Self-Taught Convolutional Neural Networks for Short Text Clustering

Jiaming Xu[a], Bo Xu[a,*], Peng Wang[a], Suncong Zheng[a],
Guanhua Tian[a], Jun Zhao[a,b], Bo Xu[a,c]

[a]*Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, P.R. China*
[b]*National Laboratory of Pattern Recognition (NLPR), Beijing, P.R. China*
[c]*Center for Excellence in Brain Science and Intelligence Technology, CAS. P.R. China*

Abstract

Short text clustering is a challenging problem due to its sparseness of text representation. Here we propose a flexible Self-Taught Convolutional neural network framework for Short Text Clustering (dubbed STC^2), which can flexibly and successfully incorporate more useful semantic features and learn non-biased deep text representation in an unsupervised manner. In our framework, the original raw text features are firstly embedded into compact binary codes by using one existing unsupervised dimensionality reduction methods. Then, word embeddings are explored and fed into convolutional neural networks to learn deep feature representations, meanwhile the output units are used to fit the pre-trained binary codes in the training process. Finally, we get the optimal clusters by employing K-means to cluster the learned representations. Extensive experimental results demonstrate that the proposed framework is effective, flexible and outperform several popular clustering methods when tested on three public short text datasets.

Keywords: Semantic Clustering, Neural Networks, Short Text, Unsupervised Learning

---

*Corresponding author. Email address: boxu@ia.ac.cn

1. Introduction

Short text clustering is of great importance due to its various applications, such as user profiling [1] and recommendation [2], for nowaday's social media dataset emerged day by day. However, short text clustering has the data sparsity problem and most words only occur once in each short text [3]. As a result, the Term Frequency-Inverse Document Frequency (TF-IDF) measure cannot work well in short text setting. In order to address this problem, some researchers work on expanding and enriching the context of data from Wikipedia [4] or an ontology [5]. 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 [6] proposed a Dirichlet multinomial mixture model-based approach for short text clustering. 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.

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) [7, 8] and Recurrent Neural Network (RNN) [9]. However, RecNN exhibits high time complexity to construct the textual tree, and RNN, using the hidden layer computed at the last word to represent the text, is a biased model where later words are more dominant than earlier words [10]. Whereas for the non-biased models, the learned representation of one text can be extracted from all the words in the text with non-dominant learned weights. More recently, Convolution Neural Network (CNN), as the most popular non-biased model and applying convolutional filters to capture local features, has achieved a better performance in many NLP applications, such as sentence modeling [11], relation classification [12], and other traditional NLP tasks [13]. 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.

We systematically introduce a simple yet surprisingly powerful Self-Taught Convolutional neural network framework for Short Text Clustering, called STC^2. An overall architecture of our proposed approach is illustrated in Figure 1. We, inspired by [14, 15], utilize a self-taught learning framework into our task. In particular, the original raw text features are first embedded into compact binary codes $\mathbf{B}$ with the help of one traditional unsupervised dimensionality reduction function. Then text matrix $\mathbf{S}$ projected from word embeddings are fed into CNN model to learn the deep feature representation $\mathbf{h}$ and the output units are used to fit the pre-trained binary codes $\mathbf{B}$. After obtaining the learned features, K-means algorithm is employed on them to cluster texts into clusters $\mathbb{C}$. Obviously, we call our approach "self-taught" because the CNN model is learnt from the pseudo labels generated from the previous stage, which is quite different from the term "self-taught" in [16]. Our main contributions can be summarized as follows:

- We propose a flexible short text clustering framework which explores the feasibility and effectiveness of combining CNN and traditional unsupervised dimensionality reduction methods.

- Non-biased deep feature representations can be learned through our self-taught CNN framework which does not use any external tags/labels or complicated NLP pre-processing.

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

This work is an extension of our conference paper [17], and they differ in the following aspects. First, we put forward a general a self-taught CNN framework in this paper which can flexibly couple various semantic features, whereas the
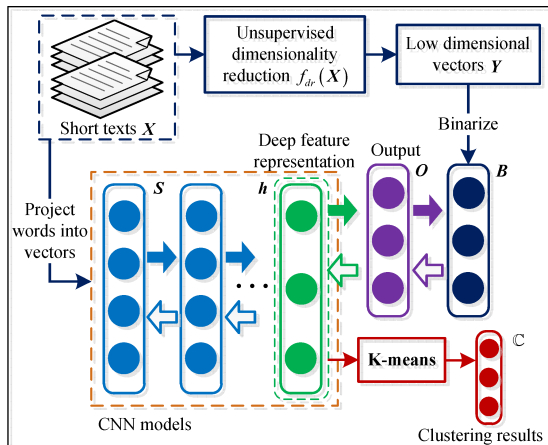
Figure 1: The architecture of our proposed STC^2 framework for short text clustering. Solid and hollow arrows represent forward and backward propagation directions of features and gradients respectively. The STC^2 framework consist of deep convolutional neural network (CNN), unsupervised dimensionality reduction function and K-means module on the deep feature representation from the top hidden layers of CNN.

conference version can be seen as a specific example of this work. Second, in this paper we use a new short text dataset, Biomedical, in the experiment to verify the effectiveness of our approach. Third, we put much effort on studying the influence of various different semantic features integrated in our self-taught CNN framework, which is not involved in the conference paper.

For the purpose of reproducibility, we make the datasets and software used in our experiments publicly available at the website[1].

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

---

[1]Our code and dataset are available: https://github.com/jacoxu/STC2

2. Related Work

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

2.1. Short Text Clustering

There have been several studies that attempted to overcome the sparseness of short text representation. One way is to expand and enrich the context of data. For example, Banerjee et al. [4] proposed a method of improving the accuracy of short text clustering by enriching their representation with additional features from Wikipedia, and Fodeh et al. [5] incorporate semantic knowledge from an ontology into text clustering. However, these works need solid NLP knowledge and still use high-dimensional representation which may result in a waste of both memory and computation time. Another direction is to map the original features into reduced space, such as Latent Semantic Analysis (LSA) [18], Laplacian Eigenmaps (LE) [19], and Locality Preserving Indexing (LPI) [20]. Even some researchers explored some sophisticated models to cluster short texts. For example, Yin and Wang [6] proposed a Dirichlet multinomial mixture model-based approach for short text clustering. Moreover, some studies even focus the above both two streams. For example, Tang et al. [21] proposed a novel framework which enrich the text features by employing machine translation and reduce the original features simultaneously through matrix factorization techniques.

Despite the above clustering methods can alleviate sparseness of short text representation to some extent, most of them ignore word order in the text and belong to shallow structures which can not fully capture accurate semantic similarities.

