Attentive Representation Learning with Adversarial Training for Short Text Clustering

Wei Zhang, Chao Dong, Jianhua Yin, and Jianyong Wang

Abstract—Short text clustering has far-reaching effects on semantic analysis, showing its importance for multiple applications such as corpus summarization and information retrieval. However, it inevitably encounters the severe sparsity of short text representation, making the previous clustering approaches still far from satisfactory. In this paper, we present a novel attentive representation learning model for shot text clustering, wherein cluster-level attention is proposed to capture the correlation between text representation and cluster representation. Relying on this, the representation learning and clustering for short text are seamlessly integrated into a unified framework. To further facilitate the model training process, we apply adversarial training to the unsupervised clustering setting, by adding perturbations to the cluster representations. The model parameters and perturbations are optimized alternately through a minimax game. Extensive experiments on three real-world short text datasets demonstrate the superiority of the proposed model over several strong competitors, verifying that adversarial training yields a substantial performance gain.

Index Terms—short text clustering, representation learning, attention mechanism, adversarial training

1 Introduction

Recent years have witnessed the fast-growing trade of short text data in various kinds of social medias, for example, Twitter, Instagram, and Sina Weibo. As a consequence, short text clustering, the task of automatically grouping multiple unlabeled documents into a number of clusters, has become increasingly important. It can benefit multiple content-centric downstream applications, such as event exploration [4], trend detection [14], to name a few. Compared with general text clustering, short text clustering is more challenging. This is because text representation in original lexical space is usually sparse and this issue is further amplified for short text [1]. Thus, the key to success of short text clustering is building an effective representation scheme for short text.

On the basis of early general clustering algorithms such as K-means [8], current developments in short text clustering mostly fall into two branches: Bayesian topic models and deep learning approaches. Topic models [3] realize probabilistic text clustering by assuming that each document is associated with a distribution over topics, and that each topic is a distribution over words. A topic usually corresponds to a cluster. To model short text, some elaborate topic models [26], [29] are presented by revising the text generation process. However, one major limitation still remains within most of these topic models. The input representation of short text is commonly based on bag-of-words assumption and one-hot encoding, which might be sparse and lack expressive ability.

- *W. Zhang and C. Dong are with the School of Computer Science and Technology, East China Normal University, Shanghai 200062, China (e-mail: zhangwei.thu2011@gmail.com; 51174500084@stu.ecnu.edu.cn).*
- *J. Yin is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100086, China (e-mail: jianyong@tsinghua.edu.cn).*
- *J. Wang is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100086, China (e-mail: jianyong@tsinghua.edu.cn).*

Manuscript received XX, XXX; revised XX, XXXX.

In order to leverage the advantage of representation learning for short text clustering, [24], [25] incorporate word embeddings [15] into the deep convolutional neural network [10] to build a three-stage framework. They aim at feeding better text representation to K-means for improving the clustering performance. However, the optimization process is partitioned into each stage separately, making the clustering process not to guide the learning of text representation and the whole framework complicated, which causes sub-optimal results. Some other deep learning approaches for general clustering problems [6], [9], [23], [30] have been trained in an end-to-end fashion. Nevertheless, they are not tailored for textual data or requires a separate step to obtain document-level representations. Specifically, hand-crafted text representations, such as term frequency-inverse document frequency (TF-IDF), are commonly taken as model input. As such, they overlook the learning of word-level semantic space.

In this paper, we concentrate on bridging the gap between representation learning and clustering for short text. Inspired by the roaring success of attention mechanism [2] in natural language processing (NLP), we provide our novel Attentive Representation Learning (ARL) model tailored for the short text clustering task. It leverages low-dimensional word embedding to build dense document-level representation with the simple mean-pooling technique. Cluster-level attention is then proposed to capture correlation between a target document and each cluster, in which the attention weights can be used as evidence to determine cluster assignments. To enable the unsupervised learning of model parameters, we propose to reconstruct the document-level representation through the weighted combination of cluster embeddings. An objective function combining the pairwise ranking loss and point-wise loss is employed to measure the reconstruction gap.

To improve the effectiveness of learning the proposed model for short text, we further introduce the adversarial training [5], [16] to our unsupervised clustering task, naming it as ARL-Adv. Adversarial training requires to feed both original real examples and intentional "adversarial examples" into the model during training process. It has obtained impressive performance in the super-

vised and semi-supervised classification tasks. Concretely, ARL-Adv associates continuous cluster embeddings with adversarial perturbations to form an additional adversarial objective function. As a supplement to the original objective function, adversarial perturbations play a role of adaptive regularization, providing the optimization process with more robustness and beneficial for short text which sometimes contain noise. The model parameters and perturbations are optimized by playing a minimax game. While the model parameters are trained to minimize the reconstruction gap in the integrated objective function, the perturbations are learned to maximize the reconstruction gap in the adversarial objective function.

