Short Text Clustering via Convolutional Neural Networks

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao,
Fangyuan Wang, Hongwei Hao
Institute of Automation, Chinese Academy of Sciences. 100190, Beijing, P.R. China
{jiaming.xu, peng.wang, boxu, guanhua.tian}@ia.ac.cn,
jzhao@nlpr.ia.ac.cn, {fangyuan.wang, hongwei.hao}@ia.ac.cn

Abstract

Short text clustering has become an increasing important task with the popularity of social media, and it is a challenging problem due to its sparseness of text representation. In this paper, we propose a Short Text Clustering via Convolutional neural networks (abbr. to STCC), which is more beneficial for clustering by considering one constraint on learned features through a self-taught learning framework without using any external tags/labels. First, we embed the original keyword features into compact binary codes with a locality-preserving constraint. Then, word embeddings are explored and fed into convolutional neural networks to learn deep feature representations, with the output units fitting the pre-trained binary code in the training process. After obtaining the learned representations, we use K-means to cluster them. Our extensive experimental study on two public short text datasets shows that the deep feature representation learned by our approach can achieve a significantly better performance than some other existing features, such as term frequency-inverse document frequency, Laplacian eigenvectors and average embedding, for clustering.

1 Introduction

Different from the normal text clustering, short text clustering has the problem of sparsity (Aggarwal and Zhai, 2012). Most words only occur once in each short text, as a result, the term frequency-inverse document frequency (TF-IDF) measure cannot work well in the short text setting. In order to address this problem, some researchers work on expanding and enriching the context of data from Wikipedia (Banerjee et al., 2007) or an ontology (Fodeh et al., 2011). However, these methods involve solid natural language processing (NLP) knowledge and still use high-dimensional representation which may result in a waste of both memory and computation time. Another way to overcome these issues is to explore some sophisticated models to cluster short texts. For example, Yin and Wang (2014) proposed a Dirichlet multinomial mixture model-based approach for short text clustering and Cai et al. (2005) clustered texts using Locality Preserving Indexing (LPI) algorithm. Yet how to design an effective model is an open question, and most of these methods directly trained based on bag-of-words (BoW) are shallow structures which cannot preserve the accurate semantic similarities.

With the recent revival of interest in Deep Neural Network (DNN), many researchers have concentrated on using Deep Learning to learn features. Hinton and Salakhutdinov (2006) use deep auto encoder (DAE) to learn text representation from raw text representation. Recently, with the help of word embedding, neural networks demonstrate their great performance in terms of constructing text representation, such as Recursive Neural Network (RecNN) (Socher et al., 2011; Socher et al., 2013) and Recurrent Neural Network (RNN) (Mikolov et al., 2011). However, RecNN exhibits high time complexity to construct the textual tree, and RNN, using the layer computed at the last word to represent the text, is a biased model (Lai et al., 2015). More recently, Convolution Neural Network (CNN), applying convolutional filters to capture local features, has achieved a better performance in many NLP applications, such as sentence modeling (Blunsom et al., 2014), relation classification (Zeng et al., 2014), and other traditional NLP tasks (Collobert et al., 2011). Most of the previous works focus CNN on solving supervised NLP tasks, while in this paper we aim to explore the power of CNN on one unsupervised NLP task, short text clustering.

To address the above challenges, we systematically introduce a short text clustering method via con-

Proceedings of NAACL-HLT 2015, pages 62–69,
Denver, Colorado, May 31 – June 5, 2015. ©2015 Association for Computational Linguistics

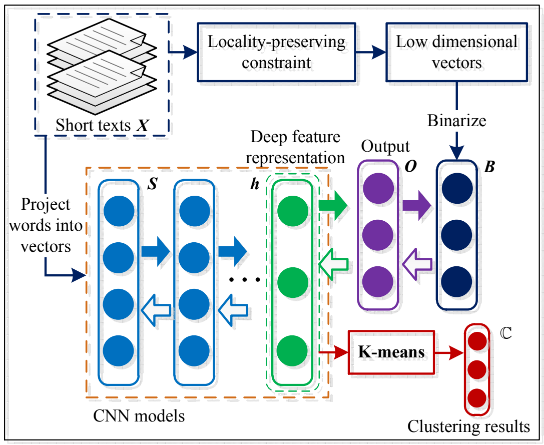

Figure 1: Architecture of the proposed short text clustering via convolutional neural networks

