JAMES: Normalizing Job Titles with Multi-Aspect Graph Embeddings and Reasoning

Michiharu Yamashita[1], Jia Tracy Shen[1], Thanh Tran[2], Hamoon Ekhtiari[3], and Dongwon Lee[1]
[1]The Pennsylvania State University, University Park, PA, USA
[2]Amazon, Cambridge, MA, USA [3]FutureFit AI, Toronto, Canada
{michiharu, jqs5443}@psu.edu, tdt@amazon.com, hamoon@futurefit.ai, dongwon@psu.edu

***Abstract*—In online job marketplaces, it is important to establish a well-defined job title taxonomy for various downstream tasks (*e*.g., job recommendation, users' career analysis, and turnover prediction). *Job Title Normalization* (*JTN*) is such a cleaning step to classify user-created non-standard job titles into normalized ones. However, solving the *JTN* problem is non-trivial with challenges: (1) semantic similarity of different job titles, (2) non-normalized user-created job titles, and (3) large-scale and long-tailed job titles in real-world applications. To this end, we propose a novel solution, named JAMES, that constructs three unique embeddings (*i*.e., *graph*, *contextual*, and *syntactic*) of a target job title to effectively capture its various traits. We further propose a multi-aspect co-attention mechanism to attentively combine these embeddings, and employ neural logical reasoning representations to collaboratively estimate similarities between messy job titles and normalized job titles in a reasoning space. To evaluate JAMES, we conduct comprehensive experiments against *ten* competing models on a large-scale real-world dataset with over 350,000 job titles. Our experimental results show that JAMES significantly outperforms the best baseline by 10.06% in Precision@10 and by 17.52% in NDCG@10, respectively. To further facilitate the acquisition of normalized job titles for job-domain applications, our JAMES API is available at: https://tinyurl.com/james-job-title-mapping.**

***Index Terms*—multi-aspect embeddings, entity mapping, representation learning, job title normalization**

I. Introduction

Background. The recent proliferation of technology has witnessed an increasing popularity of online professional platforms. These online job marketplaces connect job seekers and companies to find the best match for each other. For example, LinkedIn and Indeed, two of the largest jobs marketplace platforms, have more than 930 million users[1] and 245 million resumes[2], respectively. The vast amount of data available on job marketplaces, including resumes from job seekers and job postings from companies, has spurred companies involved in workforce development, talent intelligence, recruitment, and job search engines to utilize AI techniques to enhance their applications (*e*.g., job recommendation [1], [2], next career prediction [3], [4], and career analysis [5]). These AI-powered tools enable job seekers in finding their ideal jobs and companies in recruiting talents that match their roles. However, the workflow of such job-domain applications involves a critical step, as illustrated in Figure 1. Before building models for downstream tasks, the various entities found in raw data, especially *job titles*, must be sorted, consolidated, and normalized. For instance, a position called "systems engineer" in a company A and another position called "application programmer" in a company B may refer to the same job. Normalizing these two job titles into "software developer" (or noting their compatibility in context) is crucial for job recommendations, career trajectory analysis, and search result expansion. Therefore, the research question (RQ) we investigate is: *How can job titles be automatically normalized*? In particular, we aim to answer this RQ via the framing of the Job Title Normalization (*JTN*) (to be defined in Section 3.1).

Challenges. Although the *JTN* problem appears simple in nature, addressing it in practice poses several challenges. First, job titles often bear a semantic closeness to one another that is contingent upon the required skill sets and companies' own definitions. For example, the job title "data scientist" is prevalent today, and comprises skill sets such as mathematical modeling, statistics, and coding. However, this role can be related to "business analyst" or "data analyst" in some companies, and to "product scientist" in others. Thus, comparing job titles alone is inadequate for solving *JTN*, and it is necessary to represent them in a semantic space to ensure accurate calibration. Second, job titles collected from users' resumes are often untidy due to non-standard naming conventions and auxiliaries. The job title "software developer" in one resume can be written as "SDE" in another. Moreover, creative job titles such as "data geek" or "strategic futurist" that individuals may list on their resumes do not necessarily appear in an industry-wide job title taxonomy. Third, while an industry job taxonomy contains only a few hundred to a few thousand job titles, the number of job titles encountered on job marketplace platforms is orders of magnitude larger. Nevertheless, existing solutions have only employed either small-scale datasets or company-created datasets (as opposed to user-written), in which *JTN* was addressed through manual labeling/cleaning or text normalization procedures. For instance, Zhang et al. [6] employed a dataset with only 26 unique job titles for similar expertise job matching, while Dave et al. [7] used a dataset of 4,325 unique job titles for job and skill recommendation tasks. More recently Li et al. [8] developed a job title taxonomy containing 30,000 entries on LinkedIn for a job understanding task. However, the challenges inherent in *JTN* for the vast number of job titles still remain inadequately addressed.

[1]https://about.linkedin.com/
[2]https://www.indeed.com/about

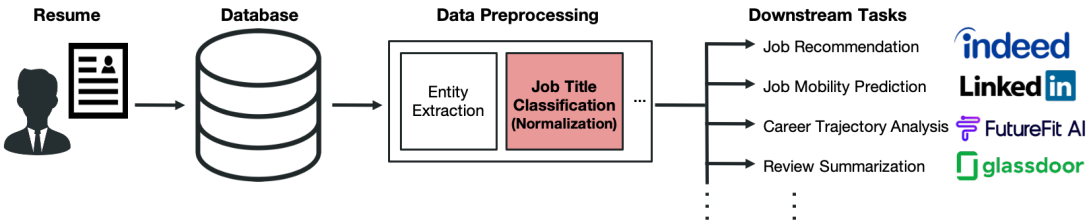

Fig. 1: Workflow of job-domain applications

Ideas. To address the aforementioned challenges in *JTN*, we propose JAMES (Job title mApping with Multi-aspect Embeddings and reaSoning), and demonstrates its effectiveness using a real-world career dataset containing more than 350,000 job titles. Specifically, JAMES considers three unique multi-aspect (*i*.e., graph, contextual, and syntactic) embeddings for candidate job titles. First, we establish a graph embedding to represent the latent *topological* job title similarity based on users' job transitions, exploiting the fact that users typically switch to similar positions or titles (*i*.e., changing from "data scientist" to "chef" is highly unlikely although possible). We use a hyperbolic graph embedding for the latent knowledge dependencies in a job title hierarchy, as it outperforms Euclidean graph embeddings on hierarchical structure datasets [9], [10]. In addition, hyperbolic graph embeddings help mitigate the problem of incomplete and inconsistent job transition patterns by providing a smaller distortion and an exponential expansion of nodes [10], [11]. Second, we leverage a pretrained BERT embedding to account for the *contextual* similarity between two candidate job titles, which can identify contextually-related job titles, as language models can measure the contextual and semantic distance. Third, we create a *syntactic* embedding to capture the string-to-string similarity between two input job titles, allowing for the detection of misspelled (*e*.g., "electric engieer") and user-created (*e*.g., "cool data scientist") job titles. Also, we design a neural collaborative reasoning [12] that takes multi-aspect embeddings as input and produces reasoning-based multi-aspect embeddings to mitigate uncertainty among standard job titles, covering job titles that are not accurately captured by either contextual or syntactic embeddings. After building multi-aspect embeddings using our large-scale resume dataset, we develop a multi-aspect co-attention mechanism that considers all three multi-aspect embeddings concurrently.

