Effectiveness of job title based embeddings on résumé to job ad recommendation

Mesut Kaya
mkaya@hum.aau.dk
Aalborg University Copenhagen
Copenhagen, Denmark

Toine Bogers
toine@hum.aau.dk
Aalborg University Copenhagen
Copenhagen, Denmark

ABSTRACT

For the task of job recommendations, common practice is to recommend job postings to job seekers, and recently different embedding techniques have been applied to solve this task. A common way to represent job seekers and job postings as embeddings is to use the whole textual data of the job postings and job seekers' resumes. Instead of using the whole textual data, textual data source like *job title, knowledge, skills, abilities* can be used as well. In this work, we present findings of a preliminary offline study, where we explore the impact of utilizing different types of embeddings to recommend job seekers to given job postings, unlike the common practice of recommending job postings to job seekers. We explore the effectiveness of using job title based embeddings compared with the embeddings based on resume and job posting full-text descriptions. Using a dataset from JobIndex —Scandinavia's largest job portals and recruitment agencies— our experimental results show that representing job seekers and job postings as embeddings by using job title text only can be at least as informative as using the full-text descriptions for most of the cases.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Personalization**.

KEYWORDS

job recommendation, job matching, recruitment, embeddings, e-recruitment, HR

1 INTRODUCTION

The popularity of online job portals has broadened the potential reach of both job seekers and companies with job vacancies by enabling them to more easily make their resumes and job postings available to each other in electronic form [3, 9, 16]. However, assessing a larger number of available positions and candidate resumes also places a greater burden on job seekers and recruiters [16]. Many consider AI technology to have the potential to reduce this manual burden, for instance, through the use of *job recommender* systems, which attempt to automatically match job seekers with companies with job vacancies [9, 16]. Recommending relevant jobs to a job seekers or suggesting a list of relevant candidates for a job vacancy is an important task of online job portals. Unlike item-to-people recommendations, job recommendation is a *reciprocal* (or people-to-people) recommendation scenario [17], in which both employer and job seeker must be satisfied with the recommendations.

RecSys in HR 2021, October 1, 2021, Amsterdam, the Netherlands

Much of the recent research on job recommender systems has focused on generating representations of both the job seekers' resumes and and the job postings [2, 5, 11, 13, 23]. These *embeddings* are then compared to each other to estimate how well a job seeker matches a given job posting and vice versa. A common way of representing job seekers and job postings as embeddings is by using the whole textual content of resumes and job postings to train word [15] or document embeddings [12]—semantic representations of each individual word or the document as a whole. Embedding words or documents in the same semantic space is necessary, because simple keyword matching between the entire text of job postings and resumes can be problematic. Both are written by different parties with different backgrounds, which can lead to a vocabulary gap [6], when the terminology used by job seekers to describe themselves differs considerably from the text in the job postings.

Embedddings can capture the semantic similarity between words, phrases, sentences and potentially documents, and to a certain extent, bridge the vocabulary gap between resumes and job postings. Job recommendation approaches can vary in the data that they use to represent both resumes and job postings in a shared latent semantic space. While some researchers generate job recommendations for job seekers using the knowledge, skills and abilities extracted from resumes and job postings [13, 19], others have computed resume embeddings using their historical interactions with job postings. First, job postings are represented as embeddings, after which job seekers' embeddings are based on the embeddings of job postings that they have previously interacted with [5, 10, 11].

One of the most common data fields present in both resumes and job postings is information about past job titles and current job titles targeted in the job postings. According to Bernard et al. [2], recruiters match job seekers with job postings by considering the past work experience and desired future opportunities of job seekers. With some experience, recruiters can carry out this task by only considering job titles without the need for going through the entire textual description. Zhang et al. [23] also states that proper job title information can potentially provide guidance for both job seekers and recruiters.

