A Neural Information Retrieval Approach for Résumé Searching in a Recruitment Agency

Brandon Grech and David Suda[a]
Department of Statistics and Operations Research, University of Malta, Msida MSD2080, Malta

Keywords: *N*-grams, Skip-grams, Word Embeddings, Document Vectors, Neural Language Model, Neural Information Retrieval.

Abstract: Finding résumés that match a job description can be a daunting task for a recruitment agency, due to the fact that these agencies are dealing with hundreds of job descriptions and tens of thousands of résumés simultaneously. In this paper we explain a search method devised for a recruitment agency by measuring similarity between résumé documents and job description documents. Document vectors are obtained via TF-IDF weights from word embeddings arising from a neural language model with a skip-gram loss function. We show that, with this approach, successful searches can be achieved, and that the number of skips assumed in the skip gram loss function determines how successful it can be for different job descriptions.

1 INTRODUCTION

Many tasks of natural language processing aim to derive a model which can understand and make sense of unstructured data, such as text, in some context. A better structure of the data can be defined if one is able to represent similarities and dissimilarities between words, phrases, paragraphs and documents being studied. This paper aims to use information retrieval methods to find similarities between job descriptions and résumé, to facilitate the process of recruiters to determine the adequate candidates for a particular job. Sifting through tens of thousands of résumés is an impractical task and, so far, people we were in contact within the local industry made use of keywords to facilitate this task. One recruitment company is now making use of the devised search method which is explained in this paper.

Measuring the similarity of document vectors has been applied in several applications. Our aim is to apply this approach as an additional tool for finding job résumés relevant to a job application. We have not found many articles that fulfill a related purpose. Cabrera-Diego et al. (Cabrera-Diego et al., 2015) evaluate the performance of various similarity indices on a large set of French job résumés. Inter-résumé proximity is once again studied by Cabrera-Diego et al. (Cabrera-Diego et al., 2019) while using a method called relevance feedback to determine whether job résumés are relevant or irrelvant for a job posting. On the other hand, Schmitt et al. (Schmitt et al., 2017) proposes a retrieval approach called LAJAM (Language Model-based Jam) with the intent of recommending job posts to applicants who have submitted their résumé. This is a very similar but opposite problem to the one we tackle. One can, of course, find other NLP literature related to document searches though not in the recruitment context (see e.g. Wei and Croft 2006, Wang and Blei 2011, Pandiarajan et al. 2014, Naveenkumar et al. 2015).

The structure of this paper is as follows. We shall first introduce the background regarding *N*-grams and skip-grams. *N*-grams consider possible sequences of *N* words within a learning set and skip-grams allow for similar length sequences with skips. We shall not be using them within a word prediction context but within the neural network structure of the word embedding stage, which is described in the third section. Word embeddings are vector representations of words which arise from neural language models, which are single-layer multi-classification neural networks applied to one-hot encoded vectors of each word and subsequent words in a sequence. We shall use the concept of *N*-grams and skip-grams both to generate our learning set and also to define the loss function of the neural network. Since we are interested in assigning document vectors to documents, we follow this with a section on how to obtain document

[a] https://orcid.org/0000-0003-0106-7947

Grech, B. and Suda, D.
A Neural Information Retrieval Approach for Résumé Searching in a Recruitment Agency.
DOI: 10.5220/0009355006450651
In *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2020)*, pages 645-651
ISBN: 978-989-758-397-1; ISSN: 2184-4313
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

vectors from word embeddings, by applying weighted averages on the vector representations of the words in the document using the term frequency inverse document frequency (TF-IDF). Cosine similarity will be then used to determine documents similarity. Finally, in the last section, we shall analyse the properties of a learning set consisting of job descriptions and résumés, and conduct information retrieval on a selection of job descriptions to assess the success rate of retrieving résumés which are relevant to the search.

