Document-based Recommender System for Job Postings using Dense Representations

Ahmed Elsafty[*] and **Martin Riedl**[†] and **Chris Biemann**[‡]
[*] XING SE, Hamburg, Germany
[†] IMS, Universität Stuttgart, Germany
[‡] Language Technology, Universität Hamburg, Germany
ahmed.elsafty@xing.com, martin.riedl@ims.uni-stuttgart.de,
biemann@informatik.uni-hamburg.de

Abstract

Job boards and professional social networks heavily use recommender systems in order to better support users in exploring job advertisements. Detecting the similarity between job advertisements is important for job recommendation systems as it allows, for example, the application of item-to-item based recommendations. In this work, we research the usage of dense vector representations to enhance a large-scale job recommendation system and to rank German job advertisements regarding their similarity. We follow a two-folded evaluation scheme: (1) we exploit historic user interactions to automatically create a dataset of similar jobs that enables an offline evaluation. (2) In addition, we conduct an online A/B test and evaluate the best performing method on our platform reaching more than 1 million users. We achieve the best results by combining job titles with full-text job descriptions. In particular, this method builds dense document representation using words of the titles to weigh the importance of words of the full-text description. In the online evaluation, this approach allows us to increase the click-through rate on job recommendations for active users by 8.0%.

1 Introduction

Recommender systems aim at providing recommendations for services that are targeted to specific users. The majority of such systems are applied in the field of e-commerce for e.g. product recommendations (Lu et al., 2015). In business-oriented networking platforms, recommender systems propose job recommendations to users.

In this deployment paper, we target the development of content-based methods for job recommendations focusing on German job advertisements. Based on our social online platform for professionals, 45% of the traffic is driven by recommendation services for job postings. Thus, improving the job recommendations is expected to result in higher user interactions.

Our online platform's infrastructure consists of several recommendation stages in order to recommend job postings to users. In this paper, we focus on the so-called More-Like-This (MLT) component that recommends job postings based on previous users interactions with other job postings. Our current system consists of an ensemble of recommendation retrieval, filtering and re-ranking stages in order to recommend relevant job postings to users. For this, it exploits metadata of a job posting like keywords, disciplines and industries in which the job is categorized.

There are multiple issues when using exact keywords or category matching for ranking job postings. First, the document collection, with over 1 million job postings, is fairly huge and too diverse to fit into the small number of available categories, e.g. 22 disciplines such as *Law* or *Media*. Second, strict word matching leads to recall issues, for instance, *J2EE Developer* will not be similar to *Software Engineer*. Thus, employing a sparse vector representation is not appropriate for retrieving similarities between job postings. In addition, due to the cold start problem (Schein et al., 2002), using solely metadata of job postings or users is not suitable, especially for new users, for which only marginal or no information exists. Furthermore, metadata can be entirely missing or incorrect (e.g. outdated or on purpose).

Consequently, we will compute similarities between job postings based on dense vector representations. Recent document embedding techniques learn meaningful syntactic and semantic relationships based on word occurrences in the text. In this paper, we use dense vector representation of documents to score similarities between job postings based on their full-text descriptions and titles. First, we create a dataset for an offline eval-

Proceedings of NAACL-HLT 2018, pages 216–224
New Orleans, Louisiana, June 1 - 6, 2018.

uation consisting of similar job postings based on user co-interactions. Then, we construct an evaluation metric based on the classification of similar and non-similar items. Testing multiple embedding models and weighting functions, the best performance is achieved when building embeddings based on the job description with an increased weight for words that appear in the job title. Finally, the model is used in an online A/B test to assert its performance on live data.

2 Related Work

Recommendation systems can be divided into three categories (Resnick and Varian, 1997): content-based, collaborative filtering and hybrid models. Content-based recommender systems use items the user positively interacted with in the past, calculate similarity scores between item pairs and rank the new recommendations accordingly (Lops et al., 2011). Collaborative filtering approach suggest items to a given user, that other similar users positively interacted with (Koren and Bell, 2015). Hybrid methods combine both techniques (Burke, 2007). To avoid cold start problems, due to missing data, we focus on content-based approach here.