Contributions. Our contributions are as follows:

- We use a large-scale, real-world, and user-generated dataset from a career platform (FutureFit AI), which comprises over 350,000 unique job titles, for the job-domain specific preprocessing task, *Job Title Normalization* (*JTN*).
- To solve the *JTN* task, we propose a novel model, JAMES, that employs multi-aspect embeddings and reasoning representations accounting for *graph*, *contextual*, and *syntactic* embeddings.
- We conduct extensive experiments and demonstrate the effectiveness of JAMES against *ten* competing baseline models. JAMES significantly outperforms the best baseline by 10.06% in Precision@10 and by 17.52% in NDCG@10, respectively. We also apply JAMES to other downstream tasks and report the findings and further implications.
- We develop and release JAMES API publicly, allowing for the acquisition of normalized job titles from job title entities.

II. Related Work

A. Job Title Classification

Previously, *JTN* was often overlooked and just addressed through manual labeling or simple data preprocessing. However, there have been several prior works that attempt to solve it as a task of job title classification [13], [14]. Wang et al. [15] proposed a CNN-based approach that developed text vectors using a job description dataset, while Zhu et al. [16] built a KNN model using Word2Vec. While such methods using job descriptions can be helpful, in the real world, it is often difficult to obtain access to all companies' job description datasets, and the applicability of such methods to user-generated job titles extracted from resumes is not well understood. Therefore, our work aims to develop a practical solution applicable to a user-generated dataset (*i*.e., resumes).

As a job entity benchmarking, Luo et al. [17] created a job transition graph using Random Walk-based vectors, and indicated the potential of job graph embedding. Zhang et al. [18] proposed Job2Vec as a job title benchmarking tool based on job records. However, both works were only validated in link and/or node prediction and not specifically designed for *JTN*, resulting in uncertainty regarding their applicability to *JTN* and normalization. Moreover, Job2Vec aimed to link job titles of the same expertise level to calibrate salaries for recruiters [18]. While these benchmarks could be used for job title clustering, they manually filtered out low-frequency words in job titles as a data preprocessing step, and limited their dataset to the IT and finance domains, which restricts the generalizability of their representations to real-world scenarios. Additionally, only 15 well-known companies such as Google and Microsoft were chosen in their IT dataset, which may not reflect a practical scenario where companies or individuals want to map all job titles from users' resumes into normalized ones. In contrast, our study focuses on a realistic and large-scale setting for *JTN*, utilizing a dataset from 165,086 unique companies across all sectors. Our approach differs from previous works as we employ contextual embeddings to capture

the potential meaning of words, and syntactic embeddings to detect misspelled and user-created words, addressing issues not considered in prior works. Furthermore, we use reasoning to obtain more robust representations.

B. Representation Learning in Job Domain

Liu et al. [19] conducted career path prediction using multiple social media. For job skill representation, Shi et al. [20] developed "Job2Skills", a market-aware skill extraction system, which considers the salient level of a skill and extracts important skill entities from job postings and target members using multi-resolution. Qin et al. [21] developed a person-job fit model that applied a word-level semantic representation for both job requirements and job seekers' experiences based on RNN. Yamashita et al. [22] proposed a long-term career path prediction from large-scale resumes with multiple embeddings. While these studies relied on job-domain entities, our JAMES can be applied to such job-domain applications to normalize job titles.

C. Hyperbolic Machine Learning

Hyperbolic geometry is a non-Euclidean geometry that focuses on spaces of constant negative Gaussian curvature. Hyperbolic space has been used to develop embedding and machine learning models for hierarchical and graph structures, due to its benefits such as embedding on smaller dimensions [10]. Poincare embedding, proposed by Nickel et al. [9], enables hierarchical data to be represented better than Euclidean embeddings. Chami et al. [11] developed a hyperbolic technique for graph convolutional networks. In the case of career trajectory datasets, they can be represented as graphs, as done by Zhang et al. [6]. However, to the best of our knowledge, our work is the first to apply hyperbolic geometry to career transition data. Since hyperbolic embeddings work well on tree-structured datasets, we consider hyperbolic embeddings to be effective for representing latent knowledge dependencies in job titles, which are often hierarchical (*e*.g., junior, senior, VP). Hence, we incorporate hyperbolic embeddings into our model and compare the baselines.

III. Preliminary

Major notations used throughout the paper are summarized in Table I. In this section, we describe a definition of the problem, Job Title Normalization (*JTN*), and our dataset.

A. Problem Definition

We formally define the *JTN* task as follows:

Job Title Normalization (*JTN*): Given a set of job titles $\mathcal{X}$ and a set of predefined and normalized job titles in a job taxonomy $\mathcal{Y}$, the *Job Title Normalization* task aims to build a function $f(\cdot)$ that produces matching probabilities from each job title $j_i \in \mathcal{X}$ to each normalized job title $v_k \in \mathcal{Y}$.

TABLE I: Definition of our notations in this paper

Notation	Definition
j	job title from resume
$\mathcal{X}$	a set of job titles from all resumes
v	normalized job title from the ground truth
$\mathcal{Y}$	a set of normalized job titles from the ground truth
$\boldsymbol{X_h}$	hyperbolic graph embeddings
$\boldsymbol{X_b}$	BERT embeddings
$\boldsymbol{X_s}$	syntactic (string similarity) embeddings
$A^{(hb)}$	affinity matrices between $\boldsymbol{X_h}$ and $\boldsymbol{X_b}$
$A^{(hs)}$	affinity matrices between $\boldsymbol{X_h}$ and $\boldsymbol{X_s}$
$A^{(bs)}$	affinity matrices between $\boldsymbol{X_b}$ and $\boldsymbol{X_s}$
$W^{(hb)}$	learnable weights between $\boldsymbol{X_h}$ and $\boldsymbol{X_b}$
$W^{(hs)}$	learnable weights between $\boldsymbol{X_h}$ and $\boldsymbol{X_s}$
$W^{(bs)}$	learnable weights between $\boldsymbol{X_b}$ and $\boldsymbol{X_s}$
K_h	attention graph for $\boldsymbol{X_h}$, acknowledging supports from the other embedding views $\boldsymbol{X_b}$ and $\boldsymbol{X_s}$ through $A^{(hb)}$ and $A^{(hs)}$
K_b	attention graph for $\boldsymbol{X_b}$, acknowledging supports from the other embedding views $\boldsymbol{X_h}$ and $\boldsymbol{X_s}$ through $\boldsymbol{X_h}$ and $A^{(hs)}$
K_s	attention graph for $\boldsymbol{X_s}$, acknowledging supports from the other embedding views $\boldsymbol{X_b}$ and $\boldsymbol{X_h}$ through $A^{(hb)}$ and $\boldsymbol{X_h}$
$\hat{\boldsymbol{X}}_{\boldsymbol{h}}$	co-attentive multi-view embeddings from $\boldsymbol{X_h}$
$\hat{\boldsymbol{X}}_{\boldsymbol{b}}$	co-attentive multi-view embeddings from $\boldsymbol{X_b}$
$\hat{\boldsymbol{X}}_{\boldsymbol{s}}$	co-attentive multi-view embeddings from $\boldsymbol{X_s}$
$\boldsymbol{X}'_{\boldsymbol{b}}$	reasoning-based representation from $\boldsymbol{X_h}$
$\boldsymbol{X}'_{\boldsymbol{s}}$	reasoning-based representation from $\boldsymbol{X_s}$

Note that during the training of the mapping function $f(\cdot)$, we consider the *JTN* task as a multi-class classification task. During the inference, with each $j_i \in \mathcal{X}$, we take the output probability distribution over all normalized job titles $v_k \in \mathcal{Y}$ as the ranking scores to output *top-k* most similar job titles $v_k \in \mathcal{Y}$.