2 N-GRAMS AND SKIP-GRAMS

By definition, an N-gram is any sequence of N words. N-gram models are probabilistic models used for estimating the probability function of words given the previous $N-1$ words. This is formally defined as follows. Consider a sequence of words $\{w_1, w_2, ...\}$. Let $\mathbf{w}_m^n = (w_m, ..., w_n)$. Then the N-**gram approximation** to the conditional probability of the next word in a given sequence of words is given by

$$P\left(w_n|\mathbf{w}_1^{n-1}\right) \approx P\left(w_n|\mathbf{w}_{n-N+1}^{n-1}\right) \quad (1)$$

When it comes to practical implementation, however, N-grams are very much subject to data sparsity, as many N-gram word combinations do not exist. One way to tackle the data sparsity problem mentioned in literature is to use skip-grams. Skip-grams are very similar to N-grams but they allow words to be skipped. By definition, K-skip N-grams for a sequence of words $w_1, w_2, \ldots$ are defined by the set

$$\left\{ w_{i_1}, w_{i_2}, \ldots, w_{i_n} | \sum_{j-1}^{N} (i_j - i_{j-1} - 1) \leq K \right\} \quad (2)$$

We now illustrate the concept of N-grams and skip-grams on the sentence 'Strong work ethic and commitment.'. The following are the:

- Possible bigrams: {strong work, work ethic, ethic and, and commitment}
- Possible 1-skip bigrams: {strong work, strong ethic, work ethic, work and, ethic and, ethic commitment, and commitment}
- Possible 2-skip bigrams: {strong work, strong ethic, strong and, work ethic, work and, work commitment, ethic and, ethic commitment, and commitment}
- Possible trigrams: {strong work ethic, work ethic and, ethic and commitment}
- Possible 1-skip trigrams: {strong work ethic, strong work and, strong ethic and, work ethic and, work ethic commitment, work and commitment, ethic and commitment}
- Possible 2-skip trigrams: {strong work ethic, strong work and, strong work commitment, strong ethic and, strong ethic commitment, strong and commitment, work ethic and, work ethic commitment, work and commitment, ethic and commitment}

(see Guthrie et al, 2016). It can be seen, in the above illustrations, how the use of skips greatly increases the N-gram learning set. The concept of N-grams and skip-grams is used in the following section within the context of neural language model.

3 WORD EMBEDDINGS

A distributional semantic model (DSM) is a model which assumes that words that occur in the same contexts tend to have similar meanings. One type of DSM are neural language models, where word vectors are modelled as additional parameters of a neural network, which approximates the conditional probability of a word given its history. Neural language models (Bengio et al., 2003) comprise an embedding layer to its distributed representation, an n-dimensional vector which characterises the meaning of the word. Given a vocabulary set V, $n < |V|$ where $|V|$ is the size of the vocabulary set. These come in the form of a single-layer (vanilla) neural network. The general process of a neural language model is as follows:

- Each available word w_i is inputted into the network in the form of a $1 \times n$-dimensional one-hot encoded vector $\mathbf{w}_i$ where the i^{t} dimension is equal to 1 and the other dimensions are set at 0;
- $\mathbf{w}_i$ is multiplied by the $n \times q$ word embedding matrix $\mathbf{W}^{(H)}$ and then transformed using an activation function $\sigma(.)$ to yield the hidden layer $\mathbf{a}_i^{(H)}$;
- $\mathbf{a}_i^{(H)}$ is then multiplied by another $q \times n$ weight matrix $\mathbf{W}^{(O)}$, which is then transformed using an output activation function $\theta(.)$, typically taken to be the softmax function presented in (4). The corresponding output is a vector of probabilities representing the likelihood that each word is likely to be the one that follows.

Note that, generally, only $\mathbf{W}^{(H)}$ is used, and $\mathbf{W}^{(O)}$ is discarded after training. The rows of $\mathbf{W}^{(H)}$ are the word embeddings of the corresponding words w_i - let us call them $\mathbf{v}_i$. These are obtained by $\mathbf{v}_i = \mathbf{w}_i \mathbf{W}^{(H)}$

We use the skip-gram model to train the neural network by minimising the error of predicting a term given one of its contexts. The loss function to optimize for obtaining $\mathbf{W}^{(H)}$ (and also $\mathbf{W}^{(O)}$) is thus defined by the negative loglikelihood as follows

$$L_{Skip-Gram} = -\frac{1}{|V|}\sum_{i=1}^{|V|}\sum_{-c\leq j\leq c, j\neq 0}\log p\left(w_{i+j}|w_i\right) \tag{3}$$

where:

- c refers to the size of the training context around each word;
- $p\left(w_{i+j}|w_i\right)$ is defined by the softmax function:

$$p\left(w_{i+j}|w_i\right) = \frac{\exp\left({\mathbf{v}^*_{i+j}}^T\mathbf{v}_i\right)}{x}\sum_{v=1}^{|V|}\exp\left(\mathbf{v}^{*T}_v\mathbf{v}_i\right) \tag{4}$$

