Enhancement of Short Text Clustering by Iterative Classification

Md Rashadul Hasan Rakib^(✉), Norbert Zeh, Magdalena Jankowska,
and Evangelos Milios

Dalhousie University, Halifax, NS, Canada
{rakib,nzeh,jankowsk,eem}@cs.dal.ca

Abstract. Short text clustering is a challenging task due to the lack of signal contained in short texts. In this work, we propose iterative classification as a method to boost the clustering quality of short texts. The idea is to repeatedly reassign (classify) outliers to clusters until the cluster assignment stabilizes. The classifier used in each iteration is trained using the current set of cluster labels of the non-outliers; the input of the first iteration is the output of an arbitrary clustering algorithm. Thus, our method does not require any human-annotated labels for training. Our experimental results show that the proposed clustering enhancement method not only improves the clustering quality of different baseline clustering methods (e.g., k-means, k-means++, and hierarchical clustering) but also outperforms the state-of-the-art short text clustering methods on several short text datasets by a statistically significant margin.

Keywords: Short text clustering · Outlier removal · Iterative classification

1 Introduction

Due to technological advances, short texts are generated at large volumes from different sources, such as micro-blogging, question answering, and social news aggregation websites. Organizing these texts is an important step towards discovering trends (e.g., political, economic) in conversations and in other data mining tasks, such as data summarization, frequent pattern analysis, and searching for and filtering information. Clustering the texts into groups of similar texts is the foundation for many of these organization strategies [1].

The lack of signal contained in short texts makes grouping of short texts based on shared topics difficult, leading to poor cohesion of texts assigned to the same cluster. The objective of our research is to improve the cohesion of clusters in a cluster partition produced by an arbitrary baseline clustering method. To achieve this, we remove outliers from each cluster and reassign them to clusters with which they have greater similarity. We demonstrate that this approach produces more accurate cluster partitions than computationally more costly state-of-the-art short text clustering methods based on neural networks [2,3].

The k-means algorithm can be viewed as an iterative classification algorithm. Starting with an initial (random) cluster partition, each iteration computes the centers of all clusters and uses these cluster centers as a classifier to reassign every input point to a new cluster. k-means-- [5] is a variation of k-means that achieves improved clustering performance by removing outliers before computing cluster centers, that is, before “training the classifier”. The classification step then assigns *all* points to their closest cluster centers, including the outliers ignored when computing cluster centers. Our method is inspired by this approach but uses a more sophisticated classifier than computing cluster centers and assigning every point to the closest cluster center. Specifically, our method follows the approach of [6] to train a classifier based on the cluster labels of the non-outliers. Iterative classification then uses the trained classifier to reassign outliers to clusters. Just as with k-means, the resulting set of clusters is the input for the next iteration or, if this is the last iteration, the final set of clusters returned by the algorithm.

Iterative classification can be applied to any set of initial clusters and is thus independent of the method used to obtain these clusters. The quality of the final set of clusters, however, does depend on the method used to compute the initial clusters. We use k-means [7], k-means-- [5] and hierarchical clustering [7] using dense and sparse similarity matrices to compute the initial clusters. k-means and k-means-- clustering are applied to the vector representations of the texts. For hierarchical clustering, we use the text similarity matrix (dense or sparse). The dense similarity matrix stores the similarity value for each text pair, whereas the sparse similarity matrix keeps a certain number of similarity values and discards the remaining ones (sets them to 0) [8].

Matrix sparsification can be performed using different criteria for choosing the values to discard. We consider two approaches here, one based on k -nearest neighbors [7] and the other based on the similarity distribution [9]. The k -nearest neighbor method keeps the k largest entries in each row. In the similarity distribution-based method, the number of similarities to keep in each row is not fixed. Instead, it is based on the distribution of the similarity values in each row, as characterized by the mean and standard deviation of these values. These sparsification methods are discussed in detail in Sect. 4.

The two main contributions of this work are as follows:

- We introduce iterative classification as a method that improves the clustering quality of different baseline clustering methods on various short text datasets and does not require human-annotated data to train the classification model. Our implementation of iterative classification and the datasets used in our experiments are publicly available.¹
- The combination of hierarchical clustering (using a sparse similarity matrix based on similarity distribution [9]) and iterative classification performs better than other clustering methods combined with iterative classification. This combination outperforms the state-of-the-art short text clustering methods by a statistically significant margin.

