Using Word Embeddings for Information Retrieval: How Collection and Term Normalization Choices Affect Performance

Dwaipayan Roy
Indian Statistical Institute, Kolkata
dwaipayan_r@isical.ac.in

Debasis Ganguly
IBM Research, Dublin, Ireland
debasis.ganguly1@ie.ibm.com

Sumit Bhatia
IBM Research, Delhi, India
sumitbhatia@in.ibm.com

Srikanta Bedathur
Indian Institute of Technology, Delhi
bedathur@cse.iitd.ac.in

Mandar Mitra
Indian Statistical Institute, Kolkata
mandar@isical.ac.in

ABSTRACT

Neural word embedding approaches, due to their ability to capture semantic meanings of vocabulary terms, have recently gained attention of the information retrieval (IR) community and have shown promising results in improving ad hoc retrieval performance. It has been observed that these approaches are sensitive to various choices made during the learning of word embeddings and their usage, often leading to poor reproducibility. We study the effect of varying following two parameters, *viz.,* i) the term normalization and ii) the choice of training collection, on ad hoc retrieval performance with word2vec and fastText embeddings. We present quantitative estimates of similarity of word vectors obtained under different settings, and use embeddings based query expansion task to understand the effects of these parameters on IR effectiveness.

ACM Reference Format:
Dwaipayan Roy, Debasis Ganguly, Sumit Bhatia, Srikanta Bedathur, and Mandar Mitra. 2018. Using Word Embeddings for Information Retrieval: How Collection and Term Normalization Choices Affect Performance. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3269206.3269277

1 INTRODUCTION

Embedding vocabulary terms as dense, real valued vectors in a low dimensional space has been shown to successfully encapsulate semantic concepts associated with them [9]. Semantic relationships captured by such *word embeddings* have been used to improve the performance of various information retrieval tasks such as document ranking [1, 5, 13], query reformulation [6, 14], relevance feedback [3, 11], and end-to-end deep neural ranking models [7, 10].

In an embedded space of words, vectors for semantically related terms lie in close proximity to each other and thus, the distance between embedding vectors of two terms can be taken as a measure of semantic relatedness between the terms. However, the choices made during the training phase can result in different representative vectors and thus, the *relative* semantic relatedness between terms is not preserved across different embedding spaces. Different studies on the application of word embeddings to IR problems have made different choices for learning the word embeddings indicating that there is a lack of established standards in this regard (which collection to use, what pre-processing to apply, etc.). On one hand, we have results that show that query-specific embeddings trained on a subset of relevant document are superior over globally trained embeddings [4], on the other hand, retrieval evaluations are conducted using embeddings learnt over an external collection such as Wikipedia [13]. The pre-TREC [8] IR community suffered from the lack of a similar set of 'best practices' (e.g. stopword removal is effective, too aggressive stemming often degrades retrieval quality). The best practices gradually became established after a few years of the TREC evaluation forum.

While efforts have been made to study the sensitivity of word embeddings to different model parameters (e.g. embedding dimensions, context window size, skip-gram v/s cbow, etc.) and their impact on ad hoc retrieval perfromance [15], little attention has been paid to investigate the effects of the nature of the *training collection* and *term normalization* used to learn the embedded vectors. In this work, we focus on this aspect and study *how the choices made with respect to the underlying corpus and its pre-processing impact the performance of downstream IR tasks*. Since the embedding models rely on context around the terms in training corpus, these choices are crucial as they directly affect the *contextual information* around each term and thus, can result in significantly different embedding spaces (Section 2). While a large, generic external corpus may be able to provide diverse contextual information to learn good embedding representations, the target corpus (i.e., the collection on which retrieval is to be executed) will be topically more relevant. Likewise, normalizing terms (via stemming) can help aggregate the contexts of different morphological forms together and thus, may help learn better representations of these terms.

