Contextually Propagated Term Weights for Document Representation

Casper Hansen
University of Copenhagen
c.hansen@di.ku.dk

Christian Hansen
University of Copenhagen
chrh@di.ku.dk

Stephen Alstrup
University of Copenhagen
s.alstrup@di.ku.dk

Jakob Grue Simonsen
University of Copenhagen
simonsen@di.ku.dk

Christina Lioma
University of Copenhagen
c.lioma@di.ku.dk

ABSTRACT

Word embeddings predict a word from its neighbours by learning small, dense embedding vectors. In practice, this prediction corresponds to a semantic score given to the predicted word (or term weight). We present a novel model that, given a target word, redistributes part of that word's weight (that has been computed with word embeddings) across words occurring in similar contexts as the target word. Thus, our model aims to simulate how semantic meaning is shared by words occurring in similar contexts, which is incorporated into bag-of-words document representations. Experimental evaluation in an unsupervised setting against 8 state of the art baselines shows that our model yields the best micro and macro F1 scores across datasets of increasing difficulty.

KEYWORDS

Word embeddings, Contextual semantics, Document representation

ACM Reference Format:
Casper Hansen, Christian Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2019. Contextually Propagated Term Weights for Document Representation . In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Word embeddings represent words as elements in a learned vector space by mapping semantically similar words to nearby points (otherwise put, by *embedding* them nearby each other). By doing so, word embeddings adopt the *Distributional Hypothesis*, which posits that words appearing in the same contexts share semantic meaning [9]. An efficient and popular implementation for learning word embeddings is word2vec [11] and the skip-gram model. Given a target word, skip-gram is trained to predict the words in the context of that target word. This prediction practically corresponds to a score (or *weight*) given to the predicted word, which can be applied to a range of tasks.

We present a novel model[1] that, given a target word, redistributes part of that word's weight (that has been computed with word embeddings) back to words occurring in similar contexts as the target word. By doing so, our model aims to simulate how semantic meaning is shared by words occurring in the same context, by sharing (or *propagating*) the semantic scores computed for these words within the neighbourhood of contextually similar words. This propagation is incorporated into bag-of-words document representations. We experimentally evaluate our model in the task of unsupervised text clustering against 3 established and 5 state of the art baselines. Our model yields the best micro and macro F_1 scores on average across all datasets. In addition, our model is efficient and robust across datasets of different inter versus intra class ratio (i.e. across datasets of increasing difficulty).

2 RELATED WORK

A simple, yet often effective practice of outputting document scores from word embeddings is by the average of the embedding of each word in a text [8]. A disadvantage of this approach is that all words are weighted equally, failing to distinguish between more and less discriminative words. Arora et al. [1] proposed an unsupervised word embedding weighting scheme using word occurrence probabilities from a large corpus. Each embedded word is weighted by: $\alpha/(\alpha + p(w))$ with $p(w)$ being the probability of word w occurring, and α being a smoothing parameter. From each averaged vector the first singular vector is subtracted, which corresponds to removing common words (e.g. `just`, `when`, and `even`) [1]. The method of Arora et al. was defined and evaluated for sentences. Kusner et al. [7] recently presented the unsupervised word mover's distance (WMD), which approximates the semantic distance between two documents by computing the earth mover's distance between the embedded words in each document. While WMD experimentally outperformed several state of the art methods in the task of text clustering, its main disadvantage is that it is computationally very costly and cannot be readily used on medium to large scale datasets.

3 CONTEXTUALLY PROPAGATED TERM WEIGHTS

Given a word embedding [10], we define as the *embedded neighbourhood* of term w_j in document d_i, the set of all similar terms having cosine similarity to w_j at least τ in the embedding space (i.e. the most similar terms thresholded by a minimum cosine similarity of

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

[1]The code is available at: https://github.com/casperhansen/CPTW

τ), where τ is a tunable threshold value (discussed in Section 3.3)[2]. We use this embedded neighbourhood to define two contextual document representations, presented next.