Dense numeric representations are commonly used to compute the similarity between content of documents (Hofmann, 2000) in order to reduce sparse count-based representations (Koren et al., 2009), which require huge amounts of memory. *Word2Vec* (Mikolov et al., 2013) has become a standard method that builds dense vector representations, which are the weights of a neural network layer predicting neighboring words. To retrieve a document representation, we compute the average of all vectors of the words in the documents. *Word2Vec* was also used for recommender systems to re-rank items based on vector correlations (Musto et al., 2015; Ozsoy, 2016). A modification that allows the usage of predicting arbitrary context in order to compute word representation is named *Word2VecF* and was introduced by Levy and Goldberg (2014). Document embedding techniques like *Doc2Vec* (Le and Mikolov, 2014) assigns each document a single vector, which gets adjusted with respect to all words in the document and all document vectors in the dataset. In an attempt to reduce *Doc2Vec* complexity and training corpus size dependencies, *Doc2VecC* (Chen, 2017) uses the same architecture as *Word2Vec*'s, except that it samples words from the document in each training iteration by creating a document vector out of their average. The vector is then used to help predicting neighboring words during training.

To our best knowledge, no dataset is available to evaluate the performance of ranking similarities between jobs. Most similar is the dataset of the RecSys 2016 task (Abel et al., 2016). However, the task of this challenge was to learn the retrieval relevant documents based on user metadata and the approaches use supervised systems. In addition, datasets for document similarity exist, but do not focus on job postings. For the task of document similarity, the 20 Newsgroups (Lang, 1995) and TREC-AP (Lewis et al., 1996) datasets are commonly used. Here the task is to assign documents to a predefined category. Thus, the task is more related to document clustering than information retrieval of similar documents. Also related are semantic text similarity tasks, where two sentences have to be scored regarding their similarity with a score between 0 and 5 (Baudiš et al., 2016). Paraphrasing is another aspect that is important for document similarity. Bernhard and Gurevych (2008) introduced a dataset for paraphrasing both questions and answers in order to enhance the results for the information retrieval.

Related work was done by Fazel-Zarandi and Fox (2009), who introduced a method for matching jobs with job seekers. Whereas this fits to the RecSys 2016 task, this does not cover job posting retrieval of similar jobs. Furthermore, supervised approaches exist that predict jobs to candidate users e.g. Poch et al. (2014). In addition, Kessler et al. (2008) introduced a dataset based on French job offers and presented a system for ranking relevant jobs to candidates based on a jobs-to-candidates similarity metric.

3 Method

We hypothesize that job offers are semantically similar if the words used in its description are semantically similar. In addition, metadata of job offers like e.g. location of employee, title or qualifications are relevant for similarity computations.

3.1 Data Retrieval

Based on our job recommendation platform, we extract user interactions (bookmarks and reply intentions) from March 2014 to March 2017 as pairs of users and jobs. First, we remove users and jobs

that have less than two interactions overall. Then, users are filtered out that have a number of overall lifetime interactions that exceeds the 99th percentile of all users. We consider such users as outliers. As click data of users is noisier than the bookmark data, we do not use clicks for the creation of this dataset.

Whereas our job recommendation platform features job postings in English and German, most users prefer German postings. This also affects our dataset, which comprises of 91% of German postings. While training semantic models for multiple languages is possible (e.g. Søgaard et al., 2017), we focus on German job postings, as found by a language detector[1].

3.2 Data Preprocessing

Before training, HTML tags, URLs and e-mails were removed using regular expressions, as early models showed a huge bias towards HR contact emails and job agencies that include boilerplate URLs in the job description footers. All special characters like non-alphabetical characters, interpunctuation and bullet points were removed. Initial semantic models required large vocabularies due to writing variations of the same word. For instance, the term *Java* occurs three times: *Java, java* and *JAVA*. Hence, we lowercase job posting texts and replace numbers with a placeholder (Abdelwahab and Elmaghraby, 2016). Finally, the document is stemmed using Snowball stemmer[2].