To sum up, our contributions are as follows:

- We present a novel attentive representation learning approach (ARL) to couple short text representation learning and clustering in a unified framework. By proposing to reconstruct document representation with the weighted combination of cluster embeddings, the model can be optimized in an unsupervised manner while cluster assignments are learned as well.
- For enhancing the effectiveness of learning the model, we introduce adversarial training by customizing adversarial examples (ARL-Adv). To our best knowledge, this is the first study to apply adversarial training into the field of unsupervised clustering setting.
- We conduct extensive experiments on three real-world short text datasets. Two of the datasets are publicly available, and the remaining one is constructed following a recent work. In comparison with several strong baselines, we demonstrate the significant improvements achieved by ARL-Adv, verifying that the adversarial training can further improve performance. To ensure the reproducibility of this paper, we make the model source code and event-related dataset released for evaluation (see Section **??** for details).

2 The Computational Model

The overall architecture of our proposed model ARL-Adv is illustrated with Figure 1. For clarity, we use different types of lines with various colors to indicate the corresponding information flow in ARL-Adv. As shown, it contains document representation with word embedding, cluster-level attention, document reconstruction, adversarial perturbation, and objective functions to be optimized. Particularly, cluster-level attention is inspired by the success of attention mechanism, which performs cluster selection directly with the representations of clusters given document features. Adversarial perturbation empowers the model with robustness. In the rest of this section, we introduce the main mathematical details of ARL-Adv, except the adversarial perturbation (shown with orange rectangles and arcs) which is later illustrated in Section 3.

2.1 Document-Level Representation

For the short text clustering problem, we have a corpus $\mathcal{D}$ containing multiple short documents, i.e., $\{D_1, D_2, \cdots, D_{|\mathcal{D}|}\}$, where $|\mathcal{D}|$ is the size of $\mathcal{D}$. Take the i-th short text $D_i \in \mathcal{D}$ for illustration. It is associated with a matrix based representation $\boldsymbol{D}_i = \{\boldsymbol{x}_t^i\}_{t=1}^{t=l_i}$, where l_i is the length of the document and $\boldsymbol{x}_t^i$ is the column vector of one-hot encoding at position t, with the size equal to the number of words in a pre-specified vocabulary V. Note that here $\boldsymbol{D}_i$ is the original representation of text, while some other strategies such as TF-IDF can be regarded as the hand-crafted feature engineering of text.

To convert each one-hot sparse representation $\boldsymbol{x}_t^i$ $(t \in \{1, \cdots, l_i\})$ to its low-dimensional dense embedding, we first adopt a look-up encoding process:

$$\boldsymbol{w}_t^i = \boldsymbol{E}\boldsymbol{x}_t^i, \tag{1}$$

where $\boldsymbol{E} \in \mathbb{R}^{K \times |V|}$ is a word embedding matrix to be optimized and K is the embedding size. Given these dense word embeddings, we then perform the mean-pooling technique to construct informative representation of the document:

$$\boldsymbol{d}_i = \frac{1}{l_i} \sum_{t=1}^{l_i} \boldsymbol{w}_t^i. \tag{2}$$

We have also tested some other document representation methods, such as attention based document representation [27], recurrent sequential representation [21], and convolutional neural network [11]. However, no significant improvements are observed, while heavy computational costs are incurred. Indeed, ARL-Adv implicitly assumes through this way that words in a document contribute almost equally to the semantic meaning of the document. In LDA, the topic for each word in a document is separately determined from their sampled topics. However, in ARL-Adv there only exists one latent topic z_m, as we will show in Equation (3). Thus, ARL-Adv is similar to the spirit of DMM [29] for handling short text, which assumes that words in the same document belong to the same topic. In addition, our local tests show that ARL-Adv does not achieve so strong performance for long text as for short text. Consequently, we deem that mean-pooling is more suitable for short text clustering.

2.2 Cluster-Level Attention

To represent the presumed M clusters for the specific corpus $\mathcal{D}$, a cluster embedding matrix $\boldsymbol{C} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_M] \in \mathbb{R}^{K \times M}$ is established. As $\boldsymbol{d}_i$ has contained useful global semantic information for the i-th document, leveraging it to select a suitable cluster can be feasible, which is the basic motivation of cluster-level attention. In general, attention function takes a query and a set of key-value pairs as inputs. Each attention score characterizes the compatibility of its corresponding key with the query [22], and is later used for the weighted combination of all values. We let both key and value correspond to the representation of cluster embedding, and regard the document embedding as query. It is intuitive to perform selection based on a probability distribution $p(z_m|\boldsymbol{d}_i)$ over each cluster z_m where $m \in \{1, 2, \cdots, M\}$. Formally, the distribution is defined as follows:

$$z_m \sim p(z_m|\boldsymbol{d}_i) = \frac{\exp(\boldsymbol{c}_m^\top \boldsymbol{d}_i)}{\sum_{m'=1}^{M} \exp(\boldsymbol{c}_{m'}^T \boldsymbol{d}_i)}. \tag{3}$$

The above computational manner could be regarded as cluster-level attention, which captures the document-cluster interaction. We denote $p(z_m|\boldsymbol{d}_i)$ as $p_{z_m}^i$ for short. By referring to the probability values $p_{z_m}^i$ for different m, it is sufficient to identify the most relevant cluster(s) given $\boldsymbol{d}_i$. Besides, each probability value could be employed to rescale the representation of the corresponding clusters for $\boldsymbol{d}_i$, yielding $\hat{\boldsymbol{C}}_i = [\hat{\boldsymbol{c}}_1^i, \hat{\boldsymbol{c}}_2^i, \cdots, \hat{\boldsymbol{c}}_M^i]$, where $\hat{\boldsymbol{c}}_m^i = p_{z_m}^i \boldsymbol{c}_m$.