¹ <https://github.com/rashadulrakib/short-text-clustering-enhancement>.

2 Related Work

2.1 Short Text Clustering

A major challenge in short text clustering is the sparseness of the vector representations of these texts resulting from the small number of words in each text. Several clustering methods have been proposed in the literature to address this challenge, including methods based on text augmentation [10,11], neural networks [2,3], topic modeling [12], and Dirichlet mixture model [4].

A recent method based on text augmentation [11] uses topic diffusion to augment each short text by finding words not appearing in the text that are related to its content. To find related words, this method determines possible topics for each text using the existing words. Then new words are added to each text; these new words are closely related to the text's topics based on the posterior probabilities of the new words given the words in the text. An earlier text augmentation method [10] finds Wikipedia articles using the short text as query string and uses the articles' titles as features.

A short text clustering method based on word embedding and a convolutional neural network called STC2-LE was proposed in [2]. It uses a convolutional neural network to learn a text representation on which clustering is performed. Another short text clustering method based on weighted word embedding and autoencoder was proposed in [3]. For each text, it calculates the average of the weighted embeddings [13] of its words. The weight of a word is calculated based on its inverse frequency in the corpus [3] which is then multiplied with its embedding to obtain weighted word embedding. After that, the embeddings of the texts are feed into an autoencoder to obtain the low dimensional representation of the texts on which clustering is performed.

Biterm topic modeling (BTM) [12] is a topic modeling approach for short texts that 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.

A short text clustering method based on a Dirichlet process multinomial mixture model called GSDPMM was proposed in [4]. GSDPMM does not partition the input into a pre-specified number of clusters. It processes the texts one by one and assigns each text to a new cluster or to one of the existing clusters based on two factors: a) a preference for a cluster with more texts and, b) a preference for a cluster whose texts share more words with the current text.

2.2 Similarity Matrix Sparsification

Sparsification of the text similarity matrix keeps the association between a text and its most similar (nearest) texts while breaking associations with less similar ones by setting the corresponding similarity scores to 0 [8]. Several similarity matrix sparsification methods have been discussed in the literature, including ones based on a global threshold [7], nearest neighbors [7], and center vectors [8].

Similarity matrix sparsification based on global threshold is the simplest sparsification method. It removes all similarity values that are below a given threshold [7]. The problem with this method is that some real clusters may be destroyed or merged because different clusters may have different similarity levels between the texts they contain.

Nearest neighbors' based methods for similarity matrix sparsification include k -nearest neighbor [7] and shared nearest neighbor [14]. k -nearest neighbor sparsification keeps only the k highest similarity scores for each text; the shared-nearest neighbor approach adds a condition that texts retaining similarity values with a particular text should share a prescribed number of neighbors.

A similarity matrix sparsification method based on the center vector was proposed in [8]. Texts are represented by *tf-idf* (term frequency-inverse document frequency) vectors and a center vector is computed by averaging these vectors. The sparsification of the similarity matrix is performed by removing similarities between all pairs of texts that are not more similar to each other than the maximum similarities of these two texts to the center vector.

3 Enhancement of Clustering by Iterative Classification

Given a collection of short texts and a partition of these texts into clusters, iterative classification modifies the given cluster partition by detecting outliers in each cluster and changing the clusters to which they are assigned. This is repeated several times, hence the term iterative in the method's name. In each iteration, we generate training and test sets containing non-outliers and outliers respectively. Then we train a classification algorithm using the training set and classify the test set using the trained model. This iterative process repeats until the stopping criterion discussed in Sect. 3.1 is satisfied. The details are shown in Algorithm 1 and are described next.

In each iteration, we choose a number P that roughly corresponds to the fraction of texts selected for the training set. P is chosen uniformly at random from an interval $[P_1, P_2]$ determined in Sect. 6.2. To generate the training set, we remove outliers from each of the K clusters defined by the current cluster labels L . To remove outliers, we use an outlier detection algorithm called Isolation Forest, which is applied to the *tf-idf* vector representations of the texts. The algorithm isolates the texts that exist in the low density region of the *tf-idf* feature space. If after removing outliers, a cluster contains more than $\frac{n}{K} \times P$ texts, then we remove texts from that cluster uniformly at random to reduce the number of texts in the cluster to $\frac{n}{K} \times P$. The reason of removing texts from each cluster is that we want each cluster to consist of roughly the same number of texts so as to reduce the bias of the classification algorithm. We add the removed texts to the test set and add the other texts to the training set. We train a classifier (Multinomial Logistic Regression) using the non-outliers and their cluster labels. Then we classify the texts in the test set using the trained classifier. This defines a new set of cluster labels of the texts in the test set and thus produces an updated cluster partition.