3.3 Ground Truth Construction

As manual annotation is expensive and time consuming – experts would have to go through N^2 jobs for completeness (where N is the sample size) – we automatically build a dataset using interactions of users from our job recommendation system. For building the dataset, we assume that two jobs are similar, if two or more users are interested in these two jobs. This assumption follows our intuition that users bookmark relevant jobs that are similar. However, this source of information can be noisy, due to random surfing, accidental clicks or when job postings are bookmarked for a friend and not for the profile owner. Hence, by selecting only jobs where several users co-interacted with, we can increase the probability that such jobs are similar.

[1] https://pypi.python.org/pypi/langdetect

[2] http://snowballstem.org/

In order to validate this assumption, a proper representative sample should be randomly selected and assessed by human experts. Since we did not have the resources for manual judgments, we compare the metadata from the job postings. For example, for 616,000 pairs of similar jobs, 70.02% of them share the same discipline. The other about 30% span across similar disciplines like e.g. *Marketing*, *Quality Assurance* and *Project Management* that have high topical overlap. However, discipline pairs exist that may not be considered as similar, like *Engineering & Technical* and *Management & Corporate Development*. Such "noise" in addition to slight diversity in bookmarked jobs is expected due to the automatic generation of the dataset. Nevertheless, such non-trivial discipline combinations have very low frequency. Better dataset construction approaches could involve increasing the number of users who co-interact with the job. Whereas this increases confidence, it decreases the dataset size drastically and could impose a bias for popular vs. rather sparingly sought disciplines.

Offline Evaluation Setup

The two jobs with the titles *Java Developer, Hamburg* and *Java Backend Developer, Stuttgart* are examples of two very similar job postings with different locations. Due to the location difference they fit to two different types of users: those who live close to Stuttgart and those close to Hamburg. For the creation of our dataset we consider the following: if there is no user co-interaction between two jobs, they will not be considered similar in the dataset. The same applies to similar jobs postings with large creation time stamp differences. For example, users that have been interested in jobs posted in 2014, might not be interested in similar jobs posted in 2017.

Inspired by the information retrieval-based evaluation approach by Le and Mikolov (2014), we created our dataset. In their approach, they created triples (s, p, n) that consists of a paragraph s, a similar paragraph p and a non-similar randomly sampled paragraph n. Inspired by this dataset, negative sampling in *Word2Vec* and cross validation, we extended the approach to construct a dataset of positive and negative samples as described in Algorithm 1. For each job, we create 10 folds of 10 similar and 40 non-similar jobs.

This algorithm returns a list of triplets consisting of the job j, a list of similar jobs Pos_f and a

Algorithm 1 Building the Evaluation Dataset

```
procedure CREATE_DATASET(jobs)
    output ← [ ]
    for j in jobs do
        for f = 1 ... 10 do
            Pos_f, Fold_f ← [ ], [ ]
            for i = 1 ... 10 do
                p_i ← random_similar_job
                Pos_f.append(p_i)
                Fold_f.append(p_i)
            for i = 1 ... 40 do
                n_i ← random_job
                Fold_f.append(n_i)
            shuffle(Fold_f)
            output.append((j, Pos_f, Fold_f))
    return output                ▷ A list of triplets
```

shuffled list $Fold_f$ of similar and non similar job postings to the job j. During evaluation, every job posting in the shuffled $Fold_f$ is compared to the corresponding job j to compute a similarity score, which is used to rearrange $Fold_f$. The precision measure is used to compare the list cutout at 10 (retrieved), and the relevant job postings in Pos_f.

Sampling "negative job postings" from the entire dataset, we reduce the chance of fetching similar job postings that our dataset did not capture. To reduce the chance of false negatives, we increase the size of the dataset by randomly generating 10 lists for each job, resulting in a dataset of 112,000 distinct job postings and 12,000 shuffled lists.

In Figure 1, we show the similarity between job titles (we translated them from German to English) based on a *Doc2VecC* model (500 dim vectors, 10 window size, 15 negative sampling, 20 iterations) using *T-SNE*. The job colored in black (*Lean Java Expert Munich*) represents the job being evaluated, and the gray ones represent similar (positive) job postings sampled from our user interactions. The remaining jobs depict non-similar (negative) jobs sampled from the entire corpus. Based on the figure we have three observations: first, most positive jobs are closest to the queried job and focus on the same topic, namely Java development. Second, some of the "negative" jobs are relevant, e.g. *FrontEnd developer* and *Teamleader in IT Development*, and have a close distance to the queried job. Third, we observe multiple clusters: for example, in the upper right corner we observe a "media management" cluster, and in the center a "project management" cluster.