(Mikolov et al. 2013). In (3), taking $c = 1$ refers to the no-skips N-gram architecture. The denominator of (4) sums over the probability of all terms in the vocabulary, however, and due to the large number of terms that are possibly present in the vocabulary, computing this can be exorbitantly costly. We use negative sampling to work around this. We replace every $p\left(w_{i+j}|w_i\right)$ in the skip-gram loss function (3) by

$$\log\sigma\left({\mathbf{v}^*_{i+j}}^T\mathbf{v}_i\right) + \sum_{k=1}^{K}E_{w_k\ P_n(w)}\left[\log\sigma\left(-{\mathbf{v}^*_{w_k}}^T\mathbf{v}_i\right)\right] \tag{5}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ and w_k draws from $P_n(w)$ using logistic regression. k is typically taken to be larger in the case of small training sets (usually from 5 up till 20) and smaller in the case of large training sets (usually from 2 up till 5). We now move on to describing the construction of document vectors, which we shall apply on our résumé and job description documents from the resulting word embeddings.

4 DOCUMENT VECTORS

Semantic compositionality (SC) refers to the issue of representing the meaning of larger texts, such as sentences, phrases, paragraphs and documents built from sequences of words. There are various ways in which documents can be represented in an q-dimensional space. The most crude way would be to use a weighted average of all the word vectors of the words in the document with respect to their number of occurrences. Some papers suggest choosing the weights by basing them on TF-IDF (Le and Mikolov, 2014). TF-IDF aims to find the most important words by decreasing the weight for frequent words which occur throughout all documents and increasing the weight for words which do not occur as regularly accross all documents or groups. We take tf_{ij} to be the number of times a term i has occurred in a document j and and we take idf_i to be the inverse document frequency given by

$$idf_i = \log\frac{m}{m_i},$$

where m refers to the total number of documents and m_i the number of documents in which a term i appears. Then the weight ω_{ij} for a word vector $\mathbf{v}_i$ in a document j is calculated by

$$\omega_{ij} = tf_{ij}idf_i.$$

Let $\mathbf{d}_i = \sum_{j=1}^{|V|}\omega_{ij}\mathbf{v}_j$ be the resulting vector representing document i. The distance between two documents $\mathbf{d}_i$ and $\mathbf{d}_j$ is also known as the similarity between documents. One approach would be to measure it using the cosine similarity

$$sim\left(\mathbf{d}_i,\mathbf{d}_j\right) = \frac{\sum_{k=1}^{N}\omega_{ki}\omega_{kj}}{\sqrt{\sum_{k=1}^{N}\omega_{ki}^2}\sqrt{\sum_{k=1}^{N}\omega_{kj}^2}} \tag{6}$$

Note that identical documents will yield a similarity of 1 in (4). Other similarity measures such as the Euclidean distance, Jaacard measure or Dice measure can also be used. The next section involves the application of the neural language model for information retrieval within the résumé searching context.

5 NEURAL INFORMATION RETRIEVAL MODEL FOR RÉSUMÉ SEARCHING

This study consists of two corpuses. The first corpus is made up of 500 randomly chosen job descriptions, while the second corpus consists of 2000 randomly chosen résumés provided by a recruitment agency. These corpuses were sourced by a recruitment agency. The job descriptions consist of a variety of jobs, varying from finance to information technology to manufacturing. These were more uniform in style as they were written by professional recruiters. For résumés, on the other hand, more variability is expected due to the authors' individuality. In this section ,we aim to determine the similarity of résumés based on the job description playing the role of the user query.

The aim of this model is to take a job description as an input, and return a list of 10 best CVs that are

