2015 IEEE 15th International Conference on Data Mining Workshops

Learning Semantic Similarity for Very Short Texts

Cedric De Boom, Steven Van Canneyt, Steven Bohez, Thomas Demeester, Bart Dhoedt
Ghent University – iMinds
Gaston Crommenlaan 8-201, 9050 Ghent, Belgium
{*cedric.deboom, steven.vancanneyt, steven.bohez, thomas.demeester, bart.dhoedt*}*@ugent.be*

***Abstract*—Levering data on social media, such as Twitter and Facebook, requires information retrieval algorithms to become able to relate very short text fragments to each other. Traditional text similarity methods such as tf-idf cosine-similarity, based on word overlap, mostly fail to produce good results in this case, since word overlap is little or non-existent. Recently, distributed word representations, or word embeddings, have been shown to successfully allow words to match on the semantic level. In order to pair short text fragments—as a concatenation of separate words—an adequate distributed sentence representation is needed, in existing literature often obtained by naively combining the individual word representations. We therefore investigated several text representations as a combination of word embeddings in the context of semantic pair matching. This paper investigates the effectiveness of several such naive techniques, as well as traditional tf-idf similarity, for fragments of different lengths. Our main contribution is a first step towards a hybrid method that combines the strength of dense distributed representations—as opposed to sparse term matching—with the strength of tf-idf based methods to automatically reduce the impact of less informative terms. Our new approach outperforms the existing techniques in a toy experimental set-up, leading to the conclusion that the combination of word embeddings and tf-idf information might lead to a better model for semantic content within very short text fragments.**

I. Introduction

On social media billions of small text messages are made public every day: own research indicates that almost every tweet is comprised of one up to approximately thirty words. To tap into this stream of extremely short text fragments, we need appropriate information retrieval algorithms. Tf-idf is an example of a traditional and very popular representation to compare texts, such as news articles, with each other [1], [2]. It relies on word overlap to find similarities, but in very short texts, in which word overlap is rare, tf-idf often fails. For this reason we need sentence representations that grasp more than just word contents.

In 2013, Mikolov et al. published three papers on the topic of distributed word embeddings to catch semantic similarities between words [3], [4], [5], which resulted in Google's *word2vec* software. Since then scientists have extensively used such embeddings to improve state-of-the-art algorithms in natural language processing, such as part-of-speech tagging [6], sentence completion [7], hashtag prediction [8], etc. There is however a lack of research and insight on how to effectively combine embeddings into a single sentence representation that contains most of its semantic information. Many authors choose to average or maximize across the embeddings in a text [8], [9], [10] or combine them through a multi-layer perceptron [6], [11], by clustering [12], or by trimming the text to a fixed length [11].

The Paragraph Vector algorithm by Le and Mikolov—also termed *paragraph2vec*—is a powerful method to find suitable vector representations for sentences, paragraphs and documents of variable length [13]. The algorithm tries to find embeddings for separate words and paragraphs at the same time through a procedure similar to word2vec. The collection of paragraphs, however, is known beforehand. This implies that finding a vector representation for a new and probably unseen paragraph—the theoretical number of different paragraphs is after all many times higher than the number of different words—requires additional training. Paragraph2vec is therefore not a fit candidate to be used in, e.g., a stream of messages as is the case with social media.

Further research is thus needed to derive optimal sentence representations based on word embeddings. By investigating and comparing the performance of several word combination approaches in a short-text matching task, we arrive at a novel technique in which we aggregate both tf-idf and word embedding signals. In this paper, we show how word embeddings can be combined into a new vector representation for the entire considered fragment, in which the impact of frequent words—i.e. with a low idf-component, and therefore mostly non-informative—is reduced with respect to more informative words. This leads to a significant increase in the effectiveness of detecting semantically similar short-text fragments, compared to traditional tf-idf techniques or simple heuristic methods to combine word embeddings. Our approach is a first step towards a hybrid method that unites word embedding and tf-idf information of a short text fragment into a distributed representation that catches most of that fragment's semantic information.

