

Deep Feature-Based Text Clustering and its Explanation

Renchu Guan^{ID}, Member, IEEE, Hao Zhang^{ID}, Yanchun Liang^{ID}, Fausto Giunchiglia,
Lan Huang^{ID}, and Xiaoyue Feng^{ID}

Abstract—Text clustering is a critical step in text data analysis and has been extensively studied by the text mining community. Most existing text clustering algorithms are based on the bag-of-words model, which faces the high-dimensional and sparsity problems and ignores text structural and sequence information. Deep learning-based models such as convolutional neural networks and recurrent neural networks regard texts as sequences but lack supervised signals and explainable results. In this paper, we propose a **deep feature-based text clustering (DFTC)** framework that incorporates pretrained text encoders into text clustering tasks. This model, which is based on sequence representations, breaks the dependency on supervision. The experimental results show that our model outperforms classic text clustering algorithms and the state-of-the-art pretrained language model, i.e., BERT, on almost all the considered datasets. In addition, the explanation of the clustering results is significant for understanding the principles of the deep learning approach. Our proposed clustering framework includes an explanation module that can help users understand the meaning and quality of the clustering results.

Index Terms—Deep learning, explanation model, feature extraction, text clustering, transfer learning

1 INTRODUCTION

CLUSTERING models attempt to classify objects based on their similarity in a valid representation. The first step in classic text clustering is to map texts into a bag-of-words-based feature vector space, which is the most commonly used text feature representation. The vector space models have been widely applied in several fields, such as document organization [1], corpus summarization [2], and content-based recommender systems [3]. Specialized clustering algorithms, such as K-means clustering, are then applied in the given feature space to group text into clusters [2]. However, the high-dimensional bag-of-words feature matrix does not record the text's sequence information or rich contextual information. Moreover, when the text is short, the bag-of-words features are sparse, making it difficult for the model to infer the semantics of the text. Several text-feature enhancement models are available for text clustering. For

example, Guan *et al.* [4] proposed a similarity metric for text clustering to capture the structural information of texts, and Song *et al.* [5] applied a concept knowledge base to extend text features and thus enhanced the semantics of the representation. However, these models are still based on feature space models and thus cannot solve the problem of poor semantic understanding.

Different from feature-based text clustering algorithms, model-based clustering algorithms view the clustering process as a generative model. For example, in the latent Dirichlet allocation (LDA) model, topics are first generated from texts; then, words in the text are generated from topics [6]. LDA can be regarded as a text clustering model because it computes a posterior topic distribution given a text's word distribution. The collapsed Gibbs sampling algorithm for the Dirichlet multinomial mixture model (GSDMM) first generates a cluster label; then, the words in the text are generated from the label [7]. These generative models consider only the words in the current text and ignore all irrelevant words in the vocabulary. Hence, a generative model avoids the processing of high-dimensional and sparse feature matrices. However, these models assume that the words in a given text are independent, and they ignore the information contained in the word sequence order, which is essential for understanding a document. For example, the two sentences "You trust him." and "You betray his trust." have entirely different semantics, but generative or bag-of-words-based models cannot distinguish the word trust because of the loss of contextual information.

Taking sequence information and contextual information into account when designing a model will facilitate the model's text understanding ability and thus improve the model's clustering performance. In recent studies, many deep learning-based text representation models have been proposed

- Renchu Guan and Yanchun Liang are with the Key Laboratory for Symbol Computation and Knowledge Engineering of the Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun 130012, China, and also with the Zhuhai Laboratory of Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, Zhuhai College, Jilin University, Zhuhai 519041, China. E-mail: {guanrenchu, ycliang}@jlu.edu.cn.
- Hao Zhang, Lan Huang, and Xiaoyue Feng are with the Key Laboratory for Symbol Computation and Knowledge Engineering of the Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun 130012, China. E-mail: haozhang17@mails.jlu.edu.cn, {huanglan, fengxy}@jlu.edu.cn.
- Fausto Giunchiglia is with the DISI, University of Trento, 38122 Trento, Italy. E-mail: fausto@disi.unitn.it.

Manuscript received 25 Dec. 2019; revised 30 Aug. 2020; accepted 29 Sept. 2020.
Date of publication 6 Oct. 2020; date of current version 7 July 2022.
(Corresponding author: Xiaoyue Feng.)
Recommended for acceptance by Q. He.
Digital Object Identifier no. 10.1109/TKDE.2020.3028943

that consider both text contextual information and sequence information [8]. The distributed representations produced by deep learning models have been successfully applied in many natural language processing (NLP) tasks [8], such as text classification [9], language recognition [10], and machine translation [11]. Several deep learning-based text clustering models have been proposed that regard a text as a sequence instead of a bag of words. For example, Xu *et al.* [12] proposed a deep convolutional neural network-based short text clustering model, but the model's supervised signals come from word co-occurrence relations. Furthermore, Wang *et al.* [13] proposed a semi-supervised deep text clustering model in which the clustering performance relies entirely on a set of given labeled instances. Due to the absence of a supervised signal in text clustering, deep learning-based models are challenging to train, and most current research is based largely on a self-taught approach to obtain clustering results [12]. Hence, these models have a poor data adaptation ability, and similarity measures are crucial to the quality of the clustering results [14].

Recent studies have demonstrated that models learned from a large-scale corpus can produce meaningful distributed embeddings for sentences. Conneau *et al.* [15] trained a bidirectional long short-term memory (BiLSTM) network on a natural language inference corpus and found that the pretrained model could produce sentence embeddings suitable for other tasks (e.g., sentence classification and image caption ranking). In addition, Peters *et al.* [16] discovered that features extracted by a pretrained neural language model are suitable for sequence tagging. In the embedded feature space, the euclidean distance between sentence embeddings is sufficient to measure the similarity of the input [15], [17]; therefore, the quality of a pretrained deep text encoder is assumed to be highly suitable for text clustering tasks.

Furthermore, as a type of transfer learning, a knowledge transfer model can be used to transfer knowledge from one domain to boost the performance in another similar domain [18]. Yosinski *et al.* [19] proposed the pretrained *AlexNet* model to transfer knowledge to other image classification tasks. Compared to training a model from scratch, transferring knowledge from a pretrained deep model is more efficient and appropriate because of the improved generalization capacity and high convergence speed. Additionally, pretrained deep models also contribute prior knowledge to new tasks. For example, Howard *et al.* [20] proposed deploying a pretrained deep language model for text classification and presented several strategies for fully utilizing pretrained models. Bidirectional Encoder Representations from Transformers (BERT), a pretrained deep bidirectional language model proposed by Google, achieved state-of-the-art (SOTA) results on a wide range of tasks, including question answering and language inference [21]. Radford *et al.*'s GPT2 model [22] was transferred and applied to several text generation tasks and achieved excellent performance. However, no studies have deployed a pretrained deep model to text clustering; hence, we introduce a pretrained deep model to text clustering.

