanism [23] which takes only two input sources, and propose a multi-aspect co-attention mechanism that works for $p$ inputs (i.e., $p \geq 2$). In this sense, our multi-aspect co-attention mechanism uses $k - 1$ views to guide the attention weights for the left-over view in parallel. Given our three view embeddings $X_h$, $X_b$, and $X_s$ in the JTM task, we demonstrate our multi-aspect co-attention as follows.

$$A_{(hh)} = \tanh(X_h W_{(hh)}^T X_h^T)$$
$$A_{(hs)} = \tanh(X_h W_{(hs)}^T X_s^T)$$
$$A_{(bs)} = \tanh(X_b W_{(hs)}^T X_s^T)$$

where $W_{(hh)}$, $W_{(hs)}$, and $W_{(bs)}$ are learnable weights. Next, we measure the weight $K_h$ for $X_h$, acknowledging supports from the other embedding views $X_b$ and $X_s$ through $A_{(hh)}$ and $A_{(hs)}$ as follows:

$$K_h = \tanh(W_h X_h + W_{bh} (A_{(hh)} X_h) + W_{sh} (A_{(hs)} X_s))$$

In a same manner, we compute the weights $K_b$ for $X_b$, and $K_s$ for $X_s$ as follows:

$$K_b = \tanh(W_b X_b + W_{bh} (A_{(hh)} X_h) + W_{bs} (A_{(bs)} X_s))$$
$$K_s = \tanh(W_s X_s + W_{sh} (A_{(hs)} X_h) + W_{bs} (A_{(bs)} X_s))$$

Then, the co-attentive multi-aspect embeddings $\hat{X}_h$, $\hat{X}_b$, and $\hat{X}_s$ can be computed as follows:

$$\hat{X}_h = \text{softmax}(K_h) \odot X_h$$
$$\hat{X}_b = \text{softmax}(K_b) \odot X_b$$
$$\hat{X}_s = \text{softmax}(K_s) \odot X_s$$

where $\odot$ is the element-wise product.

### 4.3 Reasoning-based Representations

It is unwise to map an input job title $j$ with a standard job title $\nu$ by solely looking at their similarity score. Instead, we also need to look at the similarity scores of $j$ with the rest of the standard job titles to alleviate uncertainty issues. For instance, given an input job title $j$ and a set $\mathcal{Y}$ of standardized input job titles. Assuming that the mapping score between $j$ and $\nu_l \in \mathcal{Y}$ is as high as 0.99, while the mapping scores between $j$ and the rest of other $\nu_k \in \mathcal{Y}$ ($l \neq k$) is as small as 0.01, then it is certain that $j$ maps to $\nu_l$. In another scenario,
### Table 4: Neural Logical Regularizations

| Logical Rule | Equation | Neural Logical Regularization |
|--------------|----------|-------------------------------|
| NOT          | \(\neg \text{True} = \text{False}\) | \(r_1 = \sum_{j \in X} \text{sim}(j, \neg \text{NOT}(j)) + \sum_{v \in Y} \text{sim}(v, \text{NOT}(v))\) |
| Double Negation | \(\neg(\neg j) = j\) | \(r_2 = \sum_{j \in X} (1 - \text{sim}(j, \neg \text{NOT}(j))) + \sum_{v \in Y} (1 - \text{sim}(v, \text{NOT}(v)))\) |
| OR           | \(j \lor \text{False} = j\) | \(r_3 = \sum_{j \in X} (1 - \text{sim}(\text{OR}(j, \text{False}), j)) + \sum_{v \in Y} (1 - \text{sim}(\text{OR}(v, \text{False}), v))\) |
| Identity     | \(j \lor \text{True} = j\) | \(r_4 = \sum_{j \in X} (1 - \text{sim}(\text{OR}(j, \text{True}), j)) + \sum_{v \in Y} (1 - \text{sim}(\text{OR}(v, \text{True}), v))\) |
| Annihilator  | \(j \lor j = j\) | \(r_5 = \sum_{j \in X} (1 - \text{sim}(\text{OR}(j, j), j)) + \sum_{v \in Y} (1 - \text{sim}(\text{OR}(v, v), v))\) |
| Idempotence  | \(j \lor \neg j = j\) | \(r_6 = \sum_{j \in X} (1 - \text{sim}(\text{OR}(\text{NOT}(j), \text{True}), j)) + \sum_{v \in Y} (1 - \text{sim}(\text{OR}(\text{NOT}(v), \text{True})), v))\) |
| Complementation | \(j \lor \neg j = \text{True}\) | \(r_7 = \sum_{j \in X} (1 - \text{sim}(\text{OR}(\text{NOT}(j)), \text{True})) + \sum_{v \in Y} (1 - \text{sim}(\text{OR}(\text{NOT}(v)), \text{True}))\) |