In order to study how the above choices effect the learned embeddings, we first describe two measures to compare embeddings learned under different settings (Section 2). Next, we use a word embedding-based query expansion method [11], and analyze the impact of embedding variations on the retrieval performance of this method on three different standard test collections. To study the impact of collection choice, we compare retrieval performance obtained using the embeddings trained on the target collection as well as Wikipedia (a generic, external collection). To understand the effect of term normalization, we report results using embeddings learned from stemmed and unstemmed collections. In addition, we also describe an intermediate *composition* method that can be used to simulate the effect of stemming in cases where pre-trained word vectors on unstemmed corpus are available and re-training may not be feasible. We report all results with two different word embedding models, (i) **word2vec** [9], one of the most commonly used word

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3269277

embedding model; and (ii) **fastText**, a word embedding model that utilizes sub-word information for learning term representations. We decided to study fastText as it performs implicit term normalization by composing skipgram vectors of character n-grams [2]. We compare the retrieval performance achieved using different word vector representations learned under different settings and analyze the reasons for these differences (Section 3) and conclude the paper with directions for future work (Section 4).

2 EFFECTS OF EMBEDDING VARIATIONS

Measuring Similarity between Embeddings: Word vector embeddings are obtained through training on an underlying collection of documents.More formally, in skipgram *word2vec* [9], word vectors are obtained by sliding a context window pivoted at each word position and maximizing the likelihood of generating the context given the current word. Clearly, the context words surrounding the target word can be very different for different collections resulting in different representative vectors for the same word. Given two word embedded spaces $\mathbb{E}_1$ and $\mathbb{E}_2$, we now formally describe ways of measuring the similarity between them. By the property of the objective function of the word2vec algorithm, two word vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ will have similar representations if the word y shares a fair amount of context terms with the word x. Intuitively, two embedded spaces will be similar if there is considerable overlap in the set of neighborhood (top-k most similar) vectors around each word. Formally, this can be measured by the average Jaccard similarity between the top-K neighborhoods around each word in the two embedded spaces, i.e.,

$$\sigma_k(\mathbb{E}_1, \mathbb{E}_2) = \frac{1}{|\mathbb{E}_1 \cap \mathbb{E}_2|} \sum_{\mathbf{x} \in \mathbb{E}_1 \cap \mathbb{E}_2} \frac{|N_k^{\mathbb{E}_1}(\mathbf{x}) \cap N_k^{\mathbb{E}_2}(\mathbf{x})|}{|N_k^{\mathbb{E}_1}(\mathbf{x}) \cup N_k^{\mathbb{E}_2}(\mathbf{x})|}, \quad (1)$$

where $N_k^{\mathbb{E}}(\mathbf{x})$ denotes the neighborhood set of top-k most similar vectors around word x in embedding $\mathbb{E}$. The Jaccard similarity does not take into account the relative ranks of the neighborhood word vectors. A way to incorporate the rank information then is to measure the average Spearman's correlation coefficient ρ (normalized within the range of $[0, 1]$ to be consistent with Equation 1), between the top-K neighborhoods, as follows.

$$\rho_k(\mathbb{E}_1, \mathbb{E}_2) = \frac{1}{|\mathbb{E}_1 \cap \mathbb{E}_2|} \sum_{\mathbf{x} \in \mathbb{E}_1 \cap \mathbb{E}_2} \frac{1}{2}\Big(1 + \rho\big(N_k^{\mathbb{E}_1}(\mathbf{x}), N_k^{\mathbb{E}_2}(\mathbf{x})\big)\Big) \quad (2)$$

For both metrics, a higher value indicates higher similarity between two embedded spaces. In Section 3, we use the metrics of Equations 1 and 2 to measure the similarity between embeddings learned over different collections and under different settings, and investigate the correlation between the differences in embedding spaces with differences in the performance of the downstream IR tasks. Note that, the metrics are used to measure local topology of embedding spaces to quantify how these differences may lead to differences in effectiveness of downstream tasks, (in this paper - ranking).