We propose a novel deep feature-based text clustering (DFTC) framework and explore the suitability of the deep text encoder for text clustering. In contrast to the bag-of-words model, the pretrained deep text encoder directly

processes text word by word and provides a semantic representation. Moreover, the pretrained deep text encoder solves the feature sparsity problem. We compare our model with classic text clustering models, including tf-idf-based K-means, LDA, the GSDMM, and the SOTA pretrained language model, i.e., BERT. Our model outperforms these models on almost all considered corpora. In addition, we propose a text clustering results explanation (TCRE) model that can capture the clusters' semantics and provide a qualitative evaluation of the clustering results. The TCRE model's results provide evidence of how the deep pretrained encoder-based clustering model outperforms the previously mentioned text clustering models.

The contributions of this paper are as follows:

1. We propose a novel deep feature-based text clustering framework (DFTC) that integrates sequence information and pretrained text encoders to introduce deep semantic features.
2. We propose the TCRE model which illustrates the effectiveness of the learned deep semantic features. It verifies the inverted pyramid style with indication words and their positions.
3. We show that our DFTC framework outperforms classic text clustering algorithms and SOTA pretrained language models on the considered datasets.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes our DFTC framework. Section 4 describes the TCRE model. Section 5 analyzes our model's computational complexity. Section 6 introduces the setup of our experiments. Section 7 illustrates the experimental results and presents a discussion. Finally, Section 8 concludes our paper.

2 RELATED WORK

We split our analysis of the related work into two main areas. Existing deep learning-based clustering models are surveyed, and the recurrent neural network (RNN) is introduced.

2.1 Deep Learning-Based Clustering Models

Feature transformation is a critical step for clustering models. Unlike traditional linear feature transformation methods, deep neural networks can transform data into more clustering-friendly representations due to their inherent ability to perform highly nonlinear transformations [23]. In recent years, several studies have explored the use of deep neural networks in clustering tasks. Xie *et al.* [24] designed a heuristic loss function for clustering tasks and proposed the deep embedding clustering (DEC) model. To improve the stability of the DEC model, Feng *et al.* [25] introduced an additional decoder layer into the DEC model. Yang *et al.* proposed a deep clustering network (DCN) model that combined K-means clustering and an autoencoder to learn a K-means-friendly latent space [26]. The models in these three works achieved good performance on several datasets; however, they rely heavily on the training quality of the autoencoders. Moreover, the performance of these models degrades substantially when the autoencoder collapses. Jiang *et al.* [27] proposed a deep generative model called VaDE for a data clustering task, but the model is so complex that both its time

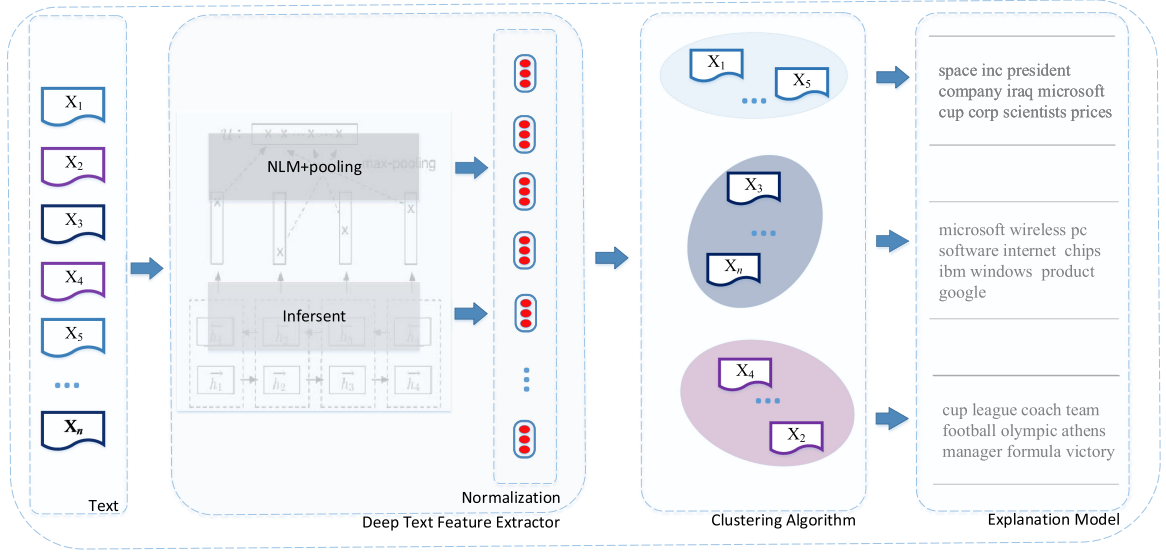


Fig. 1. The deep text clustering framework. X_i is the i th text in the given corpus.

complexity and its space complexity are untrackable. Several advances have also been made in text clustering. Xu *et al.* [12] proposed a deep learning-based short text clustering model that relies on bag-of-words signals, and Wang *et al.* [13] proposed a semi-supervised deep text clustering model in which the clustering performance relies entirely on a set of given labeled instances. However, to the best of our knowledge, no research has applied a pretrained deep learning model to text clustering and proposed an explanation of the effectiveness of the deep learning approach.

2.2 Recurrent Neural Network

In contrast to feedforward neural networks, RNNs can process variable-length sequences [28]. An RNN maintains a hidden state as the context and updates the hidden state given a token. Formally, given a sequence $(x_1, x_2, \dots, x_n)$, an RNN generates hidden states $(h_1, h_2, \dots, h_n)$ by means of the function $h_k = f(h_{k-1}, x_k)$, where f is the RNN's cell function. When training a vanilla RNN, various problems can occur, such as vanishing and exploding gradients. Thus, RNNs cannot model long dependencies. Hochreiter *et al.* [29] proposed the utilization of LSTM to mitigate these problems by introducing several gates. In later research, several minor modifications were made to the original LSTM cell [30]. In this paper, we adopt the LSTM framework described in [30]. The LSTM cell function is defined as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \quad (2)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \quad (3)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \quad (4)$$

$$h_t = o_t \tanh(c_t), \quad (5)$$

where $W_{\{xi,hi,xf,hf,xo,ho,xc,hc\}}$ are weight parameters, $b_{\{i,f,o,c\}}$ are bias vectors, and σ is a sigmoid function defined as

$\sigma(x) = \frac{1}{1+\exp(-x)}$. c_t is a memory cell that remembers previous input information and avoids the gradient vanishing problem. i_t is the input gate, which controls the input information flowing into the cell. o_t is the output gate, which controls the output information flowing from the cell. f_t is the forget gate used to control the flow of information from the previous memory cell to the next memory cell. LSTM outperforms the vanilla RNN in certain tasks, such as the neural language model [31].