B. Dataset

We obtained the dataset from a popular career platform FutureFit AI[3], which partners globally with other companies and governments to assist employees in navigating career transitions. We randomly selected over 400,000 resumes from the platform, which had at least five valid working experiences in the United States (*i*.e., a path with five nodes in a job transition graph). This was done to ensure that the job transition graph was meaningful, while also being reasonably large. Table II presents a summary of our dataset information, including the average length of words and characters in job titles.

Our dataset contains 354,168 unique job titles from 165,086 unique companies, and more than 2.7 million job transition trajectories. To solve the *JTN* task, we only extract the job seeker's basic information from their resume, such as company ID, job title, start and end dates of working, while ensuring that all private employee information is anonymized. On average, job titles have 4.15 words and 28.9 characters. The top five most frequent job titles are sales associate (0.33%), research assistant (0.23%), administrative assistant (0.23%), project manager (0.19%), and CEO (0.18%). Our dataset will be available upon request.

In comparison to previous works [6], [7], [18], [23], our dataset has a significantly larger number of job titles. Specifically, our dataset contains over 11,500 times more job titles

[3]https://www.futurefit.ai/

TABLE II: Our large-scale resume dataset

# 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

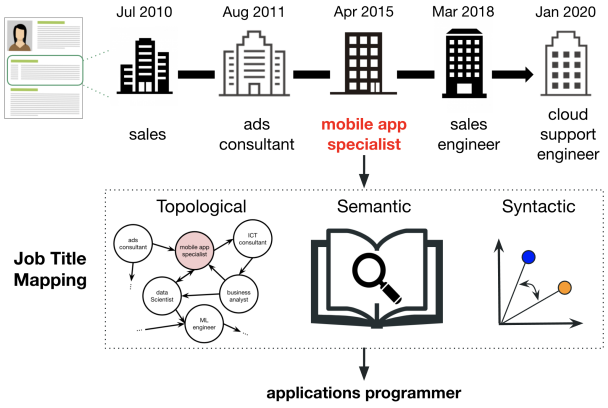

Fig. 2: Toy example of JAMES in *Job Title Normalization*.

than [6], over 69 times more job titles than [7], over 30 times more job titles than [23], and four times more job titles than the Job2Vec (Finance) dataset [18]. To create our ground truth dataset, we perform an exact search to match job titles in our dataset with the European Skills/Competences, qualifications, and Occupations (ESCO) taxonomy[4], which provides a hierarchical structure of job titles and normalized job titles as job groups. For example, "software developers" consists of "application developer", "software engineer", "software architect", etc. We remove proper nouns from the job titles, and use the matched job titles as the ground truth labels for our experiments.

IV. Our Proposed Model: JAMES

In this section, we describe our job title normalization model, JAMES. The main idea of JAMES is to learn multi-aspect representations of an input job title and produce its corresponding *top-n* normalized standard job title mappings that are predefined in a standard job title taxonomy. Figure 2 shows a toy example of how JAMES works. First, the job titles of a resume are extracted. Next, JAMES learns the multi-aspect embeddings, including graph, semantic, and syntactic embeddings, for each job title. In this example, the input job title is "mobile app speciallist", and JAMES utilizes the multi-aspect embeddings to predict the matching standard job title, resulting in "applications programmer".

Figure 3 provides a more detailed overview of JAMES. To learn the graph embeddings of the input job title, JAMES employ the state-of-the-art hyperbolic graph representation learning. To learn the semantic embeddings of the input job title, JAMES use the well-known pretrained BERT. To obtain the syntactic embeddings of the input job title, JAMES

[4]https://ec.europa.eu/esco/portal/escopedia/ESCO

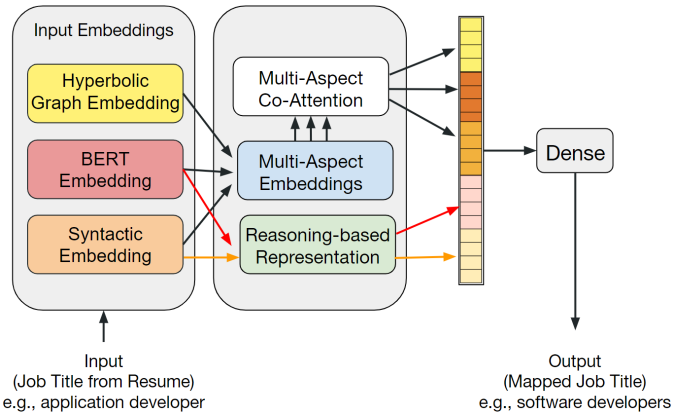

Fig. 3: Model overview of JAMES.

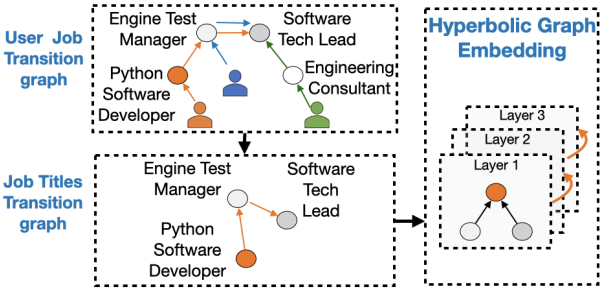

Fig. 4: Architecture for hyperbolic graph embedding.

encodes a dense embedding vector with a size equal to the number of standard job titles. Each element in the vector represents the string-based similarity score between the input job title and the corresponding standard job title. Next, we propose a multi-aspect co-attention mechanism that assigns attention scores to the three multi-aspect embeddings. We also introduce a reasoning-based module in JAMES that collaboratively reasons the multi-aspect embeddings in a reasoning space. Finally, JAMES fuses all the output embeddings to produce the *top-n* mapping normalized job titles for the input job title as outputs. We provide a detailed description of JAMES in the following subsections.

A. Multi-Aspect Embeddings

1) Hyperbolic Graph Embedding: To construct a hyperbolic graph embedding that captures the topological features of our career trajectory dataset, we first create a job transition graph, as illustrated in Figure 4. We define nodes as job titles and links as the transitions between the job titles, where each link is directed and asymmetric. For instance, if a person changes their job title from "Software Engineer" (SWE) to "Machine Learning Engineer" (MLE), the graph has a directed link from SWE to MLE.