Algorithm 1. Enhancement of Clustering by Iterative Classification

Require: D = set of n texts, L = initial cluster labels of the texts in D , K = number of clusters

Ensure: Enhanced cluster labels of the texts

- 1: $maxIteration = 50$
 - 2: $avgTextsPerCluster = n/K$
 - 3: **for** $i = 1$ to $maxIteration$ **do**
 - 4: Choose a parameter P uniformly at random from the interval $[P_1, P_2]$. (P_1 and P_2 are parameters determined in Section 6.2. P bounds the fraction of texts kept per cluster.)
 - 5: Remove outliers from each of the K clusters defined by L using an outlier detection algorithm.
 - 6: If a cluster contains more than $avgTextsPerCluster \times P$ texts, remove texts from that cluster uniformly at random so that exactly $avgTextsPerCluster \times P$ texts remain in the cluster.
 - 7: $testSet$ = texts removed in Steps 5 and 6
 $trainingSet$ = all the texts not in $testSet$
 - 8: Train a classifier using the $trainingSet$ and classify the texts in $testSet$. This assigns a new cluster label $L(t)$ to each text $t \in testSet$.
 - 9: Stop iterative classification if the per cluster text distribution becomes stable (as described in Section 3.1).
 - 10: **end for**
 - 11: **return** L
-

3.1 Stopping Criterion for Iterative Classification

Iterative classification stops when it reaches the maximum number of iterations (i.e., 50) or the sizes of the clusters become stable. Let $C_1, \dots, C_k$ and $C'_1, \dots, C'_k$ be the clusters before and after an iteration, respectively. We consider the cluster sizes to be stable if

$$\frac{1}{k} \sum_{i=1}^k ||C'_i| - |C_i|| \leq 0.05 \frac{n}{k}$$

For example, consider the problem of partitioning 100 texts into two clusters. Then the average cluster size is 50. If one iteration assigns 48 texts to the first cluster and 52 texts to the second cluster and the next iteration assigns 49 and 51 texts to these clusters, respectively, then the average absolute change of the cluster size is $\frac{1}{2}(|48 - 49| + |52 - 51|) = 1$. Since this is less than 5% of the average cluster size (50), we consider the cluster sizes to have stabilized.

4 Similarity Matrix Sparsification

4.1 k -Nearest Neighbor Sparsification

The k -nearest neighbor (k -NN) sparsification method [7] uses the number of nearest neighbors k as a parameter. A square $n \times n$ symmetric similarity matrix $S = (s_{ij})$ is the input for k -NN sparsification method. The method criterion is

to retain, for each text, exactly the k highest similarities with this text outside of the diagonal. For the text t_i , we retain a similarity (s_{ij}) between t_i and other text t_j , if s_{ij} is within the k highest similarities of t_i . However, the similarity s_{ji} between a text t_j and other text t_i may not be retained because s_{ji} may not be within the k highest similarities of t_j . Hence after applying this criterion, the resulting sparsified matrix can be a non-symmetric matrix. Therefore we symmetrize the sparsified similarity matrix by retaining both s_{ij} and s_{ji} , if any of the similarities among s_{ij} and s_{ji} is retained in the sparsified similarity matrix.

4.2 Similarity Distribution Based Sparsification

The similarity distribution based sparsification method was proposed in our previous work [9]. It sparsifies a similarity matrix based on the distribution of the similarity scores in the matrix. The input of this sparsification method is a symmetric similarity matrix for a set of n texts. The goal is to increase the signal-to-noise ratio in the matrix by keeping only the most significant similarity scores and setting less significant similarity scores to 0. Our criterion for setting entries to 0 may result in a non-symmetric matrix. Such a matrix requires symmetrization. We follow the *sparsification with exclusion* approach [7] which sets an item s_{ij} to zero only if the sparsification criterion retains neither s_{ij} nor s_{ji} .

