Improving Short Text Clustering by Similarity Matrix Sparsification

Md Rashadul Hasan Rakib
Dalhousie University
Halifax, Nova Scotia, Canada
rakib@cs.dal.ca

Magdalena Jankowska
Dalhousie University
Halifax, Nova Scotia, Canada
jankowsk@cs.dal.ca

Norbert Zeh
Dalhousie University
Halifax, Nova Scotia, Canada
nzeh@cs.dal.ca

Evangelos Milios
Dalhousie University
Halifax, Nova Scotia, Canada
eem@cs.dal.ca

ABSTRACT

Short text clustering is an important but challenging task. We investigate impact of similarity matrix sparsification on the performance of short text clustering. We show that two sparsification methods (the proposed Similarity Distribution based, and k-nearest neighbors) that aim to retain a prescribed number of similarity elements per text, improve hierarchical clustering quality of short texts for various text similarities. These methods using a word embedding based similarity yield competitive results with state-of-the-art methods for short text clustering especially for general domain, and are faster than the main state-of-the-art baseline.

CCS CONCEPTS

• **Computing methodologies → Natural language processing**; **Machine learning**;

KEYWORDS

Short Text Clustering, Similarity Matrix Sparsification, Word Embeddings

ACM Reference Format:
Md Rashadul Hasan Rakib, Magdalena Jankowska, Norbert Zeh, and Evangelos Milios. 2018. Improving Short Text Clustering by Similarity Matrix Sparsification. In *DocEng '18: ACM Symposium on Document Engineering 2018, August 28–31, 2018, Halifax, NS, Canada.* 4 pages. https://doi.org/10.1145/3209280.3229114

1 INTRODUCTION

Due to the advancement of technologies, short texts are generated at a large volume every day from different sources such as microblogging, question-answering, social news aggregation websites, etc. Therefore, proper organization of such text[1] collections is important as a way for gaining meaningful information by their users as well as a preliminary step for various text mining tasks.

Short text clustering is a well known and popular research topic in the area of text mining. The purpose of short text clustering is to support tasks such as organizing texts, discovering and visualizing topics, searching for and filtering information. Given a corpus of short texts, the task is to divide it into separate groups of similar objects. The task is challenging due to the short length of texts, which makes traditional text similarities, based on frequency or tf-idf scores of text terms, not adequate [12].

In this paper we investigate similarity matrix sparsification in the context of clustering of short texts. A similarity matrix contains similarity scores for all text pairs. Sparsification of a similarity matrix removes similarity for some pairs, with the aim of improving clustering results. Texts are clustered by an algorithm based on a similarity matrix; we choose hierarchical agglomerative clustering [9]. The major contributions of this paper are as follows:

- We propose a new similarity matrix sparsification method: the Similarity Distribution based, improving short text clustering results using various similarity measures.
- We investigate impact of various sparsification methods on short text clustering.
- We show that two sparsification methods retaining a prescribed number of similarities per text: the Similarity Distribution based and k-nearest neighbors, applied on a word embedding based similarity, lead to short text clustering competitive with state-of-the-art short text clustering methods.

2 RELATED WORK

2.1 Related Work on Short Text Clustering

STC2-LE is a short text clustering method based on word embeddings and convolutional neural network (CNN) [12]. It uses CNN to learn text representation, on which then clustering is performed.

Biterm topic modeling (BTM) is a topic modeling approach for short texts, which has been used for short text clustering [2]. It learns topics from word co-occurrence patterns (i.e., biterms). Given a topic distribution produced by BTM for each text, clustering is performed by assigning a text to its most probable topic.

DocEng '18, August 28–31, 2018, Halifax, NS, Canada
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5769-2/18/08…$15.00
https://doi.org/10.1145/3209280.3229114

[1]In this paper we use the term "text" and "document" interchangeably.

The topic diffusion method clusters short texts after corpus-based self enrichment [14]. This method adds new words to a short text, based on posterior probabilities of those new words, given all words in that text.

A method based on locality-sensitive term weighting is introduced in [15]. Distances between texts are calculated using weights of terms (e.g., words); the weights are obtained based on the property that, similar terms are tight together in terms of their own locality and well-separated from other localities.