The job transition graph is defined as $G = (V, E, W)$, where V is the set of job titles (*i*.e., nodes), E is the set of job transitions (*i*.e., links in the graph), and W is the set of link weights. The job transition weight $W_{i,j}$ is formulated as $W_{i,j} = e_{i,j} / \sum_{i=1}^{n} \sum_{j=1}^{n} e_{i,j}$, where $e_{i,j}$ is the number of transitions from node v_i to node v_j. Based on all job

(a) Euclidean embedding (b) Hyperbolic embedding

Fig. 5: Example visualizations for the Euclidean and Hyperbolic embeddings from job transition graph.

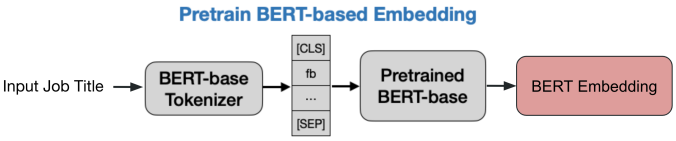

Fig. 6: Learning semantic embeddings of the input job tile via the pretrained uncased BERT-base.

transitions, we construct a graph and derive graph embeddings. To build the hyperbolic graph, we consider a head node (*i*.e., more recent user's career) as the *parent* and a tail node (*i*.e., the previous career) as the *child*, assuming that the most recent job title contains all the requirements and skill sets from the previous job titles. We then embed the nodes in hyperbolic space using Poincare embedding [9] as a hyperbolic embedding and train a Poincare ball model from the relations of nodes in the graph.

Since the Poincare ball is a Riemannian manifold, the Riemannian metric tensor is represented in the d-dimentional 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:

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

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:

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

Based on these metrics, we construct the Poincare embedding on the Poincare ball [9] with the input of the parent-child pair dataset and obtain the m-dimensional embedding. Figure 5 provides a visualization comparison between Euclidean and Poincare embeddings in a 2-dimensional ball, where the hyperbolic embedding exhibits a hierarchy of dots, while each dot in the Euclidean embedding is scattered disorderly. We output $\boldsymbol{X_h}$ as the hyperbolic graph embedding of each input job title.

2) BERT Embedding: To address the issue of different job titles referring to the same position (*e*.g., "Data Analyst" vs "Data Scientist"), we learn the semantic embeddings of the input job titles using the pre-trained BERT [24]. Specifically, we use the pre-trained DistilRoBERTa on SBERT [25] due to its efficiency and effectiveness.

The architecture for the BERT embedding is illustrated in Figure 6. For each input job title, we first tokenize it into wordpieces using the BERT-base uncased tokenizer. Then, we use the pretrained BERT-base uncased embeddings to obtain the embeddings $\boldsymbol{X_b}$ of the [CLS] token as the final representations of the input job title.

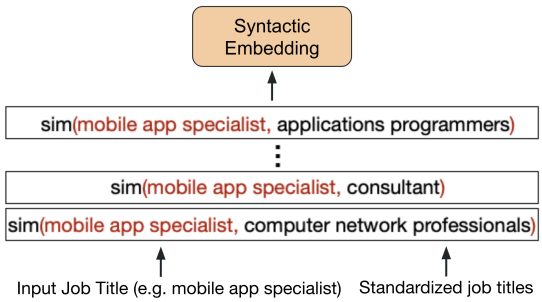

Fig. 7: Architecture for syntactic (string similarity) embedding. Given the input job title "*mobile app specialist*", we represent it by an embedding vector $\boldsymbol{X_s} \in \mathcal{R}^{|\mathcal{Y}|}$, where $\boldsymbol{X_s}$ [k] = *cosine-sim*("*mobile app specialist*", v_k), with $v_k \in \mathcal{Y}$.

3) Syntactic Embedding: To capture the syntactic representation of job titles, we use cosine similarity to score the string similarity between an input job title and all of the normalized job titles in the job taxonomy (*i*.e., ESCO). Figure 7 shows the architecture for this embedding. We calculate all pairs between job titles and their parent job titles, and define the similarity matrix as the syntactic representation.

For a given set of $\mathcal{X}$ input job titles and $\mathcal{Y}$ predefined normalized job titles in the job taxonomy, the syntactic embedding of a job title $s \in \mathcal{X}$ is a vector $\boldsymbol{X_s}$ of $\mathcal{Y}$ dimensions, where each dimension indicates the string-string cosine similarity with a character-level comparison between s and each corresponding normalized job title in $\mathcal{Y}$. For example, $\boldsymbol{X_s}[0]$ = cosine-sim(s, v_0), $\boldsymbol{X_s}[1]$ = cosine-sim(s, v_1), ..., $\boldsymbol{X_s}[|\mathcal{Y}| - 1]$ = cosine-sim$(s, v_{|\mathcal{Y}|-1})$. We define the resulting syntactic embeddings as $\boldsymbol{X_s}$.

B. Multi-Aspect Co-Attention

In the previous sections, we extract three discrete embeddings for an input job title: (i) hyperbolic graph embeddings $\boldsymbol{X_h}$, (ii) BERT embeddings $\boldsymbol{X_b}$, and (iii) syntactic embeddings $\boldsymbol{X_s}$. However, the embeddings are learned separately and may have redundant features. To address this issue, we aim to learn multi-aspect embeddings that incorporate all three embeddings and are attentive to each other. For this purpose, a traditional method is to weigh each embedding view by hierarchical attention [26], where the $\boldsymbol{X_h}$ can be used as query, $\boldsymbol{X_b}$ can be used as key/value. Then the $\boldsymbol{X_h}$ and the *attentive* $\boldsymbol{X_b}$ can be combined as a query, and $\boldsymbol{X_s}$ can be used as key/value. As the hierarchical attention is performed sequentially and is not practical for large-scale datasets with millions of job titles. Therefore, we extend the traditional co-attention mechanism [27] which takes only two input sources, and propose a multi-aspect co-attention mechanism that can work 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.

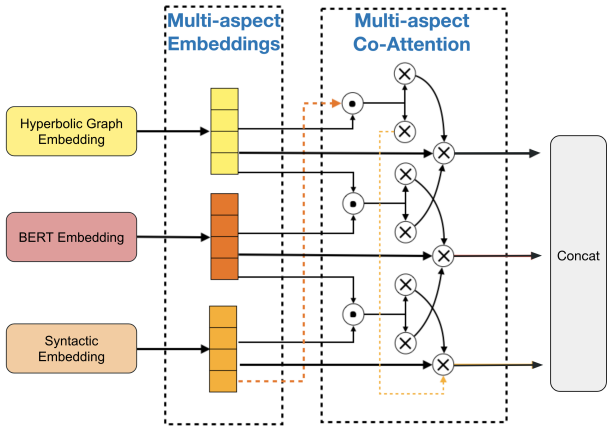

Fig. 8: Architecture for Multi-view Co-attention.