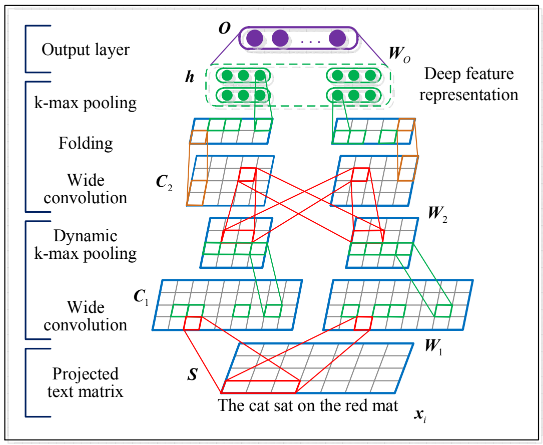

Figure 2: Dynamic convolutional neural network used for extracting deep feature representation

volutional neural networks. An overall architecture of the proposed method is illustrated in Figure 1. Given a short text collection $\mathbf{X}$, the goal of this work is to cluster these texts into clusters $\mathbb{C}$ based on the deep feature representation $\mathbf{h}$ learned from CNN models. In order to train the CNN models, we, inspired by (Zhang et al., 2010), utilize a self-taught learning framework in our work. In particular, we first embed the original features into compact binary code $\mathbf{B}$ with a locality-preserving constraint. Then word vectors $\mathbf{S}$ projected from word embeddings are fed into a CNN model to learn the feature representation $\mathbf{h}$ and the output units are used to fit the pre-trained binary code $\mathbf{B}$. After obtaining the learned features, traditional K-means algorithm is employed to cluster texts into clusters $\mathbb{C}$. The main contributions of this paper are summarized as follows:

1). To the best of our knowledge, this is the first attempt to explore the feasibility and effectiveness of combining CNN and traditional semantic constraint, with the help of word embedding to solve one unsupervised learning task, short text clustering.

2). We learn deep feature representations with locality-preserving constraint through a self-taught learning framework, and our approach do not use any external tags/labels or complicated NLP preprocessing.

3). We conduct experiments on two short text datasets. The experimental results demonstrate that the proposed method achieves excellent performance in terms of both accuracy and normalized mutual information.

The remainder of this paper is organized as follows: In Section 2, we first describe the proposed approach STCC and implementation details. Experimental results and analyses are presented in Section 3. In Section 4, we briefly survey several related works. Finally, conclusions are given in the last Section.

2 Methodology

2.1 Convolutional Neural Networks

In this section, we will briefly review one popular deep convolutional neural network, Dynamic Convolutional Neural Network (DCNN) (Blunsom et al., 2014), which is the foundation of our proposed method.

Taking a neural network with two convolutional layers in Figure 2 as an example, the network transforms raw input text to a powerful representation. Particularly, let $\mathbf{X} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^{d\times 1}\}_{i=1,2,\dots,n}$ denote the set of input n texts, where d is the dimensionality of the original keyword features. Each raw text vector $\mathbf{x}_i$ is projected into a matrix representation $\mathbf{S} \in \mathbb{R}^{d_w \times s}$ by looking up a word embedding $\mathbf{E}$, where d_w is the dimension of word embedding features and s is the length of one text. We also let $\tilde{\mathbf{W}} = \{\mathbf{W}_i\}_{i=1,2}$ and $\mathbf{W}_O$ denote the weights of the neural networks. The network defines a transformation $f(\cdot) : \mathbb{R}^{d\times 1} \rightarrow \mathbb{R}^{r\times 1}$ $(d \gg r)$ which transforms an raw input text $\mathbf{x}$ to a r-dimensional deep representation $\mathbf{h}$. There are three basic operations described as follows:

– Wide one-dimensional convolution This operation is applied to an individual row of the sentence matrix $\mathbf{S} \in \mathbb{R}^{d_w \times s}$, and yields a set of sequences $\mathbf{C}_i \in \mathbb{R}^{s+m-1}$ where m is the width of convolutional filter.

– Folding In this operation, every two rows in a feature map component-wise are simply summed. For a map of d_w rows, folding returns a map of $d_w/2$ rows, thus halving the size of the representation.

– Dymantic k-max pooling Given a fixed pooling parameter k_{top} for the topmost convolutional layer, the parameter k of k-max pooling in the l-th convolutional layer can be computed as follows:

$$k_l = \max(k_{top}, \left\lceil \frac{L-l}{L} s \right\rceil), \tag{1}$$

where L is the total number of convolutional layers in the network.