3.1 CPTW

Let $N(w_j)$ be the set of words contained in the embedded neighbourhood of word w_j (note that this includes w_j itself). Then, for each $w_k \in N(w_j)$, we define:

$$\gamma(w_k) = f(w_k, d_i)\cos(v_j, v_k) \tag{1}$$

where $\gamma(w_k)$ is our contextually propagated term weight of w_k, $f(w_k, d_i)$ is the frequency of word w_k in document d_i, and $\cos(v_j, v_k)$ is the cosine similarity between the word embeddings for word w_j and w_k. Eq. 1 computes the term weight of w_j and of each word in w_j's embedded neighbourhood. Then, based on the above term weights, we compute the contextually propagated term weights (CPTW) of document d_i, denoted CPTW(d_i), as:

$$\text{CPTW}(d_i) = \sum_{j=1}^{M} e_j \Big(\alpha_j \sum_{w_k \in N(w_j)} \gamma(w_k) \Big) \tag{2}$$

where M is the total number of unique words in the collection (such that the representation resides in $\mathbb{R}^M$), e_j is the vector with 1 at index j and zero everywhere else, $\gamma(w_k)$ is the term weight of w_k computed using Eq. 1, and α_j is a normalization constant, computed as: $\alpha_j = \left(\sum_{w_k \in N(w_j)} \cos(v_j, v_k)\right)^{-1}$. In Eq. 2, α_j ensures that all words have the same total weight independent of the number of words they are similar to. This has the benefit that a word with a larger embedded neighbourhood (larger $N(w_j)$) is not weighted higher than a word with a smaller embedded neighbourhood (smaller $N(w_j)$). When the number of unique words is considered fixed then a τ-tresholded word-to-word similarity matrix can be computed offline, such that the similarity propagation of CPTW can be trivially done efficiently using sparse vector-matrix multiplications on a traditional bag-of-words representation. Fig. 1 shows an example of CPTW in practice.

3.2 CPTW$_{\text{IDF}}$

Eq. 1 approximates the weight of term w_k according to (i) the frequency of its occurrence in d_i ($f(w_k, d_i)$), and (ii) how similar w_k is to w_j ($\cos(v_j, v_k)$). The frequency of w_k in d_i can be artificially inflated for terms that occur very often in the collection, not just in d_i (just like TF in traditional bag of words computations when not combined with IDF). To counter this effect, we introduce the following variation of $\gamma(w_k)$ that includes an IDF-like component, thus ensuring that high within-document term frequency **and** term discriminativeness in the collection are considered when computing term weights:

$$\gamma_{IDF}(w_k) = \gamma(w_k) \log\left(\frac{N}{\text{df}(w_k)} \alpha_j \cos(v_j, v_k)\right) \tag{3}$$

where $\gamma(w_k)$ is computed as in Eq. 1, df(w_k) is the number of documents in the collection that contain word w_k, N is the total number of documents in the collection, and the rest of the notation is the same as above. Note that in this case α_j is placed inside the log

[2]Note that *embedded neighbourhood* is not the same as *embedding space*: the former is derived from the similarities measured in the latter.

1: The boat is sailing on the sea
2: The ship was cruising on the ocean
3: The cat was relaxing on the couch

	Euclidean distance		
	1↔2	1↔3	2↔3
BOW	0.40	0.40	0.35
CPTW	0.16	0.49	0.40

Figure 1: Example: Our similarity propagation reduces the Euclidean distance between semantically similar texts compared to bag-of-words with frequency weighting (BOW).

because we propagate the IDF component and still need the normalization for the same reason as above. We define the CPTW$_{\text{IDF}}$ of document d_i based on the above term weights as:

$$\text{CPTW}_{\text{IDF}}(d_i) = \sum_{j=1}^{M} e_j \Big(\alpha_j \sum_{w_k \in N(j)} \gamma_{IDF}(w_k) \Big) \tag{4}$$

The purpose of the IDF component is the same as in traditional TF-IDF; however, in our approach it is computed by propagating the IDF values for each word in the embedded neighbourhood as done similarly for term frequency in Eq. 2. This means that the IDF score for each word is based on a weighted sum of all the IDF scores of the words in its embedded neighbourhood, thus taking into account the differences in IDF scores between the words in the embedded neighbourhood.

3.3 Threshold parameter τ

The optimal value of τ depends on both (i) the quality of the word embedding (to avoid dissimilar words from being represented as similar), and (ii) the similarity of the texts in the collection. If the collection consists of texts with vastly different topics, then a low τ is preferred because there is little overlap of word semantics between texts with different topics. However, if the topics are similar, then τ should be higher to avoid including semi-related words that are shared across topics. If, for the sake of explanation, we assume that no thresholding is done, then each word contains all words in its embedded neighbourhood, and thus the value of each word in the bag-of-words vector would be the sum of cosine similarities to all other words. As the cosine similarity is a dot product of unit length vectors, the distributive property of the dot product entails that the sum of cosine similarities would be, in effect, the dot product of the embedded word vector and the sum of *all* other embedded word vectors. In order to not consider this sum of all other embedded word vectors, we choose τ to define a smaller neighbourhood such that the sum only consists of words that are deemed similar enough to not introduce noise (i.e. minuscule or negative dot products).