Figure 8 shows the architecture for our multi-aspect co-attention. We start by computing three affinity matrices for three pairs of two embedding views: $A^{(hb)}$ between $\boldsymbol{X_h}$ and $\boldsymbol{X_b}$, $A^{(hs)}$ between $\boldsymbol{X_h}$ and $\boldsymbol{X_s}$, and $A^{(bs)}$ between $\boldsymbol{X_b}$ and $\boldsymbol{X_s}$. Specifically, the affinity matrices, $A^{(hb)}$, $A^{(hs)}$, and $A^{(bs)}$ are calculated as follows:

$$\begin{aligned} A^{(hb)} &= \tanh\left(\boldsymbol{X_h} W^{(hb)} \boldsymbol{X_b}^T\right) \\ A^{(hs)} &= \tanh\left(\boldsymbol{X_h} W^{(hs)} \boldsymbol{X_s}^T\right) \\ A^{(bs)} &= \tanh\left(\boldsymbol{X_b} W^{(bs)} \boldsymbol{X_s}^T\right) \end{aligned} \tag{3}$$

, where $W^{(hb)}$, $W^{(hs)}$, and $W^{(bs)}$ are learnable weights. Next, we measure the weight K_h for $\boldsymbol{X_h}$, acknowledging supports from the other embedding views $\boldsymbol{X_b}$ and $\boldsymbol{X_s}$ through $A^{(hb)}$ and $A^{(hs)}$ as follows:

$$K_h = \tanh\left(W_h \boldsymbol{X_h} + W_{bh}(A^{(hb)} \boldsymbol{X_b}) + W_{sh}(A^{(hs)} \boldsymbol{X_s})\right) \tag{4}$$

In the same manner, we compute the weights K_b for $\boldsymbol{X_b}$, and K_s for $\boldsymbol{X_s}$ as follows:

$$\begin{aligned} K_b &= tanh\left(W_b \boldsymbol{X_b} + W_{hb}(A^{(hb)^T} \boldsymbol{X_h}) + W_{sb}(A^{(bs)} \boldsymbol{X_s})\right) \\ K_s &= tanh\left(W_s \boldsymbol{X_s} + W_{hs}(A^{(hs)^T} \boldsymbol{X_h}) + W_{bs}(A^{(bs)^T} \boldsymbol{X_b})\right) \end{aligned} \tag{5}$$

Then, the co-attentive multi-aspect embeddings $\hat{\boldsymbol{X}}_{\boldsymbol{h}}$, $\hat{\boldsymbol{X}}_{\boldsymbol{b}}$ and $\hat{\boldsymbol{X}}_{\boldsymbol{s}}$ can be computed as follows:

$$\begin{aligned} \hat{\boldsymbol{X}}_{\boldsymbol{h}} &= softmax(K_h) \odot \boldsymbol{X_h} \\ \hat{\boldsymbol{X}}_{\boldsymbol{b}} &= softmax(K_b) \odot \boldsymbol{X_b} \\ \hat{\boldsymbol{X}}_{\boldsymbol{s}} &= softmax(K_s) \odot \boldsymbol{X_s} \end{aligned} \tag{6}$$

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

C. Reasoning-based Representations

Mapping an input job title j to a normalized job title v based solely on their similarity score is unwary. To alleviate uncertainty issues, it is necessary to also consider the similarity scores of j with the rest of the normalized job titles in the set $\mathcal{Y}$. For example, if the mapping score between j and a certain $v_k \in \mathcal{Y}$ is high at 0.99, while the mapping scores between j and the other $v_l \in \mathcal{Y}$ $(l \neq k)$ are low at 0.01, then it is considered *certain* that j maps to v_k. However, if the mapping score between j and $v_k \in \mathcal{Y}$ is high at 0.9, and the mapping scores between j and a few other $v_l \in \mathcal{Y}$ $(l \neq k)$ are close to (j, v_k), then there is high *uncertainty* when mapping j to v_k, even though its mapping score is the highest. Therefore, we need a mechanism that takes into account mapping scores of j with all $v_k \in \mathcal{Y}$ simultaneously.

In other words, we need a mechanism that considers collaborative supports across all the mapping scores. Specifically, in the example above, the mapping decision can be made by a reasoning procedure that checks if j is mostly similar to v_k, and totally dissimilar to the rest of the job titles $v_l \in \mathcal{Y}$ $(l \neq k)$, and concludes that j maps to v_k. Such a reasoning procedure can be represented as a logical structure, leading us to use neural collaborative reasoning [12]. Furthermore, our ablation study demonstrates that the reasoning improves the performance of *JTN*. Thus, we can represent such a reasoning procedure as a logical structure, as shown below:

$$sim(j_i, v_1) \wedge sim(j_i, v_2) \wedge sim(j_i, v_3) \rightarrow v_3 \tag{7}$$

Hence, we are inspired to design a neural collaborative reasoning module [12] 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:

$$sim(j_i, v_1) \wedge sim(j_i, v_2) \rightarrow sim(j_i, v_3) \tag{8}$$

Note that due to the lack of topological information of normalized job titles, we are not able to produce topological embeddings for normalized job titles. However, producing semantic embeddings and syntactic embeddings for normalized job titles is straight-forward 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 $v_c \in \mathcal{Y}$ with regard to the input semantic embeddings of both j_i and v_k can be defined as follows:

$$sim(j_i^{(b)}, v_1^{(b)}) \wedge \cdots \wedge sim(j_i^{(b)}, v_{|\mathcal{Y}|}^{(b)}) \rightarrow sim(j_i^{(b)}, v_c^{(b)}) \tag{9}$$

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

$$\boldsymbol{X'_b} = \neg sim(j_i^{(b)}, v_1^{(b)}) \vee \cdots \vee \neg sim(j_i^{(b)}, v_{|\mathcal{Y}|}^{(b)}) \vee sim(j_i^{(b)}, v_c^{(b)}) \tag{10}$$

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

$$\boldsymbol{X'_s} = \neg sim(j_i^{(s)}, v_1^{(s)}) \vee \cdots \vee \neg sim(j_i^{(s)}, v_{|\mathcal{Y}|}^{(s)}) \vee sim(j_i^{(s)}, v_c^{(s)}) \tag{11}$$

Figure 9 summarizes our architecture for the neural logical reasoning. With reasoning-based representations $\boldsymbol{X'_b}$ and $\boldsymbol{X'_s}$ are now established together in Equation (10) and (11), as well as co-attentive multi-aspect embeddings $\hat{\boldsymbol{X}}_{\boldsymbol{h}}$, $\hat{\boldsymbol{X}}_{\boldsymbol{b}}$, and

TABLE III: Neural Logical Regularizations. The NOT module is implemented by an one-layer MLP, and the OR module is implemented by another one-layer MLP. The *True* and *False* are logical constants in the traditional logical equations, but are learnable representations in our neural logical reasoning modules.