In this work, we compare embeddings based solely on job titles to embeddings of the entire textual description of resumes and job postings. While the scenario of recommending relevant jobs to job seekers is more common in the literature, in this paper we focus on the inverse: recommending relevant candidates for a job, in effect supporting recruitment professionals in their work [7, 20]. Our work is part of a project to improve the matching algorithms used by one of Scandinavia's largest job portals, JobIndex[1] located in Denmark. Our data is in both Danish and English and originates

[1] https://www.jobindex.dk

with this company. In addition to exploring the value of job title embeddings, we also explore the effect of using a location filter, since job seekers tend to prefer jobs that are within reasonable commuting distance from their home, just as companies prefer that their future employees live within reasonable distance from the company [13]. Finally, we also compare the performance of using pre-trained embedding models (e.g., word2vec model trained on wiki data) with models trained on the Danish/English resumes and job data.

We find that using job titles only to represent job seekers and job postings as embeddings mostly results in better recommendations compared with embeddings based on the entire texts. For word2vec models, experimental results show that training *job domain-specific* embedding models improved the performances of recommendation models that use pre-trained word2vec models. We also find that using a location filter always improves the recommendation models' performance.

The remainder of the paper is structured as follows. In Section 2, we review relevant research on using embeddings for the job recommendation. In Section 3, we present our approach to the problem. We describe our methodology, dataset and evaluation metrics used for the experiments in Section 4. Experimental results are presented and discussed in Section 5. Finally, we conclude in Section 6 with a discussion of our findings and ideas for future work.

2 RELATED WORK

Most of the early work on job recommendation has focused on content-based filtering, collaborative filtering or hybrid combinations thereof [1]. More recent research in the area of job recommendations involves representing job seekers and job postings as embeddings in a latent space [2, 5, 11, 13, 23]. It is common to represent both job seekers' resumes and companies' job postings on a shared latent space by extracting shared skill sets, competencies, occupation data [13, 19]. Alternatively, job seekers' embeddings are computed from their historical interaction data with job postings where job postings are first represented as embeddings, after which embeddings of job seekers are generated by using the embeddings of job postings that they have previously interacted with [5, 10, 11].

Unlike the common practice to recommend job postings to job seekers, in this paper we focus on the problem setting of recommending candidate job seekers for job postings. This problem is also referred as *talent search* [7, 20]. Ramanath et al. [20] present their findings about using representation learning for talent search at LinkedIn. Unlike our approach, they do not use textual data to learn the embeddings but they represent categorical entities (e.g., skill, title, company) as embeddings by using a variety of graph embedding algorithms.

As we motivated in the previous section, *job titles* make up a data type available in both job seekers' resumes and job postings. Job title information has been used as one of the important features for the task of matching resumes and job posting [14, 18]. Our focus in this paper is to examine the benefits of embedding resumes and job postings using job titles alone. To the best of our knowledge, the only work that uses embeddings of only job titles is the work by Elsafty et al. [5]. Unlike our approach, they recommend job postings to job seekers and they only create the representations of the job postings using the embedding of (1) job titles, (2) full-text only and (3) job titles and full-text combined. These three embeddings result in recommendations for similar jobs. Our recommendation scenario is inverted, in that we attempt to support recruitment professionals by recommending job seekers in relation to a specific job posting. In addition, we generate job title-based embeddings of both resumes and job postings and recommend relevant candidates based on the similarity of their resume embeddings to the job posting embedding.

Liu et al. uses job title information together with skills and location information. They parse resumes of job seekers and job postings to extract job titles, skills and location information. Then, they build job–job, skill–skill, and job–skill graphs from historical data to learn a joint representation for both job titles and skills by encoding the local neighborhood structures captured by the three graphs [13].

3 OUR APPROACH

Previous work has shown that learning embeddings on textual description of job postings can improve the accuracy of content-based recommendations [10, 11]. We also try to use different textual data from the resumes of job seekers and job postings (see Section 4.4) to create their embeddings and create recommenders that use those embeddings.

We denote the set of users as $\mathcal{U}$ and set of items as $\mathcal{I}$. In our problem setting, we consider recommending job seekers to given job postings, we define a job posting as a user $u \in \mathcal{U}$ and a job seeker as an item $i \in \mathcal{I}$. Given a textual data source that consists of a sequence of tokens $d = \{t_1, t_2, \ldots, t_m\}$, by using a pre-trained embedding *model* (we explain different type of embedding models used later in Section 4.3), we represent d as a k-dimensional vector $\vec{d}$. For simplicity, we refer to user vectors as $\vec{d_u}$, and item vectors as $\vec{d_i}$. We also represent the interaction history of a user u as $h_u = \{\vec{d_{i1}}, \vec{d_{i2}}, \ldots, \vec{d_{in}}\}$ by considering the vector representations of all items that the user has positively interacted with, and similarly we represent the interaction history of an item i as $h_i = \{\vec{d_{u1}}, \vec{d_{u2}}, \ldots, \vec{d_{ul}}\}$ by considering the vector representations of all users that has positively interacted with the item i. The recommendation task is then—given a 'reference' vector $\vec{d_u}$ that is the representation of a user u (i.e., job postings)—to find the top-N similar vectors among item vectors (i.e., job seekers).

