Tripartite Vector Representations for Better Job Recommendation

Mengshu Liu, Jingya Wang, Kareem Abdelfatah, Mohammed Korayem

CareerBuilder LLC

Greater Atlanta Area, Georgia, US

{mengshu.liu,jingya.wang,kareem.abdelfatah,mohammed.korayem}@careerbuilder.com

ABSTRACT

Job recommendation is a crucial part of the online job recruitment business. To match the right person with the right job, a good representation of job postings is required. Such representations should ideally recommend jobs with fitting titles, aligned skill set, and reasonable commute. To address these aspects, we utilize three information graphs (job-job, skill-skill, job-skill) from historical job data to learn a joint representation for both job titles and skills in a shared latent space. This allows us to gain a representation of job postings/ resume using both elements, which subsequently can be combined with location. In this paper, we first present how the representation of each component is obtained, and then we discuss how these different representations are combined together into one single space to acquire the final representation. The results of comparing the proposed methodology against different base-line methods show significant improvement in terms of relevancy.

KEYWORDS

job recommendation, vector representation, location embedding

1 INTRODUCTION

Online recruiting and job portals like Careerbuilder.com, Linkedin.com, and Indeed.com, have become the norm in the talent acquisition business. Millions of jobs are posted and even more resumes are uploaded daily. Different machine learning and information retrieval models have been applied to analyze these resumes and job descriptions, and multiple efforts have been made to match the two parties of the recruiting process. A good job/resume representation helps to improve many downstream products that in turn support the company's goal of empowering employment and helping job seekers find jobs and the training they need. Specifically, it facilitates matching job seekers and employers by improving our search and recommendation products.

The content of a job posting incorporates all aspects of a position. Entities like job title, required skills, experience, degrees, benefits, company culture, location, etc, can be extracted and normalized. Among these, title, skills, and location are the top factors when defining a job position. While a job title determines the nature of a job, skills enrich the job title, and differentiate jobs with the same title by identifying their niceties. In order to offer a good job match, an accurate representation consisting of both is needed. Rather than simply using various word representations of job title and skills, recent works [1–3] start to consider both the text content and the relationship between matching pairs. In our previous work, we propose a novel representation learning based solution, which learns job and skill vector representations into a shared latent space using three pre-processed graphs [4]. To extend this work, we consider the interconnection of both job title and skills to learn the vector representation of job postings/ resumes. By using a retrofitting model similar to the work of [5], we combine the pre-trained representations of job title and skills into one vector to represent a job or resume. In terms of location, majority of job seekers prefer jobs within reasonable commute distance. For many people, relocation is not an option, and short commute is always a big plus. Similarly, for most companies, remote employees are not preferred either. Therefore, we explicitly include a location vector in our representation. To achieve quick, accurate vector search and recommendation, we utilize FAISS [6], which is a library for efficient similarity search and clustering of dense vectors.

Our contributions in this paper are as follows:

• The vector representation proposed is applicable for both job postings and resumes. It's not only a flexible representation to obtain similar jobs or similar candidates, but also provides a direct mapping of jobs and resumes.

• We generate a more holistic vector representation jointly learned for both titles and skills which can be used in the job recommendation system.

• We incorporate the location explicitly included in our representation.

• We employ retrofitting to refine the job and skill vectors using semantic lexicons detailed in Section 3.3.
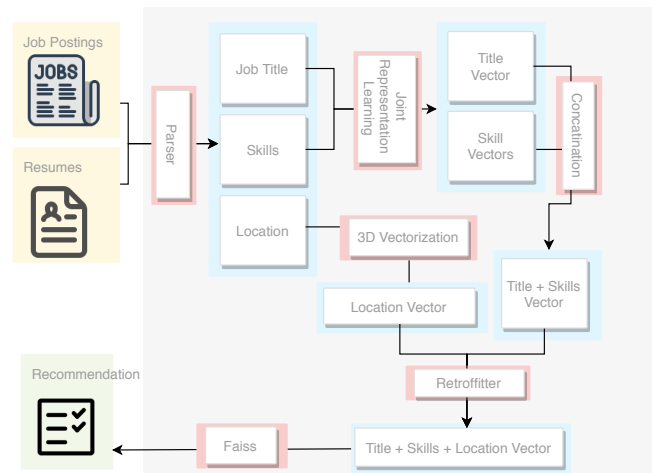

Figure 1: Workflow to generate recommendations using our tripartite vector representations.