	Logical Rule	Equation	Neural Logical Regularization.
NOT	Negation	$\neg True = False$	$r_1 = \sum_{j \in \mathcal{X}} sim(j, NOT(j)) + \sum_{v \in \mathcal{Y}} sim(v, NOT(v))$
	Double Negation	$\neg(\neg j) = j$	$r_2 = \sum_{j \in \mathcal{X}} \left(1 - sim(j, NOT(NOT(j)))\right) + \sum_{v \in \mathcal{Y}} \left(1 - sim(v, NOT(NOT(v)))\right)$
OR	Identity	$j \vee False = j$	$r_3 = \sum_{j \in \mathcal{X}} \left(1 - sim(OR(j, False), j)\right) + \sum_{v \in \mathcal{Y}} \left(1 - sim(OR(v, False), v)\right)$
	Annihilator	$j \vee True = True$	$r_4 = \sum_{j \in \mathcal{X}} \left(1 - sim(OR(j, True), True)\right) + \sum_{v \in \mathcal{Y}} \left(1 - sim(OR(v, True), True)\right)$
	Idempotence	$j \vee j = j$	$r_5 = \sum_{j \in \mathcal{X}} \left(1 - sim(OR(j, j), j)\right) + \sum_{v \in \mathcal{Y}} \left(1 - sim(OR(v, v), v)\right)$
	Complementation	$j \vee \neg j = True$	$r_6 = \sum_{j \in \mathcal{X}} \left(1 - sim(OR(j, NOT(j)), True)\right) + \sum_{v \in \mathcal{Y}} \left(1 - sim(OR(v, NOT(v)), True)\right)$

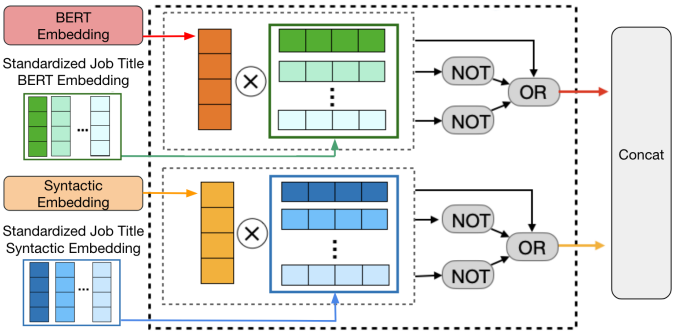

Fig. 9: Architecture for reasoning-based representation.

$\hat{\boldsymbol{X}}_{\boldsymbol{s}}$ (Equation (6)), we next fuse these embeddings to have a final representation of the input job title.

D. Fusion

We concatenate the reasoning-based representations $\boldsymbol{X}'_{\boldsymbol{b}}$ and $\boldsymbol{X}'_{\boldsymbol{s}}$, and the co-attentive multi-aspect embeddings $\hat{\boldsymbol{X}}_{\boldsymbol{h}}$, $\hat{\boldsymbol{X}}_{\boldsymbol{b}}$, and $\hat{\boldsymbol{X}}_{\boldsymbol{s}}$. Then we project the final job title embeddings into the size of all standard job titles $|\mathcal{Y}|$ and generate a class probability distribution through the *softmax* operator.

$$\hat{y} = softmax(ReLU(W([\hat{\boldsymbol{X}}_{\boldsymbol{h}}; \hat{\boldsymbol{X}}_{\boldsymbol{b}}; \hat{\boldsymbol{X}}_{\boldsymbol{s}}; \boldsymbol{X}'_{\boldsymbol{b}}; \boldsymbol{X}'_{\boldsymbol{s}}]))) \tag{12}$$

E. 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:

$$L(\theta) = -\sum_{j \in \mathcal{Y}} y_j log(\hat{y}_j) \tag{13}$$

where θ refers to all the parameters in the entire model.

In our implementation for reasoning-based representaion (Equation (10) and (11)), following [12], 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 III. The final loss function of our JAMES is defined as followings:

$$L(\theta) = -\sum_{j \in \mathcal{Y}} y_j log(\hat{y}_j) + \sum_{q=1}^{6} r_q \tag{14}$$

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

V. Empirical Validation

In this section, we present the evaluation results of our proposed JAMES model against competing baselines. We use our large-scale *JTN* dataset for the comparison, as other *JTN* datasets from [6], [7], [23] 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**: Can JAMES be useful for other downstream tasks?

A. Experimental Settings

1) Baselines: We compare JAMES against an exhaustive list of ten baseline models, including traditional simple solutions and state-of-the-art models: KNN-based [16], Word2Vec-based [28], DeepCarotene [15], Node2Vec [29], GloVe [30], NEO [13], WoLMIS [14], SBERT [31], Job2Vec [18], and Universal Sentence Encoder (USE) [32]. Note that as job descriptions are not available in our dataset, we construct the baseline models using only job titles to enable a fair comparison.

2) Evaluation protocols: To evaluate the performance of all compared models, we use two widely used ranking metrics, *Precision*@N and *NDCG*@N, 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 *log2* 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 *JTN* task. Regarding our implementation settings, we use their reported hyperparameter settings for baseline models. For the GloVe-based (word-based) models, the dimension is set to 300. For the Universal Sentence Encoder (USE), we use 512 dimensions, which is the default setting by the provider, and the SBERT's embedding size is set to 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}. During training, the number of epochs is set to 200 with early stopping. Our model and all the baselines are trained with

TABLE IV: 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 [16]	CoRR'16	0.0913	0.0871
(ii)	Word2Vec-based [28]	ECML'17	0.1254	0.0544
(iii)	DeepCarotene [15]	BigData'19	0.1255	0.0543
(iv)	Node2Vec [29]	KDD'16	0.1255	0.0609
(v)	GloVe [30]	EMNLP'14	0.3080	0.1817
(vi)	NEO [13]	ISWC'20	0.3422	0.2054
(vii)	WoLMIS [14]	IIS'18	0.3536	0.2480
(viii)	SBERT [31]	EMNLP'19	0.6121	0.4720
(ix)	Job2Vec [18]	CIKM'19	0.6122	0.4622
(x)	USE [32]	EMNLP'18	0.6619	0.4887
	JAMES	Ours	**0.7285**	**0.5743**

a batch size of 256 using the Adam optimizer and learning rate of 10^{-3}.

B. EQ1: Performance of JAMES

Table IV shows the overall performance of JAMES and the compared models on our large-scale *JTN* dataset. We observe that word-based baselines (baseline (i, ii, iii)) perform the worst. This can be attributed to two main reasons. First, word-level baselines mostly rely on word embedding techniques and do not account for contextual word semantic relationships in job titles, which results in a failure to mitigate the interdisciplinary correlation among job titles. Second, the word-level semantic baselines use additional job descriptions to enhance their performance, but job descriptions are not always publicly available in *JTN* datasets, have limited access, and are expensive to collect. Although baseline (v) performs better than the word-based baselines, its performance is still significantly lower than other models. JAMES significantly outperforms Node2Vec (baseline (iv)), indicating that using only graph representation learning is suboptimal.

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

In short, JAMES vastly outperforms all related baselines by utilizing our multi-aspect co-attentive reasoning representation. An important reason for this is that all prior models were developed using *company-generated* "pure" job titles, while our dataset is *user-generated* "impure" job titles. Compared to the best baseline, *i*.e., USE, JAMES improves *Precision@10* by 10.06% and *NDCG@10* by 17.52%, confirming the effectiveness of JAMES.