2.2 Locality-preserving Constraint

Here, we first pre-train binary code $\mathbf{B}$ based on the keyword features with a locality-preserving constraint, and choose Laplacian affinity loss, also used in some previous works (Weiss et al., 2009; Zhang et al., 2010). The optimization can be written as:

$$\begin{aligned} &\min_{\mathbf{B}} \sum_{i,j=1}^{n} S_{ij} \left\| \mathbf{b}_i - \mathbf{b}_j \right\|_F^2 \\ &s.t.\, \mathbf{B} \in \{-1,1\}^{n \times q},\ \mathbf{B}^T \mathbf{1} = \mathbf{0},\ \mathbf{B}^T \mathbf{B} = \mathbf{I}, \end{aligned} \tag{2}$$

where S_{ij} is the pairwise similarity between texts $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\|\cdot\|_F$ is the Frobenius norm. The problem is relaxed by discarding $\mathbf{B} \in \{-1,1\}^{n \times q}$, and the q-dimensional real-valued vectors $\tilde{\mathbf{B}}$ can be learned from Laplacian Eigenmap. Then, we get the binary code $\mathbf{B}$ via the media vector $median(\tilde{\mathbf{B}})$. In particular, we construct the $n \times n$ local similarity matrix $\mathbf{S}$ by using heat kernel as follows:

$$S_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}), & if\ x_i \in N_k(\mathbf{x}_j)\ or\ vice \text{ versa} \\ 0, & otherwise \end{cases} \tag{3}$$

where, σ is a tuning parameter (default is 1) and $\mathbf{N}_k(\mathbf{x})$ represents the set of k-nearest-neighbors of $\mathbf{x}$.

The last layer of CNN is an output layer as follows:

$$\mathbf{O} = \mathbf{W}_O \mathbf{h}, \tag{4}$$

where, $\mathbf{h}$ is the deep feature representation, $\mathbf{O} \in \mathbb{R}^q$ is the output vector and $\mathbf{W}_O \in \mathbb{R}^{q \times r}$ is weight matrix. In order to fit the pre-trained binary code $\mathbf{B}$, we apply q logistic operations to the output vector $\mathbf{O}$ as follows:

$$p_i = \frac{\exp(\mathbf{O}_i)}{1 + \exp(\mathbf{O}_i)}. \tag{5}$$

2.3 Learning

All of the parameters to be trained are defined as θ.

$$\theta = \{\mathbf{E}, \tilde{\mathbf{W}}, \mathbf{W}_O\}. \tag{6}$$

Given the training text collection $\mathbf{X}$, and the pre-trained binary code $\mathbf{B}$, the log likelihood of the parameters can be written down as follows:

$$J(\theta) = \sum_{i=1}^{n} \log p(\mathbf{b}_i | \mathbf{x}_i, \theta). \tag{7}$$

Following the previous work (Blunsom et al., 2014), we train the network with mini-batches by back-propagation and perform the gradient-based optimization using the Adagrad update rule (Duchi et al., 2011). For regularization, we employ dropout with 50% rate to the penultimate layer (Blunsom et al., 2014; Kim, 2014).

2.4 K-means for Clustering

With the given short texts, we first utilize the trained deep neural network to obtain the semantic representations $\mathbf{h}$, and then employ traditional K-means algorithm to perform clustering.

3 Experiments

3.1 Datasets

We test our algorithm on two public text datasets, and the summary statistics of the datasets are described in Table 1.

SearchSnippets[1]. This dataset was selected from the results of web search transaction using predefined phrases of 8 different domains (Phan et al., 2008).

[1]http://jwebpro.sourceforge.net/data-web-snippets.tar.gz.

StackOverflow[2]. We use the challenge data published in Kaggle.com[3]. This dataset consists 3,370,528 samples through July 31st, 2012 to August 14, 2012. In our experiments, we randomly select 20, 000 question titles from 20 different tags.

For these datasets, we do not remove any stop words or symbols in the text.

Dataset	C	Number	L(mean/max)	$\|V\|$
Snippets	8	12340	17.88/38	30642
Stack	20	20000	8.31/34	22956

Table 1: Statistics for the text datasets. C: the number of classes; Num: the dataset size; L(mean/max): the mean and max length of texts and $|V|$: the vocabulary size.