Collection Choice - Target vs. External Collection: Some reported studies on word embedding based approaches in IR use word vectors trained on respective document collections to improve retrieval performance [3, 5], while there are others that have used pre-trained word vectors [13]. Since the collection on which an embedding method is trained can play an important role in learning the relatedness between terms, an obvious choice is to use the underlying document collection of the retrieval system to learn the word vectors. The word vectors trained on the target retrieval collection are likely to capture the underlying term semantics specific to that collection, which can lead to improvements in IR effectiveness. The alternative is to use pre-trained vectors on large, external document collections that are often publicly available (e.g. word2vec, fastText, provide such pre-trained vectors for different languages). However, the term semantics learned with such generic collections, e.g. Wikipedia, may not capture the relevant associations between terms required to improve IR effectiveness on a particular search collection, e.g. news-wire. This choice in the experimental setup leads to our first research question, **RQ-1**: *Does it help to use word vectors trained over the target test collection instead of pre-trained word vectors on external corpora, as the former may better capture semantic relations of the target collection?*

Choices for Term Normalization: Term normalization (e.g. stemming) plays an important role in IR. An intuitive pre-processing step is thus to apply stemming before training and directly learn the word vectors for stemmed roots. During retrieval, the stemmed query terms can then be exactly matched with the embedded word vectors. However, after training word vectors on a document collection that is not pre-processed, or when working with pre-trained word vectors trained on unprocessed collections, one needs to consider how to match the embedded word vectors with the index units of an IR collection (typically stemmed). We observed that the reported studies usually lack clarity regarding this matching step and this has led to differences in experimental setup when implementing the same retrieval methodology by different research studies. For instance, Ganguly et al. [5] used pre-processing before training word vectors on the TREC Robust collection. However, when reproducing their results for a baseline, such pre-processing was not applied [13]. We propose an indexing unit composition (IUC) method to resolve this normalization difference between the index terms and the word vectors by stemming the trained word vectors and composing (vector sum) the ones that yield identical stems (representative of equivalence classes). This yields a single vector representation for each equivalence class. Another alternative to account for term normalization differences could be to use an alternative word embedding model, such as fastText [2], that implicitly performs term normalization by utilizing character n-grams. These experimental choices lead us to following research questions – **(RQ-2)** *what effect does embeddings trained on a stemmed collection have on IR performance compared to an unstemmed collection?*; and **(RQ-3)** *what effect does using approximations such as IUC or different embedding model such as fastText have on IR perfromance?*

3 EXPERIMENTS AND RESULTS

3.1 Settings

We use three different standard IR collections and learn word embeddings over them under different settings. Each embedding setting corresponds to a permutation of the following three components.

- Training algorithm, which is one of word2vec or fasttext.
- Collection on which word vectors are trained, which is one of '**target**' or '**external**'.
- Term normalization applied before/after learning word embeddings.

For the training collection option, '**target**' indicates the collection on which retrieval is to be performed. We use TREC Robust, WT10G and GOV2 collections for this purpose. The '**external**'

Collection	Normalization	Voc-size	Name
Wikipedia	Unprocessed	1259578	Wk-U
Wikipedia	Composed (Post)	996471	Wk-C
Wikipedia	Stemmed (Pre)	1031792	Wk-S
TREC Robust	Unprocessed	266673	Rb-U
TREC Robust	Composed (Post)	208760	Rb-C
TREC Robust	Stemmed (Pre)	206845	Rb-S
WT10G	Unprocessed	1532194	WT-U
WT10G	Composed (Post)	794656	WT-C
WT10G	Stemmed (Pre)	807778	WT-S
Gov2	Unprocessed	7834596	Gv-U
Gov2	Composed (Post)	2592352	Gv-C
Gov2	Stemmed (Pre)	2735818	Gv-S

Table 1: Names for different experiment settings.

option indicates that the embedding is learned on an external collection. As an external collection, we use the entire Wikipedia dump (November 2013) that was used in the Tweet contextualization track of CLEF-2013. As term normalization options, we experimented with three variations.

- No normalization of the words (denoted by '**U**') during word vector training.
- No pre-processing during word vector training, followed by a post-processing step of composing (vector-sum) of the words that yield identical stems, denoted by '**C**'.
- Pre-processing (Porter stemming) each word of a collection before training word embedding, denoted by '**S**'.