2.2. Deep Neural Networks

Recently, there is a revival of interest in DNN and many researchers have concentrated on using Deep Learning to learn features. Hinton and Salakhutdinov [22] use DAE to learn text representation. During the fine-tuning proce-

dure, they use backpropagation to find codes that are good at reconstructing the word-count vector.

More recently, researchers propose to use external corpus to learn a distributed representation for each word, called word embedding [23], to improve DNN performance on NLP tasks. The Skip-gram and continuous bag-of-words models of Word2vec [24] propose a simple single-layer architecture based on the inner product between two word vectors, and Pennington et al. [25] introduce a new model for word representation, called GloVe, which captures the global corpus statistics.

In order to learn the compact representation vectors of sentences, Le and Mikolov [26] directly extend the previous Word2vec [24] by predicting words in the sentence, which is named Paragraph Vector (Para2vec). Para2vec is still a shallow window-based method and need a larger corpus to yield better performance. More neural networks utilize word embedding to capture true meaningful syntactic and semantic regularities, such as RecNN [7, 8] and RNN [9]. 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, Long Short-Term Memory (LSTM) [27] and Gated Recurrent Unit (GRU) [28], as sophisticated recurrent hidden units of RNN, has presented its advantages in many sequence generation problem, such as machine translation [29], speech recognition [30], and text conversation [31]. While, CNN is better to learn non-biased implicit features which has been successfully exploited for many supervised NLP learning tasks as described in Section 1, and various CNN based variants are proposed in the recent works, such as Dynamic Convolutional Neural Network (DCNN) [11], Gated Recursive Convolutional Neural Network (grConv) [32] and Self-Adaptive Hierarchical Sentence model (AdaSent) [33].

In the past few days, Visin et al. [34] have attempted to replace convolutional layer in CNN to learn non-biased features for object recognition with four RNNs, called ReNet, that sweep over lower-layer features in different directions: (1) bottom to top, (2) top to bottom, (3) left to right and (4) right to

left. However, ReNet does not outperform state-of-the-art convolutional neural networks on any of the three benchmark datasets, and it is also a supervised learning model for classification. Inspired by Skip-gram of word2vec [35, 24], Skip-thought model [36] describe an approach for unsupervised learning of a generic, distributed sentence encoder. Similar as Skip-gram model, Skip-thought model trains an encoder-decoder model that tries to reconstruct the surrounding sentences of an encoded sentence and released an off-the-shelf encoder to extract sentence representation. Even some researchers introduce continuous Skip-gram and negative sampling to CNN for learning visual representation in an unsupervised manner [37]. This paper, from a new perspective, puts forward a general self-taught CNN framework which can flexibly couple various semantic features and achieve a good performance on one unsupervised learning task, short text clustering.

3. Methodology

Assume that we are given a dataset of n training texts denoted as: $\mathbf{X} = \{\mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^{d \times 1}\}_{i=1,2,...,n}$, where d is the dimensionality of the original BoW representation. Denote its tag set as $\mathbf{T} = \{1, 2, ...C\}$ and the pre-trained word embedding set as $\mathbf{E} = \{\mathbf{e}(w_i) : \mathbf{e}(w_i) \in \mathbb{R}^{d_w \times 1}\}_{i=1,2,...,|V|}$, where d_w is the dimensionality of word vectors and $|V|$ is the vocabulary size. In order to learn the r-dimensional deep feature representation $\mathbf{h}$ from CNN in an unsupervised manner, some unsupervised dimensionality reduction methods $f_{dr}(\mathbf{X})$ are employed to guide the learning of CNN model. Our goal is to cluster these texts $\mathbf{X}$ into clusters $\mathbb{C}$ based on the learned deep feature representation while preserving the semantic consistency.

As depicted in Figure 1, the proposed framework consist of three components, deep convolutional neural network (CNN), unsupervised dimensionality reduction function and K-means module. In the rest sections, we first present the first two components respectively, and then give the trainable parameters and the objective function to learn the deep feature representation. Finally, the
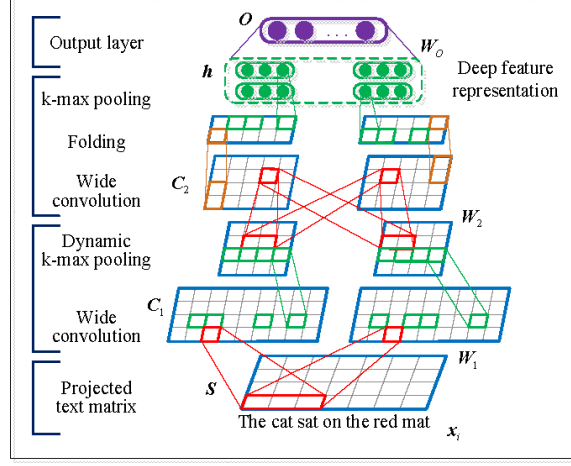
Figure 2: The architecture of dynamic convolutional neural network [11]. An input text is first projected to a matrix feature by looking up word embeddings, and then goes through wide convolutional layers, folding layers and k-max pooling layers, which provides a deep feature representation before the output layer.

last section describe how to perform clustering on the learned features.

3.1. Deep Convolutional Neural Networks

In this section, we briefly review one popular deep convolutional neural network, Dynamic Convolutional Neural Network (DCNN) [11] as an instance of CNN in the following sections, which as the foundation of our proposed method has been successfully proposed for the completely supervised learning task, text classification.

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, 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 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 $f(\cdot) : \mathbb{R}^{d \times 1} \to \mathbb{R}^{r \times 1}$ $(d \gg r)$ which transforms an input raw text $\mathbf{x}$ to a r-dimensional deep representation $\mathbf{h}$. There are three basic operations described as follows:

8

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

Folding In this operation, every two rows in a feature map are simply summed component-wisely. For a map of d_w rows, folding returns a map of $d_w/2$ rows, thus halving the size of the representation and yielding a matrix feature $\hat{\mathbf{C}} \in \mathbb{R}^{(d_w/2) \times (s+m-1)}$. Note that folding operation does not introduce any additional parameters.

Dynamic k-max pooling Assuming the pooling parameter as k, k-max pooling selects the sub-matrix $\bar{\mathbf{C}} \in \mathbb{R}^{(d_w/2) \times k}$ of the k highest values in each row of the matrix $\hat{\mathbf{C}}$. For dynamic k-max pooling, the pooling parameter k is dynamically selected in order to allow for a smooth extraction of higher-order and longer-range features [11]. 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.