2 RELATED WORK

The two major solutions of job recommendation in existing works are content-based representation learning and collaborative filtering (CF). CF builds a very high dimension sparse matrix to maintain the relationship between user and product [7, 8]. In our paper, this relationship is between each pair of job and skill. Inspired by the natural language Processing approach (NLP) of skip-gram [3], [9] proposes to learn item embedding based on interaction history in the form of a sequence. [10] presents an architecture of CF and points out the common problem of cold start. CF relies on historical interactions, therefore, when a new candidate registers or a new job is posted, no interaction record is available. Content-based method on the other hand, makes recommendations according to the characteristics of users and products as in [11].

Most recent works, including our proposed system, use hybrid approaches of both profile properties and interaction history [12]. [1, 2] jointly learns feature representation of users and their selected items in one convolutional neural network. [13] concatenates the learned sparse and dense representation of user's activity history, query, and profile as input. In this case, the recommendation problem is treated as an extreme multi-class classification problem. The classifier is a series of non-linear activation functions followed by a softmax. [14] employs a convolutional neural network to encode the user profile followed by a Long short-term memory (LSTM) [15] to encode the user's interaction history in training and continue the LSTM on time axis as a decoder to predict user's next activity in testing.

Our work is similar to the hybrid approaches in terms of learning the embedding of job titles and skills based on both their characteristics and interactions. Meanwhile, it is different that, instead of simply pairing the related jobs and skills, our profile representation models three specific relationships simultaneously between jobs and required skills. Moreover, we employ retrofitting to achieve better representation.

3 METHODOLOGY

In this section, we describe the design and methodology of our tripartite vector representation. A job posting contains important information of the hiring position. Most importantly it includes job title, skills, and location. Here, we use a combined representation vectors of job titles and skills, and obtain a vector representation for location transformed from latitude and longitude. Figure 1 shows the design of our method. Job posting and resume data go to an in-house job parser, where job title/skills/location are extracted. While titles and skills are jointly trained by a representation learning framework (Section 3.1. Location information is also vectorized and normalized (Section 3.2). In order to combine the tripartite, the title representation is first retrofitted by a list of skills associated with each job before added with the location vector (Section 3.3).

3.1 Job Title and Skill Vector

Title and skills are the defining features of any job. Embeddings for both titles and skills are learned in the same k-dimensional space [4], by utilizing three types of information networks from historical job data: (i) job-job transition network, (ii) skill-skill co-occurrence network, and (iii) job-skill co-occurrence network. Our

goal is to encode the local neighborhood structures captured by the three networks.

For the job-job transition graph, we assume that the transition between similar jobs x and y is more likely to happen than non-similar jobs x and z. Let $A_{xy}^j = \langle w_x, w_y \rangle$, the dot product of the two embedding vectors, be the affinity score between job x and job y, and D_{jj} (job-job) represents the transition relationship of job triplets (x, y, z). The objective is to learn representation W so that

$$O^{jj} = \min_{W} - \sum_{(x,y,z) \in D^{jj}} \ln \sigma(A_{xy}^{jj} - A_{xz}^{jj}) \tag{1}$$

Where sigmoid function $\sigma(v) = \frac{1}{1+e^{-v}}$ is used as the probability function which preserves the order $A_{xy}^j > A_{xz}^j$.