Very recently, Kusner et al. devised a simple method to measure the similarity between documents based on the minimal distance word embeddings have to travel from one document to another [14]. In this, the authors only consider non stop words, and evaluate their distance measure using kNN classification. We, however, will learn vector representations for documents, and we will evaluate our technique

978-1-4673-8493-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDMW.2015.86

through a newly crafted dataset of related text fragments. Also recently, Zheng and Callan created a simple algorithm based on linear regression to find relevant terms in a query [15]. The authors learned weights for each dimension in a word embedding using a supervised relevance signal. Our work is different in that we will learn weights for entire word vectors instead of separate dimensions. For this we will use tf-idf information, instead of only word embedding features. Furthermore, we will arrive at a globally applicable weighting scheme instead of a query-dependent one.

In the next section we will discuss our experimental setup and method of data collection, and explain how well a number of traditional techniques perform on our dataset. We will then use the gained insights to create more effective distributed representations by integration of the tf-idf information.

II. Experimental Set-up and Analysis

To evaluate techniques that measure semantic similarity between short text fragments, we need a reference set with couples of fragments that are semantically related, and couples which are not related. We denote the former as a *pair*, and the latter as a *non-pair*. Every couple consists of two texts, which are built up as a sequence of words. For an arbitrary couple we introduce the notation c, and the two texts in c are denoted as the sequences (c^1) and (c^2). Element j of sequence (c^1) is the vector of word j in the first text of c, denoted as $\mathbf{w}_j^1$:

$$\left(c^1\right) \triangleq \left(\mathbf{w}_1^1, \mathbf{w}_2^1, \mathbf{w}_3^1, \dots\right).$$

A vector representation for (c^1), combining the word representations contained in (c^1), is written as $\mathbf{o}^1$, and as $\mathbf{o}^2$ for (c^2) respectively.

In this paper we strive to give the initial impetus to learning sentence representations of very short text fragments mainly found on social media, but for now we will perform our experiments in a toy environment using English Wikipedia articles. These are of course very different textual media—which has some disadvantages, as we will discuss later—but Wikipedia articles have the benefit of being well-structured, which allows us to extract related texts more easily. In our experiments we use the Wikipedia dump of March 4th, 2015, after cleaning the articles by removing markup and punctuation. We convert all texts to lowercase and replace the numbers by a single character '0'. In our toy setting we require that the texts of all couples are composed of the same number of words, i.e. the length of the sequences (c^1) and (c^2) is equal to n_c. To extract a pair of texts, each containing n_c words, we take the first n_c words of a paragraph, skip the next two words, and then take the following n_c words. To extract a non-pair, we take n_c words out of two random paragraphs of different articles. This approach is closely related to the one used by Hu et al. to extract pairs and non-pairs from the Reuters corpus [7]. In total we extract five million pairs and five million non-pairs, for texts of ten, twenty, and thirty words long[1].

To represent words as a vector, we train word embeddings on the entire Wikipedia corpus. We do this through Google's *word2vec* software, using skip-gram with negative sampling, a context window of five words, and 400 dimensions. We also calculate document frequencies for every word using the same Wikipedia corpus.

We regard two text fragments to be semantically similar if their corresponding vector representations lie close to each other according to some distance measure, and dissimilar if the vectors lie farther apart. Semantic similarity between text fragments is therefore related to semantic similarity between skip-gram word embeddings, in which the cosine distance between related words is smaller compared to unrelated words. This is also the reason why we do not use paraphrase datasets, such as the Microsoft Research Paraphrase corpus or the SemEval2015 Twitter Paraphrase dataset, to perform our experiments. After all, the notion of semantic relatedness in these datasets is often too narrow: if one sentence is about Star Wars and another about Anakin Skywalker, they are semantically related although they might not be paraphrases of each other.

To verify whether our own dataset of 10 million Wikipedia couples is a valid candidate to perform similarity experiments on, we will test different techniques that try to enhance the discriminative power between pairs and non-pairs as much as possible—i.e. leading to small distances between pairs and larger distances between non-pairs. We start with techniques ranging from plain tf-idf to naive combinations of word embeddings, after which we investigate elementary mixtures of tf-idf and word embedding signals.