3.2. Unsupervised Dimensionality Reduction

As described in Figure 1, the dimensionality reduction function is defined as follows:

$$\mathbf{Y} = f_{dr}(\mathbf{X}), \tag{2}$$

where, $\mathbf{Y} \in \mathbb{R}^{q \times n}$ are the q-dimensional reduced latent space representations. Here, we take four popular dimensionality reduction methods as examples in our framework.

Average Embedding (AE): This method directly averages the word embeddings which are respectively weighted with TF and TF-IDF. Huang et al. [38] used this strategy as the global context in their task, and Socher et al. [8] and

Lai et al. [10] used this method for text classification. The weighted average of all word vectors in one text can be computed as follows:

$$\mathbf{Y}(x_i) = \frac{\sum_{i=1}^{k} \mathbf{w}(w_i) \cdot \mathbf{e}(w_i)}{\sum_{i=1}^{k} \mathbf{w}(w_i)}, \tag{3}$$

where $\mathbf{w}(w_i)$ can be any weighting function that captures the importance of word w_i in the text $\mathbf{x}$.

Latent Semantic Analysis (LSA): LSA [18] is the most popular global matrix factorization method, which applies a dimension reducing linear projection, Singular Value Decomposition (SVD), of the corresponding term/document matrix. Suppose the rank of $\mathbf{X}$ is $\hat{r}$, LSA decompose $\mathbf{X}$ into the product of three other matrices:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{4}$$

where $\mathbf{\Sigma} = diag(\sigma_1, ..., \sigma_{\hat{r}})$ and $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{\hat{r}}$ are the singular values of $\mathbf{X}$, $\mathbf{U} \in \mathbb{R}^{d \times \hat{r}}$ is a set of left singular vectors and $\mathbf{V} \in \mathbb{R}^{n \times \hat{r}}$ is a set of right singular vectors. LSA uses the top q vectors in $\mathbf{U}$ as the transformation matrix to embed the original text features into a q-dimensional subspace $\mathbf{Y}$ [18].

Laplacian Eigenmaps (LE): The top eigenvectors of graph Laplacian, defined on the similarity matrix of texts, are used in the method, which can discover the manifold structure of the text space [19]. In order to avoid storing the dense similarity matrix, many approximation techniques are proposed to reduce the memory usage and computational complexity for LE. There are two representative approximation methods, sparse similarity matrix and Nyström approximation. Following previous studies [39, 14], we select the former technique to construct the $n \times n$ local similarity matrix $\mathbf{A}$ by using heat kernel as follows:

$$A_{ij} = \begin{cases} \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}), & if \ \mathbf{x}_i \in \mathbf{N}_k(\mathbf{x}_j) \ or \ vice \ \text{versa} \\ 0, & otherwise \end{cases} \tag{5}$$

where, σ is a tuning parameter (default is 1) and $\mathbf{N}_k(\mathbf{x})$ represents the set of k-nearest-neighbors of $\mathbf{x}$. By introducing a diagonal $n \times n$ matrix $\mathbf{D}$ whose entries are given by $D_{ii} = \sum_{j=1}^{n} A_{ij}$, the graph Laplacian $\mathbf{L}$ can be computed by $(\mathbf{D} - \mathbf{A})$. The optimal $q \times n$ real-valued matrix $\mathbf{Y}$ can be obtained by solving the following objective function:

$$\begin{aligned} &\underset{\mathbf{Y}}{\arg\min} && tr(\mathbf{Y}\mathbf{L}\mathbf{Y^T}) \\ &\text{subject to} && \mathbf{Y}\mathbf{D}\mathbf{Y^T} = \mathbf{I} \\ & && \mathbf{Y}\mathbf{D}\mathbf{1} = \mathbf{0} \end{aligned} \qquad (6)$$

where $tr(\cdot)$ is the trace function, $\mathbf{Y}\mathbf{D}\mathbf{Y^T} = \mathbf{I}$ requires the different dimensions to be uncorrelated, and $\mathbf{Y}\mathbf{D}\mathbf{1} = \mathbf{0}$ requires each dimension to achieve equal probability as positive or negative).

Locality Preserving Indexing (LPI): This method extends LE to deal with unseen texts by approximating the linear function $\mathbf{Y} = \mathbf{W}_{LPI}^T \mathbf{X}$ [14], and the subspace vectors are obtained by finding the optimal linear approximations to the eigenfunctions of the Laplace Beltrami operator on the Riemannian manifold [20]. Similar as LE, we first construct the local similarity matrix $\mathbf{A}$, then the graph Laplacian $\mathbf{L}$ can be computed by $(\mathbf{D} - \mathbf{A})$, where D_{ii} measures the local density around $\mathbf{x}_i$ and is equal to $\sum_{j=1}^{n} A_{ij}$. Compute the eigenvectors $\mathbf{a}$ and eigenvalues λ of the following generalized eigen-problem:

$$\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{a} = \lambda\mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{a}. \qquad (7)$$

The mapping function $\mathbf{W}_{LPI} = [\mathbf{a}_1, ..., \mathbf{a}_q]$ can be obtained and applied to the unseen data [39].

All of the above methods claim a better performance in capturing semantic similarity between texts in the reduced latent space representation $\mathbf{Y}$ than in the original representation $\mathbf{X}$, while the performance of short text clustering can be further enhanced with the help of our framework, self-taught CNN.

11

3.3. Learning

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

$$\mathbf{O} = \mathbf{W}_O \mathbf{h}, \tag{8}$$

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 incorporate the latent semantic features $\mathbf{Y}$, we first binary the real-valued vectors $\mathbf{Y}$ to the binary codes $\mathbf{B}$ by setting the threshold to be the media vector $median(\mathbf{Y})$. Then, the output vector $\mathbf{O}$ is used to fit the binary codes $\mathbf{B}$ via q logistic operations as follows:

$$p_i = \frac{\exp(\mathbf{O}_i)}{1 + \exp(\mathbf{O}_i)}. \tag{9}$$

All parameters to be trained are defined as θ.

$$\theta = \{\mathbf{E}, \tilde{\mathbf{W}}, \mathbf{W}_O\}. \tag{10}$$

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

$$J(\theta) = \sum_{i=1}^{n} \log p(\mathbf{b}_i | \mathbf{x}_i, \theta). \tag{11}$$

Following the previous work [11], we train the network with mini-batches by back-propagation and perform the gradient-based optimization using the Adagrad update rule [40]. For regularization, we employ dropout with 50% rate to the penultimate layer [11, 41].

3.4. 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.

Dataset	C	Num.	Len.	$|V|$
SearchSnippets	8	12,340	17.88/38	30,642
StackOverflow	20	20,000	8.31/34	22,956
Biomedical	20	20,000	12.88/53	18,888

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

4. Experiments

4.1. Datasets

We test our proposed approach on three public short text datasets. The summary statistics and semantic topics of these datasets are described in Table 1 and Table 2.