4 EXPERIMENTAL EVALUATION

We experimentally evaluate our contextually propagated term weights (CPTW) and CPTW$_{\text{IDF}}$ against strong baselines for related document representation and document distance methods that use few to zero parameters, applied to text clustering. For all methods we apply k nearest neighbour (kNN) classification for evaluation purposes, in order to purely focus on the representation that each method can generate to discriminative between texts.

4.1 Data

We use 7 openly available datasets commonly used in related work [5–7, 18]. As seen in Table 1, the datasets are largely varied with respect to their vocabulary size, number of unique words in each document, and number of classes.

Dataset	#docs	unique words	#avg. unique words ± std.	#classes
bbcsport	737	13106	116.4±55.9	5
twitter	3424	8405	8.9±3.3	3
classic	7095	23624	39.3±28.1	4
amazon	8000	39852	44.47±46.9	4
reuters	7674	23109	38.9±36.9	8
20news	18846	30465	84.1±96.9	20
wiki	19981	46610	474.9±411.7	25

Table 1: Data statistics.

4.2 Baselines and Tuning

We compare our CPTW and CPTW$_{\text{IDF}}$ models against bag-of-words with frequency weighting (BOW) [16], TF-IDF [16], BM25 [14], LSI [3], LDA [2], Word Embedding Averaging (WE-AVG) [8], Smooth inverse frequency weighting (SIF) [1], and Word Mover's Distance (WMD) [7]. To tune and evaluate all models we use 5-fold cross validation, where each fold acts as testing data once, while the rest is used for training and validation. In each iteration, we use 70% of the 4 folds for training and the remaining 30% for validation. We tune the parameters of all models using a grid search of the parameter space and repeat this 3 times in each iteration (the best parameters across the 3 iterations are chosen for the test fold). For *reuters*, *20news*, and *wiki* we use the existing splits.

We use the micro and macro versions of F$_1$ for evaluation, and for validation we use the micro F$_1$ (corresponding to traditional accuracy) for choosing the best parameters. In all methods we search kNN's $k \in \{1, ..., 19\}$ as done in a similar setup by Kusner et al. [7], and also tune the following: For LSI and LDA the number of topics is searched in $\{10, ..., 1000\}$. For BM25 $k_1 \in \{1.0, ..., 2.0\}$ and $b \in \{0.5, ..., 1.0\}$. For SIF $\alpha \in \{10^{-2}, ..., 10^{-5}\}$. For CPTW and CPTW$_{\text{IDF}}$ $\tau \in \{0, ..., 1.0\}$. We use the best parameters found on the validation set when evaluating on the test set.

We use word embedding pretrained by the word2vec skip-gram model on Google News data[3]. For reproducibility purposes we use the gensim and scikit-learn python libraries[4] for all compared methods, and use the implementation of SIF provided by the authors[5]. We preprocess the datasets to remove non-alphanumeric characters, to tokenize text into lower cased word-tokens, and to remove stop words (based on the list by Salton [15]) as per [7].

4.3 Findings

Table 2 displays the performance (micro and macro F$_1$) of all methods on the 7 datasets. WMD scores are not shown for the *20news* and *wiki* datasets, because, after running WMD for 5 days, we stopped experiments with that model on the grounds of its poor efficiency. Due to this, in Table 2 we report the average scores separately for all datasets (all⊖WMD) and for the 5 datasets that WMD was run on (all⊕WMD).

CPTW$_{\text{IDF}}$ is the best performing method on both micro and macro F$_1$ on average across all datasets. CPTW is the second best performing method as per micro F$_1$ on all datasets, while WMD is the second best on the 5 datasets it could be run on. With respect to macro F$_1$, CPTW is the second best performing method on all datasets, and with WMD being the second best when only considering the 5 datasets WMD could be run on. When considering all datasets on macro F$_1$, then SIF was the best of the baselines, but with both CPTW and CPTW$_{\text{IDF}}$ performing better. The worst performing method is LDA, which is most likely because LDA works better on texts with a large number of words. Indeed LDA performs relatively well on *wiki* – the dataset with the longest texts on average – and noticeably worse than the other methods on most of the other datasets.