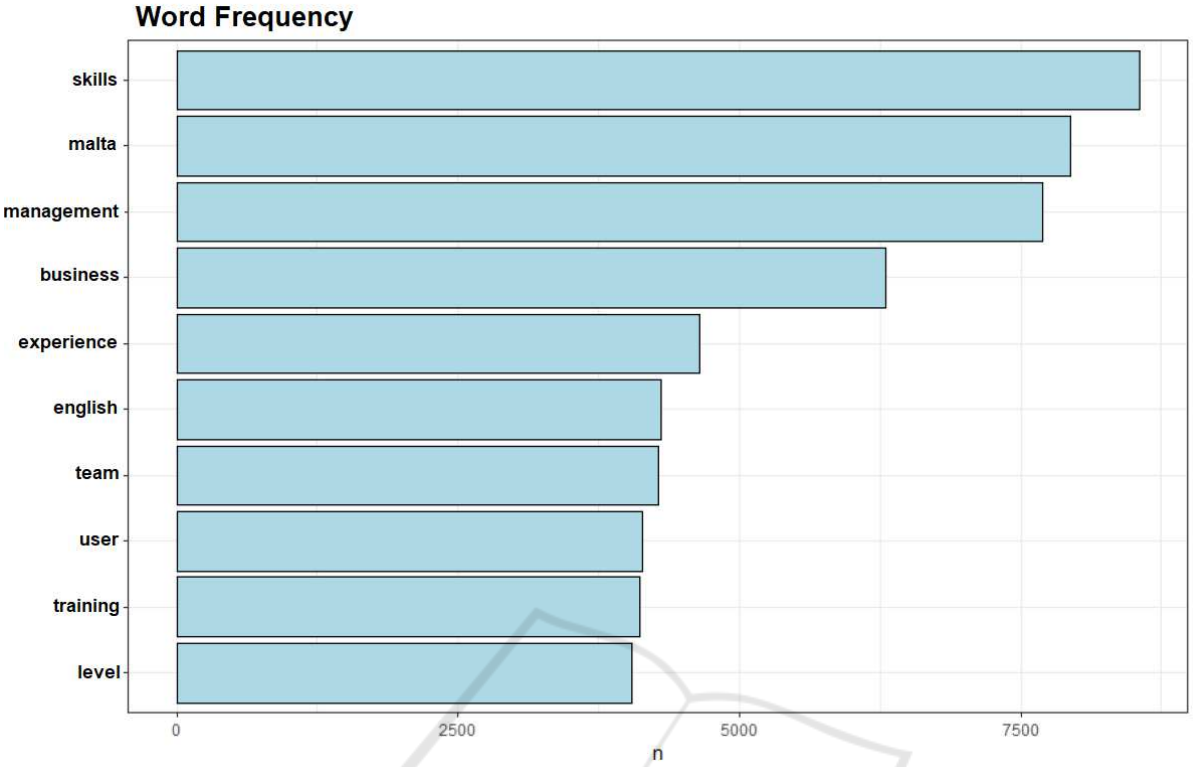

Figure 1: Word frequency: words occuring more than 4000 times throughout the whole collection of texts.

match to the job description inserted. The value 10 is chosen as it was decided to be a good number of candidates which can be shortlisted for a given vacancy. This is done using the Neural Vector Space Model (NVSM). Preemptively, all documents were preprocessed to remove any unwanted noise in the data. All words which appeared in the texts were seperated from each other if they appeared after one of these characters: "\r\n\t.,;:()?!//; and then tokenised into different terms after all the letters in each word were converted to lowercase. Stopwords were removed from the vocabulary and every word was stemmatised using the well-known Porter's suffix stripping algorithm (Porter, 1980). Also, all words having less than 3 or more than 100 characters were removed from the dataset and each stemmed word was converted to a one-hot vector.

Firstly, we shall be looking at the distribution of terms within the collection of texts made up of résumés and job descriptions. A total of 53985 words were recorded in the collection and Figure 1 shows the ones which were observed most frequently throughout the dataset. However, the most frequent words are not necessarily the most important ones as they need to be weighted depending on the number of documents they have appeared in.

It can be seen that the words in Figure 1, such as 'skills', 'business' or 'experience' would not be the most important words in a résumé. This point was highlighted in the discussion in the document vectors section regarding the use of TF-IDF weights. In fact, in Figure 2 and Figure 3, we see that the bi-grams and tri-grams with the highest TF-IDF weights for a randomly picked résumé give a lot more information about the résumé than any of these most frequently used words would.

We obtain the word embeddings for each word, where the resulting word vector for each word has dimension 300. We construct a document term matrix $\mathbb{D}$ consisting of documents as rows and terms as columns to obtain the weights for TF-IDF. Finally, we use the weights and word embeddings to derive the document vectors. A total of three models are trained, which we shall call M_0, M_1 and M_2, where the suffix denotes the number of skipped words allowed in the skip-gram architecture - this means that M_0 will consider traditional N-grams. In our case, N is taken to be 5 and c is varied according to the number of skipped words. Furthermore, the learning rate of the neural network is taken to be 0.075, k is taken to be equal to 5 in the negative sampling loss function (5). The resulting unique 5-grams in each of the models M_0, M_1 and M_2 can be seen in Table 1.