In contrast to the k -nearest neighbor 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 each text and all other texts. For each text t_i , we calculate the mean μ_i and standard deviation σ_i of similarities between t_i and all other texts. Then, 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

$$s_{ij} > \mu_i + \alpha\sigma_i, \quad (1)$$

for some global factor α . The factor α is chosen so 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 $l = \frac{n}{K} - 1$. Note that if each cluster has exactly $\frac{n}{K}$ elements and we return exactly the similarity scores between elements in the same cluster, then l is the number of non-zero non-diagonal entries in each row.

To choose the retained similarity values efficiently, we use an auxiliary value $a_{ij} = \frac{s_{ij} - \mu_i}{\sigma_i}$ for each similarity value s_{ij} . This is s_{ij} 's deviation from the mean of row i normalized by the standard deviation of row i . The criterion of Eq. 1 can be restated as: a similarity s_{ij} is to be retained if and only if $a_{ij} > \alpha$. Since we follow the *sparsification with exclusion* approach for symmetrization, we keep s_{ij} in the final symmetric matrix if the retaining criterion is fulfilled for s_{ij} or for s_{ji} . Thus, if the average number of non-zero non-diagonal entries per row is to be l , we need to return $N = \lfloor \frac{n \times l}{2} \rfloor$ entries above the main diagonal, which is achieved by choosing α to be the N^{th} largest value in $\{\max(a_{ij}, a_{ji}) | 1 \leq i < j \leq n\}$.

5 Methods for Clustering of Short Texts

5.1 k -Means and k -Means--

k -means clustering [1] is used to cluster a collection of short texts into k clusters. First, k -means clustering initializes k centers, then it assigns each text to its closest center. Then the algorithm runs for a number of iterations. In each iteration, it recomputes the cluster centers using the texts assigned to each cluster and reassigns the texts to their closest centers. This iterative process continues until the algorithm reaches the maximum number of iterations or the cluster assignments becomes stable between two successive iterations.

k -means-- [5] is a variation of k -means clustering, in which outliers are removed in each iteration of the k -means clustering before recomputing the cluster centers. To detect outliers, short texts are ranked in decreasing order using their distances to their nearest cluster centers and the d (parameter for defining the total number of outliers) most distant texts are considered as outliers and removed from the clusters so that the cluster centers will become less sensitive to outliers. This has been confirmed to improve the clustering performance.

5.2 Hierarchical Clustering Using Dense or Sparse Similarity Matrix

Hierarchical agglomerative clustering uses a symmetric matrix storing pairwise similarities between documents. Such a matrix is dense if it stores a similarity between every pair of documents. The clustering method starts with each document in its own clusters and repeatedly merges pairs of *most similar* clusters until only k (the desired numbers of clusters) clusters remain.

A dense similarity matrix provides the most detailed information about pairwise text similarities but the lowest similarity scores can be considered noise in the sense that they suggest (tenuous) connections between texts that are almost guaranteed to belong to different clusters. Setting these similarities to 0 increases the separation between clusters and produces better clustering results. We consider two sparsification methods in our experiments: k -nearest neighbor and similarity distribution based, which are discussed in Sects. 4.1 and 4.2 respectively. We form clusters based on the two resulting sparse similarity matrices using the same hierarchical clustering method as discussed above.

6 Experiments

6.1 Datasets

We used five different datasets of short texts in our experiments. The basic properties of these datasets are shown in Table 1. **SearchSnippet** is a dataset of search results from Google’s search engine, containing 12340 snippets distributed into 8 groups [15]. **SearchSnippet-test** is a subset of the SearchSnippet dataset consisting of 2280 search snippets distributed into 8 groups. **AgNews** is a subset

of a dataset of news titles [16]. It consists of 8000 texts in 4 topic categories (for each category, we randomly selected 2000 texts). **StackOverflow** is a subset of the challenge data published on Kaggle², where 20000 question titles from 20 groups were randomly selected [2]. **BioMedical** is a subset of the challenge data published on the BioASQ’s website³, where 20000 paper titles from 20 groups were randomly selected [2].