3.2 Pre-trained Word Vectors

We use the publicly available word2vec tool to train word embeddings, and the most parameters are set as same as Mikolov et al. (2013) to train word vectors on Google News setting[4], excepts of vector dimensionality using 48 and minimize count using 5. For SearchSnippets, we train word vectors on Wikipedia dumps[5]. For StackOverflow, we train word vectors on the whole corpus of the StackOverflow dataset described above which includes the question titles and post contents. The coverage of these learned vectors on two datasets are listed in Table 2, and the words not present in the set of pre-trained words are initialized randomly.

Dataset	$\|V\|$	$\|T\|$
SearchSnippets	23826 (77%)	211575 (95%)
StackOverflow	19639 (85%)	162998 (97%)

Table 2: Coverage of word embeddings on two datasets. $|V|$ is the vocabulary size and $|T|$ is the number of tokens.

3.3 Comparisons

We compare the proposed method with some most popular clustering algorithms:

- **K-means** K-means (Wagstaff et al., 2001) on original keyword features which are respectively weighted with term frequency (TF) and term frequency-inverse document frequency (TF-IDF).

- **Spectral Clustering** This baseline (Belkin and Niyogi, 2001) uses Laplacian Eigenmaps (LE) and subsequently employ K-means algorithm. The dimension of subspace is default set to the number of clusters (Ng et al., 2002; Cai et al., 2005), we also iterate the dimensions ranging from 10:10:200 to get the best performance, that is 20 on SearchSnippets and 70 on StackOverflow in our expriments.

- **Average Embedding** K-means on the weighted average of the word embeddings which are respectively weighted with TF and TF-IDF. Huang et al. (2012) also used this strategy as the global context in their task and Lai et al. (2015) used this in text classification.

3.4 Evaluation Metrics

The clustering performance is evaluated by comparing the clustering results of texts with the tags/labels provided by the text corpus. Two metrics, the accuracy (ACC) and the normalized mutual information metric (NMI), are used to measure the clustering performance (Cai et al., 2005; Huang et al., 2014). Given a text $\mathbf{x}_i$, let c_i and y_i be the obtained cluster label and the label provided by the corpus, respectively. Accuracy is defined as:

$$ACC = \frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n}, \quad (8)$$

where, n is the total number of texts, $\delta(x, y)$ is the indicator function that equals one if $x = y$ and equals zero otherwise, and $map(c_i)$ is the permutation mapping function that maps each cluster label c_i to the equivalent label from the text data by Hungarian algorithm (Papadimitriou and Steiglitz, 1998).

Normalized mutual information (Chen et al., 2011) between tag/label set $\mathbf{Y}$ and cluster set $\mathbb{C}$ is a popular metric used for evaluating clustering tasks. It is defined as follows:

$$NMI(\mathbf{Y}, \mathbb{C}) = \frac{MI(\mathbf{Y}, \mathbb{C})}{\sqrt{H(\mathbf{Y})H(\mathbb{C})}}, \quad (9)$$

[2]https://github.com/jacoxu/StackOverflow.

[3]https://www.kaggle.com/c/predict-closed-questions-on-stack-overflow/download/train.zip.

[4]https://groups.google.com/forum/#!topic/word2vec-toolkit/lxbl_MB29Ic.

[5]http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2.

where, $MI(\mathbf{Y}, \mathbb{C})$ is the mutual information between $\mathbf{Y}$ and $\mathbb{C}$, $H(\cdot)$ is entropy and the denominator $\sqrt{H(\mathbf{Y})H(\mathbb{C})}$ is used for normalizing the mutual information to be in the range of [0, 1].

3.5 Hyperparameter Settings

In our experiments, the most of parameters are set uniformly for these datasets. Following previous study (Cai et al., 2005), the parameter k in Eq. 3 is fixed to 15 when constructing the graph Laplacians in our approach, as well as in spectral clustering. For CNN model, we manually choose a same architecture for the two datasets. More specifically, in our experiments, the networks has two convolutional layers similar as the example in Figure 2. The widths of the convolutional filters are both 3. The value of k for the top k-max pooling is 5. The number of feature maps at the first convolutional layer is 12, and 8 feature maps at the second convolutional layer. Both those two convolutional layers are followed by a folding layer. We further set the dimension of word embeddings d_w as 48. Finally, the dimension of the deep feature representation r is fixed to 480. Moreover, we set the learning rate λ as 0.01 and the mini-batch training size as 200. The output size q in Eq. 4 and Eq. 2 is set same as the best dimensions of subspace in the baseline method, spectral clustering, as described in Section 3.3.