SearchSnippets[2]. This dataset was selected from the results of web search transaction using predefined phrases of 8 different domains by Phan et al. [42].

StackOverflow. We use the challenge data published in Kaggle.com[3]. The raw 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 as in Table 2.

Biomedical. We use the challenge data published in BioASQ's official website[4]. In our experiments, we randomly select 20, 000 paper titles from 20 different MeSH[5] major topics as in Table 2. As described in Table 1, the max length of selected paper titles is 53^6.

For these datasets, we randomly select 10% of data as the development set. Since SearchSnippets has been pre-processed by Phan et al. [42], we do not

---

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

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

[4]http://participants-area.bioasq.org/.

[5]http://en.wikipedia.org/wiki/Medical_Subject_Headings.

[6]http://www.ncbi.nlm.nih.gov/pubmed/207752.

SearchSnippets: 8 different domains			
business	computers	health	education
culture	engineering	sports	politics
StackOverflow: 20 semantic tags			
svn	oracle	bash	apache
excel	matlab	cocoa	visual-studio
osx	wordpress	spring	hibernate
scala	sharepoint	ajax	drupal
qt	haskell	linq	magento
Biomedical: 20 MeSH major topics			
aging	chemistry	cats	erythrocytes
glucose	potassium	lung	lymphocytes
spleen	mutation	skin	norepinephrine
insulin	prognosis	risk	myocardium
sodium	mathematics	swine	temperature

Table 2: Description of semantic topics (that is, tags/labels) from the three text datasets used in our experiments.

further process this dataset. In StackOverflow, texts contain lots of computer terminology, and symbols and capital letters are meaningful, thus we do not do any pre-processed procedures. For Biomedical, we remove the symbols and convert letters into lower case.

4.2. Pre-trained Word Vectors

We use the publicly available word2vec[7] tool to train word embeddings, and the most parameters are set as same as Mikolov et al. [24] to train word vectors on Google News setting[8], except of vector dimensionality using 48 and

---

[7]https://code.google.com/p/word2vec/.

[8]https://groups.google.com/d/msg/word2vec-toolkit/lxbl_MB29Ic/NDLGId3KPNEJ.

Dataset	$|V|$	$|T|$
SearchSnippets	23,826 (77%)	211,575 (95%)
StackOverflow	19,639 (85%)	162,998 (97%)
Biomedical	18,381 (97%)	257,184 (99%)

Table 3: Coverage of word embeddings on three datasets. $|V|$ is the vocabulary size and $|T|$ is the number of tokens.

minimize count using 5. For SearchSnippets, we train word vectors on Wikipedia dumps[9]. For StackOverflow, we train word vectors on the whole corpus of the StackOverflow dataset described above which includes the question titles and post contents. For Biomedical, we train word vectors on all titles and abstracts of 2014 training articles. The coverage of these learned vectors on three datasets are listed in Table 3, and the words not present in the set of pre-trained words are initialized randomly.

4.3. Comparisons

In our experiment, some widely used text clustering methods are compared with our approach. Besides K-means, Skip-thought Vectors, Recursive Neural Network and Paragraph Vector based clustering methods, four baseline clustering methods are directly based on the popular unsupervised dimensionality reduction methods as described in Section 3.2. We further compare our approach with some other non-biased neural networks, such as bidirectional RNN. More details are listed as follows:

K-means K-means [43] on original keyword features which are respectively weighted with term frequency (TF) and term frequency-inverse document frequency (TF-IDF).

---

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

Skip-thought Vectors (SkipVec) This baseline [36] gives an off-the-shelf encoder to produce highly generic sentence representations. The encoder[10] is trained using a large collection of novels and provides three encoder modes, that are unidirectional encoder (SkipVec (Uni)) with 2,400 dimensions, bidirectional encoder (SkipVec (Bi)) with 2,400 dimensions and combined encoder (SkipVec (Combine)) with SkipVec (Uni) and SkipVec (Bi) of 2,400 dimensions each. K-means is employed on the these vector representations respectively.

Recursive Neural Network (RecNN) In [7], the tree structure is firstly greedy approximated via unsupervised recursive autoencoder. Then, semi-supervised recursive autoencoders are used to capture the semantics of texts based on the predicted structure. In order to make this recursive-based method completely unsupervised, we remove the cross-entropy error in the second phrase to learn vector representation and subsequently employ K-means on the learned vectors of the top tree node and the average of all vectors in the tree.

Paragraph Vector (Para2vec) K-means on the fixed size feature vectors generated by Paragraph Vector (Para2vec) [26] which is an unsupervised method to learn distributed representation of words and paragraphs. In our experiments, we use the open source software[11] released by Mesnil et al. [44].

Average Embedding (AE) K-means on the weighted average vectors of the word embeddings which are respectively weighted with TF and TF-IDF. The dimension of average vectors is equal to and decided by the dimension of word vectors used in our experiments.

Latent Semantic Analysis (LSA) K-means on the reduced subspace vectors generated by Singular Value Decomposition (SVD) method. The dimension of subspace is default set to the number of clusters, we also iterate the dimensions ranging from 10:10:200 to get the best performance, that is 10 on SearchSnippets, 20 on StackOverflow and 20 on Biomedical in our experiments.

---

[10]https://github.com/ryankiros/skip-thoughts.

[11]https://github.com/mesnilgr/iclr15.

Laplacian Eigenmaps (LE) This baseline, using Laplacian Eigenmaps and subsequently employing K-means algorithm, is well known as spectral clustering [45]. The dimension of subspace is default set to the number of clusters [19, 39], we also iterate the dimensions ranging from 10:10:200 to get the best performance, that is 20 on SearchSnippets, 70 on StackOverflow and 30 on Biomedical in our experiments.

Locality Preserving Indexing (LPI) This baseline, projecting the texts into a lower dimensional semantic space, can discover both the geometric and discriminating structures of the original feature space [39]. The dimension of subspace is default set to the number of clusters [39], we also iterate the dimensions ranging from 10:10:200 to get the best performance, that is 20 on SearchSnippets, 80 on StackOverflow and 30 on Biomedical in our experiments.

bidirectional RNN (bi-RNN) We replace the CNN model in our framework as in Figure 1 with some bi-RNN models. Particularly, LSTM and GRU units are used in the experiments. In order to generate the fixed-length document representation from the variable-length vector sequences, for both bi-LSTM and bi-GRU based clustering methods, we further utilize three pooling methods: last pooling (using the last hidden state), mean pooling and element-wise max pooling. These pooling methods are respectively used in the previous works [46, 28], [47] and [10]. For regularization, the training gradients of all parameters with an $l2$ norm larger than 40 are clipped to 40, as the previous work [48].