### Figure 8: Architecture for reasoning-based representation.

![Architecture for reasoning-based representation](image)

If the mapping score between \(j\) and \(v_k \in Y\) is as high as 0.9, and the mapping scores between \(j\) and a few other \(v_l \in Y\) \((l \neq k)\) is approximately as high as \((j, v_k)\), then there is a high uncertainty when mapping \(j\) to \(v_k\), even though its mapping score is highest. Therefore, we need a mechanism to look at mapping scores of \(j\) with all \(v_k \in Y\), all at once.

In other words, we need a mechanism that considers collaborative supports across all the mapping scores. Specifically, when looking at the example above, the mapping decision can be made by a reasoning procedure like if \(j\) is mostly similar to \(v_k\), and totally not similar to the rest of the job titles \(v_l \in Y\) \((l \neq k)\), then concludes that \(j\) maps to \(v_k\). In this sense, such a reasoning procedure can be represented as a logical structure, leading us to use NCR. Furthermore, NCR improved the performance of JTM, as you can see in our ablation study, and this function does not harm the execution cost. In this sense, such reasoning procedure can be represented as a logical structure like below:

\[
sim(j, v_1) \land \sim \text{sim}(j, v_2) \land \sim \text{sim}(j, v_3) \rightarrow v_5
\]  

(7)

Hence, we are inspired to design a neural collaborative reasoning module that learns reasoning-based representations of the input job titles. In this sense, the problem of predicting \(v_2\) as a correct mapping or not with the example above (i.e., Equation (7)) is converted into the problem of deciding if the following Horn clause is True or False:

\[
\sim \text{sim}(j, v_1) \land \sim \text{sim}(j, v_2) \rightarrow \text{sim}(j, v_5)
\]

(8)

Note that due to the lack of topological information of standard job titles, we are not able to produce topological embeddings for standard job titles. However, producing semantic embeddings and syntactic embeddings for standard job titles is straightforward and follows a similar process as for input job titles. As such, we define a Horn clause for finding a mapping between the input job title \(j_i\) and a correct mapping standard job title \(v_c \in Y\) with regard to the input semantic embeddings of both \(j_i\) and \(v_c\) can be defined as follows:

\[
\sim \text{sim}(j_i, v_1) \land \cdots \land \sim \text{sim}(j_i, v_{k-1}) \rightarrow \text{sim}(j_i, v_c)
\]

(9)

Based on the De Morgan’s Law, we can re-write Equation (9) using only two basic logical operator OR (i.e., \(\lor\)) and NOT (i.e., \(\neg\)) and obtain the reasoning-based representation \(X'_b\) of \(j_i\) as follows:

\[
X'_b = \sim \text{sim}(j_i, v_1) \lor \cdots \lor \sim \text{sim}(j_i, v_{k-1}) \lor \text{sim}(j_i, v_c)
\]

(10)

Similarly, we can obtain the reasoning-based representation \(X'_s\) of \(j_i\) with regard to the syntactic embedding view as follows:

\[
\sim \text{sim}(j_i, v_1) \lor \cdots \lor \sim \text{sim}(j_i, v_{k-1}) \lor \text{sim}(j_i, v_c)
\]

(11)

Figure 8 summarizes our architecture for the neural logical reasoning. With reasoning-based representations \(X'_b\) and \(X'_s\) are now established together in Equation (10) and (11), as well as co-attentive multi-aspect embeddings \(\hat{X}_b\), \(\hat{X}_s\), and \(\hat{X}_t\) (Equation (6)), we next fuse these embeddings to have a final representation of the input job title. We describe this embedding fusion as below.