2.2 Related Work on Matrix Sparsification

One simple method of similarity matrix sparsification is to use a global threshold and remove all similarities that are below it [7]. The problem with this method is that some real clusters can be removed, because texts in different clusters may have different similarity levels [7]. Several nearest neighbors based methods have been utilized for similarity matrix sparsification, such as the k-nearest neighbors [5] and shared-nearest neighbors [6] methods. The k-nearest neighbors sparsification ignores the similarity values that are not within the top k similarities of a text; the shared-nearest neighbors approach adds a condition that texts retaining similarity values with a particular text should share a prescribed number of neighbors [7]. Another two sparsification methods relying on a center vector, obtained from all text vectors, are proposed in [3].

3 SIMILARITY MATRIX SPARSIFICATION

Sparsification of text similarity matrix keeps association between a text and its most similar (nearest) texts, while breaking associations with less similar ones [7]. We investigate the potential of various methods of sparsification, listed below, for improving short text clustering. A square $n \times n$ symmetric similarity matrix $S = (s_{ij})$ is an input to each sparsification method (n is the number of texts, s_{ij} is the similarity score between texts t_i and t_j). Some methods require additional inputs or parameters. Each sparsification method listed below retains some original values of similarities, while replacing remaining ones by zeros (self-similarities on the diagonal are always retained). A sparsification criterion may render a matrix not symmetric. Such a matrix requires symmetrization: we follow the "sparsification with exclusion" [7] approach, namely in the final matrix an element s_{ij} is set to zero only if the sparsification criterion retains neither s_{ij} nor s_{ji}.

3.1 Similarity Distribution based sparsification

3.1.1 Definition. The Similarity Distribution based (SD) sparsification is a new sparsification method that we propose. In contrast to the k-nearest neighbors method, the number of similarities to keep for each text is not fixed, instead it is based on the distribution of the similarity values between the text and all other texts. The parameter of the method, called l, is the average number of retained similarities with other texts per text (i.e., the average number of non-zero matrix elements outside of the diagonal per row) in the final, symmetric sparsified matrix.

For each text t_i, we calculate mean u_i and standard deviation δ_i of similarities between t_i and all other texts, and we sparsify similarities between t_i and other texts based on these statistics. In particular, we define the retaining criterion as follows: a similarity s_{ij} is to be retained if and only if

$$s_{ij} > u_i + \alpha\delta_i, \tag{1}$$

for some global factor α, otherwise it is to be replaced by zero.

Factor α is such that after applying the criterion and symmetrization of the matrix, the average number of non-zero elements outside of the diagonal per row is equal to the value of parameter l.

3.1.2 Algorithm. We design an algorithm to perform the SD sparsification through a search for an exact value of factor α.

For each similarity value s_{ij} in matrix S, we use an auxiliary value $a_{ij} = \frac{s_{ij}-u_i}{\delta_i}$. Using this notation, our criterion from Eq. 1 can be stated as follows: a similarity s_{ij} is to be retained if and only if $a_{ij} > \alpha$. Since we follow "sparsification with exclusion" approach for symmetrization, we will keep s_{ij} in the final symmetric matrix if the retaining criterion is fulfilled either for s_{ij} or for s_{ji} (or for both). It follows that exactly when $max(a_{ij}, a_{ji}) > \alpha$, then both s_{ij} and s_{ji} are retained in the final sparsified matrix.

Let us also notice that the condition of the average number of non-zero elements outside of the diagonal per row in the final, symmetric matrix is to be l, is equivalent to the condition that the total number of non-zero elements above the diagonal is $\left\lfloor \frac{n \times l}{2} \right\rfloor$.

Based on these observations, it can be seen that the following algorithm performs the SD sparsification. Auxiliary values a_{ij} are calculated for all elements in S. Then for each matrix element above the diagonal, i.e., for each pair of indices (i, j) such that $j > i$, the maximum of a_{ij} and a_{ji} is found, and stored in a list L; each value in L is associated with the corresponding pair of indices (i, j). List L is sorted and $\left\lfloor \frac{n \times l}{2} \right\rfloor$ largest values from L are retrieved. For each pair of indices (i, j) associated with the retrieved values, both s_{ij} and s_{ji} are retained in the final matrix; the remaining elements outside of the diagonal are set to 0. If factor α is needed explicitly, a value that is less than the smallest retrieved value from list L, and greater than or equal to the largest not retrieved value from list L, is a correct value of α.