4.4. 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 [39, 49]. Given a text $\mathbf{x}_i$, let c_i and t_i be the obtained cluster label and the label provided by the corpus, respectively. Accuracy is defined as:

$$ACC = \frac{\sum_{i=1}^{n} \delta(t_i, map(c_i))}{n}, \tag{12}$$

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 [50].

Normalized mutual information [51] between tag/label set $\mathbf{T}$ and cluster set $\mathbb{C}$ is a popular metric used for evaluating clustering tasks. It is defined as follows:

$$NMI(\mathbf{T}, \mathbb{C}) = \frac{MI(\mathbf{T}, \mathbb{C})}{\sqrt{H(\mathbf{T})H(\mathbb{C})}}, \tag{13}$$

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

4.5. Hyperparameter Settings

The most of parameters are set uniformly for these datasets. Following previous study [39], the number of nearest neighbors in Eqn. (5) is fixed to 15 when constructing the graph structures for LE and LPI. For CNN model, the networks has two convolutional layers. The widths of the convolutional filters are both 3. The value of k for the top k-max pooling in Eqn. (1) 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 Eqn. (8) is set same as the best dimensions of subspace in the baseline method, as described in Section 4.3.

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) as Huang [49]. The all subspace vectors are normalized to 1 before applying K-means and the final results reported are the average of 5 trials with all clustering methods on three text datasets.

18

	SearchSnippets	StackOverflow	Biomedical
Method	ACC (%)	ACC (%)	ACC (%)
K-means (TF)	24.75±2.22	13.51±2.18	15.18±1.78
K-means (TF-IDF)	33.77±3.92	20.31±3.95	27.99±2.83
SkipVec (Uni)	28.23±1.08	08.79±0.19	16.44±0.50
SkipVec (Bi)	29.24±1.57	09.59±0.15	16.11±0.60
SkipVec (Combine)	33.58±1.95	09.34±0.24	16.27±0.33
RecNN (Top)	21.21±1.62	13.13±0.80	13.73±0.67
RecNN (Ave.)	65.59±5.35	40.79±1.38	37.05±1.27
RecNN (Top+Ave.)	65.53±5.64	40.45±1.60	36.68±1.29
Para2vec	69.07±2.53	32.55±0.89	41.26±1.22
STC^2-AE	68.34±2.51	40.05±1.77	37.44±1.19
STC^2-LSA	73.09±1.45	35.81±1.80	38.47±1.55
STC^2-LE	**77.09±3.99**	**51.13±2.80**	**43.62±1.00**
STC^2-LPI	**77.01±4.13**	**51.14±2.92**	43.00±1.25

Table 4: Comparison of ACC of our proposed methods and three clustering methods on three datasets. For RecNN (Top), K-means is conducted on the learned vectors of the top tree node. For RecNN (Ave.), K-means is conducted on the average of all vectors in the tree. More details about the baseline setting are described in Section 4.3

4.6. Results and Analysis

In Table 4 and Table 5, we report the ACC and NMI performance of our proposed approaches and four baseline methods, K-means, SkipVec, RecNN and Para2vec based clustering methods. Intuitively, we get a general observation that (1) BoW based approaches, including K-means (TF) and K-means (TF-IDF), and SkipVec based approaches perform not well; (2) RecNN based approaches, both RecNN (Ave.) and RecNN (Top+Ave.), do better; (3) Para2vec makes a comparable performance with the most baselines; and (4) the evaluation clearly demonstrate the superiority of our proposed methods STC^2. It is an expected results. For SkipVec based approaches, the off-the-shelf encoders are

	SearchSnippets	StackOverflow	Biomedical
Method	NMI (%)	NMI (%)	NMI (%)
K-means (TF)	09.03±2.30	07.81±2.56	09.36±2.04
K-means (TF-IDF)	21.40±4.35	15.64±4.68	25.43±3.23
SkipVec (Uni)	10.98±0.93	02.24±0.13	10.52±0.41
SkipVec (Bi)	09.27±0.29	02.89±0.20	10.15±0.59
SkipVec (Combine)	13.85±0.78	02.72±0.34	10.72±0.46
RecNN (Top)	04.04±0.74	09.90±0.96	08.87±0.53
RecNN (Ave.)	50.55±1.71	40.58±0.91	33.85±0.50
RecNN (Top+Ave.)	50.44±1.84	40.21±1.18	33.75±0.50
Para2vec	50.51±0.86	27.86±0.56	34.83±0.43
STC^2-AE	54.01±1.55	38.22±1.31	33.58±0.48
STC^2-LSA	54.53±1.47	34.38±1.12	33.90±0.67
STC^2-LE	**63.16±1.56**	**49.03±1.46**	38.05±0.48
STC^2-LPI	62.94±1.65	**49.08±1.49**	**38.18±0.47**

Table 5: Comparison of NMI of our proposed methods and three clustering methods on three datasets. For RecNN (Top), K-means is conducted on the learned vectors of the top tree node. For RecNN (Ave.), K-means is conducted on the average of all vectors in the tree. More details about the baseline setting are described in Section 4.3

trained on the BookCorpus datasets [52], and then applied to our datasets to extract the sentence representations. The SkipVec encoders can produce generic sentence representations but may not perform well for specific datasets, in our experiments, StackOverflow and Biomedical datasets consist of many computer terms and medical terms, such as "ASP.NET", "XML", "C#", "serum" and "glycolytic". When we take a more careful look, we find that RecNN (Top) does poorly, even worse than K-means (TF-IDF). The reason maybe that although recursive neural models introduce tree structure to capture compositional semantics, the vector of the top node mainly captures a biased semantic while the average of all vectors in the tree nodes, such as RecNN (Ave.), can be better

	SearchSnippets	StackOverflow	Biomedical
Method	ACC (%)	ACC (%)	ACC (%)
bi-LSTM (last)	64.50±3.18	46.83±1.79	36.50±1.08
bi-LSTM (mean)	65.85±4.18	44.93±1.83	35.60±1.21
bi-LSTM (max)	61.70±5.10	38.74±1.62	32.83±0.73
bi-GRU (last)	70.18±2.62	43.36±1.46	35.19±0.78
bi-GRU (mean)	70.29±2.61	44.53±1.81	36.75±1.21
bi-GRU (max)	65.69±1.02	**54.40±2.07**	37.23±1.19
LPI (best)	47.11±2.91	38.04±1.72	37.15±1.16
STC²-LPI	**77.01±4.13**	51.14±2.92	**43.00±1.25**

Table 6: Comparison of ACC of our proposed methods and some other non-biased models on three datasets. For LPI, we project the text under the best dimension as described in Section 4.3. For both bi-LSTM and bi-GRU based clustering methods, the binary codes generated from LPI are used to guide the learning of bi-LSTM/bi-GRU models.

to represent sentence level semantic. And we also get another observation that, although our proposed STC²-LE and STC²-LPI outperform both BoW based and RecNN based approaches across all three datasets, STC²-AE and STC²-LSA do just exhibit some similar performances as RecNN (Ave.) and RecNN (Top+Ave.) do in the datasets of StackOverflow and Biomedical.