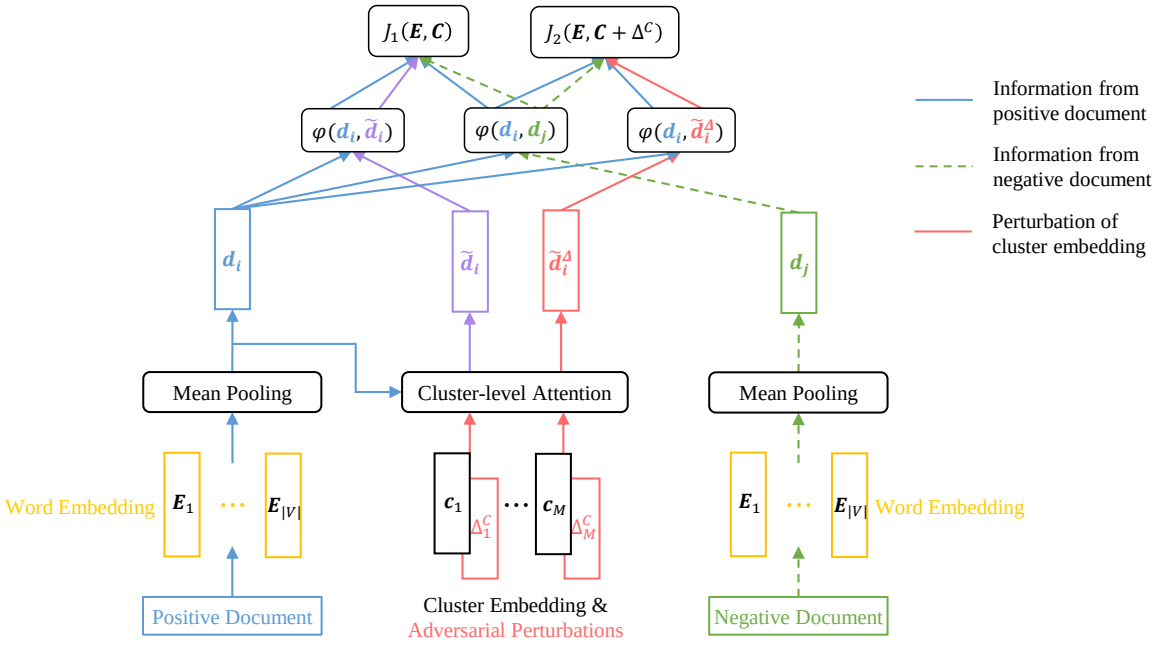

Fig. 1: The architecture of the proposed ARL-Adv. Three types of marks are used to label different information flow. $\tilde{d}_i^{\Delta}$ is the reconstruction of d_i obtained with adversarial perturbations.

2.3 Document Reconstruction with Cluster Embedding

Since text clustering is indeed an unsupervised learning problem, we leverage the cluster embedding to reconstruct the document representation, which is later utilized to guide the training of the whole model.

By assuming the reconstructed representation of the i-th document as $\widetilde{\boldsymbol{d}}_i$, we calculate it based on a linear combination of the document-dependent cluster embedding $\hat{\boldsymbol{C}}_i$:

$$\widetilde{\boldsymbol{d}}_i = \sum_{m=1}^{M} \hat{\boldsymbol{c}}_m^i. \tag{4}$$

Im reasonable, all I need is attention

Benefiting from the different contributions of clusters, we can rebuild an expressive representation of the document. For a specific document, the derivation of cluster-level attention in Equation (3) encourages those clusters that are more similar to the document representation to contribute more in the linear combination in Equation (4). When the derived reconstruction is driven to be closer to the document representation, the cluster-level attention may gradually favor one single cluster as the training goes on. The visualization in Figure **??** can partly explain the consequence. Actually, we find in the experiments that the average probabilities of the predicted maximum cluster are over 0.99 for all datasets. This means each document is highly concentrated to a single cluster, rather than an arbitrary value.

2.4 Objective Function

Given the obtained document-level representation $\boldsymbol{d}_i$ and its reconstructed representation $\widetilde{\boldsymbol{d}}_i$ from cluster embedding, we define their relevance score to be the cosine similarity between the two representations:

$$\varphi(\boldsymbol{d}_i,\widetilde{\boldsymbol{d}}_i) = \frac{\boldsymbol{d}_i^{\top}\widetilde{\boldsymbol{d}}_i}{\|\boldsymbol{d}_i\|\|\widetilde{\boldsymbol{d}}_i\|}. \tag{5}$$

We deem that when the relevance score gets larger, the result of the reconstruction is better. To achieve this, we propose a novel hybrid objective function, consisting of two parts: a pairwise ranking part and a relevance maximization part.