For initial centroids have significant impact on clustering results when utilizing the K-means algorithms, we repeat K-means for multiple times with random initial centroids (specifically, 100 times for statistical significance). The final results reported are the average of 5 trials with all clustering methods on two text datasets.

3.6 Quantitative Results

Here, we firstly evaluate the influence of the iteration number in our method. Figure 3 shows the change of ACC and NMI as the iteration number increases on two text datasets. It can be found that the performance rises steadily in the first ten iterations, which demonstrates that our method is effective. In the period of 10∼20 iterations, ACC and NMI become relatively stable on both two texts. In the following experiments, we report the results after 10 iterations.

We report ACC and NMI performance of all the clustering methods in Table 3. The experimental results show that Spectral Clustering and Average Embedding significantly better than K-means on two datasets. It is because K-means directly construct the similarity structure from the original keyword feature space while Average Embedding and Spectral Clustering extract the semantic features using shallow structure models. Compared with the best baselines, the proposed STCC extracting deep learned representation from convolutional neural network achieves large improvement on these datasets by 2.33%/4.86% and 14.23%/10.01% (ACC/NMI) on SearchSnippets and StackOverflow, respectively. Note that TF-IDF weighting gives a more remarkable improvement for K-means, while TF weighting works better than TF-IDF weighting for Average Embedding. Maybe the reason is that pre-trained word embeddings encode some useful information from external corpus and are able to get even better results without TF-IDF weighting.

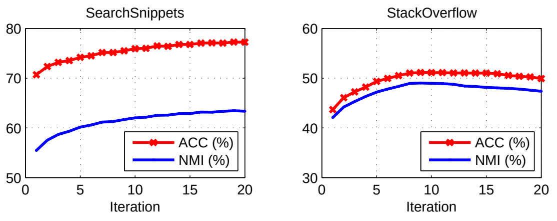

Figure 3: Influence of the iteration number on two text datasets.

In Figure 4 and Figure 5, we further report 2-dimensional embeddings using stochastic neighbor embedding (Van der Maaten and Hinton, 2008)[6] of the feature representations used in the clustering methods. We can see that the 2-dimensional embedding results of deep features representation learned from our STCC show more clear-cut margins among different semantic topics (that is, tags/labels) on two short text datasets.

4 Related Work

In this section, we review the related work from the following two perspectives: short text clustering and deep neural networks.

4.1 Short Text Clustering

There have been several studies that attempted to overcome the sparseness of short representation.

[6]http://lvdmaaten.github.io/tsne/.

	SearchSnippets		StackOverflow	
Method	ACC (%)	NMI (%)	ACC (%)	NMI (%)
K-means (TF)	24.75±2.22	9.03±2.30	13.51±2.18	7.81±2.56
K-means (TF-IDF)	33.77±3.92	21.40±4.35	20.31±3.95	15.64±4.68
Spectral Clustering	63.90±5.36	48.44±2.39	27.55±0.93	21.03±0.37
Spectral Clustering (best)	74.76±5.08	58.30±1.97	37.17±1.62	26.27±0.86
Average Embedding (TF-IDF)	62.05±5.27	46.64±1.87	37.02±1.29	35.58±0.84
Average Embedding (TF)	64.63±4.84	50.59±1.71	37.22±1.57	38.43±1.13
STCC	**77.09±3.99**	**63.16±1.56**	**51.13±2.80**	**49.03±1.46**

Table 3: Comparison of ACC and NMI of clustering methods on two short text datasets. For Spectral Clustering, the dimension of subspace are set to the number of clusters, and Spectral Clustering (best) get the best performance by iterating the dimensions ranging from 10:10:200. More details about the baseline setting are described in Section 3.3

One way is to expand and enrich the context of data. For example, Banerjee et al. (2007) proposed a method of improving the accuracy of short text clustering by enriching their representation with additional features from Wikipedia, and Fodeh et al. (2011) incorporate semantic knowledge from an ontology into text clustering. Another way is to explore some sophisticated models to cluster short text. For example, Yin and Wang (2014) proposed a Dirichlet multinomial mixture model-based approach for short text clustering and Cai et al. (2005) applied the LPI algorithm for text clustering. Moreover, some studies both focus the above two streams. For example, Tang et al. (2012) proposed a novel framework which performs multi-language knowledge integration and feature reduction simultaneously through matrix factorization techniques. However, the former works need solid NLP knowledge while the later works are shallow structures which can not fully capture accurate semantic similarities.