For every couple c we create a tf-idf representation for both (c^1) and (c^2), and calculate the cosine similarity between (c^1) and (c^2). Figure 1 shows a histogram plot of the number of couples as a function of their cosine similarity, for both pairs and non-pairs separately, and for texts of 20 words long. We see that there are many couples having a very low cosine similarity, which is due to the very short length of the text fragments. There are many more pairs having a larger cosine similarity than there are non-pairs, but non-related texts can also exhibit relatively large similarity values, which is due to coincidental overlap of mostly non-informative words.

As for word embeddings, we create two traditional sentence representations as a baseline. In a first representation we take the mean of all word embeddings in the text:

$$\forall \ell \in \{1,2\} \colon \mathbf{o}^\ell = \frac{1}{n_c} \sum_{j=1}^{n_c} \mathbf{w}_j^\ell, \tag{1}$$

in which $\mathbf{w}_j^\ell$ represents the *word2vec* word embedding vector of the j'th word in text sequence ℓ. In a second

[1]The dataset can be obtained upon request.

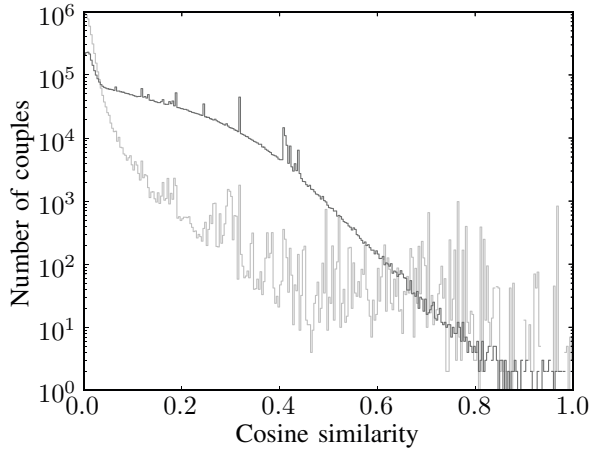

Figure 1. Histogram plot of the number of couples as a function of their cosine similarity using tf-idf, for both pairs (dark grey) and non-pairs (light grey).

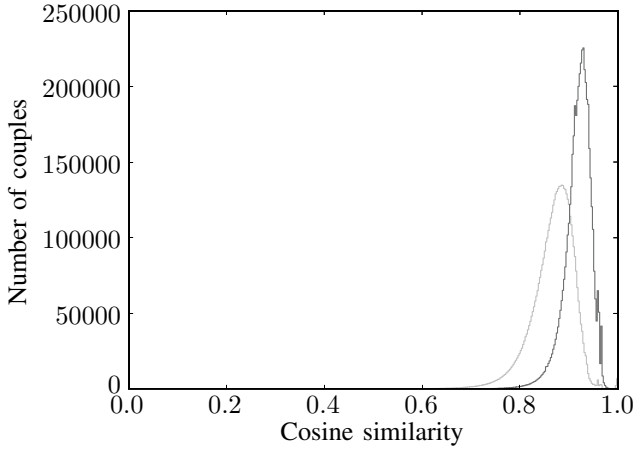

Figure 2. Histogram plot of the number of couples as a function of their cosine similarity using the mean of the word embeddings, for both pairs (dark grey) and non-pairs (light grey).

representation, we take for each dimension the maximum across all embeddings:

$$\forall \ell \in \{1,2\}, k \in \{1,\ldots,400\}\colon \mathbf{o}_k^\ell = \max_j \mathbf{w}_{j,k}^\ell. \quad (2)$$

Between two such representations we then calculate the cosine similarity as before.