Table 1. Summary of the short text datasets

Dataset	#Clusters	#Texts	Average #words/text
SearchSnippet	8	12340	17.03
SearchSnippet-test	8	2280	17.18
AgNews	4	8000	22.61
StackOverflow	20	20000	8.23
BioMedical	20	20000	12.88

6.2 Experimental Setup

Experimental Setup for Iterative Classification. We preprocessed the texts by removing stop words and converting them to lowercase. Then we transformed each text into the *tf-idf* vector representation for a given text collection.

Each iteration of the iterative classification algorithm picks some percentage P of each cluster as the training set and reassigns the remaining texts to clusters based on a classifier trained using this training set; P is chosen uniformly at the random from some interval $[P_1, P_2]$. To justify this approach and to determine optimal choices for P_1 and P_2 , we ran preliminary experiments using a representative dataset (SearchSnippet-test). Specifically, we considered choosing P uniformly at random from the interval $[P_1, P_2]$ or choosing a fixed percentage P in every iteration. For the former method, we determined the optimal combination of P_1 and P_2 ($P_1 = 0.5$ and $P_2 = 0.95$). For the later, we determined the optimal choice of P ($P = 0.6$). Choosing P uniformly at random from the interval $[0.5, 0.95]$ resulted in cluster accuracy of 82.21 for the representative dataset. Choosing a fixed percentage $P = 0.6$ in every iteration resulted in cluster accuracy of 80.25. Thus we chose $P_1 = 0.5$ and $P_2 = 0.95$ and chose P uniformly at random from this interval in all experiments.

Experimental Setup for Clustering. To perform clustering, we used the preprocessed texts described in Sect. 6.2. Then, texts were represented as vectors using pretrained word embeddings (i.e., Glove [17] and BioASQ [18]). The Glove

² <https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow/download/train.zip>.

³ <http://participants-area.bioasq.org/>.

embedding⁴ was trained using the Glove method [17] on Wikipedia dumps. The BioASQ embedding⁵ was trained using the Word2Vec method [13] on abstracts of biomedical publications. We used the Glove embedding for all datasets except the biomedical dataset since these datasets contained terms related to general domains such as search snippets. For the biomedical dataset, the BioASQ embedding was more appropriate due to its specific focus on biomedical terms.

We represented each text by the average of the vectors of all words in the text. Then, we applied the five different clustering methods described in Sect. 5 to the text vectors. For the k-means and k-means-- clustering algorithms, we used the text vectors as the points to be clustered. For hierarchical clustering, we constructed the dense similarity matrix by computing similarities between the vectors using cosine similarity for all the text pairs. After that, we sparsified the dense similarity matrix using the k -NN and similarity distribution based (SD) sparsification methods. Then we applied hierarchical agglomerative clustering using dense (HAC) and sparse similarity matrices (HAC- k -NN and HAC- SD).

6.3 Results

In our experiments, we use five datasets of short texts which are SearchSnippet, SearchSnippet-test, AgNews, StackOverflow, and BioMedical. We used accuracy (ACC) and normalized mutual information (NMI) as the evaluation measures for different clustering algorithms (as in [2]). The clustering results (ACC, NMI) of these datasets are shown in Table 2. The last two rows of Tables 2a and 2b show the ACC and NMI scores obtained using the state-of-the-art short text clustering methods STC2-LE [2] and SIF-Auto [3]. The ACC and NMI scores of five clustering algorithms both before and after iterative classification for the five datasets are shown in these two Tables. The results with or without the $_{IC}$ suffix are the results with or without iterative classification. The best result (ACC, NMI) for each dataset is shown in bold.

To compensate for the dependence of k-Means, k-Means-- on the choice of cluster seeds, we ran the k-Means and k-Means-- clustering algorithms 20 times on the same dataset and performed iterative classification on the clustering obtained in each run. After that, we calculated the mean and standard deviation of the 20 clustering results (ACC, NMI) obtained by k-Means, k-means--, k-Means- $_{IC}$ and k-means-- $_{IC}$ for each dataset. We ran hierarchical agglomerative clustering (HAC), HAC- k -NN, and HAC- SD only once since HAC is deterministic. However, the enhancement of the clustering obtained by iterative classification varies between runs since the training and test sets are chosen randomly in each iteration. So, we ran iterative classification 20 times on the clustering obtained using HAC, HAC- k -NN and HAC- SD , and again calculated the mean and standard deviation of each of the 20 clustering results obtained by HAC- $_{IC}$, HAC- k -NN- $_{IC}$ and HAC- SD - $_{IC}$ for each dataset.