As for the first part, our model seeks to minimize a margin based pairwise ranking loss [19]. Specifically, for the i-th document, we sample several other documents from the corpus and regard them as a pseudo negative document set $\mathcal{N}_i$. And for the document $D_j \in \mathcal{N}_t$, we can calculate $\varphi(\boldsymbol{d}_i,\boldsymbol{d}_j)$ based on Equation (5), where the embedding $\boldsymbol{d}_j$ is obtained with the same procedure as shown in Equation (2). Then we formulate the pairwise ranking loss for the t-th document as follows:

$$\mathcal{L}_1(i;\boldsymbol{E},\boldsymbol{C}) = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\max(0,\gamma-\varphi(\boldsymbol{d}_i,\widetilde{\boldsymbol{d}}_i)+\varphi(\boldsymbol{d}_i,\boldsymbol{d}_j)), \tag{6}$$

where $|\mathcal{N}_i|$ denotes the number of pseudo negative documents, and γ is the margin between positive and negative ones, empirically set to 1. We can see minimizing the above loss would make the reconstructed $\widetilde{\boldsymbol{d}}_i$ having a larger relevance score with the original representation $\boldsymbol{d}_i$ than negative samples at the given margin.

As a supplement, we adopt the relevance maximization of $\varphi(\boldsymbol{d}_i,\widetilde{\boldsymbol{d}}_i)$. To keep consistent with the optimization direction of $\mathcal{L}_1$, we define the following loss function:

$$\mathcal{L}_2(i;\boldsymbol{E},\boldsymbol{C}) = -\varphi(\boldsymbol{d}_i,\widetilde{\boldsymbol{d}}_i), \tag{7}$$

which can be regarded as a pointwise loss function, aiming to predict d_i as similar as possible given the combination of all cluster embeddings. By fusing $\mathcal{L}_1(i)$ and $\mathcal{L}_2(i)$, it ensures that $\varphi(\boldsymbol{d}_i,\widetilde{\boldsymbol{d}}_i)$ not only has larger relevance than those of pseudo negative document pairs, but also a large value itself. Taking all

the short documents in the corpus $\mathcal{D}$ into consideration, we define the overall optimization function as follows:

$$J_1(\boldsymbol{E},\boldsymbol{C})=\sum_{i=1}^{|\mathcal{D}|}\big(\mathcal{L}_1(i;\boldsymbol{E},\boldsymbol{C})+\mathcal{L}_2(i;\boldsymbol{E},\boldsymbol{C})\big). \tag{8}$$

From a whole perspective, our model contains word embedding $\boldsymbol{E}$ and cluster embedding $\boldsymbol{C}$ as its parameters. The documents are directly encoded into a low dimensional space through its word embeddings, so that no extraordinary parameters are required. All the traits make our ARL-Adv a relatively simple but highly effective model, which is later demonstrated in the experiments.

3 Adversarial Training

We propose to apply adversarial training to facilitate the representation learning process of ARL-Adv. In essence, adversarial training provides a new objective function based on adversarial perturbations to complement the original optimization procedure. This requires the unsupervised algorithm to perform well on both original samples and adversarial samples, thus benefiting the model in its robustness.

Since typical adversarial perturbations take continuous values and are added to real-valued vectors like images, they are not directly applied to discrete tokens like words. Following [17], we add continuous adversarial perturbations to the embedding level of clusters. It is worth noting that we do not add adversarial perturbations to the word embedding matrix. The reason is that the number of words extends far beyond that of clusters, so that adding perturbations to words instead of clusters entails more parameters. It also causes a change in the information flow of our model, according to Figure 1. We empirically validate that applying perturbations on words does not achieve the same or better performance than adding perturbations to cluster embedding, which is shown in Table 4. Formally, we define the cluster-level adversarial perturbations in ARL-Adv as $\Delta^C \in \mathbb{R}^{K\times M}$. Consequently, we propose our new objective function as formulated below:

$$J_2(\boldsymbol{E},\boldsymbol{C}+\Delta^C)=\sum_{i=1}^{|\mathcal{D}|}\big(\mathcal{L}_1(i;\boldsymbol{E},\boldsymbol{C}+\Delta^C)+\mathcal{L}_2(i;\boldsymbol{E},\boldsymbol{C}+\Delta^C)\big). \tag{9}$$

The optimization of the above objective function can be viewed from two aspects. For learning Δ^C, the optimal condition is as follows:

$$\Delta^{C*}=\arg\max_{\Delta^C} J_2(\boldsymbol{E},\boldsymbol{C}+\Delta^C). \tag{10}$$

In contrary, the optimization of $\boldsymbol{E}$ and $\boldsymbol{C}$ is to minimize J_2. Typically, a norm-based constraint should be adopted to restrict the scale of Δ^C.

By integrating the two objective functions, J_1 and J_2, we define the final optimization target:

$$J(\boldsymbol{E},\boldsymbol{C},\Delta^C)=J_1(\boldsymbol{E},\boldsymbol{C})+\alpha J_2(\boldsymbol{E},\boldsymbol{C}+\Delta^C), \tag{11}$$

$$\boldsymbol{E}^*,\boldsymbol{C}^*,\Delta^{C*}=\arg\min_{\boldsymbol{E},\boldsymbol{C}}\max_{\Delta^C} J(\boldsymbol{E},\boldsymbol{C},\Delta^C), \tag{12}$$

where α controls the relative strength of J_2. As a complement to J_1, J_2 plays a role similar to adaptive regularization methodology, stabilizing the training of the proposed model. The optimization of Equation (11) involves playing a minimax game. At each step, the worst-case perturbation to cluster embedding is first identified to increase the value of J_2 as much as possible. Afterwards, $\boldsymbol{E}$ and $\boldsymbol{C}$ are optimized to be robust to such intentional perturbations in J_2, while retaining good performance in J_1. This procedure confirms to the noisy scenario of short text.