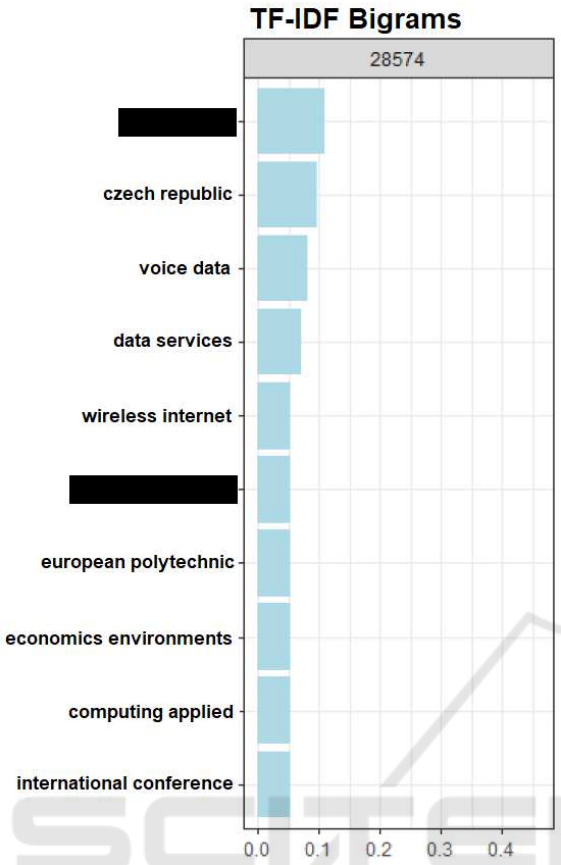

Figure 2: Top bi-grams for randomly selected résumé. Potentially sensitive information blotted in black.

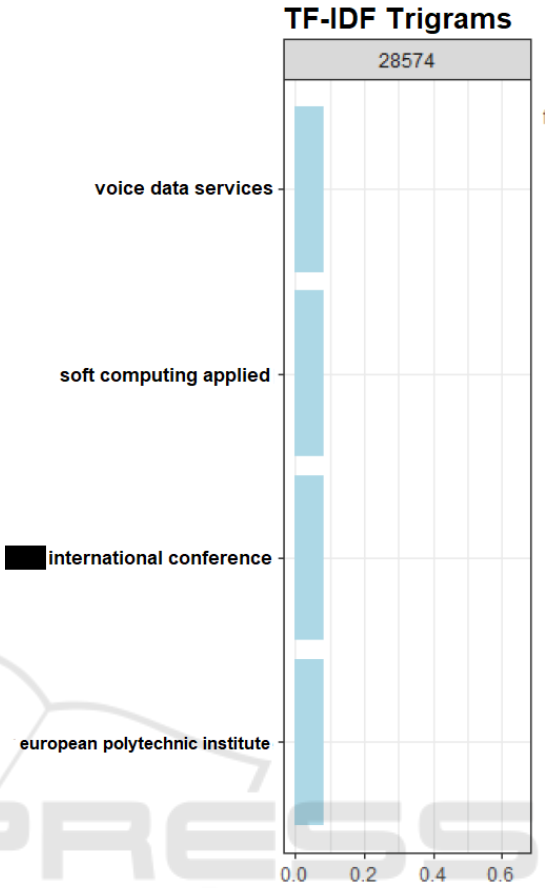

Figure 3: Top tri-grams for a randomly selected résumé. Potentially sensitive information blotted in black.

Table 1: Unique 5-grams for M_0, M_1 and M_2.

Model	Unique 5-grams
M_0	48655
M_1	847190
M_2	1216294

For our model, we take three job descriptions pertaining to three different jobs - we shall call them Job 1, Job 2 and Job 3. Job 1 relates to a sales executive role, Job 2 relates to a business development role and job 3 relates to a software development role. Figure 2 shows histograms of the résumés' similarities with each query using M_0, M_1 and M_2. The higher the similarity, the more relevant the document is to the inputted query according to our model. It can be noted that the overall similarity between documents increases at the number of skips is increased. This can be expected since more permutations are available within each document. The performance of these models is evaluated and compared using the $F1$-score. The $F1$-score combines precision and recall to give an overall measure of the model's accuracy via the equation

Table 2: Precision and Recall Illustrative Example.