⁴ <http://nlp.stanford.edu/data/glove.42B.300d.zip>.

⁵ bioasq.lip6.fr/tools/BioASQword2vec/.

Table 2. ACC and NMI of different clustering methods, their corresponding enhancements by iterative classification, and state-of-the-art methods for short text clustering. Δ indicates that this method is statistically significantly inferior to its corresponding enhancement obtained by iterative classification. * indicates that this method is statistically significantly inferior to *HAC_{SD}IC*.

Clustering Methods	Datasets				
	Search Snippet	Search SnippetTest	AgNews	Stack Overflow	Bio Medical
	ACC(%)	ACC(%)	ACC(%)	ACC(%)	ACC(%)
<i>HAC_{SD}</i>	82.69 Δ	89.47 Δ	81.84 Δ	64.80 Δ	40.13 Δ
<i>HAC_{SD}IC</i>	87.67$\pm$0.63	92.16$\pm$0.85	84.52$\pm$0.50	78.73$\pm$0.17	47.78$\pm$0.51
<i>HAC_k-NN</i>	79.08 Δ *	87.14 Δ *	76.83 Δ *	58.11 Δ *	39.75 Δ *
<i>HAC_k-NN_{IC}</i>	83.19* $\pm$ 0.61	90.76* $\pm$ 1.79	81.83* $\pm$ 0.35	70.07* $\pm$ 0.11	46.17* $\pm$ 1.10
<i>HAC</i>	76.54 Δ *	77.06 Δ *	76.56 Δ *	61.64 Δ *	38.86 Δ *
<i>HAC_{IC}</i>	80.63* $\pm$ 0.69	83.92* $\pm$ 2.66	81.13* $\pm$ 1.22	67.69* $\pm$ 2.12	46.13* $\pm$ 0.92
<i>k</i> -Means	63.89 Δ * $\pm$ 1.15	63.22 Δ * $\pm$ 1.79	58.17 Δ * $\pm$ 1.87	41.54 Δ * $\pm$ 2.16	36.92 Δ * $\pm$ 0.81
<i>k</i> -Means _{IC}	83.13* $\pm$ 0.69	82.84* $\pm$ 2.32	78.06* $\pm$ 3.13	69.89* $\pm$ 1.52	43.50* $\pm$ 1.38
<i>k</i> -means--	47.42 Δ * $\pm$ 1.13	61.96 Δ * $\pm$ 1.98	62.48 Δ * $\pm$ 2.13	43.77 Δ * $\pm$ 0.39	39.95 Δ * $\pm$ 1.21
<i>k</i> -means-- _{IC}	79.77* $\pm$ 2.67	75.29* $\pm$ 2.79	77.45* $\pm$ 3.49	69.25* $\pm$ 1.88	45.61* $\pm$ 3.19
STC2-LE	78.29* $\pm$ 2.72			53.81* $\pm$ 3.37	44.81* $\pm$ 1.72
SIF-Auto	79.13* $\pm$ 1.27			59.85* $\pm$ 1.81	55.73$\pm$1.97