We further replace the CNN model in our framework as in Figure 1 with some other non-biased models, such as bi-LSTM and bi-GRU, and report the results in Table 6 and Table 7. As an instance, the binary codes generated from LPI are used to guide the learning of bi-LSTM/bi-GRU models. From the results, we can see that bi-GRU and bi-LSTM based clustering methods do equally well, no clear winner, and both achieve great enhancements compared with LPI (best). Compared with these bi-LSTM/bi-GRU based models, the evaluation results still demonstrate the superiority of our approach methods, CNN based clustering model, in the most cases. As the results reported by Visin et al. [34], despite bi-directional or multi-directional RNN models perform

	SearchSnippets	StackOverflow	Biomedical
Method	NMI (%)	NMI (%)	NMI (%)
bi-LSTM (last)	50.32±1.15	41.89±0.90	34.51±0.34
bi-LSTM (mean)	52.11±1.69	40.93±0.91	34.03±0.28
bi-LSTM (max)	46.81±2.38	36.73±0.56	31.90±0.23
bi-GRU (last)	56.00±0.75	38.73±0.78	32.91±0.40
bi-GRU (mean)	55.76±0.85	39.84±0.94	34.27±0.27
bi-GRU (max)	51.11±1.06	**51.10±1.31**	32.74±0.34
LPI (best)	38.48±2.39	27.21±0.88	29.73±0.30
STC^2-LPI	**62.94±1.65**	49.08±1.49	**38.18±0.47**

Table 7: Comparison of NMI of our proposed methods and some other non-biased models on three datasets. For LPI, we project the text under the best dimension as described in Section 4.3. For both bi-LSTM and bi-GRU based clustering methods, the binary codes generated from LPI are used to guide the learning of bi-LSTM/bi-GRU models.

a good non-biased feature extraction, they yet do not outperform state-of-the-art CNN on some tasks.

In order to make clear what factors make our proposed method work, we report the bar chart results of ACC and MNI of our proposed methods and the corresponding baseline methods in Figure 3 and Figure 4. It is clear that, although AE and LSA does well or even better than LE and LPI, especially in dataset of both StackOverflow and Biomedical, STC^2-LE and STC^2-LPI achieve a much larger performance enhancements than STC^2-AE and STC^2-LSA do. The possible reason is that the information the pseudo supervision used to guide the learning of CNN model that make difference. Especially, for AE case, the input features fed into CNN model and the pseudo supervision employed to guide the learning of CNN model are all come from word embeddings. There are no different semantic features to be used into our proposed method, thus the performance enhancements are limited in STC^2-AE. For LSA case, as we known, LSA is to make matrix factorization to find the best subspace ap-
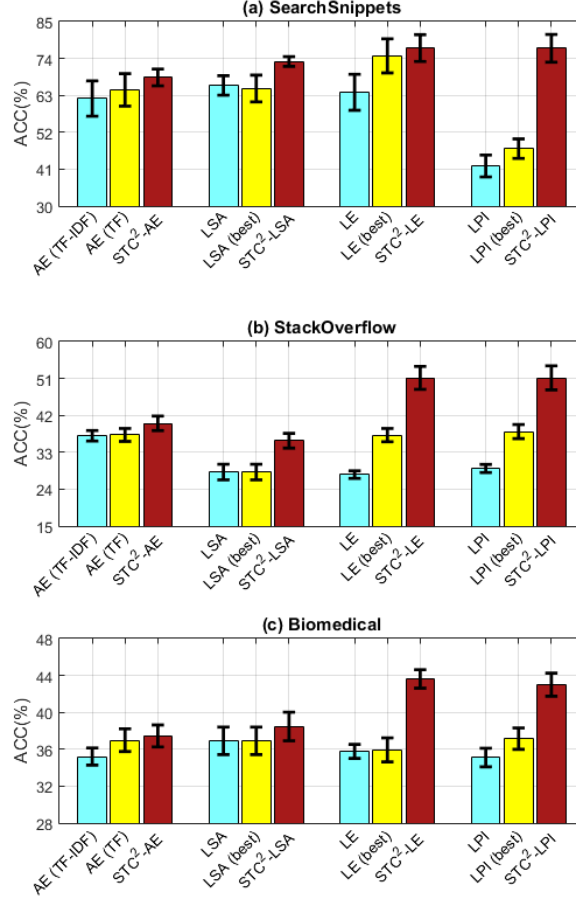
Figure 3: ACC results on three short text datasets using our proposed STC^2 based on AE, LSA, LE and LPI.

proximation of the original feature space to minimize the global reconstruction error. And as [25, 53] recently point out that word embeddings trained with word2vec or some variances, is essentially to do an operation of matrix factorization. Therefore, the information between input and the pseudo supervision in CNN is not departed very largely from each other, and the performance enhancements of STC^2-AE is also not quite satisfactory. For LE and LPI case, as we known that LE extracts the manifold structure of the original feature space, and LPI extracts both geometric and discriminating structure of the original feature space [39]. We guess that our approach STC^2-LE and STC^2-LPI achieve enhancements compared with both LE and LPI by a large margin, because both
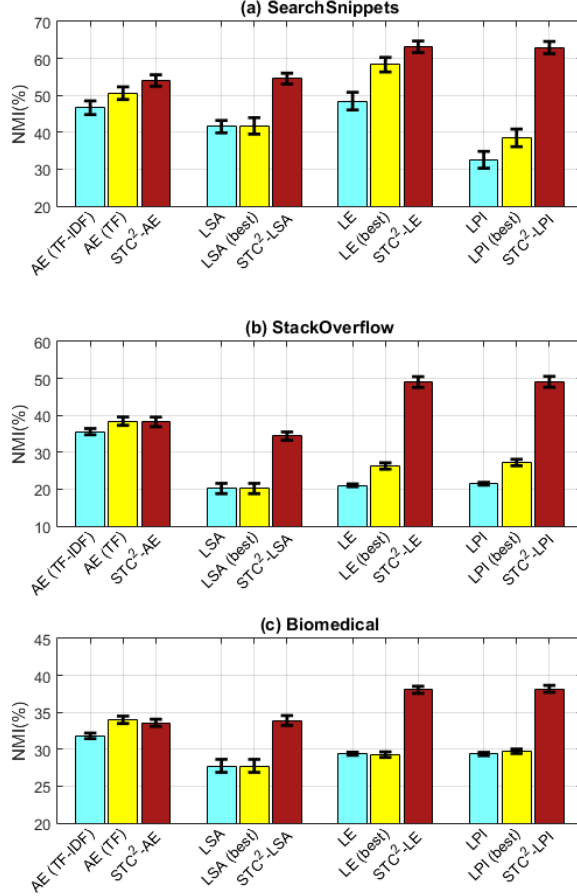
Figure 4: NMI results on three short text datasets using our proposed STC^2 based on AE, LSA, LE and LPI.