Table 1 summarizes different collections and settings used for learning different embedding variants. We used the publicly available implementations of word2vec and fastText and set the embedding vector dimensions to 200 for all settings. In case of word2vec, the context window size for the skipgram model was set to 10.

Query Expansion with Word Vectors: Next, to study the effect of different embedding variants on retrieval performance, we employ the embedded vectors for query expansion (QE) as proposed by Roy et al. [12]. Specifically, given a query Q, the expanded query is computed as $Q' = Q \bigcup_{t \in Q} N_k(\mathbf{t})$, where $N_k(\mathbf{t})$ denotes the set of k-nearest neighbors of term t as obtained from the embedded space. The cosine-similarity values of these nearest neighbors are used as their weights in Q'. We used Lucene for indexing and retrieval and implementing the above model. After initial experimentation on the model presented in [12], we set $k = 120$, i.e. add 120 most similar terms to obtain the expanded query. Further, the value of the Jelinek-Mercer smoothing parameter in the language model, λ, was set to 0.6. While there exists more advanced approaches that utilize word vectors for improving retrieval performance in more involved ways (e.g. [5, 11, 13]), the chosen approach employs word vectors in a straight-forward manner (directly using the similarities between words as weights for QE) making it easier to interpret and analyze the observed differences in performance directly in terms of the neighborhoods of the word vectors.

3.2 Results and Discussions

Table 2 reports the similarity metrics ρ_k and σ_k (Equations 1 and 2) between embeddings learned using different collections and settings. Note how the embeddings learned from different collections differ significantly as measured by the two parameters. For e.g., the σ_k values between Wk-U and the three collections (Rb-U,WT-u, Gv-U) lie between 0.56 and 0.60 indicating that the semantic concepts associated with the terms as captured by different collections vary significantly. Further, even when comparing the embeddings learnt over same document collection, we observe that normalization choices lead to very different embedding spaces. For example, word2vec embeddings for Rb-C and Rb-S have a σ_k of 0.6057 indicating that even for the same underlying corpus, different normalization approaches can result in very different embedding spaces. Similar observations can be made for other settings and together, they lend significant weight to our initial hypothesis that *embedding spaces can differ appreciably with variations in term normalization, stemming, etc. even if the underlying corpus remains same.*

Next, we report the results of the word vector based QE approach on the three test collections in Table 3. We observe from the table that for word2vec embeddings, using the target corpus and some form of normalization (either composition (C) or stemming (S)) for learning embeddings, in general, helps in achieving significantly better performance than using embeddings learned from an external, un-normalized corpus. Further, note that from Table 2, the space of embedded word vectors trained on the unstemmed version of an external collection exhibits lower similarity to the target corpus than that trained on a processed version (composed or stemmed). For example, ρ(**Wk-U**, **Rb-U**) is lower than ρ(**Wk-S**, **Rb-S**) (same applies for σ values). This indicates that the retrieval performance obtained using embeddings learned from unprocessed collection should be lower than that obtained with the processed collection. This can be verified from Table 3, which shows that word2vec QE MAP values for TREC-Rb with unprocessed Wikipedia (0.2280) is lower than that of composed and stemmed Wikipedia (0.2477 and 0.2496 respectively).

An intriguing observation is that this trend of getting better IR effectiveness with word2vec embeddings learned from normalized, target collections does not held when fastText is used for learning embeddings. As we observe from Table 3, IR perfromance is better using fastText embeddings learned using un-normalized, external collection. One exception here is the Gov2 collection, where higher IR performance is achieved using target collection. We speculate that the reason for this observed behavior could be the way embeddings are learned by fastText. Unlike word2vec that works at individual word level, fastText first learns the embedded representations of character n-grams before combining them to obtain vectors for words. Therefore, it performs term normalization in an *implicit* manner and applying stemming before running fastText has the effect of reducing the number of character n-grams and altering their contexts, potentially leading to noisier representations of words.