3.1 Recommendation Models

To obtain the final vector representation of users $\vec{d_u}$ (job postings) and items $\vec{d_i}$ (job seekers) and to perform the recommendation task, we define the following models:

- *EMB*: Job seekers' representations are based on textual data from their resumes while job postings' representations are based on the textual data from the job postings. In other words, given a user u with its textual data $d_u = \{t_1, t_2, \ldots, t_m\}$, $\vec{d_u}$ is computed by using d_u and any given embedding model. Similarly, for a given item i its vector representation is computed by using d_i and any given embedding model. This variant can be used for cold-start scenario since it does not require any interaction data between job seekers and job postings.

- *EMBUH*: This variant calculates embeddings based on the user history (EMBUH). Given an item i and its textual data d_i, $\vec{d}_i$ is computed using d_i and any given embedding model. After computing all item embeddings in $\mathcal{I}$, for a given user u, its representation is the average of the item embeddings in h_u (i.e., a job posting's embedding is the average[2] of all the resumes' embedding vectors that had a positive interaction with the job ad).
- *EMBUHL*: This variant is similar to EMBUH, but is limited to using only the latest user history (EMBUHL). The only difference is, instead of considering all items in h_u, we only consider the embedding of the item that has the latest interaction with the user u. This variant is similar to the previous work that considers the *recency* of the interactions [10, 11].
- *EMBIH*: This variant calculates embeddings based on the item history (EMBIH). Given a user u and its textual data d_u, $\vec{d}_u$ is computed using d_u and any given embedding model. After computing all user embeddings in $\mathcal{U}$, for a given item i, its representation is the average of the user embeddings in h_i (i.e., a job seeker's embedding is the average of all the job postings' embedding vectors that it had a postive interaction with).
- *EMBIHL*: This variant is similar to EMBIH, but uses only the latest user history (EMBIHL). The difference is, we only consider the embeddding of the user that has the most recent interaction with the item.

3.2 Location filter

As we will explain in more detail in Section 4.1, both job postings and resumes come with a set of geo-location identifiers. We denote the set of location IDs that a job seeker i is interested in as l_i. Similarly, we denote the set of location IDs that is provided in a job posting u as l_u. To apply location filter to any of the recommendation models explained above, we simply take the intersection of l_i and l_u. Then, for a given user u, we filter out all candidate items where the resume and job posting location IDs are disjoint sets, i.e., where the intersection is empty.

4 METHODOLOGY

In this section, we explain the details of our offline evaluation methodology, by describing our dataset, the different embedding models we tested in our experiments and the textual data used to created the embeddings, as well as the baseline recommender algorithms we compared them against.

4.1 Dataset

We extract our data from the data dump provided by JobIndex. The dataset contains both job postings and anonymized resumes in a semi-structured textual format. While most resumes and job postings are in Danish (89.45% of resumes and 81.95% of job postings), a small portion are in English (10.55% of resumes and 18.05% of job postings).

[2]A comparison with other methods for combining item embeddings into a single user embedding vector is left as future work.

4.1.1 Metadata. Each job posting contains a textual description of the job in question, an appetizer text that summarizes the job posting, the job title of the job posting and a list of of location IDs corresponding to the location of the company offering the position. Each resume contains a resume text in which job seekers describe themselves, a list of job titles that they are interested in or have previous work experience in, and a list of location IDs corresponding to the geographic area(s) they prefer to work in.