Rank	Judgment	Precision	Recall
1	R	1.0	0.1
2	N	0.50	0.1
3	R	0.66	0.2
4	N	0.50	0.2
5	R	0.60	0.3
6	R	0.66	0.4
7	N	0.57	0.4
8	R	0.63	0.5
9	N	0.55	0.5
10	N	0.50	0.5

$$F1 = 2\left(\frac{precision \times recall}{precision + recall}\right) \tag{7}$$

(7) is based on the first 10 documents each model returns as being the most similar to each of the three job descriptions. Precision and recall are calculated in a cumulative manner as shown in Table 2. Relevant documents are denoted by R, whereas non-relevant ones are denoted by N. The cumulative precision is

Figure 4: Similarity histograms for models M_0 (upper row), M_1 (middle row) and M_2 (bottom row) for Job 1, Job 2 and Job3.

Table 3: $F1$ scores for Jobs 1, 2 and 3 given models M_0, M_1 and M_2.

Job	M_0	M_1	M_2
1	0.4615	0.667	0.667
2	0.75	0.947	0.947
3	0.889	0.571	0.571

calculated as the proportion of the number of relevant documents up to that point. On the other hand, recall is calculated by incrementing 0.1 (1/10) every time a relevant document is found. The final $F1$ score is calculated on the precision and recall obtained in the tenth row. Hence, for the illustrative example in Table 2 we have $F1 = 2\frac{0.5\times0.5}{0.5+0.5} = 0.5$.

As one can see in Table 3, the best fit when considering only standard N-grams (model M_0, no skips) is Job 3 (the software development role), with an $F1$ score of 0.889. Job 2 (the business development role) also had a decent $F1$ score of 0.75. Job 1 (the sales executive role) had a very poor $F1$ score on the other hand. On the other hand, for model M_1, the $F1$ score for Job 1 and Job 2 is greatly improved while the $F1$ score for Job 3 is much inferior. No difference in results were given, on the other hand, between model M_1 and model M_2. We can also see that the description for Job 1 performed the poorest in the résumé search when taking into account the $F1$ score for all models.

6 DISCUSSION

It can be noted that different models can perform better for different applications. The no skips model performs better for highly specific job descriptions such as the software development role. On the other hand, when considering less specific job descriptions, models with skips tend to return more relevant documents. It is therefore advisable to experiment with different settings for K in the word embedding model when performing the search.

As a side note, it was observed that the model did not perform well if the job descriptions included job titles of other roles such as, for example, "... *reporting directly to the chief executive officer*...". This is because the applied model does not look at the the order in which words are presented, but rather at the collection of terms within each document. Hence, when using this model, recruiters need to take into consideration that some words might be related more to other job descriptions than to the one in question, as this can lead the information retrieval system to return seemingly irrelevant documents, and this issue may need to be rectified.

REFERENCES

Bengio, Y., Ducharme, R., Vincent, P. and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137-1155.

Cabrera-Diego, L. A., Durette, B., Lafon, M., Torres-Moreno, J. M., and El-Bèze, M. (2015). How Can We Measure the Similarity Between Résumés of Selected Candidates for a Job?. In *Proceedings of the 11th International Conference on Data Mining (DMIN'15)*, pages 99-106.

Cabrera-Diego, L. A., El-Bèze, M., Torres-Moreno, J.M., and Durette, B. (2019). Ranking Rsums Automatically Using only Rsums: A Method Free of Job Offers. *Expert Syst. Appl.*, 123: 91-107.

Guthrie, D., Allison, B., Liu, W., Guthrie, L., and Wilks, Y. (2006). A Closer Look at Skip-Gram Modelling. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006), Genoa, Italy.*

Le, Q. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. In *ICML'14 Proceedings of the 31st International Conference on International Conference on Machine Learning*, pages (II)1188-(II)1196, ACM.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3136-3144, NIPS.

Naveenkumar, D.S.R., Kranthi Kiran, M., Thammi Reddy, K. and Sreenivas Raju, V. (2015). Applying NLP Techniques to Semantic Document Retrieval Application for Personal Desktop. In *Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1*, pages 385-392, Springer.

Pandiarajan, S., Yazhmozhi, V. M. and Praveen Kumar, P. (2014). Semantic Search Engine Using Natural Language Processing. *Advanced Computer and Communication Engineering Technology*, 351:561-571, Springer.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130137.

Schmitt, T., Gonard, F., Caillou, P., and Sebag, M. (2017). Language Modelling for Collaborative Filtering: Application to Job Applicant Matching. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1226-1233, IEEE.

Wei, X., and Croft, W. B. (2006). LDA-based Document Models for Ad-Hoc Retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178-185, ACM.

Wang, C. and Blei, D.M. (2011). Collaborative Topic Modeling for Recommending Scientific Articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 448-456, ACM.