of LE and LPI get useful semantic features, and these features are also different from word embeddings used as input of CNN. From this view, we say that our proposed STC has potential to behave more effective when the pseudo supervision is able to get semantic meaningful features, which is different enough from the input of CNN.

Furthermore, from the results of K-means and AE in Table 4-5 and Figure 3-4, we 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 re-

sults without TF-IDF weighting. Meanwhile, we find that LE get quite unusual good performance than LPI, LSA and AE in SearchSnippets dataset, which is not found in the other two datasets. To get clear about this, and also to make a much better demonstration about our proposed approaches and other baselines, we further report 2-dimensional text embeddings on SearchSnippets in Figure 5, using t-SNE[12] [54] to get distributed stochastic neighbor embedding of the feature representations used in the clustering methods. We can see that the results of from AE and LSA seem to be fairly good or even better than the ones from LE and LPI, which is not the same as the results from ACC and NMI in Figure 3-4. Meanwhile, RecNN (Ave.) performs better than BoW (both TF and TF-IDF) while RecNN (Top) does not, which is the same as the results from ACC and NMI in Table 4 and Table 5. Then we guess that both "the same as" and "not the same as" above, is just a good example to illustrate that visualization tool, such as t-SNE, get some useful information for measuring results, which is different from the ones of ACC and NMI. Moreover, from this complementary view of t-SNE, we can see that our STC^2-AE, STC^2-LSA, STC^2-LE, and STC^2-LPI show more clear-cut margins among different semantic topics (that is, tags/labels), compared with AE, LSA, LE and LPI, respectively, as well as compared with both baselines, BoW and RecNN based ones.

From all these results, with three measures of ACC, NMI and t-SNE under three datasets, we can get a solid conclusion that our proposed approaches is an effective approaches to get useful semantic features for short text clustering.

5. Conclusions

With the emergence of social media, short text clustering has become an increasing important task. This paper explores a new perspective to cluster short texts based on deep feature representation learned from the proposed self-taught convolutional neural networks. Our framework can be successfully accomplished
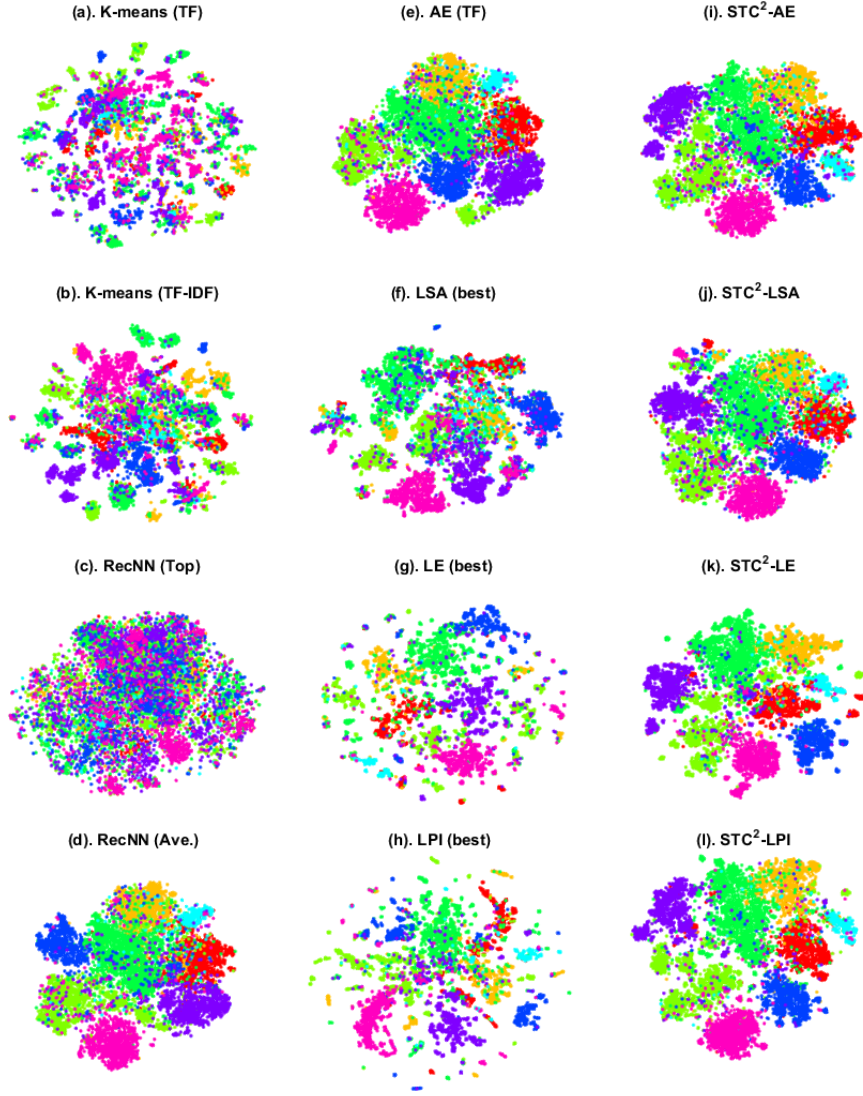
---

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

Figure 5: A 2-dimensional embedding of original keyword features weighted with (a) TF and (b) TF-IDF, (c) vectors of the top tree node in RecNN, (d) average vectors of all tree node in RecNN, (e) average embeddings weighted with TF, subspace features based on (f) LSA, (g) LE and (h) LPI, deep learned features from (i) STC^2-AE, (j) STC^2-LSA, (k) STC^2-LE and (l) STC^2-LPI. All above features are respectively used in K-means (TF), K-means (TF-IDF), RecNN (Top), RecNN (Ave.), AE (TF), LSA(best), LE (best), LPI (best), and our proposed STC^2-AE, STC^2-LSA, STC^2-LE and STC^2-LPI on SearchSnippets. (Best viewed in color)

26

without using any external tags/labels and complicated NLP pre-processing, and and our approach is a flexible framework, in which the traditional dimension reduction approaches could be used to get performance enhancement. Our extensive experimental study on three short text datasets shows that our approach can achieve a significantly better performance. In the future, how to select and incorporate more effective semantic features into the proposed framework would call for more research.