The optimization procedure is summarized in Algorithm 1. As usual, stochastic gradient descent algorithms such as Adam [12] are leveraged to learn the word and clustering embeddings. ARL-Adv calculates the gradients of $\boldsymbol{E}$ and $\boldsymbol{C}$ over Equation (11) through back propagation, and update the parameters accordingly. The learning of adversarial perturbation Δ^C follows the approximated linearizing methodology proposed in [5], leading to fast analytic form solutions. With the adding of L2 norm to each column of Δ^C, i.e., $\|\Delta^C_m\| \le \epsilon$ $(m \in \{1,2,\cdots,M\})$, we can easily derive the following updating rule for the perturbation of each cluster embedding:

$$\Delta^C_m=\epsilon\frac{\boldsymbol{g}^C_m}{\|\boldsymbol{g}^C_m\|},\ \text{ where }\ \boldsymbol{g}^C_m=\alpha\frac{\partial J_2}{\partial \boldsymbol{C}_m}. \tag{13}$$

After finishing the training of ALR-Adv, the cluster assignments are determined based on the attention weights over the learned document embedding matrix $\boldsymbol{C}$, while the adversarial perturbations Δ^C are ignored.

Algorithm 1: Adversarial training for the proposed model.

Input : Short text corpus $\mathcal{D}$, cluster number M, adversarial strength α, adversarial norm ϵ, and other hyper-parameters such as embedding size K.
Output: Word embedding $\boldsymbol{E}$, cluster embedding $\boldsymbol{C}$, cluster-level attention weights $p(z_m|\boldsymbol{d}_i)(\forall m \in \{1,2,\cdots,M\}$ and $i \in \{1,2,\cdots,|\mathcal{D}|\})$.

```
begin
    Initialize word embedding E and cluster embedding C;
    for iter = 1 to max_iter do
        Sample a mini-batch D_batch ∈ D;
        for i ∈ D_batch do
            Constructing document pairs (i, j), where j ∈ D_batch;
            Learning adversarial perturbations Δ^C through Equation 13;
            Updating word embedding E and cluster embedding C by gradient descent of Equation 12;
```

4 Experiments

4.1 Datasets

We utilize three real-world short text datasets . A brief introduction of each dataset and some common text preprocessing procedures are provided below.

- **TREC**: The dataset is from the Text REtrieval Conference[1] on 2011-2015 tweet tracks. They are organized by their corresponding queries, and evaluated into several relevance levels. We retain tweets labeled relevant or highly-relevant to their queries to ensure the quality of labels.

1. http://trec.nist.gov/data/microblog.html

TABLE 1: Detailed statistics of each dataset.

Dataset	#Cluster	#Document	Vocabulary	Avg Len
TREC	128	11142	3503	7.87
GoogleNews	152	11109	6555	6.23
Event	69	26619	3314	8.78

- **GoogleNews**: This dataset is composed of groups of news titles and snippets clawed from Google News[2]. Manual observation has confirmed its favorable grouping quality.
- **Event**: Following [18], we perform extraction of event-related tweets from an off-the-shelf tweet dataset clawed in 2016[3]. Prior knowledge about the events, including the time window, relevant entities and keywords, is fetched from Wikipedia[4].

The first two datasets are publicly available[5], and have been adopted in previous studies [28], [29]. For preprocessing, we utilize typical procedures in text processing, which consists of converting words into lowercase, removing stop words and irregular words, and applying Porter stemming. In addition, words with frequency below 5 are discarded. The detailed statistics of all datasets can be found in Table 1. *#Cluster* and *#Document* refer to the actual number of clusters and documents for each dataset. *Vocabulary* is the number of remaining tokens after preprocessing. *Avg Len* stands for the average length of documents counted in words.

4.2 Comparison Methods

4.2.1 Baselines

We select some representative baselines for comprehensive comparison. They are categorized into state-of-the-art methods for short text clustering (Short), conventional text clustering methods (Conventional), and some advanced deep learning based general clustering models (General). Detailed configurations of the baselines are given below for clarity.

- **LDA** [3]: LDA is a classical and standard generative statistical model which learns topic or cluster distribution for each document.
- **K-means**: It is an early method for general clustering problem, relying on hand-crafted textual features when applied to text data. Two types of feature representations are adopted in our experiments, i.e., TF-IDF and its low-dimensional representation obtained by Principal Component Analysis (PCA).
- **VaDE** [9][6]: VaDE extends canonical variational auto-encoding approaches to support clustering task, by utilizing the idea of the Gaussian mixture model.
- **DEC** [23][7]: It is a deep embedded clustering model that leverages autoencoder, with TF-IDF features as input to map documents into low-dimensional embeddings. Then the mapping function and cluster embedding are refined based on the idea of self-training. Similar to VaDE, DEC has specified transforming and parameter settings for text data.
- **STCC** [24], [25][8]: This model mainly consists of three separate steps. It first trains a convolutional neural network with the help of autoencoder. Afterwards, the well-trained model is employed to get document embedding, which is fed into K-means for final clustering.
- **BTM** [26][9]: BTM regards bi-grams as the representation of short text, and generates them conditioned on different topics.
- **GSDMM** [29][10]: GSDMM is tailored for short text clustering and assumes that each document is generated by one topic, which is fundamentally different from LDA that generates one word from one topic.
- **STC** [6][11]: STC adopts self-training inspired by DEC and using Smoothed Inverse Frequency (SIF) to compute a weighted average of pre-trained word embeddings in a stage independent of optimizing its clustering method.