3 THE DEEP FEATURE-BASED TEXT CLUSTERING FRAMEWORK

Given a corpus $C = \{x_1, x_2, \dots, x_n\}$, where x_i is a sentence or paragraph, our objective is to group the texts into several clusters. The framework of our *DFTC* model is shown in Fig. 1. For each text x_i , our framework first uses a pretrained text encoder to extract features. We adopt two pretrained deep text encoders, namely, the language model and the language inference model *InferSent*, both of which are based on LSTM. In the second step, a feature normalization module employs normalization techniques (e.g., layer normalization) to ensure the features' numerical stability and to ensure that the feature vectors satisfy specific qualities, such as conforming to a normal distribution. In the last step, the normalized features are fed into the selected clustering algorithm, such as K-means. After obtaining the cluster partition results, the explanation model produces representative words for each cluster.

3.1 The Deep Text Feature Extractor

We consider two deep feature extractors: the neural language model and *InferSent*. We will introduce both extractors as follows.

The goal of the language model is to estimate the probability function of a sequence of words from a large unlabeled corpus. Given a sequence of n words, $t_1, t_2, \dots, t_n$, the probability of the sequence can be written as

$$p(t_1, t_2, \dots, t_n) = \prod_{k=1}^N p(t_k | t_1, t_2, \dots, t_{k-1}), \quad (6)$$

where $p(t_k | t_1, t_2, \dots, t_{k-1})$ is the probability of the current word t_k given the word sequence $t_1, t_2, \dots, t_{k-1}$. Most neural language models [32] are built from an LSTM network and are trained to predict the next word given the previous words. In step k , the current time-series state $\vec{h}_k$ is modeled by the function $\vec{h}_k = f(\vec{h}_{k-1}, x_k)$, where f is the LSTM cell function and x_k is the word representation of word t_k . The probability function $p(t_k | t_1, t_2, \dots, t_{k-1})$ can be estimated by the *softmax* function. However, due to the unstable gradient problem of LSTM, a backward language model can supplement the complementary information neglected by the forward language model. In contrast to forward language models, backward language models predict the previous word given the following words. The probability function of the sequence can be decomposed as

$$p(t_1, t_2, \dots, t_n) = \prod_{k=1}^N p(t_k | t_n, t_{n-1}, \dots, t_{k+1}). \quad (7)$$

$\overleftarrow{h}_k = f(\overleftarrow{h}_{k+1}, x_k)$ is used to estimate $p(t_k | t_n, t_{n-1}, \dots, t_{k+1})$, which is similar to the forward language model. Due to the complementarity between the forward language model and the backward language model, we have $h_k = [\vec{h}_k, \overleftarrow{h}_k]$ as the k th token's representation, where $[\cdot, \cdot]$ is the concatenation operator.

Due to the variability of the sentence or document length, we cannot directly feed the context features $h_1, h_2, \dots, h_n$ into the subsequent modules. Hence, we must fuse the context features into a fixed-size feature vector. In this paper, we adopt three feature fusion strategies.

- 1) *Max-pooling* selects the maximum value over each dimension of the d -dimensional hidden context feature vectors $h_1, h_2, \dots, h_n$ to build a text representation, as shown in Equation (8). Max-pooling regards the highest value as the most important feature

$$h[k] = \max_{i=1 \dots n} h_{ik}. \quad (8)$$

- 2) *Mean-pooling* averages the d -dimensional hidden context feature vectors to the feature vector, as shown in Equation (9). The idea of mean-pooling is that all context feature vectors can represent the whole text, and the average of these vectors will reduce the noise in the model

$$h[k] = \frac{1}{n} \sum_{i=1 \dots n} h_{ik}. \quad (9)$$

- 3) The *last-time* context feature vector captures the semantics of the whole text sequence. Hence, we can concatenate the forward language model's last feature $\vec{h}_n$ and the backward language model's last feature $\overleftarrow{h}_1$ into a new feature vector, $h = [\vec{h}_n, \overleftarrow{h}_1]$. h is then fed into the following module.

InferSent is another sentence representation model that can provide meaningful sentence embeddings given a set of

sentences [15]. In contrast to the neural language model, which is trained on an unlabeled corpus, *InferSent* is trained on a labeled natural language inference (NLI) corpus in a supervised manner. Then, the learned knowledge is transferred to other tasks. The goal of the NLI task is to determine whether a pair of sentences are entailed, contradictory, or neutral. In the training phase, the *InferSent* model first encodes two sentences into two-sentence embeddings, fuses these embeddings into a single embedding, and finally feeds the embedding into a 3-way classifier. The model is trained in an end-to-end manner using stochastic gradient descent (SGD) on the Stanford Natural Language Inference (SNLI) dataset. *InferSent* adopts the BiLSTM model with max-pooling as its sentence encoder and achieves a distinguished transfer performance on many NLP tasks, such as text classification and sentiment analysis. Because *InferSent* is trained on sentence information instead of paragraph or document information, we split paragraphs into sentences and average the sentences' *InferSent* embeddings to model a paragraph.

3.2 The Feature Normalization Module

We use the feature normalization function $\bar{h} = f(h)$ to ensure that the features conform to various characteristics, such as normality and stability. We introduce three normalization strategies: identity normalization, standard normalization, and layer normalization. These normalization strategies are exchangeable.