[3]https://code.google.com/archive/p/word2vec/
[4]https://radimrehurek.com/gensim/ and https://scikit-learn.org/stable/
[5]https://github.com/PrincetonML/SIF

4.4 Influence of τ

We further investigate the stability of the threshold τ of our model. Specifically, we investigate to what extent (or if at all) optimal values of τ correlate with a large inter versus intra class ratio in the embedded neighbourhood of a sample document (the intra versus inter class ratio indicates classification difficulty as explained next). We conduct this analysis only with CPTW$_{\text{IDF}}$, our best performing method in Table 2.

We define *inter vs intra class ratio* (IICR) as the ratio of average Euclidean distances from each point to its closest inter and intra class neighbours. Intra class neighbours refer to the points close to each other within the same class, and inter class neighbours refer to the points close to each other across different classes. The ratio of average Euclidean distances for the inter and intra class neighbours shows how similar the points from one class are compared to the other classes. We study how IICR varies when τ is changed, and compute IICR as an average over all classes in a dataset for a specific τ as follows:

$$\text{distance}_{c_1}^{\text{inter}} = \frac{1}{|c_1|} \sum_{v_1 \in c_1} \sum_{v_2 \in N_k(v_1, C-\{c_1\})} ||v_1 - v_2||_2 \quad (5)$$

$$\text{distance}_{c_1}^{\text{intra}} = \frac{1}{|c_1|} \sum_{v_1 \in c_1} \sum_{v_2 \in N_k(v_1, c_1)} ||v_1 - v_2||_2 \quad (6)$$

$$IICR = \frac{1}{|C|} \sum_{c_1 \in C} \frac{\text{distance}_{c_1}^{\text{inter}}}{\text{distance}_{c_1}^{\text{intra}}} \quad (7)$$

where C represents the set of classes, and $N_k(v_1, c_1)$ is the k closest points to v_1 from the c_1 class. An IICR score close to 1 means that inter and intra class samples are highly similar and thus hard to correctly classify. A higher IICR score than 1 means that inter and intra class samples are more dissimilar, which makes classification fundamentally easier. We consider the closest points, instead of all points, because the space should be transformed such that points become more similar to close intra class points, and become more dissimilar to the closest inter class points. We expect a large value of IICR to correlate well with an appropriate value of τ for a specific dataset, because we reason that our model should disambiguate semantically related and unrelated documents better. We choose the best performing k on each dataset.

We compute the IICR for all datasets when varying τ and plot it with varying τ (Fig. 2), where the legend in the graphs shows the optimal averaged τ across the cross validations per dataset. Generally, the graphs show that maximum ratios are correlated with τ values close to the optimal found in the cross validation, except for *wiki*. This follows our intuition of our model being able to disambiguate the documents such that intra class samples become more similar compared to inter class samples. This means that

F_1	BM25		BOW		TF-IDF		LDA		LSI		WE-AVG		WMD		SIF		CPTW		CPTW_{IDF}	
	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro
bbcsport	.970	.968	.980	.979	**.986**	**.987**	.864	.864	.976	.979	.915	.917	.976	.978	.927	.931	.980	.979	.985	.986
twitter	.711	.470	.715	.522	.728	.540	.642	.353	.721	.508	.721	.537	.704	.445	.714	.495	.724	.532	**.732**	**.550**
classic	.849	.860	.935	.935	.947	.950	.840	.845	.921	.920	.926	.933	**.963**	**.964**	.931	.937	.955	.958	.957	.960
amazon	.866	.866	.892	.893	.899	.898	.762	.761	.834	.832	.915	.915	.909	.910	.914	.914	.899	.900	**.918**	**.918**
reuters	.925	.859	.894	.818	.873	.814	.916	.833	.955	.864	.956	.885	**.961**	**.910**	.958	.888	.950	.877	.942	.894
20news	.467	.470	.650	.649	.661	.662	.488	.482	.565	.572	.641	.634	–	–	.644	.636	.661	.659	**.703**	**.699**
wiki	.392	.148	.396	.139	.395	.135	.389	.156	**.417**	.207	.407	.213	–	–	.409	.214	.407	**.217**	.403	.173
all⊖WMD	.740	.663	.780	.705	.784	.712	.700	.613	.770	.697	.783	.719	–	–	.785	.716	.797	.732	**.806**	**.740**
all⊕WMD	.864	.805	.883	.829	.887	.838	.805	.731	.881	.821	.887	.837	.903	.841	.889	.833	.902	.849	**.907**	**.862**

Table 2: Micro and macro F_1. all⊖WMD is the average score for all datasets, and all⊕WMD is the average score for all datasets where WMD could be run. Bold denotes best scores.