All the above models are tuned to be suitable for the experimental datasets to ensure a fair comparison. For reducing the impact of noises, all results in our experiments are averaged over 10 runs. We also tried GaussianLDA[12] and GANMM[13], which are introduced in Section **??** However, the results show that GaussianLDA could not obtain competitive clustering performance than other baselines. And training GANMM incurs a heavy computational burden, especially when the cluster number is not small. Thus we omit their details in our experiments.

4.2.2 Variants of the proposed model

To verify the effectiveness of the main components of the proposed model, we consider some variants of the full model, the details of which are listed as follows.

- **ARL-Adv**: This is the full version of our proposed model, which adopts optimization target as Equation (11) and enables adversarial training.
- **ARL**: Only J_1 is employed to train the model in this setting. By comparing ARL with the full model ARL-Adv, we can verify the effectiveness of adversarial training for the unsupervised clustering task.
- **ARL-Adv(no train w)**: This method optimizes parameters other than the Word embedding matrix. Comparison against it can show the necessity of learning word-level semantic representation.
- **ARL-Adv(no train c)**: Cluster embedding matrix is not learned in this model, hoping to partially verify the benefit of combining representation learning and clustering.
- **ARL-Adv w/o** $\mathcal{L}_1$: The pairwise ranking loss $\mathcal{L}_1$ is removed from ARL-Adv. That is, the documents doss not interact with their negative samples.
- **ARL-Adv w/o** $\mathcal{L}_2$: The supplementary pointwise loss $\mathcal{L}_2$ is not considered by ARL-Adv. Both ARL-Adv w/o $\mathcal{L}_2$ and ARL-Adv w/o $\mathcal{L}_2$ still perform adversarial learning.
- **ARL-Random**: This variant takes random noise as perturbations with the same scale level as intentional adversarial perturbations.
- **ARL-Adv(word)**: It shares a very similar spirit to ARL-Adv, except that adversarial perturbation is added to word embeddings instead of cluster embeddings.

For ARL-Adv and its variants, word embeddings are pre-trained on each dataset with word2vec[14] for 200 epochs. We have

2. http://news.google.com
3. https://archive.org/
4. https://en.wikipedia.org
5. https://github.com/jackyin12/MStream.
6. https://github.com/slim1017/VaDE
7. https://github.com/piiswrong/dec
8. https://github.com/jacoxu/STC2
9. https://github.com/xiaohuiyan/BTM
10. https://github.com/jackyin12/GSDMM
11. https://github.com/hadifar/stc_clustering
12. https://github.com/rajarshd/Gaussian_LDA
13. https://github.com/eyounx/GANMM
14. https://radimrehurek.com/gensim/models/word2vec.html

TABLE 2: The default settings for ARL-Adv.

K	$\lvert\mathcal{D}_{batch}\rvert$	α	ϵ
300	64	1.0	50.0

also tried to leverage the power of BERT to initialize the word embeddings and fine-tune BERT along with the training of ALR-Adv. However, no improved performance is observed in the local tests of several hyper-parameter settings. Moreover, we initialize centroids of the clusters by performing K-means on document embeddings, the same as [23] does. Without loss of generality, the default hyper-parameters are set to the ones shown in Table 2. K is the dimenstion of embeddings, and $|\mathcal{D}_{batch}|$ is the size of a batch. Unless otherwise stated, the results will be reported under this setting.

4.3 Evaluation Metrics

We adopt three commonly used metrics for text clustering performance evaluation, i.e., normalized mutual information (NMI) [20], adjusted rand index (ARI) [7], and clustering accuracy (ACC) [23]. Suppose ν_i denotes the number of documents in the i-th true topic and $\widetilde{\nu}_j$ represents the number of documents in the j-th predicted cluster. We use $\bar{\nu}_{ij}$ to denote the number of documents simultaneously appearing in two clusters. In addition, $\mathcal{M}$ denotes the set of all possible one-to-one mappings between the generated clusters and real topics. For an efficient search of the best mapping, Hungarian algorithm [13] can be adopted. Formally, NMI, ARI and ACC can be formulated as:

$$\mathrm{NMI} = \frac{\sum_{ij} \bar{\nu}_{ij} \log \frac{|\mathcal{D}| \cdot \bar{\nu}_{ij}}{\nu_i \widetilde{\nu}_j}}{\sqrt{(\sum_i \nu_i \log \frac{\nu_i}{|\mathcal{D}|})(\sum_j \widetilde{\nu}_j \log \frac{\widetilde{\nu}_j}{|\mathcal{D}|})}}, \tag{14}$$

$$\mathrm{ARI} = \frac{\sum_{ij} \binom{\bar{\nu}_{ij}}{2} - \left[\sum_i \binom{\nu_i}{2} \sum_j \binom{\widetilde{\nu}_j}{2}\right] / \binom{|\mathcal{D}|}{2}}{\frac{1}{2}\left[\sum_i \binom{\nu_i}{2} + \sum_j \binom{\widetilde{\nu}_j}{2}\right] - \left[\sum_i \binom{\nu_i}{2} \sum_j \binom{\widetilde{\nu}_j}{2}\right] / \binom{|\mathcal{D}|}{2}}, \tag{15}$$

$$\mathrm{ACC} = \max_{m \in \mathcal{M}} \frac{\sum_{i=1}^{|\mathcal{D}|} \mathbf{1}\{l_i = m(c_i)\}}{|\mathcal{D}|}. \tag{16}$$