Acknowledgments

References

References

[1] J. Li, A. Ritter, E. Hovy, Weakly supervised user profile extraction from twitter, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 165–174.

[2] J. Wang, Q. Li, Y. P. Chen, Z. Lin, Recommendation in internet forums and blogs, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Uppsala, Sweden, 2010, pp. 257–265.

[3] C. C. Aggarwal, C. Zhai, A survey of text clustering algorithms, in: Mining Text Data, Springer, 2012, pp. 77–128.

[4] S. Banerjee, K. Ramanathan, A. Gupta, Clustering short texts using wikipedia, in: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2007, pp. 787–788.

[5] S. Fodeh, B. Punch, P.-N. Tan, On ontology-driven document clustering using core semantic features, Knowledge and information systems 28 (2) (2011) 395–421.

[6] J. Yin, J. Wang, A dirichlet multinomial mixture model-based approach for short text clustering, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 233–242.

[7] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, C. D. Manning, Semi-supervised recursive autoencoders for predicting sentiment distributions, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2011, pp. 151–161.

[8] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the conference on empirical methods in natural language processing (EMNLP), Vol. 1631, Citeseer, 2013, p. 1642.

[9] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, S. Khudanpur, Extensions of recurrent neural network language model, in: Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, IEEE, 2011, pp. 5528–5531.

[10] S. Lai, L. Xu, K. Liu, J. Zhao, Recurrent convolutional neural networks for text classification, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[11] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014.

[12] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, Relation classification via convolutional deep neural network, in: Proceedings of COLING, 2014, pp. 2335–2344.

[13] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, The Journal of Machine Learning Research 12 (2011) 2493–2537.

[14] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, ACM, 2010, pp. 18–25.

[15] G. Lin, C. Shen, D. Suter, A. v. d. Hengel, A general two-step approach to learning-based hashing, in: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE, 2013, pp. 2552–2559.

[16] R. Raina, A. Battle, H. Lee, B. Packer, A. Y. Ng, Self-taught learning: transfer learning from unlabeled data, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 759–766.

[17] J. Xu, P. Wang, G. Tian, B. Xu, J. Zhao, F. Wang, H. Hao, Short text clustering via convolutional neural networks, in: Proceedings of NAACL-HLT (workshop), 2015, pp. 62–69.

[18] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, R. A. Harshman, Indexing by latent semantic analysis, JAsIs 41 (6) (1990) 391–407.

[19] A. Y. Ng, M. I. Jordan, Y. Weiss, et al., On spectral clustering: Analysis and an algorithm, Advances in neural information processing systems 2 (2002) 849–856.

[20] X. He, P. Niyogi, Locality preserving projections, in: Neural Information Processing Systems, Vol. 16, MIT, 2004, pp. 153–160.

[21] J. Tang, X. Wang, H. Gao, X. Hu, H. Liu, Enriching short text representation in microblog for clustering, Frontiers of Computer Science 6 (1) (2012) 88–101.

[22] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[23] J. Turian, L. Ratinov, Y. Bengio, Word representations: a simple and general method for semi-supervised learning, in: Proceedings of the 48th annual meeting of the association for computational linguistics, Association for Computational Linguistics, 2010, pp. 384–394.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.

[25] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014) 12.

[26] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1188–1196.

[27] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[28] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014).

[29] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.

[30] A. Graves, A.-R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 6645–6649.

[31] L. Shang, Z. Lu, H. Li, Neural responding machine for short-text conversation, arXiv preprint arXiv:1503.02364.

[32] K. Cho, B. van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv preprint arXiv:1409.1259.

[33] H. Zhao, Z. Lu, P. Poupart, Self-adaptive hierarchical sentence model, arXiv preprint arXiv:1504.05070.

[34] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, Y. Bengio, Renet: A recurrent neural network based alternative to convolutional networks, arXiv preprint arXiv:1505.00393.

[35] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.

[36] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, S. Fidler, Skip-thought vectors, in: Advances in Neural Information Processing Systems, 2015, pp. 3276–3284.

[37] X. Wang, A. Gupta, Unsupervised learning of visual representations using videos, arXiv preprint arXiv:1505.00687.

[38] E. H. Huang, R. Socher, C. D. Manning, A. Y. Ng, Improving word representations via global context and multiple word prototypes, in: Proceedings

of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, Association for Computational Linguistics, 2012, pp. 873–882.

[39] D. Cai, X. He, J. Han, Document clustering using locality preserving indexing, Knowledge and Data Engineering, IEEE Transactions on 17 (12) (2005) 1624–1637.

[40] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, The Journal of Machine Learning Research 12 (2011) 2121–2159.

[41] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2014.

[42] X.-H. Phan, L.-M. Nguyen, S. Horiguchi, 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, ACM, 2008, pp. 91–100.

[43] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al., Constrained k-means clustering with background knowledge, in: ICML, Vol. 1, 2001, pp. 577–584.

[44] G. Mesnil, T. Mikolov, M. Ranzato, Y. Bengio, Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews, arXiv preprint arXiv:1412.5335.

[45] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering., in: Advances in Neural Information Processing Systems, Vol. 14, 2001, pp. 585–591.

[46] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, R. Ward, Deep sentence embedding using the long short term memory

network: Analysis and application to information retrieval, arXiv preprint arXiv:1502.06922.

[47] D. Tang, B. Qin, T. Liu, Document modeling with gated recurrent neural network for sentiment classification, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1422–1432.

[48] S. Sukhbaatar, J. Weston, R. Fergus, et al., End-to-end memory networks, in: Advances in neural information processing systems, 2015, pp. 2440–2448.

[49] P. Huang, Y. Huang, W. Wang, L. Wang, Deep embedding network for clustering, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, IEEE, 2014, pp. 1532–1537.

[50] C. H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity, Courier Corporation, 1998.

[51] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E. Y. Chang, Parallel spectral clustering in distributed systems, Pattern Analysis and Machine Intelligence, IEEE Transactions on 33 (3) (2011) 568–586.

[52] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, S. Fidler, Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 19–27.

[53] Y. Li, L. Xu, F. Tian, L. Jiang, X. Zhong, E. Chen, Word embedding revisited: A new representation learning and explicit matrix factorization perspective, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15), AAAI, 2015.

[54] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (2579-2605) (2008) 85.