### 4.4 Fusion

We first concatenate reasoning-based representations \(X'_b\) and \(X'_s\) and the co-attentive multi-aspect embeddings \(\hat{X}_b\), \(\hat{X}_s\), and \(\hat{X}_t\). Then we produce the output prediction by project the final job title’s embeddings into the size of all standard job titles \(|Y|\) and generate class probability distribution through the \(\text{softmax}\) operator.

\[
y = \text{softmax}(\text{ReLU}(W([\hat{X}_b; \hat{X}_s; \hat{X}_t; X'_b; X'_s])))
\]

(12)

### 4.5 Learning Objective

We use the categorical cross-entropy as the loss function to train our JAMES. The categorical cross-entropy loss function is defined as following:

\[
L(\theta) = -\sum_{j \in Y} y_j \log(\hat{y}_j)
\]

(13)

where \(\theta\) refers to all the parameters in the entire model.

In our implementation for reasoning-based representation (Equation (10) and (11)), following [7] the OR module is implemented by a multi-layer perceptron (MLP) with one hidden layer, and the NOT/NEGATION module is also implemented by another multi-layer perceptron. To explicitly guarantee that these OR and NOT modules implement the expected logic operations, we constraints
them with logical regularization as defined in Table 4. The final loss function of our JAMES is defined as followings:

$$L(\theta) = - \sum_{j=1}^{N} y_j \log(\hat{y}_j) + \sum_{q=1}^{6} r_q$$

where $\sum_{q=1}^{6} r_q$ is the summation of all six neural logical regularizations that are defined in Table 4.

5 EMPIRICAL VALIDATION

In this section, we evaluate the effectiveness of JAMES model against competing baselines. In the comparison, we use our large-scale JTM dataset, as other JTM datasets from [8, 19, 36] are not publicly available. We attempt to answer the following Evaluation Questions (EQ):

- EQ1: How does JAMES perform against the baselines?
- EQ2: Which components in JAMES are more helpful?
- EQ3: How complex is JAMES?

5.1 Set-Up

We compare JAMES against a exhaustive list of ten baseline models relevant to the JTM task: KNN-based [38], Word2Vec-based [3], DeepCarotene [32], Node2Vec [14], GloVe [27], NEO [13], WoLMIS [2], BERT-based [30], Job2Vec [35], and Universal Sentence Encoder (USE) [5]. The details of baselines and our implementation settings are described in the Appendix A. Note that since job description is not available in our dataset, we operate baseline models using only job titles. Next, to evaluate all compared models, we use Precision@N and NDCG@N – two widely used ranking metrics, with N being the top-N results produced by each model. Precision@N accounts for the number of relevant results among top-N output candidates, while NDCG@N applies an increasing discount of $\log_2$ to items at lower ranks. We divide the dataset into 64%, 16%, and 20%, where we train for 64% using 16% as a validation, and then test for 20% for the JTM task.

5.2 EQ1: Performance of JAMES

We observe that word-based baselines (baseline (i, ii, iii)) perform the worst. The reasons are two folds. First, word-level baselines mostly utilize word embedding technique. Thus, they do not account for contextual word semantic relationships in job titles, and could not mitigate the interdisciplinary correlation across job titles. Second, word-level semantic baselines used additional job description to boost their performance. However, we assure that job description is not always publicly available in JTM datasets. In addition, collecting job description from career platforms is expensive and has access limitation. Although baseline (v) is better than them, the performance is still much lower than JAMES. JAMES significantly outperforms Node2Vec (baseline (iv)). This indicates that solely using graph representation learning is suboptimal.

More applied and career-specific models (baseline (vi, vii)) achieve a higher performance compared to word-level semantic models and topological baseline as they are able to deal with both messy and interdisciplinary job titles, although JAMES still outperforms them. The sentence-level semantic based models (baseline (viii, ix, x)) perform very well in contrast with other baselines because they can extract and represent semantic meanings using pretrained models.