Figure 2 shows a histogram plot for the mean of the embeddings, and for texts of 20 words long. The graph shows two curves with a stretched tail towards lower cosine similarities. We see that the mode of the non-pairs curve lies more to the left than the mode of the pairs curve, but still close to each other. Our hypothesis is that this is due to overlap in non-informative but frequently occurring words, such as articles and prepositions. Such words, however, contribute little to the semantic meaning of a text, and by reducing the influence of such words, we want to accentuate the true semantics of a text fragment. By reducing this coincidental similarity, we intend to shift the non-pairs stronger toward lower similarities than the pairs, hence increasing the resolution between both.

Since less informative terms are common to many sentences, they mostly have a high document frequency as well. We therefore implement the mean and max techniques again, but this time we only use the top 30% of the words with the highest idf component. In a final technique we use all words, but we weigh each word vector with its idf value, after which we take the mean.

As word embeddings contain both positive and negative numbers, we also test the influence of the sign in these embeddings. In a first experiment we take, instead of the maximum, the minimum across all word embeddings in a text. In a second experiment we test whether extremes, either positive or negative, are important indicators for semantic similarity. We therefore simply concatenate the maximum vector and the minimum vector to form a new vector representation.

To evaluate the power of the previously described techniques to discriminate between pairs and non-pairs, we calculate two performance metrics: optimal split error and Jensen-Shannon (JS) divergence. We obtain the former using the optimal threshold for which the number of misclassified couples is minimal. The latter is a symmetric measure expressing the similarity between two probability distributions, based on the well-known—but asymmetric—KL divergence. The lower the optimal split error or the higher the JS divergence, the better a technique can distinguish pairs from non-pairs. Table I shows the results, for texts of 10, 20 and 30 words.

As for the traditional techniques, the max approach works best for 10 and 20 words, but as the number of words increases to 30, tf-idf performs best, which is logical since word overlap rises with a growing number of words, as can be seen in Figure 1 as well. The min approach performs almost as well as the max approach. By concatenating the minimum and maximum vectors, we see that split error and JS divergence are improved by a large margin. We can thus conclude that the sign in word embeddings holds complementary semantic information. By incorporating document frequency information, we do better than all previous techniques for texts of 10 words long. But for longer texts the mean approach performs best, while the min and max combinations achieve worse results than when using the complete text. The best performing techniques are the idf-weighed mean approach and the approach taking the mean of 30% of the word vectors with the highest idf-components, as they perform similarly across the different word lengths.

We also investigate the influence of the used distance metric. We test cosine distance, Euclidean distance, L_3-norm, L_4-norm and Bray-Curtis distance, which are nor-

Table I
COMPARISON OF DIFFERENT WORD VECTOR AGGREGATION TECHNIQUES WITH COSINE SIMILARITY.

	10 words		20 words		30 words	
	Split error	JS divergence	Split error	JS divergence	Split error	JS divergence
Tf-idf	36.15%	0.11946	20.09%	0.35991	12.55%	0.54468
Mean	30.67%	0.16186	21.05%	0.34109	16.33%	0.45534
Max	28.27%	0.20831	19.06%	0.40030	15.18%	0.49882
Min	28.89%	0.19694	19.54%	0.38696	15.69%	0.48592
Min/max	26.89%	0.22946	16.86%	0.45004	12.76%	0.56253
Mean, top 30% idf	23.69%	0.30044	15.86%	**0.48260**	**12.42%**	**0.56456**
Max, top 30% idf	26.56%	0.24028	20.63%	0.36809	16.66%	0.45522
Min/max, top 30% idf	25.43%	0.25761	19.13%	0.39775	14.91%	0.49927
Mean, idf weighed	**23.35%**	**0.30400**	**15.77%**	0.48028	12.43%	0.56028

Table II
COMPARISON OF DIFFERENT DISTANCE METRICS FOR TEXTS OF 20 WORDS LONG.

	Split error	JS divergence
Mean, cosine	21.05%	0.34109
Mean, Euclidean	**19.55%**	**0.37788**
Mean, L_3	19.62%	0.37511
Mean, L_4	19.77%	0.37061
Mean, Bray-Curtis	21.22%	0.33775

malised between 0 and 1. We use texts of 20 words long and the mean of the embeddings. Table II shows the results, again expressed in terms of split error and JS divergence. Euclidean distances perform best in our tests, so we continue to use Euclidean distances hereafter.