4.2 Deep Neural Networks

With the recent revival of interest in DNN, many researchers have concentrated on using Deep Learning to learn features. Hinton and Salakhutdinov (2006) use DAE to learn text representation. During the fine-tuning procedure, they use backpropagation to find codes that are good at reconstructing the word-count vector.

Recently, researchers propose to use external corpus to learn a distributed representation for each word, called word embedding (Turian et al., 2010), to improve DNN performance on NLP tasks. The skip-gram and continuous bag-of-words models of (Mikolov et al., 2013) propose a simple single-layer architecture based on the inner product between two word vectors, and Jeffrey Pennington et al. (2014) introduce a new model for word representation, called GloVe, which captures the global corpus statistics.

Based on word embedding, neural networks can capture true meaningful syntactic and semantic regularities, such as RecNN (Socher et al., 2011; Socher et al., 2013) and RNN (Mikolov et al., 2011). However, RecNN exhibits high time complexity to construct the textual tree, and RNN, using the layer computed at the last word to represent the text, is a biased model. Recently, CNN, applying convolving filters to local features, has been successfully exploited for many supervised NLP learning tasks as described in Section 1. This paper, to our best knowledge, is the first time to explore the power of CNN and word embedding to solve one unsupervised learning task, short text clustering.

5 Conclusions

In this paper, we proposed a short text clustering based on deep feature representation learned from CNN without using any external tags/labels and complicated NLP pre-processing. As experimental study shows that STCC can achieve significantly better performance than the baseline methods.

Acknowledgments

We thank anonymous reviewers for their comments, and this work is supported by the National Natural

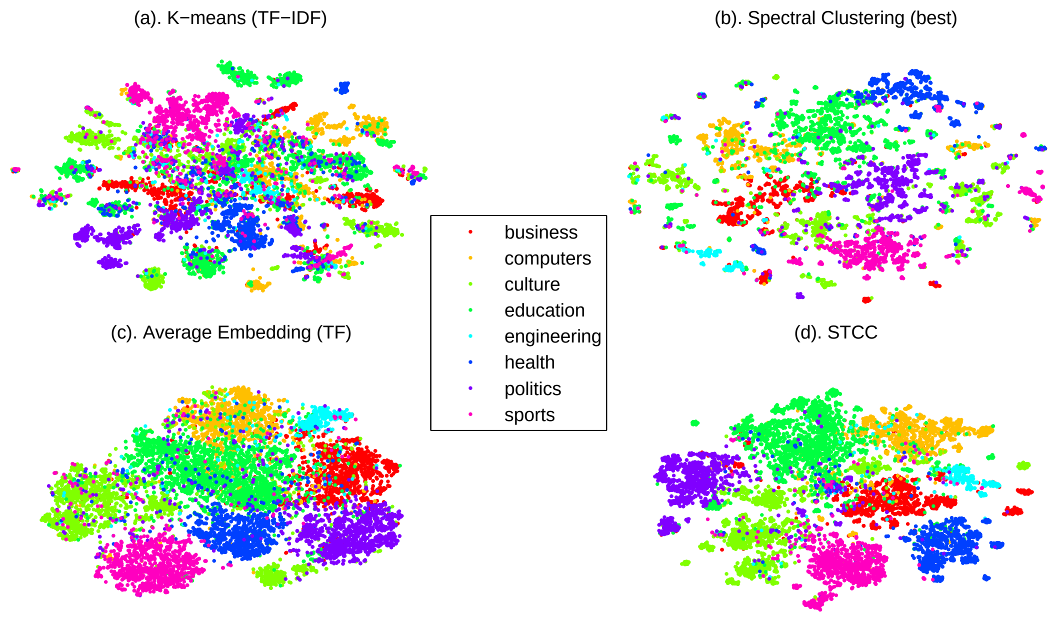

Figure 4: A 2-dimensional embedding of original keyword features weighted with TF-IDF (a), Laplacian eigenvectors (b), average embeddings weighted with TF (c) and deep learned features (d) which are respectively used in K-means (TF-IDF), Spectral Clustering (best), Average Embeddings (TF) and the proposed STCC methods on SearchSnippets. (Best viewed in color)

Science Foundation of China (No. 61203281 and No. 61303172) and Hundred Talents Program of Chinese Academy of Sciences (No. Y3S4011D31).