Table 5: Precision@10 and NDCG@10 of JAMES and baseline models on our dataset. The best results are in bold, the best baseline’s performance is underlined.

| Model       | Venue       | Precision@10 | NDCG@10 |
|-------------|-------------|--------------|---------|
| (i) KNN-based [38] | CoRR’16     | 0.0913       | 0.0871  |
| (ii) Word2Vec-based [3] | ECML’17       | 0.1254       | 0.0544  |
| (iii) DeepCarotene [32] | BigData’19   | 0.1255       | 0.0543  |
| (iv) Node2Vec [14] | KDD’16       | 0.1255       | 0.0609  |
| (v) GloVe [27] | EMNL’14      | 0.3080       | 0.1817  |
| (vi) NEO [13] | ISWC’20      | 0.3422       | 0.2054  |
| (vii) WoLMIS [2] | IIS’18       | 0.3536       | 0.2480  |
| (viii) BERT-based [30] | EMNL’19       | 0.6121       | 0.4720  |
| (ix) Job2Vec [35] | CIKM’19      | 0.6122       | 0.4622  |
| (x) USE [5] | EMNL’18      | 0.6619       | 0.4887  |
| JAMES      | This Paper   | 0.7285       | 0.5743  |

In short, JAMES vastly improved over all related baselines by our multi-aspect co-attentive reasoning representation. One important reason is that all prior models were developed using company-generated “clean” job titles while our dataset is user-generated “dirty” job titles. Compared to the best baseline, i.e., USE, JAMES improves Precision@10 by 10.06% and NDCG@10 by 17.52%. These results confirm the effectiveness JAMES.

5.3 EQ2: Ablation Study

Next, we conduct an ablation study to answer EQ2. Given that the BERT-based model (i.e., baselines vii and x) yielded relatively good performance and that BERT embedding is an essential feature of ours, we measure the performance of removing each single component of JAMES except for the BERT embedding. We show the results in Table 5. From Table 5, we observe two findings. First, hyperbolic graph embeddings have a significant contribution in the JTM performance of JAMES. For example, JAMES reduces Precision@10 by 18.28% and NDCG@10 by 19.62% when removing this component from JAMES. Because messy and overlapping job titles (i.e., Challenge 1-3 in Introduction) are the major issues in JTM task, hyperbolic graph embeddings can be effective for such job titles. Second, co-attention (CoAtt) and reasoning-based representation (Reasoning) increase the performance from the simple concatenation model by 4.70% by Precision@10 and 7.47% by NDCG@10, which shows the effectiveness of multi-aspect co-attention and fusion of co-attentive multi-aspect embeddings and reasoning-based representations. To see the performance in other job-driven downstream tasks, we explore an additional experiment as following section. Overall, removing each of components in JAMES reduces JAMES performance. All of those results show the effectiveness of our JAMES.

5.4 EQ3: Complexity of JAMES

Even though our model outperforms baselines in accuracies, we need to consider our model’s complexity for practical use. Thus, we compare the required execution time of JAMES with three most competitive baselines–USE, Job2Vec, and Node2Vec. In this setting, we use the feature after we pretrain and cache all embeddings for our dataset. Table 7 shows each execution time, which is the average of 10 runs. Even though JAMES takes a bit longer time than others,
Table 6: Ablation study experiments for JAMES.

| Model          | Precision@10 | NDCG@10 |
|----------------|--------------|---------|
| JAMES          | 0.7285       | 0.5743  |
| - CoAtt        | 0.6996 (↓1.43%) | 0.5630 (↓2.01%) |
| - CoAtt - Reasoning ($) | 0.6958 (↓4.70%) | 0.5344 (↓7.47%) |
| - * XSim       | 0.6273 (↓16.13%) | 0.4859 (↓18.19%) |
| - * XHGE       | 0.6159 (↓18.28%) | 0.4801 (↓19.62%) |
| - * XHGE - XSim | 0.6121 (↓19.02%) | 0.4720 (↓21.67%) |

Table 7: Average execution time of JAMES and baselines after caching the embeddings.

| Method         | Time (sec) |
|----------------|------------|
| USE [5]        | 1.080      |
| Node2Vec [14]  | 1.274      |
| Job2Vec [35]   | 1.350      |
| JAMES          | 1.365      |

JAMES still takes around one second which is practically effective to implement the system accounting for our performance.