Finally, for both word2vec and fastText, the post-processing step of composing unprocessed (whole) words into indexing units (stems) yields results that lie between the two extremes of learning the embeddings on unprocessed and stemmed collections. Given that in many cases training embedings on target corpus may not be feasible due to time and resource constrains, our experiments suggests that in such cases, it may be a better trade-off to use easily available pre-trained word vectors trained on large unprocessed collections (such as Google-News) and applying compositions such as proposed in this paper (IUC) to make use of advantages offered by term-normalization.

4 CONCLUSIONS AND FUTURE WORK

We investigated the impact of term normalization and collection choices in learning word embeddings and their effect on ad hoc retrieval performance. We proposed two metrics for measuring the similarities between the embedded spaces of word vectors obtained under different settings. Using embeddings trained over different collections and under different settings for the query expansion task, we found that: a) small differences in settings can lead to considerable differences in the embedded spaces of word vectors; b) these differences can lead to considerable variations in the effectiveness of downstream task of ad hoc IR, c) there is no 'clear winner' among the term normalization alternatives, since

Word2vec	Rb-U	WT-U	Gv-U
Wk-U	0.0911 \ 0.5828	0.0600 \ 0.5643	0.1319 \ 0.6040

Word2vec	WT-C	WT-S	Wk-C	Wk-S
WT-C		0.4853 \ 0.8315	0.1249 \ 0.5919	0.1521 \ 0.6063
WT-S			0.1355 \ 0.5993	0.1694 \ 0.6180
Wk-C				0.3170 \ 0.7002

FastText	Rb-U	WT-U	Gv-U
Wk-U	0.1295 \ 0.5899	0.1889 \ 0.6286	0.1787 \ 0.5744

FastText	WT-C	WT-S	Wk-C	Wk-S
WT-C		0.0669 \ 0.5595	0.0455 \ 0.5611	0.0315 \ 0.5539
WT-S			0.1247 \ 0.5914	0.1549 \ 0.6083
Wk-C				0.3714 \ 0.7339

Word2vec	Rb-C	Rb-S	Wk-C	Wk-S
Rb-C		0.1396 \ 0.6057	0.0827 \ 0.5773	0.0794 \ 0.5757
Rb-S			0.1085 \ 0.5881	0.1282 \ 0.5997
Wk-C				0.3161 \ 0.7022

Word2vec	Gv-C	Gv-S	Wk-C	Wk-S
Gv-C		0.3388 \ 0.7412	0.1237 \ 0.5903	0.1451 \ 0.6101
Gv-S			0.0970 \ 0.5804	0.1190 \ 0.5927
Wk-C				0.3094 \ 0.6989

FastText	Rb-C	Rb-S	Wk-C	Wk-S
Rb-C		0.4002 \ 0.7659	0.1182 \ 0.5994	0.0911 \ 0.5837
Rb-S			0.1054 \ 0.5912	0.1071 \ 0.5940
Wk-C				0.3924 \ 0.7532

FastText	Gv-C	Gv-S	Wk-C	Wk-S
Gv-C		0.3086 \ 0.7484	0.0349 \ 0.5484	0.2304 \ 0.6572
Gv-S			0.0431 \ 0.5832	0.0380 \ 0.5505
Wk-C				0.3892 \ 0.7411

Table 2: σ_k and ρ_k (below and above diagonal of each cell, respectively) values between different embedding spaces. A value of $k = 120$ (identical to #QE terms) is used to compute the similarities. Since it is not possible to compute the inter-embedding similarities between an unstemmed space and a stemmed one without the application of IUC, some comparisons do not exist in the table, e.g.between the pair WT-C and WT-U