Similarly, for the skill-skill graph, coexisting skills x and y which appear on the same job posting or the same resume are closer to each other than non-coexisting skills such as x and z. Let D^{ss} (skill-skill) be the set of training triplets of skills with coexisting relationship, our objective here is

$$O^{ss} = \min_{W'} - \sum_{(x,y,z) \in D^{ss}} \ln \sigma(A_{xy}^{ss} - A_{xz}^{ss}) \tag{2}$$

Moreover, for the job-skill graph, if skill y^s appears on the advertisement of job x^j, its embedding vector is more similar to x^j than non-related skill z^s. Given the set of training triplets D^{js} (job-skill), our desired vector representation of jobs W and skills W' are learned according to the objective

$$O^{js} = \min_{W W'} - \sum_{(x^j, y^s, z^s) \in D^{js}} \ln \sigma(A_{xy}^{js} - A_{xz}^{js}) \tag{3}$$

Finally, to achieve high quality job and skill embedding, we optimize this joint objective function

$$O(W, W') = \min_{W, W'} O^{jj} + O^{ss} + O^{js} + \lambda \cdot (\| W \|_F^2 + \| W' \|_F^2) \tag{4}$$

where λ is the coefficient of the l_2 regularization term $\| \cdot \|_F^2$ to avoid over-fitting.

Vectors of dimension size 50 are obtained for 4325 unique job titles and 6214 skills, using joint Bayesian Personalized Ranking (BPR) [16].

3.2 Location Vector

Location is another key factor in a job posting. Commute time matters when people are choosing a position. The common practice when dealing with location is to pre-filter or post-filter the recommendations with a fixed radius. This method has a few downsides: 1. Some jobs are less sensitive to distance than others. For example, people are more willing to commute longer with a highly compensated job than a minimum wage part-time one; 2. The system is more difficult to implement because of the extra layer of filtering. 3. Specifically for our case where Faiss is used for similarity search, only one index file is needed if location is embedded in the vector.

Before latitude and longitude can be added to our embedding model, they need to be transformed since they are not on the same scale as the title and skill vector. Geo-locations are three dimensional in its nature, and to represent location in a similar fashion as title and skill vectors, we did a transformation of latitude and longitude, as shown in Figure 2:

$$x = \cos(\theta) \times \cos(\phi) \quad (5)$$

$$y = \cos(\theta) \times \sin(\phi) \quad (6)$$

$$z = \sin(\theta) \quad (7)$$

Where θ represents the latitude and ϕ is the longitude.

Thus, location is represented as a normalized three dimensional vector, which can be later combined with the title and skill vector.
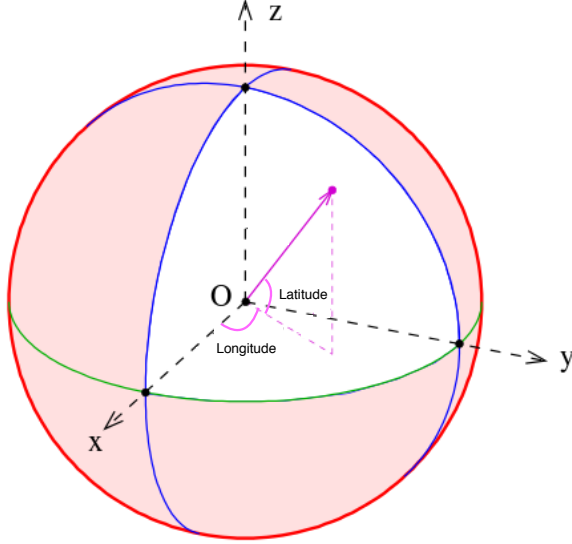

Figure 2: Latitude and longitude can be converted into a three-dimensional coordinates.

3.3 Combination of title, skill, location

Job postings with the same title might require different skill sets at different companies in different industries. The next natural question is how to combine the title and skill vectors together to a personalized vector for a specific job, given both title and skill vectors are trained in the same latent space.

To assemble the vectors back into one to represent the job posting, we apply the retrofitting method [5] to combine the vectors of the job title and all skills of the job posting. This method adjusts the position of the job title based on the skills appeared in the job posting. For example, consider one job posting looking for a web developer, and a person has a recent title of JavaScript developer. The two job titles are similar, though still different. If JavaScript is listed as a top skill in the job posting, the distance between the job and the resume will be shortened. This gives the person a better place in ranking, even though the job title is not a perfect match.

In Faruqui's work [5], they implemented a retrofitting method to adjust any pre-trained embedding using semantic lexicons:

$$q_i = \frac{\sum_{j:(i,j)} \beta_{ij} q_j + \alpha \hat{q}_i}{\sum_{j:(i,j)} \beta_{ij} + \alpha_i} \quad (8)$$

where q_i is the modified vector, $\hat{q}_i$ is the initial vector, q_j are the neighbors. This is derived by minimizing the distance between initial q_i and neighbors. And for their case, the results converge after 10 iterations.