6 PRACTICAL IMPACT

In this section, we present the applications of JAMES to demonstrate its practical impact. We summarize the talent acquisition market and its potential, and apply JAMES to two downstream career-domain tasks (i.e., link prediction and job mobility prediction).

6.1 Talent Acquisition Market

Online professional platforms and career-domain companies help other companies to acquire talented employees. The talent solutions and acquisition technologies have significant impacts for these career-domain companies. For instance, roughly 65% of LinkedIn’s annual revenue comes from talent solutions business [12]. CareerBuilder invests $300 millions for talent acquisition for three years5, as talent acquisition market is expected to be worth of billions of dollars and to expand rapidly. In this context, JTM is an essential task. Just as a plethora of online advertising technologies led to the rapid growth of high-tech companies such as Google and Facebook, if JTM improves the dataset quality, the job-person matching would become more robust, which eventually impacts business revenue largely. In addition, with the enhancement of talent solution technology itself, job seekers will be able to find more suitable jobs faster, and recruiters will be able to create candidate lists more easily.

6.2 Link Prediction

Link prediction is one of the most common tasks for graphs and networks [4, 17, 37]. In this part, to show the effectiveness of the job title’s multi-aspect embeddings learned by JAMES, we present an additional capability of JAMES in the link prediction task. We adopt Node2Vec [14], a state-of-the-art and widely-used link prediction model as a baseline for comparison. We also use Word2Vec, GloVe, USE, and Job2Vec as representative baselines.

Table 8: AUC in link prediction task.

| Method                      | AUC  |
|-----------------------------|------|
| Word2Vec-based [3]          | 0.5648 |
| GloVe [27]                  | 0.6278 |
| USE [5]                     | 0.8370 |
| Node2Vec [14]               | 0.8743 |
| Job2Vec [35]                | 0.9431 |
| JAMES                       | 0.9957 |

Table 9: Job Mobility Prediction

| Method                      | MAP@10 |
|-----------------------------|--------|
| No JTM (unpreprocessed) + NEMO [19] | 0.5349 |
| Job2Vec [35] + NEMO [19]     | 0.6418 |
| USE [5] + NEMO [19]          | 0.6529 |
| JAMES + NEMO [19]            | 0.7013 |

Given a large-scale job transition graph built on our dataset, we perform following steps to generate training/development/testing sets for the link prediction task. First, we randomly remove 20% of the total number of links in the graph, considering them as the positive links in the testing set. We also randomly sample the same amount of negative links in the graph for the testing set. Next, for the remaining of 80% positive links in the graph, we randomly remove 20% of the positive links and, randomly sample the same amount of negative links to formulate a development set. The rest of the graph is kept as a training set. Then, given an employee node embedding, a job-title node embedding obtained by JAMES, and the link/edge that connects these two nodes, we follow [14] and use different binary operators (i.e., Average, Hadmard, Weighted-L1, and Weighted-L2) that operate on the employee and job-title node embeddings to obtain the link embedding. We select the best binary operator using the development set. As for reporting the performance metric, we choose AUC as it is mostly adopted by prior studies on the link prediction task.

The performance of JAMES and compared methods is shown in Table 8. The best baseline is Job2Vec. We observe that JAMES outperforms all of the baselines. JAMES relatively improves Job2Vec by 5.59% of AUC. This result shows that the output embeddings of the job title from our JAMES model is effective not only for JTM task, but also for job-driven downstream tasks.

6.3 Job Mobility Prediction.

To see the effect of JAMES in the dataset improvement from the JTM, we conduct a job mobility prediction with additional supports of JAMES. The job mobility prediction task is one of the most essential downstream tasks in career-based problems, and needs JTM as a preprocessing. We first reuse the same dataset as in Table 2, and then apply JAMES to preprocess the dataset, converting each input job title with the top-1 matching standard job title. In addition to our JAMES, we use Job2Vec, USE as baselines. We also prepare the unpreprocessed dataset. Next, the output dataset is used to conduct a job mobility prediction. Given a user’s sequence of job trajectories, the job mobility prediction task predicts the next job title for the user. For the prediction model, we adopt NEMO [19] as it is the state-of-the-art model. For the evaluation, we report mean

5https://press.careerbuilder.com/2020-09-15-CareerBuilder-Commits-to-Three-Year-300M-Talent-Acquisition-Investment
average precision at 10 (MAP@10) as the compared metric. Note that we only use the job title transition information and cut other features from NEMO as they are expensive to collect and are not available in our dataset. We compare how the result change by the JTM methods.