C. EQ2: Ablation Study

We conducted an ablation study to address EQ2. Since SBERT (*i*.e., baselines viii and x) yielded relatively good performance and BERT embedding is an essential feature of our model, we evaluated the performance of removing each single component of JAMES except for the BERT embedding. Table V presents the results. We observed two main findings. First, hyperbolic graph embeddings have a significant contribution to the *JTN* performance of JAMES. Removing this component from JAMES without co-attention and reasoning led to a reduction of *Precision@10* by 18.28% and *NDCG@10* by 19.62%. This demonstrates the effectiveness of hyperbolic graph embeddings for the major issues in *JTN* task, such as overlapping and messy job titles (i.e., Challenge 1-3 in Introduction). Second, co-attention (CoAtt) and reasoning-based representation (Reasoning) improved the performance from the simple concatenation model by 4.70% in *Precision@10* and 7.47% in *NDCG@10*, showing the effectiveness of multi-aspect co-attention and fusion of co-attentive multi-aspect embeddings and reasoning-based representations. In summary, the removal of each component in JAMES reduced its performance, indicating the effectiveness of our model.

TABLE V: Ablation study experiments for JAMES.

Model	Precision@10	NDCG@10
JAMES	**0.7285**	**0.5743**
- CoAtt ($=\alpha$)	0.6996 ($\downarrow$ 4.13%)	0.5630 ($\downarrow$ 2.01%)
- α - Reasoning ($=\beta$)	0.6958 ($\downarrow$ 4.70%)	0.5344 ($\downarrow$ 7.47%)
- β - Syntactic	0.6273 ($\downarrow$ 16.13%)	0.4859 ($\downarrow$ 18.19%)
- β - Hyperbolic Graph ($=\kappa$)	0.6159 ($\downarrow$ 18.28%)	0.4801 ($\downarrow$ 19.62%)
- κ - Syntactic	0.6121 ($\downarrow$ 19.02%)	0.4720 ($\downarrow$ 21.67%)

D. EQ3: Other Downstream Tasks

To assess the performance of JAMES in other job-domain downstream tasks, we conducted additional experiments as follows.

1) Link Prediction: Link prediction is one of the most common tasks for graphs and networks [33]–[35]. In this part, to see the effectiveness of the multi-aspect embeddings learned by JAMES, we present an additional capability of JAMES in the link prediction task. We compare JAMES with Node2Vec, Word2Vec, GloVe, USE, and Job2Vec as representative baselines.

To generate the training/development/testing sets for the link prediction task, we randomly removed 20% of the total number of links in the graph, considering them as the positive links in the testing set, and sampled the same amount of negative links in the graph for the testing set. We also randomly removed 20% of the positive links in the remaining 80% positive links and sampled the same amount of negative links to form a development set. The rest of the graph was kept as a training set. Then, we followed [29] and used different binary operators (*i*.e., Average, Hadmard, Weighted-L1, and Weighted-L2) to obtain the link embedding from the employee node embedding, the job-title node embedding obtained by JAMES, and the link/edge that connects these two nodes. We selected the best binary operator using the development set and reported the performance metric using AUC.

Table VI shows the performance of JAMES and the compared methods. The best baseline is *Job2Vec*, and we observe that JAMES outperforms all of the baselines. Specifically, JAMES relatively improves Job2Vec by 5.59% of AUC. This

TABLE VI: AUC in link prediction task.

Method	AUC
Word2Vec-based [28]	0.5648
GloVe [30]	0.6278
USE [32]	0.8370
Node2Vec [29]	0.8743
Job2Vec [18]	0.9431
JAMES	**0.9957**

TABLE VII: Job Mobility Prediction

Method	MAP@10
No *JTN* (unpreprocessed) + NEMO [23]	0.5349
Job2Vec [18] + NEMO [23]	0.6418
USE [32] + NEMO [23]	0.6529
JAMES + NEMO [23]	**0.7013**

result demonstrates that the multi-aspect embeddings from our JAMES model are effective not only for *JTN* task but also for link prediction.

2) Job Mobility Prediction.: Job mobility prediction is an essential job-domain downstream task [3], [23], [36], [37]. It involves predicting a user's next job titles based on their sequence of job trajectories, wherein *JTN* is conducted for preprocessing the dataset. Specifically, we use JAMES to preprocess our resume dataset, by converting each input job title with the *top-1* matching standard job title. To compare with our JAMES, we use Job2Vec, USE as baselines. We also prepare the unpreprocessed dataset. For the job mobility prediction model, we adopt NEMO [23] as it is the state-of-the-art model, and evaluate its performance using mean average precision at 10 (MAP@10) as the metric. Note that we only consider the job title transition information for our evaluation, as other features are costly to collect and are not available in our dataset. We compare the impact of the *JTN* methods on the model's performance.

Table VII shows the performance of the job mobility prediction task using the JAMES and baselines for preprocessing. The results demonstrate that the JAMES has a considerable impact on the job mobility prediction model's performance. Its multifaceted mapping approach assists job mobility prediction models in learning more effectively, resulting in a performance improvement. Although the accuracy improvement alone does not provide insights into user behavior, the effectiveness of JAMES in improving performance suggests its potential applicability in various job-domain downstream tasks.

VI. JAMES API and Use Cases

To make JAMES accessible as a public resource, we release a RESTful API for JAMES. This API allows users to input any textual job title entity and receive the corresponding normalized job titles based on the public taxonomy (*i*.e., ESCO). Users are able to obtain up to the top-5 most relevant normalized job titles, which helps individuals or organizations in preprocessing and cleansing their job title datasets for job-domain downstream tasks. The API provides the output predicted by JAMES if the input job entity is included in our graph. Otherwise, the API employs textual embeddings to get the output so that users can use any text and retrieve relevant normalized job titles. We also built a demo website that users can touch and see the results more intuitively. Figure 10 shows the screenshot of JAMES web app that uses our API behind.

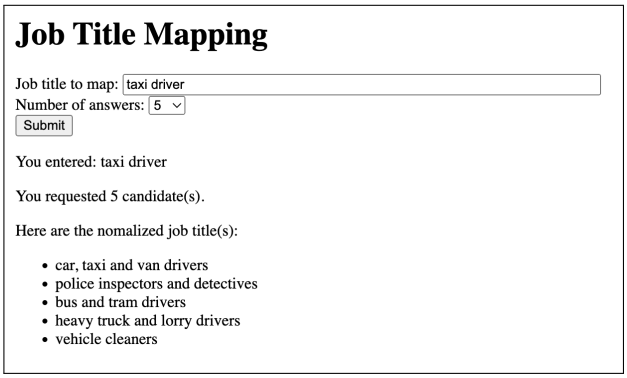

Fig. 10: JAMES API demo page. This shows an example of the job title normalization for "taxi driver"

Our JAMES API has the potential for a wide range of applications and use cases, as listed below:

- *Recruitment platforms*: By integrating job titles, JAMES API enables recruiters and job seekers to more easily compare job titles and requirements across different companies and industries.
- *Resume standardization*: JAMES API can be incorporated to automatically normalize job titles in users-uploaded resumes on online platforms, facilitating the matchmaking between job seekers with job postings.
- *Market research*: JAMES API is also beneficial in market and economic research for tracking job trends and analyzing job requirements across different industries and regions, as economic researchers typically need to clean job titles for their analysis.
- *Search query expansion*: By mapping all variations of an entity to a single normalized form, JAMES API can be used to improve the relevance of job search results, expanding the search to include all documents that mention the normalized job titles.