- 1) *Identity normalization* is an identity function, i.e., $f(h) = h$, given the feature vector h . In this paper, we utilize normalization as the baseline for comparison with the other feature normalization methods.
- 2) *Standard normalization* is a commonly used feature normalization method that applies $\bar{h}_i = \frac{h_i}{\|h_i\|}$ to transform an input feature vector into a vector with one norm. After the transformation, the euclidean distance between two feature vectors is equivalent to the cosine distance between them.
- 3) *Layer normalization* is implemented primarily to avoid the covariate shift problem when training a neural network [33]. For some feature embedding $h_i = (h_{i1}, h_{i2}, \dots, h_{im})$, which is an m -dimensional vector, layer normalization utilizes Equation (10) to normalize the input feature, where ϕ_i is the mean of the elements in h_i , as shown in Equation (11), and σ_i is the standard variance, as shown in Equation (12). After the transformation, each element of $\bar{h}_i$ represents a sampling from the same normal distribution

$$\bar{h}_i = \frac{h_i - \phi_i}{\sigma_i}, \quad (10)$$

$$\phi_i = \frac{1}{n} \sum_{j=1}^N h_{ij}, \quad (11)$$

$$\sigma_i = \sqrt{\frac{1}{n} \sum_{j=1}^N (h_{ij} - \phi_i)^2}. \quad (12)$$

3.3 The Clustering Algorithm

Our deep text clustering framework is suitable for most data clustering algorithms. Due to the brevity of the K-means algorithm, we apply the classic K-means clustering algorithm to the extracted features in this research. Other clustering algorithms, such as affinity propagation and self-organizing feature maps, are also suitable in our framework.

Given n extracted features $\{h_1, h_2, \dots, h_n\}$, K-means clustering is used to partition the n feature points into K groups. The objective function of the K-means algorithm is

$$L = \sum_{i=1}^K \sum_{j=1}^n t_{ij} \|h_j - \mu_i\|^2, \quad (13)$$

where μ_i is the center of the i th cluster and t_{ij} identifies whether the i th data point belongs to the j th cluster. Hence, the value of t_{ij} is 0 or 1, and $\sum_{j=1}^K t_{ij} = 1$. Directly minimizing the objective function is an NP-hard problem because of the discrete value of t_{ij} . The most commonly used approximate algorithm is EM iteration. In the E-step, each point is assigned to the nearest cluster center according to the distance between the data point and cluster center μ_i , after which the value of t_{ij} is identified. In the M-step, each cluster's center is computed by the following formula:

$$\mu_i = \frac{\sum_{j=1}^n t_{ij} h_j}{\sum_{j=1}^n t_{ij}}. \quad (14)$$

The E-step and M-step alternate iteratively until the algorithm converges.

4 THE EXPLANATION MODEL

Because of the unsupervised nature of text clustering algorithms, we cannot be directly aware of each cluster's meaning. The most common explanation method for a text clustering algorithm is to compute the word frequency distribution for each cluster and use the most frequent words in a cluster to represent the cluster's semantics. We represent this kind of traditional word frequency explanation model as *FREQ*. However, one problem with this model is that high-frequency words are often common among several clusters. For example, *said* will be one of the highest-frequency words for each cluster when we cluster a news corpus. Moreover, a naive method can also introduce noise.

To solve these two problems, we introduce a novel model to adjust every word's weight for each cluster adaptively.

In this study, our proposed *TCRE* model is illustrated in Algorithm 1. The inputs of the algorithm are the corpus and clustering results. Every text in a given corpus is labeled with a class ID given the clustering results, and these labels can be regarded as pseudolabels of the texts. The main idea of our algorithm is to use a logistic classifier to fit the associations between texts and pseudolabels. The algorithm includes two parts. In the first part (lines 3 to 9), the *TCRE* model maps the text in the corpus into bag-of-words features. In contrast to ordinary text classification, 0-1 features are the inputs of the classifier instead of tf-idf features. If a word exists in a text, the feature value of the word is 1; otherwise, the feature value is 0. Stop words and low-frequency words are removed because they do not provide meaningful

information. In the second part (lines 11 to 23), the *TCRE* model acquires indication words that express every cluster's meaning. The prediction function of the logistic regression classifier is shown in Equation (15). The weights w_l of the logistic regression classifier for a cluster can be regarded as the scores of the words in the l th cluster

$$p(y_i = l | x_i) = \frac{\exp(w_l \cdot x_i)}{\sum_{j=1}^C \exp(w_j \cdot x_i)}. \quad (15)$$

The higher the score of a word in a cluster, the more important that word is in the cluster. For each cluster, the *TCRE* model selects the top n words with the highest scores as indication words. The explanation results can then be used to measure the quality of the clustering results and help a user understand the semantics of a cluster partition.

Algorithm 1. Procedure of the *TCRE* Model

Input: *Corpus* is the corpus, and *clusterResult* is the pseudolabel list.

Output: *indWordsList* contains the list of indication words for every cluster.

- 1: Let $n = \text{size}(\text{Corpus})$.
 - 2: Let *featList* = [].
 - 3: {Map the text into 0-1 bag-of-words features.}
 - 4: **for** $i \in [1, n]$ **do**
 - 5: Split the text *Corpus*[i] into tokens.
 - 6: Filter out stop words and low-frequency words.
 - 7: Map the remaining tokens into 0-1 feature vector *feat*.
 - 8: Append the feature vector *feat* into *featList*.
 - 9: **end for**
 - 10: {Obtain Indication Words for Every Cluster.}
 - 11: Train the logistic regression classifier *lgClassifier* on training data (*featList*, *clusterResult*).
 - 12: {*weightList* is the weight list whose length is the number of labels.}
 - 13: Let *weightList* = *lgClassifier.weights*
 - 14: Let *indWordsList* = []
 - 15: **for** *weight* $\in$ *weightList* **do**
 - 16: Let *tmpWeight* = []
 - 17: **for** $w \in \text{weight}$ **do**
 - 18: Append $\text{abs}(w)$ into *tmpWeight*
 - 19: **end for**
 - 20: Let *tmpIdx* = $\text{argsort}(\text{tmpWeight})[:n]$
 - 21: Map *tmpIdx* into indication words *indWords*
 - 22: Append *indWords* into *indWordsList*
 - 23: **end for**
 - 24: **return** *indWordsList*
-

5 COMPUTATIONAL COMPLEXITY ANALYSIS

To analyze the complexity of the proposed model in detail, we present the complexity of the *DFTC* framework in each step. The first and most complicated step in our framework is to achieve an effective text representation in the deep text feature extractor. The neural language model and the *InferSent* representation model are both dependent on the LSTM network. Assuming that the dimension of the LSTM model is d and the average text length is ns , the computational complexity of one layer of the LSTM is $O(ns \cdot d^2)$, and we assume that the height of our multiple-layer LSTM model is

nl . Hence, our model achieves the text representation by $O(ns \cdot nl \cdot d^2)$. The complexities of the three normalization methods are $O(ns \cdot d)$. For the clustering part, assuming that there are nd text snippets, the complexity of the K-means algorithm is $O(nd \cdot d \cdot I)$, where I is the number of iterations. Hence, the total complexity of the DFTC model is $O(nd \cdot ns \cdot nl \cdot d^2 + nd \cdot ns \cdot d + nd \cdot d \cdot I)$. Because $ns \cdot nl \cdot d^2$, $ns \cdot d$ and $d \cdot I$ are constants, the complexity is $O(nd)$.