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer.

Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. 2007. Clustering short texts using wikipedia. In *SIGIR*, pages 787–788. ACM.

Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591.

Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *ACL*.

Deng Cai, Xiaofei He, and Jiawei Han. 2005. Document clustering using locality preserving indexing. *Knowledge and Data Engineering, IEEE Transactions on*, 17(12):1624–1637.

Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. 2011. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Samah Fodeh, Bill Punch, and Pang-Ning Tan. 2011. On ontology-driven document clustering using core semantic features. *Knowledge and information systems*, 28(2):395–421.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, pages 873–882. Association for Computational Linguistics.

Peihao Huang, Yan Huang, Wei Wang, and Liang Wang. 2014. Deep embedding network for clustering. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 1532–1537. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*.

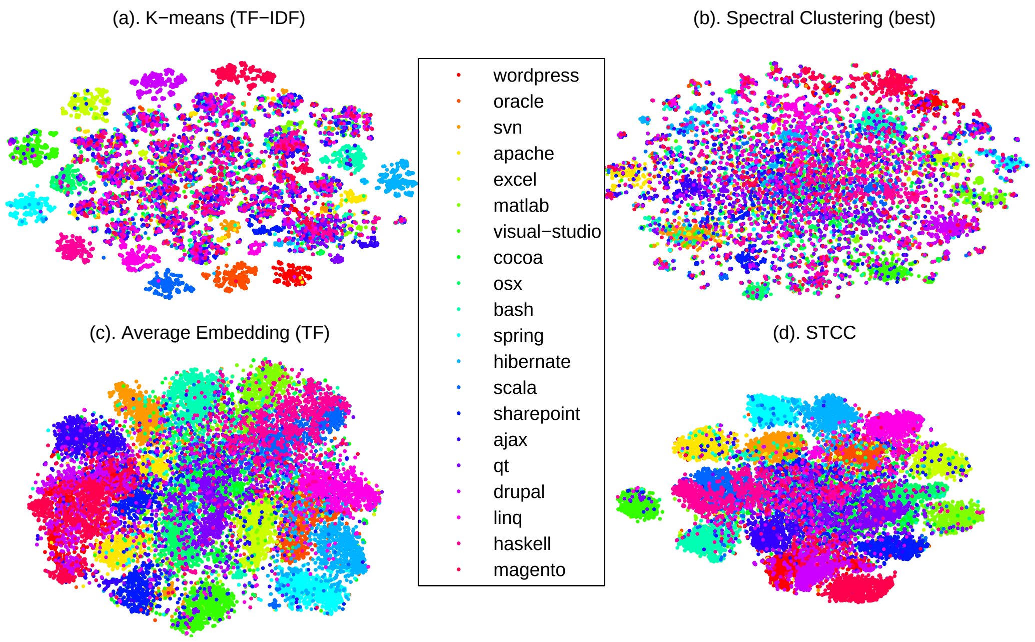

Figure 5: A 2-dimensional embedding of original keyword features weighted with TF-IDF (a), Laplacian eigenvectors (b), average embeddings weighted with TF (c) and deep learned features (d) which are respectively used in K-means (TF-IDF), Spectral Clustering (best), Average Embeddings (TF) and the proposed STCC methods on StackOverflow. (Best viewed in color)

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan H Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. 2002. On spectral clustering: Analysis and an algorithm. *NIPS*, 2:849–856.

Christos H Papadimitriou and Kenneth Steiglitz. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *EMNLP*, 12.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW*, pages 91–100. ACM.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, page 1642. Citeseer.

Jiliang Tang, Xufei Wang, Huiji Gao, Xia Hu, and Huan Liu. 2012. Enriching short text representation in microblog for clustering. *Frontiers of Computer Science*, 6(1):88–101.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85.

Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. 2001. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584.

Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *NIPS*, pages 1753–1760.

Jianhua Yin and Jianyong Wang. 2014. A dirichlet multinomial mixture model-based approach for short text clustering. In *SIGKDD*, pages 233–242. ACM.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught hashing for fast similarity search. In *SIGIR*, pages 18–25. ACM.