III. LEARNING SEMANTIC SIMILARITY

As became clear from the data analysis, combining knowledge from both tf-idf and word embeddings can be beneficial. Using only the portion with the highest idf component of all words clearly reduces split error and improves JS divergence. After all, low-idf words have no clear-cut semantic meaning, and since these words are present in many sentences, there is more coincidental overlap between non-related sentences. Removing these words—or lowering their influence—from a text representation thus succeeds in pulling apart the average similarity between pairs and between non-pairs.

In this section we investigate how we can learn to optimally weigh words in a short text. This way we intend to do better than just taking the top-idf words or weighing these words with their idf component, in order to maximize the average distance between pairs and non-pairs. As before, we perform the experiments in a toy setting on couples of short Wikipedia texts, with as many pairs as non-pairs. We divide the total dataset into a training set $\mathcal{D}$ of 1.5 million couples, a test set $\mathcal{T}$ of 1.5 million couples and a validation set $\mathcal{V}$ of 2.0 million couples. Since we describe ongoing research and present the first steps towards a flexible hybrid technique, we only consider texts of 20 words long in this section, and varying the fragment length will be suggested as future work.

We implement the following learning procedure. For every couple c in the training set we sort the words in both texts (c^1) and (c^2) according to their document frequency—i.e. the word with the lowest document frequency comes first—arriving at $(c^{1'})$ and $(c^{2'})$. Next we multiply the word embedding vector of each word $\mathbf{w}_j^{1'}$ and $\mathbf{w}_j^{2'}$ with an importance factor i_j; these importance factors are global weights that will be learned. Finally, we take the mean of these weighed embeddings to obtain a fixed-length vector $\mathbf{o}^1$ for (c^1) and $\mathbf{o}^2$ for (c^2):

$$\forall \ell \in \{1,2\}\colon \mathbf{o}^\ell = \frac{1}{n_c}\sum_{j=1}^{n_c} i_j \cdot \mathbf{w}_j^{\ell'}. \tag{3}$$

We take the mean since it is the best performing technique in the third part of Table I. Figure 3 illustrates the entire procedure of calculating a vector representation for a sentence using the importance factor method. We see that first the words in the sentence are sorted according to their idf-component; next, their 400-dimensional word embedding vectors are multiplied by importance factors, and finally the mean is taken.

To learn the importance factors, we define a loss function as a function of any couple c that minimizes the distance between the vectors of a pair, and maximizes the distance between the vectors of a non-pair:

$$f(c) \triangleq \begin{cases} d(\mathbf{o}^1, \mathbf{o}^2) & \text{if } c \text{ is a pair} \\ -d(\mathbf{o}^1, \mathbf{o}^2) & \text{if } c \text{ is a non-pair} \end{cases} \tag{4}$$

with $d(\cdot)$ a distance function of choice. We use a squared Euclidean distance as distance function:

$$d(\mathbf{o}^1, \mathbf{o}^2) = \sum_{j=1}^{n_c} (\mathbf{o}_j^1 - \mathbf{o}_j^2)^2. \tag{5}$$

We then optimize the following objective as a function of

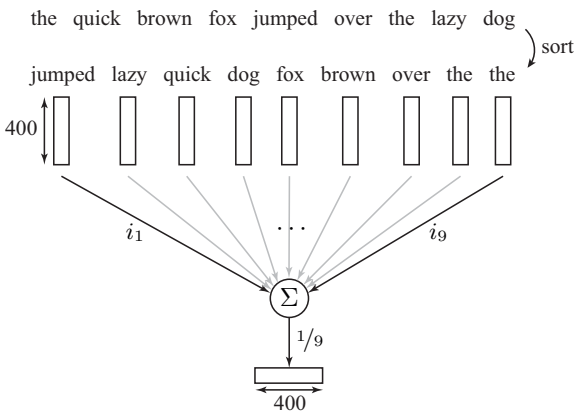

Figure 3. Illustration of the importance factor approach for a toy sentence of nine words long.