3.2 k-Nearest Neighbors sparsification

The k-nearest neighbors (k-NN) method uses the number of nearest neighbors k as a parameter. The method criterion is to retain, for each text, exactly k highest similarities with this text outside of the diagonal. After this criterion is applied, symmetrization is required.

3.3 Center based sparsification

The *Center* method ($\bar{\phi}$) and the *Modified Center* method ($\hat{\phi}$) [3] are not parameterized. The Center method uses the mean vector c of all text vectors. A similarity between two texts t_i, t_j is retained if and only if it is higher than the maximum of the similarities between t_i, c, and t_j, c. The Modified Center method uses a modified version of vector c. No additional symmetrization is needed.

4 EXPERIMENTS

4.1 Datasets

Five datasets of short texts have been used. **SearchSnippets** is a dataset of results' snippets from Google search engine, containing

12340 snippets distributed into 8 groups [11]. **SearchSnippets-test** is a subset of the SearchSnippets dataset consisting of only test dataset. This dataset has 2280 snippets distributed into 8 groups. **StackOverflow** is a subset of the challenge data published in Kaggle[2], where 20000 question titles from 20 groups were randomly selected [12]. **AgNews** is a subset of a dataset of news titles [13]. It consists of 8000 texts in 4 topic categories (for each category we randomly selected 2000 texts). **BioMedical** is a subset of the challenge data published in BioASQ's website[3], where 20000 paper titles from 20 groups were randomly selected [12].

4.2 Experimental Settings

4.2.1 Similarity Measures. We perform experiments for five text similarity measures. **Word Embedding based** similarity is the cosine similarity for text vectors, which are averages of embeddings of words of a given text. We use three types of pretrained word embeddings (leading to three different similarities): *Glove embeddings*[4] trained by Glove method [10] on Wikipedia dumps, *Word2Vec embeddings*[5] trained by Word2Vec method [8] on Google News, and *BioASQ embeddings*[6] trained by Word2Vec method on the abstracts of biomedical publications. BioASQ similarity is applied only on BioMedical dataset. **GTM** [4] is a text similarity based on a word pair similarity that utilizes information of co-occurrence of the given two words in Google tri-gram corpus [1]. **Tf-Idf** similarity is cosine similarity for text vectors in the bag-of-words vector space model with term frequency-inverse document frequency weights.

4.2.2 Clustering of Similarity Matrices. Clustering based on a similarity matrix (without or with sparsification) is performed by hierarchical agglomerative clustering with Ward criterion using fastcluster implementation [9]. Ward criterion has been chosen because in our preliminary experiments it performs best comparing to other linkage methods. Number of clusters for a dataset is set to the number of classes (different labels) in the data.

4.2.3 Number of Retained Elements in Similarity Matrix. The SD method and the k-NN method, described in Section 3, each require a parameter related to the number of retained elements per row (parameter l of the former method is the average number of retained elements outside of the diagonal per row in the final matrix; parameter k of the latter method is the fixed number of retained elements outside of the diagonal before symmetrization).

We set these parameters based on an intuition that the number of similarities to retain per text can depend on the average number of similarities between a text and other texts in the same cluster. If n is the number of texts, and C is the number of clusters to produce, then, on average, the number of similarities between a text and other members of the same cluster will be $M = n/C - 1$. We perform experiments with setting the parameter of each method to M and to $2M$. Both methods performed better for the latter value in most similarity/dataset combinations. Thus we fixed parameters l and k of these two methods to $2M$ (as defined above, M depends on the number of texts and the number of clusters).

[2]https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow/download/train.zip
[3]http://participants-area.bioasq.org/
[4]http://nlp.stanford.edu/data/glove.42B.300d.zip
[5]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
[6]bioasq.lip6.fr/tools/BioASQword2vec/

4.3 Results

4.3.1 Impact of Similarity Matrix Sparsification. We first evaluate whether considered sparsification methods improve clustering results, using accuracy (ACC) and normalized mutual information (NMI) as evaluation measures (as in [12]). We consider a sparsification method to improve clustering for a given combination of dataset and similarity, if both evaluation measures increase after the method is applied to the similarity matrix.

The Center method does not improve clustering for any combination of dataset and similarity, and often decreased evaluation measures considerably. The Modified Center method rarely improves clustering only in 3 out of 16 dataset/similarity combinations[7] and is never the best-performing sparsification.