Intuitively, if we add two vectors together, the result vector will lie in between the initial two vectors. Adjusting vectors by adding the vectors of neighboring words will bring words similar in meanings closer to each other. For example, in Figure 3, $\hat{Smile}$ and $\hat{Tears}$ are the initial vectors for two very different words, so they are further apart in direction (cosine similarity). After the tuning using a neighboring word: $happy$, the new vectors for smile and tears are closer than before, since the tears are in the context of "happy tears".
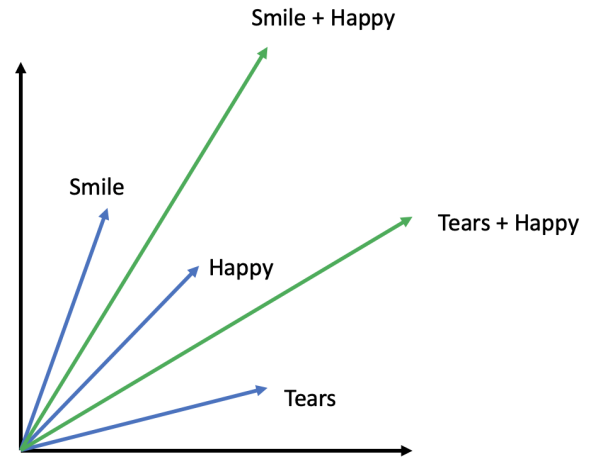

Figure 3: Addition of embedding vectors.

Similarly, we applied the above operation to our job posting data, and generated a combined representation of the job posting using title and the skills. Jobs with similar set of skills as well as the same title will have a higher rank than the ones with the same title but less overlapping skills. Rather than going through several iterations as in the original paper, we only apply the updates once, as skill vectors are not updated in the process. We can view this one time update as fine tuning of the title vector, or the calculation of the "job + title" vector, shown as below, where n is the number of skills in a job posting, which are given the same weight:

$$q_{\text{job}} = \frac{n \cdot q_{\text{title}} + \sum q_{\text{skill}}}{2n} \quad (9)$$

Location vector is then concatenated with the new vector. The weight of the location vector can be adjusted based on how the jobs are sensitive to distance.

$$q_{\text{job+location}} = [q_{\text{job}}, x, y, z] \quad (10)$$

4 RESULTS

To test and evaluate our proposed system, we use a real dataset (jobs and users) via CareerBuilder.com. CareerBuilder operates the

largest job posting board in the U.S. and has an extensive growing global presence, with millions of job postings, more than 60 million actively searchable resumes, over one billion searchable documents, and more than a million searches per hour.

Each job/ resume is parsed into job title, skill list, and location (Figure 1). We then translate these three parts into their vector representations, which are later combined to the final job posting vector. We have also curated a list of top skills for each job title, in case skills are not available. The curation is based on millions of resumes and job postings, as well as human knowledge. We use this skill list to generate a vector if no skill is given. This provides a more accurate representation of the job than just using a job title. These vectors all go into our recommendation engine for different recommendation products.

We calculate the vectors on 300K actual job postings on the Careerbuilder Website, using 6 different models (FastText, Word2Vec, Glove-6B-300 , Glove-840B-300 [17–19], and our proposed retroffiter model with and without location embedding), so each job posting has 6 vectors for comparison. For each of these vectors, we calculate the top 50 similar vectors using FAISS. For job posting data, a flat index method is used. For resume data, which is much larger in size, we compress them to 16 blocks in 64-dimension and build the inverted indexing of size 65535. Four metrics are computed for evaluation: 1) Distance, which is the geographic distance between the recommended job and the input. 2) In-range job counts, which is the number of jobs within 50 miles in the top 50 recommended jobs. 3) Title match rate, which is the percentage of recommended jobs with the same job title as the input. 4) Title coverage, which measures the title match rate within 50 miles. The results for different models are computed and compared.