4 Offline Evaluation

In this section, we first report results that are computed based on full-text job descriptions. Then, we exploit the performance using the job titles. To complete our experiments we show results for the combination of job titles and job descriptions.

In our experiments, we use commonly used hyperparameters (Siencnik, 2015; Levy et al., 2015; Yao et al., 2017). We tested different combinations of window size (2, 5, 10), model choice (skip-gram vs. continuous bag of words) and number of dimensions (100, 250, 500) and picked the following hyperparameters for the rest of the experiments: skip-gram model with vector size of 500, window size of 10, 15 words for negative sampling, 20 iterations and a threshold for the minimum count of 5.

Due to the ranking nature of the task, we report results based on the precision at 10 (P@10) score considering the ten highest ranked jobs. Since we have 10 positive similar job postings in each list, the P@10 can be interpreted as an average percentage of jobs in the top 10 which are actually similar and can have a maximum value of 100%.

Full-Text Job Description: As a baseline we represent each job as a word vector of TF-IDF scores based on the job description and use the cosine similarity for re-ranking the jobs (see Table 1). This baseline performs lowest with a P@10 score of 8.69% showing that such a sparse representation is insufficient to identify similarities between documents.

Model	Stemmed	Doc. Context	TF-IDF weights	P@10
TF-IDF				08.69 %
Word2Vec				54.84 %
Word2Vec	*			56.22 %
Word2VecF	*	*		61.12 %
Word2VecF	*	*	*	62.81 %
Doc2VecC	*			62.73 %
Doc2VecC	*		*	**64.23 %**

Table 1: Precision scores of word embedding models using full-text description only.

Using *Word2Vec*, we achieve a score of 54.85%, demonstrating that dense representations perform much better on our dataset than using sparse word representations. Stemming the documents yields to a further improvement (+1.38) and reduces the

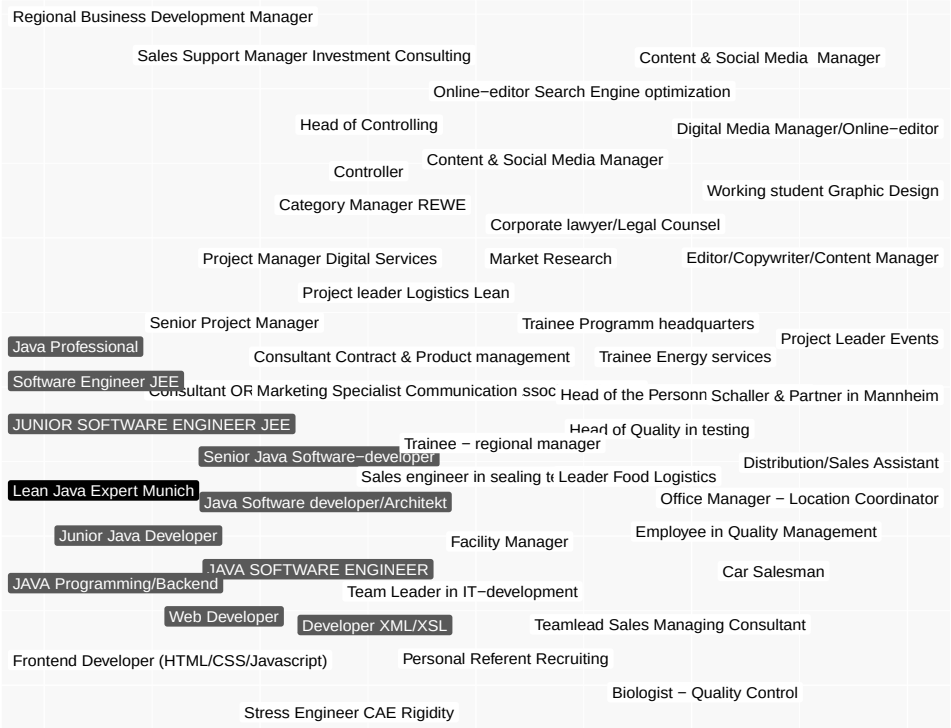

Figure 1: T-SNE representation of a sampled list after a model evaluation (job titles are translated from German to English).