4.1.2 Interaction Data. The dataset also contains data on interactions between job seekers and job postings. When recruiters associated with the job portal identify job seekers as relevant candidates for a job, they can contact them by sending them a message containing a short introduction, argumentation for why they should apply for the job, and an embedding version of the job posting. In addition, each contact email has a link that makes it easy for job seekers to provide feedback on the recommendation. Job seekers can respond to the recruiters either positively or negatively. Our dataset contains 268,442 resumes and 426,226 job postings. On average, 18.4 job seekers are contacted per job posting. Out of those, 10.9 job seekers do not respond, 5.1 of them respond negatively, and only 2.4 of them respond positively. In this work, we consider positive interactions only and we filter out the negative responses[3]. In addition, we filter out all jobs with less than 5 positive responses to make sure that each user has at least one item in validation and test set (see the following section for more details). Our final dataset contains 98,172 resumes and 45,173 job postings. It contains a total of 303,758 positive interactions with an average of 6.7 user actions (users are job postings) and 3.1 item actions (items are job seekers). The sparsity of the dataset is 99.99%.

4.1.3 Dataset Splits. After pre-processing the dataset, we partition the interactions into training, validation and test sets such that 60% of each user's interactions are in the training set, 20% of them are in the validation set and 20% are in the test set. We use the timestamp information to keep the most recent interactions in the test set. Note that, since we use positive interactions only, all items in the test set are considered as relevant items in the experiments.

4.2 Baselines

As our two baseline algorithms— following [10]— we use item-based KNN (**ItemKNN**) [22] together with a popularity-based recommender (**Pop**) algorithms on the interaction data. We also tried BPR [21] and NeuMF [8], but optimizing their hyperparameters was not feasible due to time constraints.

We use the Python library RecBole[4] to run our experiments. It has a hyper-parameter optimization module as well. For ItemKNN, we optimize the number of nearest-neigbors parameter m using the validation set. The value that optimized the MRR metric (See Section 4.5) on the validation sets was $m = 500$.

[3]We leave the exploration of negative response data as future work, because there are three different types of negative responses: (1) a job seeker is not interested in the job, (2) they are not interested in working for that specific company, or (3) they have found a job elsewhere. The latter is not necessarily a negative response as the job seeker may still find the job relevant for them.

[4]https://github.com/RUCAIBox/RecBole

4.3 Embedding Models

We use publicly available pre-trained models as well as train our own embedding models on the textual data of the resumes and job postings in our dataset.

4.3.1 Pre-trained word2vec embeddings. We use a 100-dimensional word2vec skipgram model [15] trained on the Danish CoNLL17 corpus[5].

4.3.2 Self-trained domain-specific word2vec embeddings. We also train our own word2vec model using the Gensim[6] library on the resume text from the resumes and appetizer text from the job postings. Similar to the CoNLL17-based word2vec model, we set the embedding size to 100 and window size to 10.

4.3.3 Self-trained domain-specific doc2vec embeddings. Similar to our self-trained domain-specific word2vec embeddings trained on our dataset, we also train a doc2vec model on this data. Again, we set the embedding size to 100 and we use Gensim's implementation of doc2vec. For word2vec-based models, a document d's embedding $\vec{d}$ is computed by taking the average of the each token's $t \in d$ word embedding. For doc2vec, d is computed by taking it as a document and the model returns a single document embedding vector $\vec{d}$.

4.4 Textual data used for embeddings

Now, we will explain different ways (in terms of the textual data used) to represent users and items as embedding vectors.

4.4.1 Embeddings of job titles. Resumes contain a field called **job titles** that is a list of job titles the user aspires to or has held previously. For job postings as well, there is the **jobtitle** field that contains the job title of the position in question. Most of these job titles are in Danish, but some of them are in English. We use a Python language detection package[7] to detect the language of the text, after which we translate the English text to Danish where necessary using another Python package[8]. Then, using any of the models explained in the previous section, job title textual data of the resumes and job postings are represented as k-dimensional embedding vectors.

4.4.2 Embeddings of resume text and appetizer text. Resumes contain a field called **resume text** where job seekers write a description text about themselves. For the job postings as well there is an **appetizer** text that summarizes the job description. Similar to the job titles, we detect English text and translate it to Danish where necessary. Both job seekers (based on their resumes) and job postings are then represented as embedding vectors similar to how this was done for job titles.

4.5 Evaluation Metrics

We use a set of metrics commonly used for evaluating recommender systems to compare our different algorithms. We use their Recbole implementation[9].