VII. Conclusion

In this paper, we proposed a novel job title normalization model, JAMES, toward creating a fine-grained job taxonomy using a real-world and large-scale career trajectory dataset, which contains more than 350K job titles. Our approach utilized multi-aspect embeddings (*i*.e., graph, semantic, and syntactic embedding), multi-aspect co-attention, and reasoning-based representation to address the challenges of the *Job Title Normalization* (*JTN*) task effectively. We conducted extensive experiments comparing JAMES to ten baseline models on the *JTN* task and performed an ablation study. Furthermore, we conducted additional experiments on practical downstream tasks, such as link prediction and job mobility prediction, to assess the practical impact of our approach. Our results showed that: (1) JAMES outperformed all baseline models in the *JTN*

task, and (2) JAMES was effective in job-domain downstream tasks. Finally, we release JAMES as an API for public use, which is useful for job-domain downstream tasks.

ACKNOWLEDGMENT

The work was in part supported by NSF awards #1934782 and #1909702, and PSU CSRAI seed grant 2021.

REFERENCES

[1] K. Kenthapadi, B. Le, and G. Venkataraman, "Personalized job recommendation system at linkedin: Practical challenges and lessons learned," in *Proceedings of the ACM conference on recommender systems (RecSys)*, 2017, pp. 346–347.

[2] I. Paparrizos, B. B. Cambazoglu, and A. Gionis, "Machine learned job recommendation," in *Proceedings of the ACM conference on recommender systems (RecSys)*, 2011, pp. 325–328.

[3] Q. Meng, H. Zhu, K. Xiao, L. Zhang, and H. Xiong, "A hierarchical career-path-aware neural network for job mobility prediction," in *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2019, pp. 14–24.

[4] H. Xu, Z. Yu, J. Yang, H. Xiong, and H. Zhu, "Dynamic talent flow analysis with deep sequence prediction modeling," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 31, no. 10, pp. 1926–1939, 2018.

[5] H. Li, Y. Ge, H. Zhu, H. Xiong, and H. Zhao, "Prospecting the career development of talents: A survival analysis perspective," in *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2017, pp. 917–925.

[6] L. Zhang, T. Xu, H. Zhu, C. Qin, Q. Meng, H. Xiong, and E. Chen, "Large-scale talent flow embedding for company competitive analysis," in *Proceedings of the Web Conference (WWW)*, 2020, pp. 2354–2364.

[7] V. S. Dave, B. Zhang, M. Al Hasan, K. AlJadda, and M. Korayem, "A combined representation learning approach for better job and skill recommendation," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2018, pp. 1997–2005.

[8] S. Li, B. Shi, J. Yang, J. Yan, S. Wang, F. Chen, and Q. He, "Deep job understanding at linkedin," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2020, pp. 2145–2148.

[9] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," 2017, pp. 6341–6350.

[10] O.-E. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," 2018, pp. 5350–5360.

[11] I. Chami, R. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, p. 4869, 2019.

[12] H. Chen, S. Shi, Y. Li, and Y. Zhang, "Neural collaborative reasoning," in *Proceedings of the Web Conference (WWW)*, 2021, pp. 1516–1527.

[13] A. Giabelli, L. Malandri, F. Mercorio, M. Mezzanzanica, and A. Seveso, "Neo: A tool for taxonomy enrichment with new emerging occupations," in *International Semantic Web Conference*. Springer, 2020, pp. 568–584.

[14] R. Boselli, M. Cesarini, S. Marrara, F. Mercorio, M. Mezzanzanica, G. Pasi, and M. Viviani, "Wolmis: a labor market intelligence system for classifying web job vacancies," *Journal of Intelligent Information Systems*, vol. 51, no. 3, pp. 477–502, 2018.

[15] J. Wang, K. Abdelfatah, M. Korayem, and J. Balaji, "Deepcarotene-job title classification with multi-stream convolutional neural network," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 1953–1961.

[16] Y. Zhu, F. Javed, and O. Ozturk, "Semantic similarity strategies for job title classification," *arXiv preprint arXiv:1609.06268*, 2016.

[17] H. Luo, S. Ma, A. J. B. Selvaraj, and Y. Sun, "Learning job representation using directed graph embedding," in *Proceedings of the International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019, pp. 1–5.

[18] D. Zhang, J. Liu, H. Zhu, Y. Liu, L. Wang, P. Wang, and H. Xiong, "Job2vec: Job title benchmarking with collective multi-view representation learning," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2019, pp. 2763–2771.

[19] Y. Liu, L. Zhang, L. Nie, Y. Yan, and D. Rosenblum, "Fortune teller: predicting your career path," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 30, no. 1, 2016.

[20] B. Shi, J. Yang, F. Guo, and Q. He, "Salience and market-aware skill extraction for job targeting," in *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2020, pp. 2871–2879.

[21] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, and H. Xiong, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2018, pp. 25–34.

[22] M. Yamashita, Y. Li, T. Tran, Y. Zhang, and D. Lee, "Looking further into the future: Career pathway prediction," *ACM WSDM Workshop on Computational Jobs Marketplace 2022*, 2022.

[23] L. Li, H. Jing, H. Tong, J. Yang, Q. He, and B.-C. Chen, "Nemo: Next career move prediction with contextual embedding," in *Proceedings of the International Conference on World Wide Web Companion*, 2017, pp. 505–513.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[25] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[26] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the Conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2016, pp. 1480–1489.

[27] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, pp. 289–297, 2016.

[28] R. Boselli, M. Cesarini, F. Mercorio, and M. Mezzanzanica, "Using machine learning for labour market intelligence," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 330–342.

[29] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 855–864.

[30] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[31] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[32] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar *et al.*, "Universal sentence encoder for english," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 169–174.

[33] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, "Link prediction techniques, applications, and performance: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 553, p. 124289, 2020.

[34] L. Cai and S. Ji, "A multi-scale approach for graph link prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 34, no. 04, 2020, pp. 3308–3315.

[35] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, pp. 5165–5175, 2018.

[36] L. Zhang, D. Zhou, H. Zhu, T. Xu, R. Zha, E. Chen, and H. Xiong, "Attentive heterogeneous graph embedding for job mobility prediction," in *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2021, pp. 2192–2201.

[37] C. Wang, H. Zhu, Q. Hao, K. Xiao, and H. Xiong, "Variable interval time sequence modeling for career trajectory prediction: Deep collaborative perspective," in *Proceedings of The ACM Web Conference (WWW)*, 2021, pp. 612–623.