Our explanation model includes two steps. The first step is to build a linear relation between different clustering results and each word in the corpus. In this step, we adopt a logistic model. LIBLINEAR's logistic implement has a time complexity of $O(n)$, where n is the size of the corpus. Assuming the number of word features is V , constructing the model will involve a time complexity of $O(nV)$. The second step is to find indication words. In this step, the time complexity is $O(V)$. Hence, the TCRE model's time complexity is $O(nV + V) \approx O(nV)$.

6 EXPERIMENTAL SETUP

We first introduce five corpora and three evaluation metrics. Then, classic text clustering algorithms and the SOTA pre-trained language model, i.e., BERT, are described.

6.1 Datasets

We evaluate our model on five corpora: *AG news*, *DBpedia*, *Yahoo! Answers*, *R2*, and *R5*. The corpora *AG news*, *DBpedia*, and *Yahoo! Answers* were collected and constructed by Zhang *et al.* [34]. Because of the large sizes of the three corpora, directly performing experiments on the original corpora would be time-consuming; therefore, we adopted abbreviated versions of the datasets. Following previous research [13], we randomly selected 1,000 instances for each class in each dataset. In our preliminary experiments, we found that the sampled balanced corpora resulted in a performance similar to that achieved with the original data. The corpora *R2* and *R5* were extracted from the corpus *Reuters-21578* by us. We introduce these corpora as follows:

AG news is a news categorization corpus. Zhang *et al.* [34] constructed this corpus by choosing the top four categories from AG's corpus of news articles on the web.¹ These texts are gathered from more than 2,000 news sources by ComeToMyHead for more than one year of activity. Each text in the *AG news* corpus includes the original title and content. There are four categories in the corpus: *World*, *Sports*, *Business*, and *Sci/Tech*.

The *DBpedia* ontology classification corpus was constructed by selecting several classes from the knowledge base *DBpedia*'s ontology by Zhang [34]. Each text snippet in the corpus is an entity's description, and its label is the entity's ontological class label. The corpus contains 14 nonoverlapping classes: *Company*, *Educational Institution*, *Artist*, *Athlete*, *Office Holder*, *Means of Transportation*, *Building*, *Natural Place*, *Village*, *Animal*, *Plant*, *Album*, *Film*, and *Written Work*.

Yahoo! Answers is a topic classification corpus extracted from the *Yahoo! Answers Comprehensive Questions and Answers* version 1.0 dataset through the *Yahoo! Webscope* program by

Zhang [34]. Each text in the corpus includes a question and its corresponding answers. There are ten categories: *Society&Culture*, *Science&Mathematics*, *Health*, *Education&Reference*, *Sports*, *Business&Finance*, *Entertainment&Music*, *Family&Relationships*, *Computer&Internet*, and *Politics&Government*.

The *Reuters-21578* corpus² was initially collected and labeled by the Carnegie Group and Reuters. The corpus contains 21,578 documents grouped into 135 categories. Different from other corpora, this corpus is highly unbalanced. The largest category includes thousands of items, whereas the smallest category has only a few. Following previous research [35], we constructed two clustering corpora, *R2* and *R5*, which include the two and five largest categories, respectively. The categories in corpus *R2* are *earn* and *acq*. The categories in corpus *R5* are *earn*, *acq*, *cude*, *trade*, and *money-fx*. In the following experiments, we use these two unbalanced corpora to evaluate our model.

6.2 Evaluation Metrics

The clustering performance is evaluated by comparing the clustering results with the given labels. We adopt three commonly used evaluation metrics: the clustering accuracy (ACC) [36], normalized mutual information (NMI) [36], and adjusted Rand index (ARI) [37].

ACC is defined as

$$ACC = \max_m \frac{\sum_{i=1}^n \mathbf{1}\{l_i = m(c_i)\}}{n}, \quad (16)$$

where l_i is the ground-truth label of text i and c_i is the label predicted by the clustering algorithm. $m()$ is a one-to-one mapping between the cluster labels and ground-truth labels. The $\mathbf{1}\{\}$ function outputs 1 when the equation in curly brackets is true and outputs 0 otherwise. This accuracy metric takes a cluster assignment from an unsupervised algorithm and a ground-truth assignment and then finds the best matching between them. The function $m()$ can map the cluster label into its best-matched ground-truth label. The best mapping can be efficiently computed via the Hungarian algorithm [24]. The intent of the *ACC* function is to compute the best matching accuracy between the two groups of labels, and the Hungarian algorithm can be used to efficiently compute the best match.

NMI is defined as

$$NMI(L, C) = \frac{MI(L, C)}{\sqrt{H(L)H(C)}}, \quad (17)$$

where L is the ground-truth label and C is the label predicted by the clustering algorithm. $MI(L, C)$ is the mutual information between L and C and is used to measure the relevance between them. $H(\cdot)$ represents entropy. In this function, $\sqrt{H(L)H(C)}$ is used to normalize $MI(L, C)$ to the range of $[0, 1]$.

1. http://groups.di.unipi.it/gulli/AG_corpus_of_news_articles.
html

2. <http://www.daviddlewis.com/resources/testcollections/reuters21578>

TABLE 1
Clustering Ability Comparison Among the Traditional Models and Our Proposed Model

	Avoids the high-dimensional problem	Avoids the sparsity problem	Contains sequence information	Uses transferred semantic knowledge
tf-idf+KM	-	-	-	-
LDA [6]	✓	-	-	-
LDA+KM [6]	✓	-	-	-
GSDMM [7]	✓	✓	-	-
DEC [24]	✓	✓	-	-
IDEC [25]	✓	✓	-	-
STC [12]	✓	✓	✓	-
DFTC	✓	✓	✓	✓

ARI is defined as

$$ARI = \frac{\sum_{i,j} \binom{n_{i,j}}{2} - [\sum_i \binom{n_{i,\cdot}}{2} \sum_j \binom{n_{\cdot,j}}{2}] / \binom{n}{2}}{\frac{1}{2} [\sum_i \binom{n_{i,\cdot}}{2} + \sum_j \binom{n_{\cdot,j}}{2}] - [\sum_i \binom{n_{i,\cdot}}{2} \sum_j \binom{n_{\cdot,j}}{2}] / \binom{n}{2}}, \quad (18)$$

where n is the number of all instances, $n_{i,j}$ is the number of instances appearing in the i th predicted label and j th ground-truth label, $n_{i,\cdot}$ is the number of i th predicted label instances, and $n_{\cdot,j}$ is the number of j th ground-truth label instances. The function *ARI* computes the similarity between the ground-truth labels and the clustering algorithm-predicted labels and takes values in the range of (0, 1).