[5] http://vectors.nlpl.eu/repository/
[6] https://pypi.org/project/gensim/
[7] https://pypi.org/project/langdetect/
[8] https://pypi.org/project/googletrans/
[9] For the formulation of the metrics considered in Recbole, please refer to https://recbole.io/docs/recbole/recbole.evaluator.metrics.html

For a user u:

- **Recall** measures the proportion of u's relevant test set items that are in the top-N results.
- **MRR** measures whether u finds an item that is in u's relevant test set in the earlier ranks of the top-N results.
- **Normalized Discounted Cumulative Gain (nDCG)** measures the extent to which u finds u's relevant test set items in the earlier ranks of the top-N results. Like MRR, nDCG is sensitive to the rank of items. Unlike MRR, it takes into account all of the items in the top-N results that are in u's relevant test set items.

The results reported in the following section are the macro-averaged means of each metric, i.e., we first compute them for each user $u \in \mathcal{U}$, after which we take the mean over all users. Given the difficulty of the task and the sparsity of the data, we report all results at the top-100 cut-off in the following sections.

5 RESULTS

In this section, we report our empirical findings of extensive experiments. Table 1 shows results for experiments that use location filter. We do not show the results for the experiments without applying location filter since for all cases applying location filter always improves the performance of the recommendation models, as described in more detail in Section 5.4. The top section of Table 1 shows the results for **Pop** and **ItemKNN** baselines, of which the latter outperforms the former. The middle section of Table 1 shows the results of embeddings of job titles only and the bottom section shows the results for the embeddings based on the full-text representations. Vertically, Table 1 shows the results of three different types of embedding models—pre-trained vs. self-trained domain-specific word2vec embeddings and self-trained domain-specific doc2vec embeddings.

Overall, we find that the **ItemKNN** baseline almost always outperforms the embedding-based recommendation models. These results are in line with previous work by Lacic et al. [10], who also used doc2vec-based embeddings. They found that a user-based KNN algorithm outperformed a doc2vec approaches in terms of accuracy. However, collaborative filtering algorithms perform worse in cold-start scenarios without any interaction data, so our experimental setup and data filtering protocol most likely benefits the **ItemKNN** baseline. In a real-world scenario, recruitment professionals are tasked with finding relevant candidates for a new job posting, which is a classic example of a cold-start scenario. We therefore believe that exploring and comparing the different embedding-based models is still valuable.

5.1 Impact of job title-based recommendation

Except for EMBIH, embeddings computed based on job titles only outperforms embeddings computed based on the full-text representations. This supports our claim that representing both job seekers and job postings using only job titles can be valuable to a certain extent.

5.2 Pretrained vs. domain-specific embeddings

For word2vec based models, we observe that, as expected, training our own word2vec embeddings on the textual data of job postings

Table 1: Results for recommending resumes to jobs using different type of embeddings by applying location filter. Pop and ItemKNN baseline results are shown at the top of the table. The results are repeated for Pop and ItemKNN for pretrained word2vec, word2vec on job data and doc2vec on job data multi columns. The highest value for each metric is highlighted in bold for each block. The second highest value for each metric is highlighted in italic for each block.

Algorithm	Recall	MRR	nDCG	Recall	MRR	nDCG	Recall	MRR	nDCG
Baseline algorithms									
Pop	0.0212	0.002	0.0052	0.0212	0.002	0.0052	0.0212	0.002	0.0052
ItemKNN	**0.1608**	**0.0256**	**0.0495**	*0.1608*	**0.0256**	**0.0495**	**0.1608**	**0.0256**	**0.0495**
Embeddings by using job title data only									
	pretrained word2vec			word2vec on job data			doc2vec on job data		
EMB	0.0909	0.0121	0.0256	0.1161	0.01	0.0284	0.0315	0.0033	0.0082
EMBUH	*0.1496*	*0.0178*	*0.0406*	**0.1638**	*0.0151*	*0.0409*	0.0654	0.0059	0.0161
EMBIH	0.0981	0.0108	0.0257	0.1137	0.0084	0.0271	0.0473	0.0048	0.0121
EMBUHL	0.1017	0.0136	0.0286	0.1071	0.0097	0.0265	0.05	0.0047	0.0125
EMBIHL	0.0468	0.0111	0.017	0.0589	0.0063	0.0155	0.0212	0.0025	0.0058
Embeddings by using job seeker and job posting textual data									
	pretrained word2vec			word2vec on job data			doc2vec on job data		
EMB	0.0331	0.0032	0.0084	0.0596	0.0053	0.0146	0.0367	0.0036	0.0093
EMBUH	0.0441	0.004	0.011	0.0723	0.0069	0.0181	0.0151	0.0014	0.0038
EMBIH	0.1208	0.0129	0.0318	0.1322	0.0135	0.0341	*0.157*	*0.0173*	*0.042*
EMBUHL	0.0304	0.0028	0.0075	0.0439	0.0037	0.0106	0.0156	0.0015	0.0039
EMBIHL	0.0412	0.0061	0.012	0.0608	0.0067	0.0159	0.0437	0.0048	0.0115