Results of the sparsification methods are in the top part of Table 1. The best result (ACC, NMI) for each combination of dataset and similarity is denoted bold. We can observe that the SD method and the k-NN method improve clustering performance in most cases (in all except for 2 and 6 dataset/similarity combinations, respectively). For datasets other than BioMedical, the best hierarchical clustering results are obtained with the SD sparsification on Glove similarity. For BioMedical data, the best hierarchical clustering results are achieved also with the SD sparsification, but with BioASQ similarity.

4.3.2 Comparison with State-of-the-Art Methods. Table 1 (the bottom part) shows clustering performance (ACC and NMI) by four state-of-the-art methods for short text clustering, described in Section 2: STC2-LE [12], Corpus-based Topic Diffusion [14], Locality-Sensitive Term Weighting [15] and BTM [2]. For BTM, we perform experiments ourselves, and we report the best results over parameter search for two parameters of the method, α and β. For the other three state-of-the-art methods, we list previously published results [12, 14, 15] for datasets, for which they are available, because an implementation is either unavailable or (for STC2-LE) it can not be run on a new set.

For datasets other than BioMedical, results by the SD and k-NN methods with Glove similarity are competitive with state-of-the-art methods. Namely, the SD method with Glove similarity produces the best NMI on these sets, and it produces ACC that is best on three sets (on SearchSnippets-test it is slightly lower than that by the Topic Diffusion method). The results of the k-NN method with Glove similarity are on those sets also usually higher than those of state-of-the-art methods (the exceptions are ACC on SearchSnippets-test and NMI on SearchSnippets).

For BioMedical data, the SD and k-NN sparsification methods perform best with BioASQ similarity. With respect to both ACC and NMI, they are outperformed on this set by STC2-LE.

We compare run-time of clustering with the SD and k-NN sparsification with that of STC2-LE. The sparsification based methods are much faster than STC2-LE by one (SD) or two (k-NN) orders of magnitude, as shown in Table 2.

[7]Center based methods cannot be applied on GTM similarity, because they require vector representation of texts, which is not produced by GTM.

Table 1: Accuracy and NMI for short text clustering. The top part is for hierarchical clustering using similarity matrix, with the best result for each combination of similarity and dataset bold. The bottom part is for state-of-the-art previous methods for short text clustering. The best overall result for each dataset is shaded.

Method			SearchSnippets-test		SearchSnippets		StackOverflow		AgNews		BioMedical	
Hierarchical Agglomerative Clustering	Similarity	Sparsification	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
	Glove	Without	0.7706	0.7099	0.7654	0.5941	0.6164	0.5446	0.7656	0.5282	0.3536	0.2901
		SD	**0.8947**	**0.7873**	**0.8269**	**0.6376**	**0.6480**	**0.5948**	**0.8184**	**0.5457**	0.3690	0.3071
		k-NN	0.8714	0.7642	0.7908	0.6051	0.5811	0.5406	0.7683	0.5243	**0.3965**	**0.3155**
	Word2Vec	Without	0.6653	0.6209	0.6448	0.4725	0.2996	0.2589	0.7427	0.4574	0.2383	0.1731
		SD	**0.8096**	**0.6860**	**0.7610**	**0.5496**	0.3404	**0.2873**	**0.7887**	**0.5255**	0.2638	0.2096
		k-NN	0.7531	0.6614	0.7201	0.5066	**0.3470**	0.2792	0.7843	0.4931	**0.2803**	**0.2128**
	Tf-Idf	Without	0.2719	0.2683	0.4201	0.2988	0.3736	0.2909	0.4381	0.2375	0.2677	**0.2430**
		SD	**0.4763**	**0.4349**	**0.5180**	**0.3612**	**0.4389**	**0.3198**	**0.6621**	**0.3716**	**0.3041**	0.2410
		k-NN	0.4442	0.3759	0.4863	0.3401	0.4190	0.2913	0.6043	0.3478	0.2941	0.2264
	GTM	Without	0.7842	0.6854	0.7126	0.4881	**0.6334**	0.5864	0.5577	0.3248	0.3513	0.2929
		SD	**0.8153**	**0.6933**	**0.7136**	**0.4903**	0.6285	**0.5944**	**0.7197**	**0.3847**	**0.3750**	**0.3042**
		k-NN	0.8061	0.6808	0.6833	0.4557	0.6261	0.5601	0.6083	0.3730	0.3582	0.2931
	BioASQ	Without	x	x	x	x	x	x	x	x	0.3886	0.3101
		SD	x	x	x	x	x	x	x	x	**0.4013**	**0.3351**
		k-NN	x	x	x	x	x	x	x	x	0.3975	0.3219
state-of-the-art	STC2-LE		x	x	0.7709	0.6316	0.5114	0.4903	x	x	0.4362	0.3805
	Topic Diffusion		0.9001	0.5009	x	x	x	x	x	x	x	x
	Loc.-Sens. Term Weight.		0.8751	0.5824	x	x	x	x	x	x	x	x
	BTM		0.7267	0.5905	0.5087	0.3520	0.3350	0.2447	0.5368	0.3117	0.2885	0.2091