(a) ACC results

Clustering Methods	Datasets				
	Search Snippet	Search SnippetTest	AgNews	Stack Overflow	Bio Medical
	NMI(%)	NMI(%)	NMI(%)	NMI(%)	NMI(%)
<i>HAC_{SD}</i>	63.76 Δ	78.73 Δ	54.57 Δ	59.48 Δ	33.51 Δ
<i>HAC_{SD}IC</i>	71.93$\pm$1.04	85.55$\pm$1.09	59.07$\pm$0.84	73.44$\pm$0.35	41.27$\pm$0.36
<i>HAC_k-NN</i>	60.51 Δ *	76.42 Δ *	52.43 Δ *	54.06 Δ *	32.19 Δ *
<i>HAC_k-NN_{IC}</i>	65.49* $\pm$ 0.97	83.17* $\pm$ 1.17	56.02* $\pm$ 0.86	68.88* $\pm$ 0.43	38.78* $\pm$ 0.53
<i>HAC</i>	59.41 Δ *	70.99 Δ *	52.82 Δ *	54.46 Δ *	31.01 Δ *
<i>HAC_{IC}</i>	63.61* $\pm$ 1.09	77.49* $\pm$ 1.11	56.57* $\pm$ 1.23	61.76* $\pm$ 1.35	38.50* $\pm$ 0.61
<i>k</i> -Means	43.75 Δ * $\pm$ 1.31	51.54 Δ * $\pm$ 0.92	35.26 Δ * $\pm$ 2.01	38.01 Δ * $\pm$ 2.12	33.71 Δ * $\pm$ 0.29
<i>k</i> -Means _{IC}	66.27 Δ * $\pm$ 1.00	76.88 Δ * $\pm$ 2.64	52.32 Δ * $\pm$ 2.47	69.84 Δ * $\pm$ 0.66	38.08 Δ * $\pm$ 0.81
<i>k</i> -means--	47.43 Δ * $\pm$ 1.65	49.73 Δ * $\pm$ 2.15	39.68 Δ * $\pm$ 1.15	41.89 Δ * $\pm$ 0.86	34.49* $\pm$ 1.93
<i>k</i> -means-- _{IC}	63.01* $\pm$ 1.69	71.11* $\pm$ 2.40	51.05* $\pm$ 3.63	69.64* $\pm$ 1.28	35.63* $\pm$ 2.82
STC2-LE	64.72* $\pm$ 1.37			49.51* $\pm$ 1.63	38.42* $\pm$ 0.87
SIF-Auto	57.72* $\pm$ 1.43			55.59* $\pm$ 1.23	47.21$\pm$1.19

(b) NMI results

Impact of Iterative Classification. We evaluated whether iterative classification improves the initial clustering obtained using different clustering algorithms. We consider iterative classification to improve the clustering for a given dataset if both ACC and NMI are increased using iterative classification.

Table 2 shows that iterative classification improves the initial clustering of short texts in terms of both ACC and NMI. For most of the datasets, the best clustering ACC and NMI were obtained by applying iterative classification to the clustering obtained by HAC with SD sparsification (*HAC_{SD}*). The reason is that *HAC_{SD}* [9] produces better initial clustering than other clustering methods for these datasets and the enhancement of clustering depends on the initial clustering.

Comparison with State-of-the-Art Methods. Our second comparison aims to assess how the results of iterative classification in conjunction with the different clustering methods compare to state-of-the-art short text clustering methods, specifically STC2-LE [2] and SIF-Auto [3]. Table 2a and 2b show that *HAC_{SD-IC}* and *HAC_{k-NN-IC}* outperform STC2-LE⁶ for the SearchSnippet, StackOverflow and BioMedical datasets in terms of ACC and NMI. It is also shown that *HAC_{SD-IC}*, *HAC_{k-NN-IC}*, *HAC_{IC}*, *k-Means_{IC}*, and *k-means--_{IC}* outperform SIF-Auto for the SearchSnippet and StackOverflow datasets in terms of ACC and NMI. However, on the Biomedical dataset, the performance of SIF-Auto is better than any clustering method and its corresponding enhancement by iterative classification.

Statistical Significance Testing of Clustering Performance. Our third comparison aims to investigate whether the clustering improvements achieved by iterative classification are statistically significant. In particular, we perform two investigations: a) whether the improved results achieved by iterative classification are statistically significantly better than the results of their corresponding clustering methods. b) whether the improved results achieved by our best clustering method *HAC_{SD-IC}* are statistically significantly better than the results of different clustering methods (with or without iterative classification and state-of-the-art methods). For significance testing, we performed a two-tailed paired t-test (with significance level $\alpha = 0.05$) using the pairwise differences of clustering results (ACC, NMI) of 20 runs obtained by different pairs of clustering methods.

On all datasets except the BioMedical dataset, and for all clustering methods tested, the enhancement by iterative classification is statistically significantly better than the base clustering method, and the former are statistically significantly inferior to our method *HAC_{SD-IC}*. For the BioMedical dataset, the ACC and NMI scores achieved by *HAC_{SD-IC}* are statistically significantly better than that of STC2-LE. However, SIF-Auto outperforms *HAC_{SD-IC}* on the BioMedical dataset.