training time, due to the smaller vocabulary size. Combining the stemmed representation with context information – we use the document IDs and compute the representation using *Word2VecF* – we achieve improvements of +4.9 points in comparison to the standard *Word2Vec* approach. In this setting, we predict the document ID for every word (unlike predicting its neighbors in *Word2Vec*). Such a "window" can be seen as a global context over the entire document, which performs better than using *Word2Vec* local context. By extending this model with *TF-IDF* scores, the performance is boosted by another +1.69 points to a score of 62.81%. In addition, we compute similarities with *Doc2VecC* using stemmed job descriptions as documents. This method performs best among the single tested models (62.73%), and scores highest when combined with *TF-IDF* weights achieving a score of 64.32%.

Job Title: Whereas the models mentioned above use the job description, most users click on jobs based on the title. Thus, we investigate building document vectors using solely title information using *Word2Vec* with stemmed words and *TF-IDF* weights. This experiment should reveal whether computational efforts can be reduced by using less information.

As shown in Table 2, *Word2Vec* using title vector yields a P@10 of 58.79%. Whereas these results are lower, they are still impressive, as we only have one "sentence" with an average of 4.8 words. In addition, we consider job titles as documents and use *Doc2Vec*. Given the small sentence size, it can be trained in reasonable time. In our experiments, we test this model with various dimensions (100, 250, 500) and keep the other parameters fixed.[3] Testing the effect of *Doc2Vec* on titles that have not been seen before, we achieve a low precision of 20.66%. This was tested by dropping the document vectors generated for our dataset after training, and using the model to infer the doc-

Model	P@10
Word2Vec – 500 dim.	58.79%
Doc2Vec – 100 dim.	59.87 %
Doc2Vec – 250 dim.	60.03 %
Doc2Vec – 500 dim.	**61.23 %**
Doc2Vec – 500 dim. – Inferred	20.66 %

Table 2: Results using the title with various embeddings.

[3]We use a distributed bag of words model, window size of 10, minimum word count of 5 and a sampling rate of 1e-5.

ument vectors again. When predicting vectors for unseen documents, the model infers the title vector based on its words, however, information loss is to be expected. This implies that the model cannot be efficiently used in an online scenario or in a pipeline of streaming jobs since the entire model has to be retrained on the full data to obtain a better word coverage.

Title weighted description: Next, we combine *Doc2VecC* word vectors of the description weighted by the *TF-IDF* values with weights, indicating if a word is contained in the title. For the combination we use the following formula:

$$D(w_1, \ldots, w_k) = \frac{\sum_{i=1}^{k} TF\text{–}IDF(w_i) * V(w_i) * \lambda(w_i)}{\sum_{i=1}^{k} TF\text{–}IDF(w_i) * \lambda(w_i)}$$

$$\lambda(w_i) = \begin{cases} c, & \text{if } w_i \in \text{title}, c > 1.0 \\ 1, & \text{otherwise} \end{cases}$$

with $\lambda(w_i) = c$ with the constant $c > 1.0$ if w_i is contained in the title and $\lambda(w_i) = 1.0$ if the word w_i is not contained in the title.

When constructing the document vector D containing k words, all word vectors $V(w_i)$ are multiplied by their corresponding scalar *TF-IDF* values and the constant c if the word appears in the title. Then, the vectors are summed up and divided over the weights to calculate the weighted average. Based on findings in the previous section, we already know that the title provides enough information to distinguish jobs. Thus, weighting title words higher when averaging pulls the document vector a bit closer to the title in the vector space. Using $c = 5$, we achieve result with a precision score of 73.05%. It shows that by choosing a proper weighting function, we can achieve better results than changing the entire model. In industry, often not the best performing system is used, but the one which can also be applied efficiently to new and unseen data. Since word vectors are precomputed, document vectors can be computed online in the platform pipeline, such that vectors of new documents are available when needed by the recommender services.