6.3 Compared Methods

We compare our model with the text clustering algorithms listed below.

tf-idf-based K-means. In this paper, we choose the 2,000 most frequently used words (after removing stop words) as features. The baseline uses K-means on tf-idf features to group text into clusters.

LDA [6]. We consider three K values, 30, 40 and 50, where K is the number of topics. Two approaches can be followed to utilize LDA for clustering. The first is selecting the topic with the highest topic probability as a text's predicted label. The second is to use the topic distribution as the feature and apply a data clustering model, such as K-means, to group the texts. According to Griffiths *et al.*'s research [38], setting the LDA model parameters as $\alpha = K/50$ and $\beta = 0.1$ generally yields good model quality. We follow these settings in this paper.

GSDMM [7]. The GSDMM regards text clustering as a Dirichlet multinomial mixture model that is solved by Gibbs sampling. Following the original paper on the model, we set the GSDMM hyperparameters to $\alpha = 0.1$ and $\beta = 0.1$. Similar to LDA, we consider several K values: 30, 40 and 50.

DEC [24]. Xie *et al.* built a self-taught loss for a deep clustering model called DEC, which was not designed specifically for text clustering; hence, we built bag-of-words features for the DEC model. We follow the default configuration of the DEC model in the original paper.

IDEC [25]. The IDEC model is a modified version of the DEC model with an additional decoder after the middle hidden layer. The decoder makes the training process more

stable. We adopt the default configuration of the IDEC model from the original paper.

STC [12]. The STC model is a deep short text clustering model that utilizes a convolutional neural network to learn representations from bag-of-words features. The STC model obtains cluster partitions by employing K-means to cluster the learned representations.

BERT [21]. The BERT model is a pretrained language model proposed in 2019. It is based on the Transformer model and has obtained SOTA performance on several NLP tasks far beyond the performance of existing CNN or RNN models. To fully evaluate our model, we utilize the *Bert-base* model for comparison. We adopt the pretrained BERT model as a text embedding extractor, which contains 12 Transformer blocks ($L=12$). For each block, the hidden layer size H is 768, and the number of self-attention heads is 12. The total number of parameters in BERT is approximately 110 M. And its fine-tuning step is omitted. Before feeding the text into the BERT model, we transform the text into lowercase and tokenize it using WordPiece.

The clustering ability comparison results among these models are shown in Table 1, where a "✓" symbol indicates that the clustering model can avoid the high-dimensional problem, can avoid the sparsity problem, contains sequence information, or uses transferred semantic knowledge; a "-" symbol indicates that the model cannot or does not.

6.4 Pretrained Models

We introduce a neural language model and *InferSent* as the feature extractor for our text clustering framework. For the neural language model, we adopt the pretrained language model *ELMo* [16],³ which contains two BiLSTM layers with a residual connection from the first layer to the second layer. The dimension of each BiLSTM layer is 4,096. The final output of the BiLSTM is projected into a 1,024-dimensional representation that is fed into the prediction layer. Conneau *et al.* [15] trained the *InferSent* model on the SNLI dataset and released a pretrained model,⁴ the encoder of which is a BiLSTM max-pooling network. Fixed word representations are fed into the 4,096-dimensional BiLSTM network, and a max-pooling layer is used to transform the intermediate representations into 4,096-dimensional vectors.

3. <https://allennlp.org/elmo>

4. <https://github.com/facebookresearch/InferSent>

TABLE 2
Experimental Results From the Compared Clustering Models and Our Deep Clustering Model on the Five Datasets