Table 1 shows that by explicitly including latitude and longitude in our embedding, the average distance of recommended jobs is reduced by 90%. All top 50 jobs are within a reasonable proximity of the original input job.

Table 2 shows the number of jobs within 50 miles in the top 50 jobs increased dramatically for retrofitter with location embedding. Both table 1 and table 2 show that the base-line methods are not location sensitive, even though location is included in the text file. Implicitly having location in the embedding is far from enough, and the advantage of explicitly including location in the embedding model is prevailing.

Table 3 shows title match rate and coverage. Our retrofitter model with no location embedding has a much higher title match rate and coverage than the base-line methods as well as the retrofitter model with location. However, the drop in performance in our location retrofitter model purely comes from the limited relevant jobs in close proximity, which is a trade-off we have to make. A perfect job match from ten thousand miles away is simply not a perfect job for most people. Even so, our retrofitter model still has a better results in both measurements than the base-line methods.

5   CONCLUSION AND FUTURE WORK

In this paper, we discuss the representation model which can be used for a recommendation system and it is currently being utilized within CareerBuilder. Three facets of a job posting are considered: job title, job skills, and location. While job title carries the most

Model	Average	Median	STD
FastText	940.85	760.69	819.33
W2V-300	936.46	754.67	818.58
Glove-6B-300	959.54	785.40	823.14
Glove-840B-300	942.22	763.72	816.83
Retrofitter-no loc	902.918	727.087	836.275
Retrofitter-loc	**90.417**	**70.177**	**95.673**

Table 1: Distance statistics for different representations.

Model	Average	Median
FastText-300	5.637	3.0
W2V-300	5.67	3.0
Glove-6B-300	5.108	3.0
Glove-840B-300	5.503	3.0
Retrofitter-no loc	1.614	1.0
Retrofitter-loc	**21.420**	**20.0**

Table 2: Number of jobs with a distance less than 50 miles in the top 50 jobs.

Model	Carotene-Match (%)	Coverage (%)
FastText-300	11.0	3.1
W2V-300	11.2	3.1
Glove-6B-300	10.6	3.0
Glove-840B-300	10.6	3.0
Retrofitter-no loc	**68.9**	**14.7**
Retrofitter-loc	**14.1**	**5.5**

Table 3: This table shows two metrics. First, it shows the percentage of jobs with the same title match. Second, it shows the coverage which means the percentage of recommended jobs within the 50 mi with the same job title.

weight in determining what a job is, skill set defines the nuances which differs from job to job. Most job seekers are also very sensitive to the location of a job. In our model, we encompass all three of these aspects, and are able to give location sensitive, highly related job recommendations to our users.

There are a number of improvements being worked on such as : Develop an inductive learning framework to accommodate newly emerged job titles and skills, as the current model is transductive, and representation vectors only exist if it is in the input graph; Incorporate more features in the job representation such as education and previous experience; Adjust the location embedding in a more quantifiable way to control the radius of recommended jobs; Combine the current representation model with other models to provide a better results; Apply different weights to skills based on their importance to the job title.

REFERENCES

[1] Wenyi Huang, Zhaohui Wu, Chen Liang, Prasenjit Mitra, and C Lee Giles. A neural probabilistic model for context based citation recommendation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[2] Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. User modeling with neural network for review rating prediction. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

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

[5] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, 2015.

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[7] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM, 2015.

[8] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244. ACM, 2015.

[9] Oren Barkan and Noam Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[10] Lili Wu, Sam Shah, Sean Choi, Mitul Tiwari, and Christian Posse. The browsemaps: Collaborative filtering at linkedin. In *RSWeb@ RecSys*, 2014.

[11] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[12] Kyo-Joong Oh, Won-Jo Lee, Chae-Gyun Lim, and Ho-Jin Choi. Personalized news recommendation using classified keywords to capture user preference. In *16th International Conference on Advanced Communication Technology*, pages 1283–1287. IEEE, 2014.

[13] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198. ACM, 2016.

[14] Liangyue Li, How Jing, Hanghang Tong, Jaewon Yang, Qi He, and Bee-Chung Chen. Nemo: Next career move prediction with contextual embedding. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 505–513. International World Wide Web Conferences Steering Committee, 2017.

[15] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[19] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.