Table 9 shows the job mobility prediction performance, indicating that the JTM task has a substantial impact on the model performance. One primal reason is that our JAMES captures multiple aspects of job title (i.e., topologically, semantically, and syntactically). Such multifaceted mapping helps job mobility prediction models learn relatively easily. Some examples of the differences in mapping results are presented in the Appendix C. Although we do not yet know exactly how the accuracy improvement will affect the user behavior, we plan to use JAMES and perform A/B testing on several downstream tasks including the job mobility prediction.

7 CONCLUSION

In this paper, we proposed a novel job title mapping model JAMES toward making a granular job taxonomy using a real-world and large-scale career trajectory dataset, which contains more than 350K job titles. JAMES is based on multi-aspect embeddings (i.e., topological, semantic, and syntactic embedding), multi-aspect co-attention, and reasoning-based representation to effectively deal with the challenges of the Job Title Mapping (JTM) task. To present the effectiveness of our proposal, we conducted extensive experiments to compare the performance of JAMES against ten comparable baselines on the JTM classification task and we did ablation study. Furthermore, we conducted additional experiments using practical downstream tasks (i.e., link prediction and job mobility prediction) to see the practical impact. We showed that: (1) JAMES outperformed all baseline models in the JTM task, and (2) JAMES was effective in career-domain downstream tasks. For real-world applications, JAMES can be utilized in two scenarios. First, JAMES can serve as a binary classifier and produce a mapping probability score for two input job titles. Second, JAMES can return the most relevant job group from a job title as input. Those applications will be publicly available as a demo in the future.