5 Online Evaluation

The existing recommender system uses *Elasticsearch*[4] to retrieve job postings based on the user's metadata, then exploits the user's previous interactions with job postings to rank the recommendations in a doc-to-doc similarity fashion via TF-IDF. This is used as a ranking baseline. For our online evaluations, we use the retrieval module from *Elasticsearch*, and plug our fastest and best performing job representation (title weighted *Doc2VecC*) model into a new system to re-rank the retrieved documents.

Before we performed the online evaluation, we analyzed whether the results with the *A/B* test will differ using different semantic representation, to prove whether the *A/B* test will lead to any meaningful result. For this, we re-rank the same retrieved recommendations for 2000 users sampled from the most active users on the platform.

As shown in Table 3, the intersected (common) recommendations (μ) between the two systems does not exceed 36% for all K ranks in the recommendation lists, with a decreasing standard deviation (σ). This reveals that the changes have huge impact on the rankings.

Top *K*	**Intersection**		**Avg Distance (km)**	
	μ	σ	$Existing$	New
4	30.1%	32.16%	287	179
10	35.5%	27.89%	293	188
20	35.4%	25.69%	325	195
50	34.1%	21.28%	336	192

Table 3: Pre-analysis for the A/B test. We show the mean and standard deviation of common recommendations returned by the systems on different ranks K, and the average distance of job postings to the user in kilometers (km).

In addition, we analyze the average distance in kilometers (km) of the recommended job postings to the user's location. The new model favors to rank jobs with closer distance at higher position: the top 4 recommendations are 30% closer and even 60% closer for the top 50 jobs. This is an important finding, as we hypothesize that users prefer jobs that are closer to their location. Job locations are usually included in the title, allowing vectors of cities to contribute higher in the title weighted averaging approach.

To perform the *A/B* test, we conduct a controlled experiment by selecting active users (with at least a single previous job interaction) and split them into two groups: one group gets job posting recommendations ranked by the *Elasticsearch*, and

[4] https://www.elastic.co

the second group gets job posting recommendations ranked by our best system (title weighted *Doc2VecC* model).

First, we apply an *A/A* test (Kohavi et al., 2009) to test for any split bias: both groups get recommendations from the existing system for 30 days. Then, the *A/B* test is conducted over the period of 20 days. The success metric is the Click-Through-Rate (CTR), which is the percentage of clicked items considering all items that have been shown to the users. Thus, the more items users interact with, the higher the CTR and the more successful is the algorithm.

	Group 1	**Group** 2
A/A test	20.000%	19.986%
A/B test	20.000%	21.600%

Table 4: Results of the A/A and A/B test with masked CTR to comply with the platform's policy.

Table 4 shows the results for the *A/A* and *A/B* test. To keep the true numbers proprietary to the company, we masked the absolute CTR values by normalizing group 1's real CTR to 20% and changing the clicks and group 2's CTR accordingly to preserve the ratios without showing confidential numbers. The *A/A* test shows negligible difference between the splits (-0.07%), showing no bias between the two groups. The experimental group 2 has a very noticeable relative difference of +8.00% more clicks per received recommendations using the title weighted description model.

To exemplify the difference between both systems, we show in Table 5 the top recommendations for a postdoctoral researcher who showed interest in three *Deep Learning Engineer* positions. Most of the recommendations of the existing system are software engineer associated job postings, while the new system suggests research oriented job postings with topics similar to the user's previous interactions like *data science*.

In contrast to offline evaluations, deploying models in productive pipelines must adhere to certain metrics, like request response time. As the recommender ranks over 300 jobs against multiple interactions per request, it shows a +9.90% increase in average response time compared to the existing indexed *Elasticsearch* model. While the new system's response time lies within our acceptable ranges, it could be improved by reducing the model's vector dimensionality at the cost of its performance.

	Existing System	**New System**
1	IT project leader	Deep Learning in Autonomous cars
2	Software Engineer (Automotive)	Data Scientist
3	Senior Software Engineer (smart cars)	PhD researcher in Medical Imaging
4	Senior IT Consultant	Computer Linguist/ Analytics
5	Java Software Engineer	Researcher in single-cell Bioinformatics

Table 5: Ranked output from the existing and new system for a user with interest in *machine learning*.