Table 2: Run-time in seconds of sparsification based clustering with Glove embeddings and STC2-LE

Method	SearchSnippets	StackOverflow	BioMedical
SD	137	648	613
k-NN	54	67	63
STC2-LE	8277	8401	8594

5 CONCLUSIONS AND FUTURE WORK

We have shown that similarity matrix sparsification by our new proposed SD (Similarity Distribution based) method and by the k-NN (k-nearest neighbors) method improve short text clustering results for various similarity measures in most cases. The most promising results are obtained with word embedding based similarities: for sparsification based clustering, the best results are obtained using the SD sparsification with either Glove embedding or biomedical word embedding similarity. The results of clustering with the SD and k-NN sparsifications with word embeddings based similarities are competitive with state-of-the-art short text clustering methods. It is only on one dataset, the set of biomedical data, that these sparsification based methods are outperformed with respect to both ACC and NMI by another system STC2-LE; however this neural network based system is more time expensive.

In future work we will consider ways of improving the sparsification based short text clustering of special texts such as biomedical. We intend to investigate the use of biomedical terms (words, multi-word phrases) in documents as a possible approach. We also plan a rigorous analysis of the parameters of the SD and k-NN methods and a recommendation of the best way of parameter selection for these methods.

REFERENCES

[1] T. Brants and A. Franz. 2006. Web 1T 5-gram corpus version 1.1. *Linguistic Data Consortium* (2006).

[2] X. Cheng, X. Yan, Y. Lan, and J. Guo. 2014. BTM: Topic Modeling over Short Texts. *IEEE Transactions on Knowledge and Data Engineering* 26, 12 (2014), 2928–2941.

[3] T. Gollub and B. Stein. 2010. Unsupervised Sparsification of Similarity Graphs. In *Classification as a Tool for Research.* 71–79.

[4] A. Islam, E. Milios, and V. Kešelj. 2012. Text similarity using google tri-grams. In *Proceedings of the 25th Canadian conference on Advances in Artificial Intelligence.* 312–317.

[5] A. K. Jain and R. C. Dubes. 1988. *Algorithms for Clustering Data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[6] R. A. Jarvis and E. A. Patrick. 1973. Clustering Using a Similarity Measure Based on Shared Near Neighbors. *IEEE Trans. Comput.* 22, 11 (1973), 1025–1034.

[7] V. Kumar. 2000. *An Introduction to Cluster Analysis for Data Mining.* Technical Report. Dept. of Computer Science, Univ. of Minnesota, Minneapolis, MN. https://www-users.cs.umn.edu/~hanxx023/dmclass/cluster_survey_10_02_00.pdf

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013).

[9] D. Müllner. 2013. fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software* 53, 9 (2013), 1–18.

[10] J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.* 1532–1543.

[11] X. Phan, L. Nguyen, and S. Horiguchi. 2008. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. In *Proceedings of the 17th International Conference on World Wide Web.* 91–100.

[12] J. Xu, B. Xu, P. Wang, S. Zheng, G. Tian, J. Zhao, and B. Xu. 2017. Self-Taught convolutional neural networks for short text clustering. *Neural Networks* 88 (2017), 22 – 31.

[13] Xiang Z. and Yann L. 2015. Text Understanding from Scratch. (2015). http://arxiv.org/abs/1502.01710

[14] C. Zheng, C. Liu, and H. Wong. 2018. Corpus-based topic diffusion for short text clustering. *Neurocomputing* 275 (2018), 2444 – 2458.

[15] C. Zheng, S. Qian, W. Cao, and H. Wong. 2017. Locality-Sensitive Term Weighting for Short Text Clustering. In *Neural Information Processing.* 434–444.