REFERENCES

[1] Steven Bird. 2006. NLTk: the natural language toolkit. In Proceedings of the COLING/AACL 2006 Interactive Presentation Sessions. 69–72.
[2] Roberto Boselli, Mirko Cesarini, Stefania Marrara, Fabio Mercorio, Mario Mezzanzanica, Gabriella Pasi, and Marco Viviani. 2018. WoLiMS: a labor market intelligence system for classifying web job vacancies. Journal of Intelligent Information Systems 51, 3 (2018), 477–502.
[3] Roberto Boselli, Mirko Cesarini, Fabio Mercorio, and Mario Mezzanzanica. 2017. Using machine learning for labour market intelligence. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 330–342.
[4] Lei Cai and Shuiwang Ji. 2020. A multi-scale approach for graph link prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 3308–3315.
[5] Danial Cev, Yindes Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal Sentence Encoder for English. In EMNLP: System Demonstrations. 169–174.
[6] Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. NeurIPS 32 (2019), 4669.
[7] Hanxiong Chen, Shaoyun Shi, Yong Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In Proceedings of the Web Conference 2021. 1516–1527.
[8] Yixin Chen, Chik-hiu S Dave, Baichuan Zhang, Mohammad Al-Hassan, Khalidah Aljadda, and Mohammad Karamy. 2018. A combined representation learning approach for better job and skill recommendation. In CIKM. 1997–2005.
[9] Thomat H Davenport and DJ Patil. 2012. Data scientist. Harvard business review 90, 5 (2012), 70–78.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
[11] Octavian-Eugen Ganea, Gary Beiglmeier, and Thomas Hofmann. 2018. Hyperbolic Neural Networks. In NeurIPS. 5350–5360.
[12] Sahin Cem Geyrik, Qi Guo, Bo Hu, Cagri Ozcgul, Keten Thakkar, Xianren Wu, and Krishnam R Kenthapadi. 2018. Talent search and recommendation systems at LinkedIn: Practical challenges and lessons learned. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. 1533–1534.
[13] Anna Giabelli, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Andrea Stevese. 2020. NEO: A Tool for Taxonomy Enrichment with New Emerging Occupations. In International Semantic Web Conference. Springer, 564–584.
[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In KDD. 855–864.
[15] Karveb Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In International Conference on Machine Learning. PMLR, 4116–4126.
[16] Krishnam R Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized job recommendation system at LinkedIn: Practical challenges and lessons learned. In RecSys. 346–347.
[17] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Biswaraj Biswas. 2020. Link prediction techniques, applications, and performance: A survey. Physica A: Statistical Mechanics and its Applications 553 (2020), 124289.
[18] Huayu Li, Yong Ge, Hengshu Zhu, Hui Xiong, and Hongke Zhao. 2017. Prospecting the career development of talents: A survival analysis perspective. In KDD. 917–925.
[19] Liangyue Li, How Jing, Hangsong Tong, Jaewon Yang, Qi He, and Bee-Chung Chen. 2017. Nemo: Next career move prediction with contextual embedding. In WWW Companion. 505–513.
[20] Shan Li, Baoxia Shi, Jaewon Yang, Ji Yan, Shuai Wang, Fei Chen, and Qi He. 2020. Deep Job Understanding at LinkedIn. In SIGIR. 2145–2148.
[21] Yifeng Li, Fang-Xiang Wu, and Alioune Ngom. 2018. A review on machine learning principles for multi-view biological data integration. Briefings in bioinformatics 19, 2 (2018), 325–340.
[22] Ye Liu, Lunming Zhang, Liqiang Nie, Yan Yan, and David Rosemann. 2016. Fortune teller: predicting your career path. In AAAI. Vol. 30.
[23] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. Advances in neural information processing systems 29 (2016), 289–297.
[24] Haiyan Luo, Shichuan Ma, Anand Joseph Bernard Selvaraj, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In EMNLP. 1532–1543.
[25] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In KDD. 14–24.
[26] Maximilian Nickel and Douwe Kiela. 2017. Poincare embeddings for learning hierarchical representations. In NeurIPS. 6341–6350.
[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In EMNLP. 1532–1543.
[28] Baoxue Shi, Jaewon Yang, Feng Guo, and Qi He. 2020. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In SIGIR. 25–34.
[29] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In EMNLP. https://arxiv.org/abs/1908.10084
[30] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In EMNLP. https://arxiv.org/abs/1908.10084
[31] Baoxue Shi, Jaewon Yang, Feng Guo, and Qi He. 2020. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In SIGIR. 25–34.
[32] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In EMNLP. https://arxiv.org/abs/1908.10084
[33] Baoxue Shi, Jaewon Yang, Feng Guo, and Qi He. 2020. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In SIGIR. 25–34.
[34] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In EMNLP. https://arxiv.org/abs/1908.10084
[35] Crockett, Xuewen, and Krishnam R Kenthapadi. 2020. Large-Scale Talent Flow Embedding for Company Competitive Analysis. In TheWebConf 2020. 2534–2534.
[36] Mohsen Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. Advances in Neural Information Processing Systems 31 (2018), 5165–5175.
[37] Yuxin Zhu, Faizan Javed, and Ozgur Otturk. 2016. Semantic similarity strategies for job title classification. arXiv preprint arXiv:1609.06268 (2016).
A EXPERIMENTAL SETTINGS

A.1 Baseline Methods

We compare JAMES against the exhaustive list of ten baseline models which are relevant to the JTM task.

(i) **KNN-based** [38]: It uses KNN with Word2Vec embeddings, where similar job titles are in the same cluster.

(ii) **Word2Vec-based** [3]: It uses bag-of-words or Word2Vec embeddings from words and job descriptions of web job vacancies, then apply SVM and CNN. We choose CNN as one representative model.

(iii) **DeepCarotene** [32]: It uses the pretrained Word2Vec embeddings to obtain character, word and job description embeddings, then fuse three embedding types using a CNN model.

(iv) **Node2Vec** [14]: As a graph representation model, we build a transition graph and use Node2Vec to learn node embeddings, and utilize a CNN.

(v) **GloVe** [27]: We obtain GloVe embeddings of two input job titles and utilize a CNN model as in DeepCarotene [32].

(vi) **NEO** [15]: It uses Word2Vec and GloVe for words and sub-words for the input job title and standard job title, and measures hierarchical semantic score and cosine similarity score between them.