6 Conclusion and Future Work

In this paper, we have introduced a new method for automatically creating datasets for the offline evaluation of job posting similarities. Using such a silver standard dataset, we have evaluated the performance of different dense vector representations of documents in order to identify the most promising setup. Building dense representations based on full-text job descriptions yields the best results. However, computing representations for novel job postings becomes computational expensive, as the model has to be recomputed, as estimating representations for new documents results in much lower results. Building models from titles, the scores only slightly decrease, however, the computation of new models is much faster. In our experiments, we observe the best performance with a combined model, using the words within the title for weighting words in the description that allows to compute new representations in an online scenario. With this model, we yield a substantial 8% relative increase in CTR over the platform's previous system component.

In future work, we want to extend the weighting scheme by integrating ontology and keyword information in order to improve the similarity search.

References

Omar Abdelwahab and Adel Elmaghraby. 2016. UofL SemEval-2016 task 4: Multi domain word2vec for twitter sentiment classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval at NAACL-HLT*, pages 164–170, San Diego, CA, USA.

Fabian Abel, András A. Benczúr, Daniel Kohlsdorf, Martha Larson, and Róbert Pálovics. 2016. RecSys challenge 2016: Job recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 425–426, Boston, MA, USA.

Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivỳ. 2016. Sentence pair scoring: Towards unified framework for text comprehension. volume Arxiv.

Delphine Bernhard and Iryna Gurevych. 2008. Answering Learners' Questions by Retrieving Question Paraphrases from Social Q&A Sites. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, EANL '08, pages 44–52, Columbus, OH, USA.

Robin Burke. 2007. In *The Adaptive Web*, chapter Hybrid Web Recommender Systems, pages 377–408.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. In *Proceedings of the International Conference on Learning Representations, ICLR*, Toulon, France.

Maryam Fazel-Zarandi and Mark S. Fox. 2009. Semantic Matchmaking for Job Recruitment: An Ontology-Based Hybrid Approach. In *Proceedings of the 3rd International SMR2 2009 Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web, collocated with the 8th International Semantic Web Conference*, pages 3111–3119, Washington DC, USA.

Thomas Hofmann. 2000. Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In *Advances in Neural Information Processing Systems 12*, pages 914–920. Denver, CO, USA.

Rémy Kessler, Nicolas Béchet, Mathieu Roche, Marc El-Bèze, and Juan-Manuel Torres-Moreno. 2008. Automatic profiling system for ranking candidates answers in human resources. In *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, pages 625–634, Monterrey, Mexico.

Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. 2009. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181.

Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):42–49.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, Tahoe City, CA, USA.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, pages 1188–1196, Beijing, China.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 302–308, Baltimore, MD, USA.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96*, pages 298–306, Zurich, Switzerland.

Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105.

Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. 2015. Recommender system application developments. *Decision Support Systems*, 74(C):12–32.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems, NIPS*, pages 3111–3119, Lake Tahoe, NV, USA.

Cataldo Musto, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. 2015. Word embedding techniques for content-based recommender systems: An empirical evaluation. In *Poster Proceedings of the 9th ACM Conference on Recommender Systems, RecSys*, Vienna, Austria.

Makbule G. Ozsoy. 2016. From word embeddings to item recommendation. *CoRR*, abs/1601.01356.

Marc Poch, Núria Bel, Sergio Espeja, and Felipe Navio. 2014. Ranking Job Offers for Candidates: Learning Hidden Knowledge from Big Data. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation LREC'14*, pages 2076 – 2082, Reykjavik, Iceland.

Paul Resnick and Hal R. Varian. 1997. Recommender systems. *Communications*, 40(3):56–58.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, Tampere, Finland.

Scharolta Katharina Siencnik. 2015. Adapting word2vec to named entity recognition. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA*, pages 239–243, Vilnius, Lithuania.

Anders Søgaard, Yoav Goldberg, and Omer Levy. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL, pages 765–774, Valencia, Spain.

Yao Yao, Xia Li, Xiaoping Liu, Penghua Liu, Zhaotang Liang, Jinbao Zhang, and Ke Mai. 2017. Sensing spatial distribution of urban land use by integrating points-of-interest and Google word2vec model. *International Journal of Geographical Information Science*, 31(4):825–848.