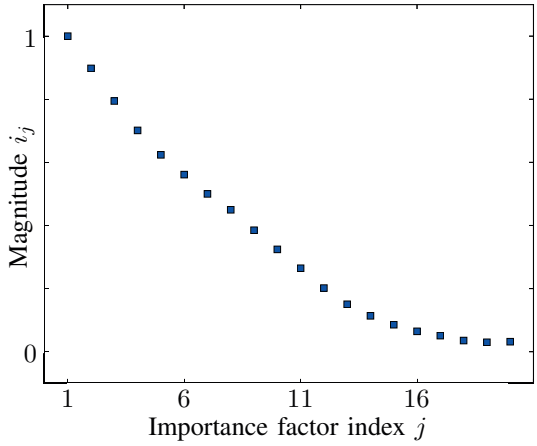

Figure 4. Plot of the importance factor magnitudes.

the importance factors:

$$J(i_1, \ldots, i_{n_c}) = \frac{1}{|\mathcal{D}|} \sum_{c \in \mathcal{D}} f(c) + \lambda \sum_{j=1}^{n_c} i_j^2. \qquad (6)$$

To minimize this objective function we use stochastic gradient descent with batches of 100 couples, a learning rate of 0.1, a momentum of 0.9, and a regularization constant λ of 0.0015. We start the optimization with all importance factors equal to 0.5. Thanks to the large amount of couples in the training set, we can stop the optimization after one epoch of training. By then, the factors have settled to an optimum and the procedure has seen all training couples exactly once, thereby reducing chances of overfitting.

Figure 4 shows a plot of the importance factors that were learned through the earlier-described optimization procedure. We clearly notice that the importance factors steadily decrease in magnitude; words with a low document frequency therefore weigh much more than words with a high document frequency, which confirms our hypothesis. The factors at the end are very close to zero.

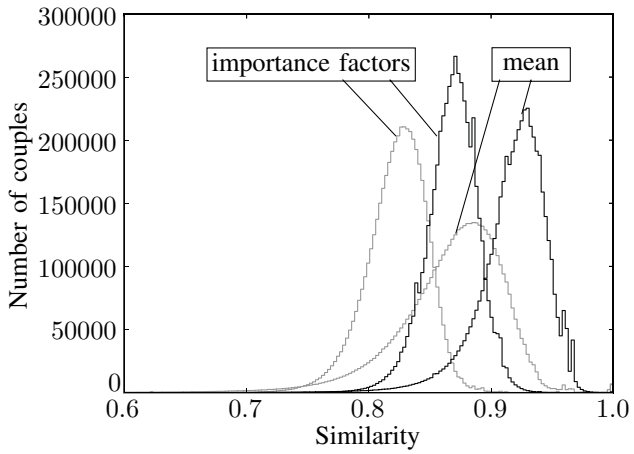

Figure 5. Comparison between mean embeddings and the importance factor approach, for both pairs (dark grey) and non-pairs (light grey).

To compare the performance of our importance factor approach to the earlier-described combination techniques, we calculate the optimal split point between pairs and non-pairs on our validation set for each of the techniques. Using these optimal split points, we then calculate the final split error rate on our test set. As a distance metric we use a normalized Euclidean distance, except for tf-idf for which we used the standard cosine distance. Table III shows the error rates for the different techniques we tested. We notice that the importance factor technique outperforms the other approaches by approx 2.0% in error rate. This is a significant decrease, as $p < 0.001$ in a two-tailed binomial test. We compare our importance factor approach with the plain mean technique on the complete dataset in Figure 5, in which we see that in our approach the variance of the curves is smaller and that there is less overlap between the pairs and non-pairs.

IV. Discussion and Conclusion

We gained insight in the power of tf-idf and several word embedding aggregation techniques to relate pairs of very short texts to each other. We also learned how to optimally combine knowledge from both tf-idf and word embeddings to maximize the separation between pairs and non-pairs. The best performing traditional technique is a concatenation of maximum and minimum vectors, and for a large number of words tf-idf produces comparable results. Our importance factor approach, however, significantly outperforms all other techniques by a large margin.