7 Conclusion and Future Work

We have demonstrated that iterative classification enhances the clustering of short texts for various short text datasets based on initial clusters obtained using such as k-means, k-means--, hierarchical agglomerative clustering (HAC), HAC using *k*-NN and *SD* sparsification methods. The most promising results were obtained by applying iterative classification to the clustering obtained by HAC using the proposed *SD* sparsification (*HAC_{SD-IC}*). Experimental results show that *HAC_{SD-IC}* outperforms a state-of-the-art short text clustering method based on convolutional neural network (STC2-LE) on all the datasets in terms

⁶ We were unable to reproduce the clustering for other short text datasets using STC2-LE and SIF-Auto.

of ACC and NMI. Moreover, HAC-SD-IC outperforms another state-of-the-art short text clustering method based on autoencoder (SIF-Auto) in terms of ACC and NMI on several short text datasets. The proposed clustering enhancement method advances the state of the art in short text clustering, which is important in the following practical contexts such as social media monitoring, product recommendation, and customer feedback analysis. The proposed method is a generic clustering enhancement approach for short texts where various classification algorithms, initial clustering and number of clusters can be easily integrated.

In the future, we will apply our clustering enhancement algorithm to long documents to investigate whether iterative classification leads to performance improvements. We also plan to use phrase similarity as a basis for computing text similarity so as to obtain better text similarity scores, since the performance of clustering algorithms depends on the quality of individual text similarity scores.

References

1. Aggarwal, C.C., Zhai, C.: A survey of text clustering algorithms. In: Aggarwal, C., Zhai, C. (eds.) *Mining Text Data*, pp. 77–128. Springer, Boston (2012). https://doi.org/10.1007/978-1-4614-3223-4_4
2. Xu, J., et al.: Self-taught convolutional neural networks for short text clustering. *Neural Netw.* **88**, 22–31 (2017)
3. Hadifar, A., Sterckx, L., Demeester, T., Develder, C.: A self-training approach for short text clustering. In: *Proceedings of the 4th Workshop on Representation Learning for NLP*, Italy, pp. 194–199 (2019)
4. Yin, J., Wang, J.: A model-based approach for text clustering with outlier detection. In: *32nd International Conference on Data Engineering (ICDE)*, pp. 625–636 (2016)
5. Shekhar, S., Lu, C.-T., Zhang, P.: A unified approach to detecting spatial outliers. *GeoInformatica* **7**(2), 139–166 (2003)
6. Li, M., Cheng, Y., Zhao, H.: Unlabeled data classification via support vector machines and k-means clustering. In: *Proceedings of the International Conference on Computer Graphics, Imaging and Visualization*, pp. 183–186 (2004)
7. Kumar, V.: An introduction to cluster analysis for data mining. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN (2000). https://www-users.cs.umn.edu/hanxx023/dmclass/cluster_survey_10-02_00.pdf
8. Gollub, T., Stein, B.: Unsupervised sparsification of similarity graphs. In: Locarek-Junge, H., Weihs, C. (eds.) *Classification as a Tool for Research*, pp. 71–79. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-10745-0_7
9. Rakib, M.R.H., Jankowska, M., Zeh, N., Milios, E.: Improving short text clustering by similarity matrix sparsification. In: *Proceedings of the ACM Symposium on Document Engineering 2018*, Halifax, NS, Canada, pp. 50:1–50:4 (2018)
10. Banerjee, S., Ramanathan, K., Gupta, A.: Clustering short texts using Wikipedia. In: *Proceedings of the 30th International ACM Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, pp. 787–788 (2007)

11. Zheng, C., Liu, C., Wong, H.: Corpus-based topic diffusion for short text clustering. *Neurocomputing* **275**, 2444–2458 (2018)
12. Cheng, X., Yan, X., Lan, Y., Guo, J.: BTM: topic modeling over short texts. *IEEE Trans. Knowl. Data Eng.* **26**(12), 2928–2941 (2014)
13. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: *Proceedings of Workshop at ICLR 2013*, January 2013
14. Kanj, S., Bröls, T., Gazut, S.: Shared nearest neighbor clustering in a locality sensitive hashing framework. *J. Comput. Biol.* **25**(2), 236–250 (2018)
15. Phan, X., Nguyen, L., Horiguchi, S.: 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*, Beijing, China, pp. 91–100 (2008)
16. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems*, USA, pp. 649–657 (2015)
17. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1532–1543 (2014)
18. Du, Y., Pan, Y., Wang, C., Ji, J.: Biomedical semantic indexing by deep neural network with multi-task learning. *BMC Bioinform.* **19**(20), 502 (2018)