and job seekers' resume data improves recommendation performance over generic pre-trained embeddings. The EMBIH model is again the only exception to this, where the pre-trained word2vec model slightly outperforms using trained word2vec model on the job data.

5.3 Word- vs. document-level embeddings

For the embeddings based on job titles only, using word2vec models to represent job seekers and job postings as embeddings for all cases performs better than using doc2vec models. Compared with documents containing a lot of tokens (e.g., job posting description), job titles tend to contain only a few tokens and can even consist of a single token, such as "economist". That is why doc2vec may not be suitable for job title-based embeddings compared with word2vec.

For the embeddings based on full-text data, for only EMBIH using doc2vec models performs better than using word2vec models. For all EMBIHL, EMBUH and EMBUHL using word2vec models performs better than using doc2vec models. Compared with documents containing only a few tokens (e.g., job titles), job posting or job seeker description contain more tokens. One would expect to have better results for using doc2vec models. However, for some cases using word2vec models on textual data containing more tokens can perform better than doc2vec models. We plan to investigate this behaviour further in future work.

5.4 Impact of location filter

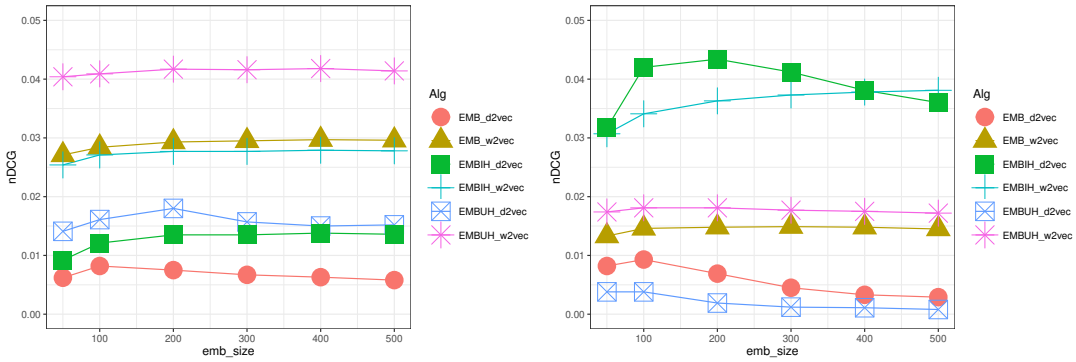
Although the results are not shown for the experiments without applying the location filter, for all configurations and recommendation models, applying location filter always increases the values of all evaluation metrics (for most of the cases at least 50% improvement over without applying the location filter) . These results are in line with the claim that companies are more likely to prefer job seekers that are within reasonable commute distance and similarly job seekers are more likely to prefer jobs that are posted by companies that are within reasonable commute distance [13].

5.5 Impact of recency

For all cases, EMBUHL and EMBIHL always perform worse than EMBUH and EMBIH. In other words, using the *recency* of the interactions is performing worse than considering all historical interactions. These results are different than the previous work that considers the effect of recency of the interactions [10, 11] (similar to us, they also consider the latest interaction to model the recency). We note that, unlike our problem setting of recommending job seekers to job postings, they recommend job postings to job seekers.