Settings			MAP		
Method	**Domain**	**Normalization**	**Rb**	**WT10G**	**Gov2**
No QE			0.2355	0.1472	0.2072
Word2vec	External	Unprocessed (U)	0.2280	0.1562^{*}	0.2326^{*T}
Word2vec	External	Composed (C)	0.2477^{*U}	0.1610^{*U}	0.2341^{*}
Word2vec	External	Stemmed (S)	0.2496^{*UC}	0.1632^{*U}	$\mathbf{0.2363}^{*UC}$
Word2vec	Target	Unprocessed (U)	0.2272^{*}	0.1565^{*}	0.2139^{*}
Word2vec	Target	Composed (C)	0.2486^{*U}	$\mathbf{0.1712}^{*UE}$	0.2299^{*U}
Word2vec	Target	Stemmed (S)	$\mathbf{0.2520}^{*UCE}$	0.1692^{*U}	0.2313^{*U}
FastText	External	Unprocessed (U)	$\mathbf{0.2502}^{*CST}$	$\mathbf{0.1655}^{*CST}$	0.2321^{*S}
FastText	External	Composed (C)	0.2445^{*}	0.1570^{*}	0.2292^{*}
FastText	External	Stemmed (S)	0.2432^{*}	0.1553^{*}	0.2260^{*}
FastText	Target	Unprocessed (U)	0.2452^{*}	0.1537^{*}	$\mathbf{0.2489}^{*SE}$
FastText	Target	Composed (C)	0.2473^{*U}	0.1554^{*S}	0.2451^{*E}
FastText	Target	Stemmed (S)	0.2463^{*}	0.1533^{*}	0.2436^{*E}

Table 3: Embedding based QE [12] effectiveness with various settings on standard IR collections. A '$*$' indicates significance (paired t-test with 95% confidence) w.r.t 'No-QE'; U, C and S indicate significance w.r.t *U*nprocessed, *C*omposed and *S*temmed, respectively. For any method-normalization pair, E and T respectively indicates significant differences between the *E*xternal and *T*arget collections.

we observed that word2vec generally works well on a stemmed collection, whereas fastText on an unprocessed collection, and this can be attributed to the inherent characteristics of the embedding algorithms; since fastText is seen to perform better for other tasks when trained on unprocessed collection [2], this is the first effort to validate the claim for ad-hoc IR; d) our proposed post-processing term normalization by composing vectors of words yielding the same stem produces more stable results (lying between the two extremes of unprocessed and stemmed) across the different word embedding algorithms; e) composition based post term normalization can be a good choice when working with pre-trained word vectors trained with word2vec, since it outperforms the results with unprocessed word vectors on three standard IR collections; f) no 'clear winner' among the word vector training algorithms, since the results produced by word2vec are slightly better than those obtained with fastText on TREC-Rb and WT10G, whereas fastText word vectors produces better results on Gov2. In future, we plan to solidify these observations to offer general best practices for a range of different neural IR methods (e.g. DRRM[7]) as well as experiment using large datasets (e.g. Common Crawl).

REFERENCES

[1] Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. 2016. Analysis of the Paragraph Vector Model for Information Retrieval. In *Proc. of ICTIR'16*. 133–142.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL* 5 (2017), 135–146.

[3] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *Proc. of ACL'16*.

[4] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *Proc. of ACL'16*. 367–377.

[5] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J. F. Jones. 2015. Word Embedding based Generalized Language Model for Information Retrieval. In *Proc. of SIGIR'15*. 795–798.

[6] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context- and Content-aware Embeddings for Query Rewriting in Sponsored Search. In *Proc. of SIGIR '15*. 383–392.

[7] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proc. of CIKM'16*. 55–64.

[8] Donna K. Harman (Ed.). 1992. *Overview of TREC-1*. NIST.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proc. NIPS '13*. 3111–3119.

[10] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match using Local and Distributed Representations of Text for Web Search. In *Proc. of WWW'17*. 1291–1299.

[11] Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth J. F. Jones. 2016. Word Vector Compositionality based Relevance Feedback using Kernel Density Estimation. In *Proc. of CIKM'16*. 1281–1290.

[12] Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using Word Embeddings for Automatic Query Expansion. In *Proc. of NeuIR-2016 Workshop, collocated with SIGIR*.

[13] Hamed Zamani and W. Bruce Croft. 2016. Embedding-based Query Language Models. In *Proc. of ICTIR'16*. 147–156.

[14] Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed Representations. In *Proc. of SIGIR '15*. 575–584.

[15] Guido Zuccon, Bevan Koopman, Peter Bruza, and Leif Azzopardi. 2015. Integrating and Evaluating Neural Word Embeddings in Information Retrieval. In *Proc. of ADCS '15*. 12:1–12:8.