(vii) **WoLMS** [2]: It uses the bag-of-words representation from words and job descriptions of web job vacancies, then apply SVM, Random Forest, etc. We choose Random Forest as one representative model.

(viii) **BERT-based** [30]: It first extracts contextual embeddings of the input job titles, and then, fetches contextual embeddings into a CNN model to classify if two input job titles are similar or not.

(ix) **Job2Vec** [35]: It is state-of-the-art multi-view learning model for job title link benchmarking. Job2Vec uses graph representation on job titles, semantic representation from job descriptions, and job transition balance and duration.

(x) **Universal Sentence Encoder (USE)** [5]: It encodes input job titles into a high dimensional vector, then is used for the job title classification.

A.2 Implementation settings

For baseline models, we use their reported hyperparameter settings. The dimension of the GloVe-based (word-based) models was 300 as it is the largest dimension from the provider in the public page\(^6\). For the Universal Sentence Encoder (USE), we use 512 dimensions, which is the default of the provider, and the BERT-based model’s embedding size is 768 for the same reason. For Node2Vec, we choose 128 accounting for execution time on a large-scale graph dataset.

For our model, we vary the embedding size from \{128, 256, 512\}. The number of epoch is set to 200 with early stopping (i.e., we stop training if the model’s performance is not improved in 20 continuous epochs). Our model and all the baselines are trained with a batch size of 256 using Adam optimizer and learning rate of \(10^{-3}\).

\(^6\)https://nlp.stanford.edu/projects/glove/

B GRAPH EMBEDDING VISUALIZATION

Figure 9 is the visualization comparison between Euclidean and Poincare embeddings in a 2-dimentional ball. We observe the differences between the two representation spaces. In Euclidean embedding, each dot is scattered disorderly, whereas hyperbolic embedding shows a hierarchy of dots. Thus, we adopt the hyperbolic graph embedding as it can capture the implicit job transition hierarchy and topological knowledge dependencies.

C CASE STUDIES

Although our extensive experimental results revealed the effectiveness of JAMES, we show some case studies based on the prediction result by JAMES to explore the mapping result more. In this section, we compare JAMES with Universal Sentence Encoder (USE)[5] and Job2Vec [35] as the best and second-best baselines, respectively. Figure 10 shows 20 case studies from our dataset. The green-colored cells are the results successfully predicted (i.e., true positives), and the light-green ones are not true positives but can be considered as reasonably correct answers while the rest of the cells are incorrect answers. Here, we can observe patterns of each model prediction.

As we see the case where JAMES succeeded but other baselines failed, USE tends to be determined by the meaning of job titles, and Job2Vec predicted a logical result for the relation of job titles. For example, the ground truth of “market manager” is “sales and marketing managers”. JAMES correctly predicted it while USE’s result is “retail and wholesale trade managers” which is based on the semantic features, and Job2Vec’s result is “supply, distribution and related managers” which is one of the popular transition of “market manager”. Thus, we claim that JAMES can predict the job title in a more balanced way.

On the other hand, we can observe how JAMES fails. When the inputs are “children’s social worker” or “youth guidance worker”, JAMES predicts “child care workers”. Although the ground truth is “social work and counselling professionals”, we can carefully say this failure is acceptable intuitively considering how they work as such roles. JAMES gets more real-world wise prediction result. Finally, when we see the results that all the models fail, there is a difference of how each model fails. For instance, the ground truth of “assistant principal” is “education managers”, and JAMES predicts “university and higher education teachers”. This result is reasonable here as well because “assistant principal” is a bit vague definition and we can consider “education teachers” is almost the same with “education managers”. In that case, there is a possibility that the publicly available taxonomy itself, which we use as a ground truth,
Figure 10: 20 cases of prediction results by JAMES, Universal Sentence Encoder (USE) [5] and Job2Vec [35]. The green colored cells are true positives, and the light-green ones can be considered as reasonably correct answers, whose semantic meanings are compatible to true positives. By and large, JAMES produces the best result.

is not perfect and we can develop a new mapping using via a hierarchical clustering directly from our multi-aspect embeddings.