5.6 Impact of embedding size

To explore the effect of embedding dimension, we run experiments by varying the embedding size of the word2vec and doc2vec models from the set $\{50, 100, 200, 300, 400, 500\}$. We report the performance of different recommenders using nDCG in the Fig. 1. Fig. 1a shows the results for job title based embeddings, and Fig. 1b shows the results for full-text based embeddings. For each of the recommendation algorithms EMB, EMBUH and EMBIH, we show the results for both w2vec and doc2vec based embeddings. Mostly, for both cases all recommenders reach their peak point for the embedding sizes of 100 or 200.



(a) Job title based embeddings **(b) Full-text based embeddings**

Figure 1: nDCG values for varying embedding sizes for both word2vec and doc2vec models.

Table 2: Top-1 job seeker recommendations generated for 2 different job postings for job title-based embeddings using word-level embedding models. Recommendations are based on embeddings using job title data only. We note that, all Danish job titles are translated to English and presented here.

IT Support Analyst			Head of HR and Personnel		
EMB	EMBUH	EMBIH	EMB	EMBUH	EMBIH
IT Architect	IT Associate	Bank employee	Consultant	HR Director	HR Business Partner
IT Administrator	IT Consultant	IT employee	Staff and HR	HR Business Partner	HR Chief
IT system consultant	IT Supporter	Finance employee	Personnel assistant	HR Chief	HR Consultant
IT Supporter			Staff member	HR Manager	HR Manager
			consulting assistant	Head of Administration	Leader
				Personnel Manager	Personnel Manager
					Personnel Development Manager

5.7 Example recommendations

Table 2 shows the top recommended job seeker for two different job postings. The top part of the Table 2 shows the job title of the two job postings. The bottom part of the table shows the job titles of the top recommended job seeker (a list of job titles that the job seeker is interested in or have previous work experience in) for 3 different recommendation models, i.e., EMB, EMBUH and EMBIH. For both examples, we can see the effect of using embeddings in terms of capturing the semantic similarity between job titles, e.g., "Head of HR" and "HR Director".

One point of interest are the recommendations generated by the EMBIH algorithm for the job posting describing an "IT Support Analyst" position. Here, we see that the list of recommended job titles include "Bank employee", "IT employee" and "Finance employee". As we explained in Section 3.1, EMBIH calculates the embedding of job seeker with job title "IT Support Analyst" based on the item history, i.e., job postings that the job seeker had a positive interaction with. When we analyze the job seeker's historical interactions, we can see that the job seeker was mostly interested in "IT" positions of some banks.

6 DISCUSSION & CONCLUSIONS

In this work, we explored the impact of embeddings for tackling the problem of recommending job seekers to job postings. We did extensive offline evaluation to compare different recommender models by using embeddings based on different type of textual data source. Specifically, we focused on the effect of using job titles only in comparison with using full-text descriptions of job postings and resumes of job seekers. We find that generating embeddings by using only job titles typically results in better recommendations with respect to different evaluation metrics, as compared with embeddings based on the full-text representations of job seekers and job postings. Experimental results also show that using location filter always improves the recommendations.

We note that this is a preliminary study to explore the effect of job title based embeddings for the task of recommending relevant candidates for an open job posting. There are some limitations of our current work:

- **Job titles vs. industry.** Although job title is a valuable data source, we note that we are aware of some of the limitations related to it. First, job postings or job seekers' resumes can have meaningless titles or no title at all [2]. Second, job title data can be messy data since it can contain a lot of subjective and non-standard naming conventions [23]. Third, using job title for recommending job seekers to job postings can be problematic for some specific cases. Consider a job title "Project Manager" for two different job postings, and let's assume that the *industry* information is different for the two, e.g., one job posting is for IT and the other for Finance. Without using the industry information, embeddings based

on job title only will have the same matching degree for a job seeker with job title "Project Manager" from any industry to this specific job postings. We are planning to use job title based embeddings together with embeddings based on other textual data source like *industry* from both job postings and job seekers' resumes in the future.

- **Multilinguality.** As we explain in Section 4.4, we translate English text to Danish where necessary. If the quality of the translation is poor, embeddings may not be representative for the translated text for the semantic similarity. We leave the effect of using multilingual embeddings as future work.
- **Alternative embedding models.** In this work, we investigated word2vec and doc2vec models to represent job seekers and job postings as embeddings. We leave the effect of using BERT-based embeddings [4] as future work.