Higher values evaluated by NMI, ARI and ACC indicate better quality in clustering outputs. They all equal to 1 when a perfect match is achieved in cluster assignments on the whole corpus. Both NMI and ARI penalize unnecessary split of documents from the same true cluster to several predicted clusters. ARI further penalizes undesirably merging of documents from different true clusters into the same predicted cluster, making it a more rigorous metric.

4.4 Model Comparison

We report evaluation results in Table 3 for all baselines and ARL-Adv on the adopted datasets. The cluster numbers here are set to the same as in Table 1. From a whole perspective, all the methods do not perform exactly the same on the three metrics. For example, K-means(TF-IDF) gains better results than LDA in terms of NMI and ACC, but not in ARI. This phenomenon shows the necessity of adopting all the three metrics. Since they focus on different properties of the clustering results, using all of them can provide more comprehensive comparisons from distinct perspectives.

As expected, the models belonging to the "Conventional" category do not show competitive performance than other types of baselines, since they are not tailored for short text and do not fully leverage the power of representation learning. LDA is the standard topic modeling approach for general text modeling. Yet, it works poorly here because assigning clusters to words in the same short text can aggravate sparsity, as the corresponding results demonstrate. Although PCA converts the sparse representation of TF-IDF to a continuous low-dimensional space, K-means(PCA) shows no improvements over K-means(TF-IDF), indicating that introducing representation that is independent of the model training process may not help.

Both DEC and VaDE are deep learning based approach, which can learn low-dimensional representation of targets using their original representation, such as the pixel values of images and TF-IDF based representations of texts. Therefore, though they are proposed for general clustering tasks, they may not be limited to texts. They are able to yield better results in most cases than the category of "Conventional" methods.

Among the baselines originally proposed for short text clustering, STCC performs not very well. This shows that dividing the representation learning and clustering process into separate stages tends to be sub-optimal. The reason may lies in that representation learning process lacks proper guidance from the feedback of clustering if they are not learned together. By comparison, GSDMM and BTM perform much better than STCC. In particular, GSDMM outperforms the general deep learning based clustering models in most cases. We attribute this phenomenon to the proper assumption in GSDMM that documents are generated from only one topic, along with its proper approximation in posterior distribution. STC exhibits well performance on TREC (still worse than ARL-Adv) but not so well results on the other two datasets. This is because STC cannot optimize word embeddings when training its clustering method.

On the whole, our full model ARL-Adv consistently achieves best results among all the adopted baselines in terms of the three metrics. This might be attributed to the following reasons: (1) ARL-Adv utilizes word embedding to obtain document embedding and learns word embedding with the optimization of whole model rather than learning it separately; (2) ARL-Adv combines the representation learning and short text clustering in an end-to-end learning fashion; (3) adversarial training adopted in ARL-Adv can effectively improve clustering performance. The last part of Table 3 shows that ARL-Adv improves ARL consistently in the three datasets, especially in GoogleNews and Event, verifying the third reason. In the next section, we empirically demonstrate the benefit of learning word embedding and cluster embedding by ablation study on the model.

4.5 Analysis of the Proposed Model

Table 4 shows the results of ablation study, so we can investigate the role of key components in our proposed model. By first comparing ARL-Adv(no train w) and ARL-Adv(no train c) with ARL-Adv, we can see their performance drops significantly, showing that the learning of word representation and cluster representation, accompanied with the automatic selection of cluster, is indeed indispensable. Besides, ARL-Adv(no train c) behaves better than ARL-Adv(no train w), indicating the training of word embedding is more critical in this sense.

The second part of Table 4 shows the relative influence of the pairwise-ranking loss and pointwise loss. On TREC, the two losses

TABLE 3: Performance comparison over all methods.

Type	Dataset	TREC			GoogleNews			Event		
	Metrics	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
Conventional	LDA	0.7514	0.5897	0.6120	0.7197	0.5074	0.5878	0.6348	0.4557	0.4984
	K-means(TF-IDF)	0.8376	0.3682	0.6463	0.7941	0.2316	0.5897	0.6790	0.3290	0.4473
	K-means(PCA)	0.8289	0.3460	0.6267	0.7724	0.2011	0.5547	0.6680	0.3150	0.4332
General	DEC	0.8664	0.5839	0.6993	0.8505	0.5088	0.6705	0.7423	0.4264	0.5320
	VaDE	0.8712	0.6632	0.7170	0.8389	0.5318	0.6287	0.6620	0.2927	0.4078
Short	STCC	0.8172	0.5758	0.6455	0.7947	0.5262	0.6122	0.6567	0.3797	0.3959
	BTM	0.8759	0.6920	0.7182	0.8656	0.6310	0.7023	0.6781	0.3834	0.4704
	GSDMM	0.8746	0.7453	0.7512	0.8700	0.6782	0.7278	0.7572	0.4937	0.5039
	STC	0.8906	0.7372	0.7589	0.8667	0.6471	0.7169	0.6431	0.3170	0.3982
	ARL	0.9261	0.8428	0.8347	0.8995	0.7897	0.8057	0.8243	0.6078	0.6156
	ARL-Adv	**0.9305**	**0.8486**	**0.8402**	**0.9054**	**0.8214**	**0.8303**	**0.8450**	**0.6728**	**0.6607**

TABLE 4: Ablation study of the proposed model.