In this paper we have laid out the first steps towards a flexible and hybrid technique to combine word embeddings with tf-idf information. Since this is still ongoing research, a few remarks need, however, to be made. First, our experimental set-up is close to a toy setting. Wikipedia is a completely different textual medium than a social platform such as Twitter, in which the used language is full of slang, hashtags and spelling errors. In future work we will therefore

Table III
COMPARISON OF ERROR RATES ON THE TEST SET, INDICATING A SIGNIFICANT REDUCTION FOR THE IMPORTANCE FACTOR APPROACH.

	Error rate
Tf-idf	19.60%
Mean	19.43%
Max	19.05%
Min/max	16.78%
Mean, top 30% idf	17.02%
Max, top 30% idf	18.05%
Min/max, top 30% idf	16.40%
Mean, idf weighed	24.00%
Mean, importance factors	**14.44%**

adapt our novel technique to texts found in social media posts.

Secondly, our current approach is limited to texts of a fixed length, while all other techniques in Table III do not suffer from this restriction. This forms a strong constraint on the applicability of the technique. However, current research shows promise that the approach can be extended to texts of arbitrary length as well, but this still needs to be investigated further in future work.

In other future work we will discuss yet other combination schemes of word embeddings and tf-idf, and experiment with the number of dimensions in the embeddings. We will also investigate and compare the performance of LSI, topic models such as LDA, and other state-of-the-art document distance measures based on word embeddings.

V. ACKNOWLEDGMENTS

Cedric De Boom is funded by a Ph.D. grant of Ghent University, Special Research Fund (BOF) and of the Flanders Research Foundation (FWO).
Steven Van Canneyt and Steven Bohez are funded by a Ph.D. grant of the Agency for Innovation by Science and Technology in Flanders (IWT).

REFERENCES

[1] C. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*. Cambridge University Press, Apr. 2009.

[2] P. Achananuparp, X. Hu, and X. Shen, "The Evaluation of Sentence Similarity Measures," in *DaWaK 2008: International Conference on Data Warehousing and Knowledge Discovery*, Jul. 2008.

[3] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," in *Proceedings of NAACL HLT*, Apr. 2013.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *NIPS 2013: Advances in neural information processing systems*, Oct. 2013.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proceedings of Workshop at ICLR*, Jan. 2013.

[6] F. Godin, B. Vandersmissen, A. Jalalvand, W. De Neve, and R. Van de Walle, "Alleviating Manual Feature Engineering for Part-of-Speech Tagging of Twitter Microposts using Distributed Word Representations," in *Workshop on Modern Machine Learning and Natural Language Processing, NIPS 2014*, Oct. 2014.

[7] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional Neural Network Architectures for Matching Natural Language Sentences," in *NIPS 2014: Advances in Neural Information Processing Systems*, 2014, pp. 2042–2050.

[8] J. Weston, S. Chopra, and K. Adams, "#TagSpace: Semantic embeddings from hashtags," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[9] C. N. dos Santos and M. Gatti, "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts," in *COLING 2014, the 25th International Conference on Computational Linguistics*, Dublin, Jul. 2014, pp. 69–78.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *The Journal of Machine Learning Research*, vol. 12, Feb. 2011.

[11] L. Kang, B. Hu, X. Wu, Q. Chen, and Y. He, "A Short Texts Matching Method using Shallow Features and Deep Features," in *Third CCF Conference, NLPCC 2014*, Nov. 2014.

[12] X. Zhang and Y. Lecun, "Text Understanding from Scratch," *arXiv.org*, Feb. 2015.

[13] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *arXiv.org*, May 2014.

[14] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From Word Embeddings To Document Distances." *ICML*, pp. 957–966, 2015.

[15] G. Zheng and J. Callan, "Learning to Reweight Terms with Distributed Representations," in *the 38th International ACM SIGIR Conference*. New York, New York, USA: ACM Press, 2015, pp. 575–584.