ACKNOWLEDGEMENTS

This research was supported by the Innovation Fund Denmark, grant no. 0175-000005B.

REFERENCES

[1] Shaha T Al-Otaibi and Mourad Ykhlef. 2012. A survey of job recommender systems. *International Journal of Physical Sciences* 7, 29 (2012), 5127–5142.

[2] Timothé Bernard, Tanguy Moreau, Clément Viricel, Paul Mougel, Christophe Gravier, and Frédérique Laforest. 2020. Learning Joint Job Embeddings Using a Job-Oriented Asymmetrical Pairing System. In *ECAI 2020*. IOS Press, 1970–1977.

[3] Tanya Bondarouk and Chris Brewster. 2016. Conceptualising the Future of HRM and Technology Research. *The International Journal of Human Resource Management* 27, 21 (2016), 2652–2671.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[5] Ahmed Elsafty, Martin Riedl, and Chris Biemann. 2018. Document-based recommender system for job postings using dense representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*. 216–224.

[6] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The Vocabulary Problem in Human-System Communication. *Communications of the ACM* 30, 11 (1987), 964–971.

[7] Sahin Cem Geyik, Qi Guo, Bo Hu, Cagri Ozcaglar, Ketan Thakkar, Xianren Wu, and Krishnaram Kenthapadi. 2018. Talent search and recommendation systems at LinkedIn: Practical challenges and lessons learned. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1353–1354.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[9] Ugur Karaboga and Pelin Vardarlier. 2020. Examining the Use of Artificial Intelligence in Recruitment Processes. *Bussecon Review of Social Sciences (2687-2285)* 2, 44 (Dec 2020), 1–17. https://doi.org/10.36096/brss.v2i4.234

[10] Emanuel Lacic, Dominik Kowald, Markus Reiter-Haas, Valentin Slawicek, and Elisabeth Lex. 2017. Beyond accuracy optimization: On the value of item embeddings for student job recommendations. *arXiv preprint arXiv:1711.07762* (2017).

[11] Emanuel Lacic, Markus Reiter-Haas, Tomislav Duricic, Valentin Slawicek, and Elisabeth Lex. 2019. Should we embed? A study on the online performance of utilizing embeddings for real-time job recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 496–500.

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

[13] Mengshu Liu, Jingya Wang, Kareem Abdelfatah, and Mohammed Korayem. 2019. Tripartite vector representations for better job recommendation. *arXiv preprint arXiv:1907.12379* (2019).

[14] Malte Ludewig, Michael Jugovac, and Dietmar Jannach. 2017. A Light-Weight Approach to Recipient Determination When Recommending New Items. In *Proceedings of the Recommender Systems Challenge 2017*. 1–6.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).

[16] Paolo Montuschi, Valentina Gatteschi, Fabrizio Lamberti, Andrea Sanna, and Claudio Demartini. 2014. Job Recruitment and Job Seeking Processes: How Technology Can Help. *IT Professional* 16, 5 (Sep 2014), 41–49. https://doi.org/10.1109/MITP.2013.62

[17] Iván Palomares, Carlos Porcel, Luiz Pizzato, Ido Guy, and Enrique Herrera-Viedma. 2021. Reciprocal Recommender Systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation. *Information Fusion* 69 (2021), 103–127.

[18] Ioannis Paparrizos, B Barla Cambazoglu, and Aristides Gionis. 2011. Machine learned job recommendation. In *Proceedings of the fifth ACM Conference on Recommender Systems*. 325–328.

[19] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 25–34.

[20] Rohan Ramanath, Hakan Inan, Gungor Polatkan, Bo Hu, Qi Guo, Cagri Ozcaglar, Xianren Wu, Krishnaram Kenthapadi, and Sahin Cem Geyik. 2018. Towards deep and representation learning for talent search at LinkedIn. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2253–2261.

[21] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[22] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[23] Denghui Zhang, Junming Liu, Hengshu Zhu, Yanchi Liu, Lichen Wang, Pengyang Wang, and Hui Xiong. 2019. Job2Vec: Job title benchmarking with collective multi-view representation learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2763–2771.