Dataset	TREC			GoogleNews			Event		
Metrics	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
ARL-Adv(no train w)	0.9001	0.7257	0.7631	0.8699	0.6483	0.6991	0.7039	0.3783	0.4450
ARL-Adv(no train c)	0.9029	0.7755	0.7954	0.8783	0.7335	0.7688	0.7740	0.6230	0.6072
ARL-Adv w/o $\mathcal{L}_1$	0.9279	0.8408	0.8368	0.9005	0.7881	0.8044	0.6826	0.3915	0.4631
ARL-Adv w/o $\mathcal{L}_2$	0.9286	0.8441	0.8380	0.9033	0.8120	0.8240	0.8176	0.6244	0.6297
ARL	0.9261	0.8428	0.8347	0.8995	0.7897	0.8057	0.8243	0.6078	0.6156
ARL-Random	0.9250	0.8395	0.8330	0.8980	0.7813	0.7998	0.8174	0.6005	0.6064
ARL-Adv(word)	0.9271	0.8458	0.8358	0.9014	0.7925	0.8074	0.8294	0.6387	0.6307
ARL-Adv	**0.9305**	**0.8486**	**0.8402**	**0.9054**	**0.8214**	**0.8303**	**0.8450**	**0.6728**	**0.6607**

play similar roles and removing either of them does not obviously damage the performance. In contrary, the pairwise ranking loss is more crucial for the clustering performance in the other two datasets, and adding pointwise loss could strengthen the clustering of short text.

The third part of Table 4 further explores the effect of cluster-level adversarial perturbations, by comparing it with cluster-level random perturbations (ARL-Random) and word-level adversarial perturbations (ARL-Adv(word)). We can infer from the results that: (1) ARL-Random obtains slightly worse results than ARL. This reveals that simply adding random perturbations without learning may not bring more useful information to the model. (2) ARL-Adv(word) performs slightly better than ARL, but not as well as ARL-Adv, especially in GoogleNews and Event. For example, ARL-Adv improves ARL-Adv(word) by 2.3% and 3% in terms of ACC in the two datasets, respectively.

5 Conclusion

In this paper, we have developed the novel model ARL-Adv. It fuses the representation learning and short text clustering in an end-to-end fashion through the proposed cluster-level attention. Adversarial perturbations are further added to cluster embeddings, enhancing the robustness and effectiveness of model training through a minimax game. Extensive studies on three real-life datasets show that ARL-Adv can achieve superior performance, even compared with the state-of-the-arts for short text clustering and the recently developed deep learning based clustering models. Further analysis into ARL-Adv is performed to explain the contributions of its components.

References

[1] C. C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer, 2012.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022, 2003.

[4] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. C. Aggarwal, and J. Huang. STREAMCUBE: hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream. In *ICDE*, pages 1561–1572, 2015.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. 2015.

[6] A. Hadifar, L. Sterckx, T. Demeester, and C. Develder. A self-training approach for short text clustering. In *Repl4NLP, the 4th Workshop on Representation Learning for NLP*, pages 1–6, 2019.

[7] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[8] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[9] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *IJCAI*, pages 1965–1972, 2017.

[10] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665, 2014.

[11] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[13] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics*, 2(1-2):83–97, 1955.

[14] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *SIGMOD*, pages 1155–1158, 2010.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[16] T. Miyato, A. M. Dai, and I. Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR*, 2017.

[17] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. In *ICLR*, 2016.

[18] A. Ritter, E. Wright, W. Casey, and T. Mitchell. Weakly supervised extraction of computer security events from twitter. In *WWW*, pages 896–905, 2015.

[19] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2(1):207–218, 2014.

[20] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *JMLR*, 3(Dec):583–617, 2002.

[21] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent

neural network for sentiment classification. In *EMNLP*, pages 1422–1432, 2015.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[23] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.

[24] J. Xu, P. Wang, G. Tian, B. Xu, J. Zhao, F. Wang, and H. Hao. Short text clustering via convolutional neural networks. In *NAACL*, pages 62–69, 2015.

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

[26] X. Yan, J. Guo, Y. Lan, and X. Cheng. A biterm topic model for short texts. In *WWW*, pages 1445–1456, 2013.

[27] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489, 2016.

[28] J. Yin, D. Chao, Z. Liu, W. Zhang, X. Yu, and J. Wang. Model-based clustering of short text streams. In *SIGKDD*, pages 2634–2642, 2018.

[29] J. Yin and J. Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *SIGKDD*, pages 233–242, 2014.

[30] Y. Yu and W. Zhou. Mixture of gans for clustering. In *IJCAI*, pages 3047–3053, 2018.