	AG news			DBpedia			Yahoo! Answers			R2			R5		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
tfidf+KM	56.8	33.0	30.0	56.7	64.0	58.8	31.9	18.7	16.5	73.9	34.8	34.8	52.2	41.0	41.0
LDA [6]	17.7	27.4	17.4	60.2	66.0	59.4	28.9	23.3	19.0	17.9	34.8	15.6	25.6	42.1	23.6
LDA+KM [6]	55.0	31.7	27.1	59.3	61.8	56.7	36.1	27.4	23.0	85.3	49.9	48.5	72.0	50.7	48.1
GSDMM [7]	43.5	42.5	30.4	68.0	75.2	71.7	49.6	36.1	34.2	74.5	62.3	45.3	57.6	59.5	44.5
DEC [24]	47.0	12.4	11.5	58.8	68.1	66.1	14.2	1.2	1.1	76.0	37.4	36.9	37.9	22.2	18.5
IDEC [25]	63.5	29.5	29.4	64.6	74.0	71.7	22.0	10.3	9.0	77.7	32.3	31.5	46.1	34.8	29.5
STC [12]	64.6	39.2	38.8	54.9	61.4	60.4	37.7	23.6	21.4	77.9	35.7	36.5	57.1	40.8	37.5
BERT+Max+I+KM	48.0	22.4	22.4	44.3	52.5	52.3	13.5	1.5	1.3	72.7	26.7	26.7	44.2	29.6	29.6
BERT+Max+LN+KM	82.7	56.9	56.9	76.3	80.6	80.5	33.6	17.8	17.6	84.2	49.7	49.7	57.1	45.9	45.9
BERT+Max+N+KM	82.3	56.4	56.3	68.8	75.1	75.0	26.0	12.2	12.0	84.8	49.2	49.2	69.3	50.5	50.5
BERT+Mean+I+KM	80.4	54.3	54.3	70.8	74.6	74.5	40.4	26.5	26.4	84.4	50.0	50.0	67.8	53.7	53.6
BERT+Mean+LN+KM	80.4	53.8	53.7	70.4	74.7	74.6	42.3	27.6	27.5	84.5	49.4	49.4	67.6	53.5	53.5
BERT+Mean+N+KM	80.4	53.9	53.8	73.1	74.7	74.6	42.4	27.7	27.6	84.5	49.4	49.4	67.6	53.5	53.5
BERT+Last+I+KM	34.0	5.3	5.3	69.5	71.6	71.5	19.6	6.0	5.8	85.6	49.0	49.0	55.1	40.7	40.6
BERT+Last+LN+KM	35.3	5.6	5.5	69.0	72.1	72.0	19.3	5.9	5.7	85.6	49.2	49.1	54.5	40.9	40.8
BERT+Last+N+KM	35.4	5.7	5.6	69.8	72.3	72.3	18.3	5.0	4.9	85.6	49.2	49.1	54.5	40.9	40.9
LM+Max+I+KM	70.7	42.6	42.2	57.4	65.4	64.8	22.1	10.9	10.5	82.3	43.9	43.9	55.7	45.9	45.8
LM+Max+LN+KM	81.6	57.3	56.7	73.5	79.0	77.8	43.1	29.6	29.3	84.5	48.6	47.3	56.7	46.4	41.1
LM+Max+N+KM	80.3	56.8	56.0	67.8	75.5	74.1	42.9	28.5	27.4	84.6	50.2	50.2	57.3	46.5	46.5
LM+Mean+I+KM	80.7	59.1	58.2	75.4	80.2	79.1	46.6	32.8	31.4	84.3	49.8	48.4	52.5	45.7	45.6
LM+Mean+LN+KM	82.2	60.5	60.0	74.8	80.1	79.0	50.3	34.9	34.4	84.8	50.6	49.2	69.7	61.5	53.8
LM+Mean+N+KM	82.1	60.3	60.0	76.0	80.2	79.2	51.1	35.0	34.4	84.7	50.4	49.0	69.6	61.5	61.5
LM+Last+I+KM	33.8	5.5	4.6	56.7	57.5	56.5	16.4	4.3	3.9	62.1	21.7	21.7	49.9	31.0	30.9
LM+Last+LN+KM	33.1	5.1	4.2	57.2	58.4	57.4	16.5	4.3	3.9	62.1	21.7	21.7	50.1	31.2	31.2
LM+Last+N+KM	33.1	5.1	4.2	56.9	57.9	57.0	16.5	4.2	3.8	62.1	21.7	21.7	50.1	31.2	31.2
InferSent+I+KM	67.4	40.4	40.2	56.8	61.4	60.6	29.1	16.8	16.3	84.0	49.3	48.0	40.4	29.4	25.2
InferSent+LN+KM	80.8	53.5	53.4	77.6	77.9	77.0	39.2	24.7	24.1	85.0	51.0	49.6	59.1	43.0	37.1
InferSent+N+KM	80.7	53.5	53.4	73.5	75.1	74.2	39.2	24.1	23.4	85.2	50.5	49.1	59.1	41.8	36.1

We apply LDA and GSDMM to 30, 40 and 50 topics. We run each model 10 times and report the average of the experimental results. For the compared methods, we report the best experimental results of each configuration. Boldface indicates the best experimental results in each group.

7 EXPERIMENTAL RESULTS & DISCUSSION

In this section, we report our model's experimental results and explain the clustering results. In Section 7.1, we report the experimental results and compare our model with other models. In Section 7.2, we visualize deep text features by t -SNE, which illustrates the effectiveness of the pretrained text encoder. In Section 7.3, we report the transformed deep text features' clustering performance. In Section 7.4, we explain the clustering results obtained by our proposed TCRC model. The indication words discovered by our model can illustrate the meaning of every cluster.

7.1 Comparison With Other Methods

Table 2 presents the results of all models on all five datasets. KM represents the K-means clustering model. LM and *InferSent* represent the neural language model and the *InferSent* model, respectively. I, LN, and N represent the identity normalization, layer normalization and standard normalization feature transformation strategies, respectively. For each dataset, we evaluate the clustering results with three metrics. Hence, we obtain 15 metrics for the five datasets. The clustering models are divided into three groups: classic bag-of-words and generative models, BERT-based models, and the *DFTC* models.

As shown in Table 2, for 12 of the 15 metrics, our model outperforms the classic bag-of-words and generative models, including tf-idf, LDA, GSDMM, DEC, IDEC, and STC. These experimental results illustrate the effectiveness of introducing contextual information. Furthermore, our model outperforms BERT on 11 metrics, which demonstrates the

effectiveness of the *DFTC* models. We consider several configurations for our deep clustering framework; among them, LM+Mean+N+KM achieves the best performance on all the datasets except R2. For example, this configuration achieves an accuracy of 82.2 percent, which is 27.2 percent higher than that obtained by the best compared method, *STC*. *InferSent*+LN+KM achieves similar performance on *AG news*, *Dbpedia* and R2 but worse performance on *Yahoo! Answers* and R5. The GSDMM is the most robust of the four compared models, but its performance is still far from that of our deep clustering model, especially on *AG news* and *Dbpedia*. Because most of these existing text clustering algorithms are based on bag-of-words models, the feature space cannot fully construct the semantic space of the raw text, and the loss of subsequent information will induce the loss of semantic information. In addition, text data are notoriously high-dimensional: as the size of the corpus increases, so does the size of the vocabulary. The bag-of-words model cannot fully utilize long-tailed words; thus, its representation ability is minimal. In contrast, our framework is based on a deep pretrained model that can infer text semantics by contextual information. Pretraining the model from a large-scale corpus will introduce new transferred knowledge. Our model is insensitive to clustering algorithms. Although the K-means clustering algorithm is adopted, our model also outperforms most of the SOTA text clustering algorithms.

Text data contain some low-frequency words, including slang, misspelled words, and other uncommon words. Traditional text clustering methods cannot effectively process sentences or documents with too many low-frequency words. These outlier text data will influence text clustering

algorithms' performance. For our text clustering model, there are two mechanisms for processing these abnormal data. First, a pretrained deep model can infer an unknown word's meaning by its context; in contrast, traditional text clustering cannot perceive a word's contextual information because bag-of-words features lose subsequent text information. Second, the neural language model also considers character-level information, which considers a word's lexical spelling information. For example, "good" and its misspelled word "goodd" are considered to have similar semantics.

For the neural language model, as shown in the previous section, three methods are used to fuse variable-length features into fixed-sized features. Mean-pooling produces better experimental results than max-pooling and last-time. Last-time has the worst performance because an RNN cannot adequately model a sequence's long-distance dependencies utilizing the last-time feature.