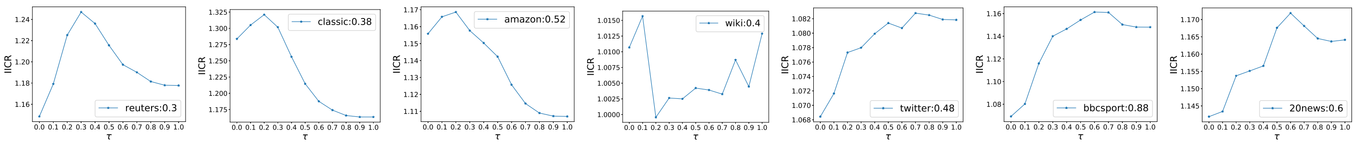

Figure 2: Each graph shows the IICR as a function of τ. The legend shows the best (average) τ from validation.

our approach was able to make documents of the same class more similar when compared to other classes. For *wiki* most τ values perform very similar for the best k, and when considering the ratios we see a similar trend where all ratios are nearly identical.

5 CONCLUSION

We presented *Contextually Propagated Term Weights*, or *CPTW* that propagate the weight of a word (computed via embeddings) to words occurring in similar contexts. The redistribution of weight has the effect of generalizing the semantics of a text, leading to improved discriminative power. CPTW has low computational cost: state of the art word embeddings are used that can be precomputed efficiently offline, and the propagation itself can be performed efficiently using sparse matrix multiplications based on a precomputed matrix of word similarities.

Experimental evaluation against 8 baselines on 7 well-known datasets shows that CPTW yields the best micro and macro F1 scores on average across all datasets. Most notably, CPTW outperforms strong embedding based methods such as word mover's distance (WMD) [7] and smooth inverse filtering (SIF) [1]; likewise, the experiments show that CPTW's notion of embedded neighbourhood is robust across datasets of different inter vs intra class ratio (i.e. across datasets of increasing classification difficulty). Using the ratio of average euclidean distance between inter and intra class kNN samples as a measure of classification difficulty of a dataset, we found that the value of the (sole) parameter τ in CPTW chosen by cross validation was close to the τ maximizing this ratio.

Our work complements recent efforts to employ word embeddings in novel ways that are both (i) computationally efficient, and (ii) semantics-sensitive in that discriminative power is increased by exploiting semantic [12, 13, 17] and structural [4] similarities. Along these lines, future research directions include investigating whether, and to what extent, multiple similarity thresholds may improve the overall discriminative power of the method.

ACKNOWLEDGMENTS

Partly funded by Innovationsfonden DK, DABAI (5153-00004A).

REFERENCES

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.
[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.
[3] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41, 6 (1990), 391–407.
[4] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. 2019. Neural Speed Reading with Structural-Jump-LSTM. In *ICLR*.
[5] Ganesh J, Manish Gupta, and Vasudeva Varma. 2016. Doc2Sent2Vec: A Novel Two-Phase Approach for Learning Document Representation. In *SIGIR*. ACM, 809–812.
[6] Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *CIKM*. ACM, 1411–1420.
[7] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From Word Embeddings To Document Distances. In *ICML*. 957–966.
[8] Guy Lev, Benjamin Klein, and Lior Wolf. 2015. In defense of word embedding for generic text representation. In *International Conference on Applications of Natural Language to Information Systems*. Springer, 35–50.
[9] Christina Lioma, Jakob Grue Simonsen, Birger Larsen, and Niels Dalum Hansen. 2015. Non-Compositional Term Dependence for Information Retrieval. In *SIGIR*. 595–604.
[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.
[12] Navid Rekabsaz, Mihai Lupu, and Allan Hanbury. 2017. Exploration of a threshold for similarity based on uncertainty in word embedding. In *ECIR*. 396–409.
[13] Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Guido Zuccon. 2016. Generalizing translation models in the probabilistic relevance framework. In *CIKM*. ACM, 711–720.
[14] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109–126.
[15] Gerard Salton. 1971. The SMART retrieval system-experiments in automatic document processing. *Englewood Cliffs* (1971).
[16] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.
[17] Dongsheng Wang, Quichi Li, Lucas Chaves Lima, Jakob Grue Simonsen, and Christina Lioma. 2019. Contextual Compositionality Detection with External Knowledge Bases and Word Embeddings. In *WWW International Workshop of Deep Learning for Graphs and Structured Data Embedding*.
[18] Dell Zhang, Jun Wang, Emine Yilmaz, Xiaoling Wang, and Yuxin Zhou. 2016. Bayesian performance comparison of text classifiers. In *SIGIR*. 15–24.