Our framework performs feature transformation before feeding the features into the clustering algorithm. Layer normalization is the most effective strategy for the configuration of max-pooling-based feature fusion. Compared with the LM+Max+I+KM and *InferSent*+I+KM configurations, LM+Max+LN+KM and *InferSent*+LN+KM achieve substantial performance improvements because every element value of a transformed feature is very large, and layer normalization normalizes these values to reduce the covariate shift. For the mean-pooling-based configurations, standard normalization and layer normalization achieve only small performance improvements because the mean-pooling strategy attempts to consider all the time inputs and because averaging operations can provide a robust feature representation.

For the *Yahoo! Answers* and *R2* datasets, our proposed deep model does not achieve ideal performance. Each item in the *Yahoo! Answers* dataset contains a question and several different answers that are not semantically correlated. Directly inputting these features into an LSTM encoder fails to fully account for the sequence semantics. Moreover, the text in *Yahoo! Answers* contains some nonstandard Internet language, such as *goodd* and *BTW*. The *InferSent* feature extractor is trained on a normative corpus, and the neural language model is pretrained on a large-scale Internet corpus. Hence, the LM-based clustering models achieve better clustering performance than the *InferSent*-based clustering models in this case.

Other deep learning-based clustering algorithms, namely, DEC, IDEC and STC, do not perform better than our model because these three models rely on bag-of-words features, which ignore the sequential and structural information of the text. Moreover, the DEC and IDEC models are dependent on an autoencoder; however, autoencoder training is not a stable process, and the performance of the encoder may degrade.

7.2 Feature Visualization

The clustering experiment results from the previous section show how our clustering model outperforms bag-of-words-based and generative model-based text clustering models, mainly because the distributed text representation built by a deep model puts similar texts in nearby positions and the euclidean distance between text features represents a semantic relation. To verify our explanation, we visualize the deep text encoder features and tf-idf features using a commonly

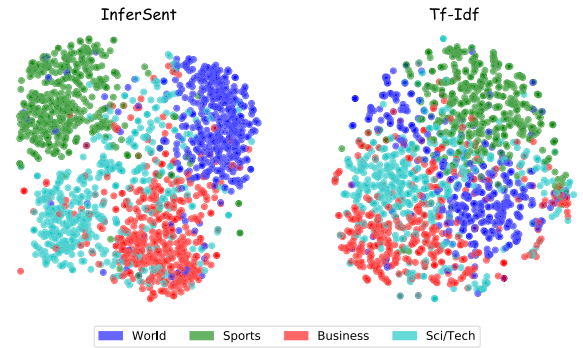


Fig. 2. *t*-SNE visualization for *InferSent* and tf-idf features.

used visualization method, *t*-SNE [39], which maps high-dimensional features into 2D features. For the deep text encoder features, we use the *InferSent*+LN configuration. Fig. 2 shows the feature visualization results of our selected *AG news* dataset. Following the original paper in which *t*-SNE was proposed, we adopt a perplexity value between 5 and 50; we ultimately choose 30 as the value by visualizing the results. In addition, we employ a total of 1,000 iterations. *t*-SNE is stopped upon reaching the maximum number of iterations or when there is no change in the KL-divergence. The learning rate is 200.0. To visualize the result in 2D space, the output dimension of *t*-SNE is 2. In Fig. 2, blue, green, red, and cyan represent *World*, *Sports*, *Business*, and *Sci/Tech*, respectively. The tf-idf feature points are mixed in the center of the right plot, and it is difficult to distinguish the different clusters. By contrast, the *InferSent* feature points from the same cluster remain together in the left plot, which clearly demonstrates that the texts represented by deep text encoder features are easier to distinguish among the clusters.

7.3 Clustering Using Transformed Deep Text Features

To further verify the effectiveness of the deep features, we use two feature selection algorithms, namely, a stacked denoising autoencoder and principal component analysis (PCA), to distill semantic information from the extracted features and then feed the distilled features into the clustering algorithm. In this experiment, we select the outputs of LM+Max+LN, LM+Max+N, LM+Mean+I, LM+Mean+LN, LM+Mean+N, *InferSent*+LN, and *InferSent*+N as the input features because these configurations achieve the ideal performance in the abovementioned experiments.

The dimensionality of the stacked denoising autoencoder is $d-1200-1200-d$, where d is the dimension of the input features. We adopt the same architecture for all feature configurations. As shown in Table 3, the LM+Mean+LN+AE+KM configuration achieves an accuracy of 85.1 percent on the *AG news* dataset, which is 3.5 percent higher than the accuracy achieved by the model after removing the autoencoder. However, for most configurations, deploying the autoencoder to distill features does not further improve the performance. In addition, introducing an autoencoder into our *DFTC* framework increases the complexity. For PCA, we select 300 as the dimension of the reduced features for all configurations on all datasets. Different from the autoencoder, PCA achieves robust and ideal performance. However, compared to the experimental results in

TABLE 3
Experimental Results From PCA- and AE-Transformed Features for the Five Datasets

	AG news			DBpedia			Yahoo! Answers			R2			R5		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
LM+Max+LN+AE+KM	81.0	57.2	56.5	71.8	77.5	76.4	42.2	28.7	28.4	84.2	49.6	48.3	56.5	46.0	45.9
LM+Max+N+AE+KM	81.9	56.0	55.8	66.3	73.0	71.6	42.0	27.3	26.5	84.3	48.6	48.6	56.0	44.2	38.3
LM+Mean+I+AE+KM	80.8	59.1	58.2	75.1	80.8	79.6	52.1	36.7	35.0	84.1	49.4	49.3	52.4	46.1	39.8
LM+Mean+LN+AE+KM	85.1	62.6	62.4	75.6	80.6	79.5	48.2	33.7	32.7	84.6	50.3	50.3	69.5	61.4	61.3
LM+Mean+N+AE+KM	80.8	57.6	56.8	72.2	76.7	75.7	51.9	33.4	32.9	84.6	50.3	50.3	67.4	57.4	49.8
<i>InferSent</i> +LN+AE+KM	81.3	54.8	54.8	70.5	75.3	73.7	39.5	23.9	23.4	84.8	50.5	49.0	58.9	43.0	37.2
<i>InferSent</i> +N+AE+KM	79.7	51.7	51.5	67.9	72.3	70.9	37.1	22.7	21.9	85.0	50.8	49.4	58.8	42.9	37.1
LM+Max+LN+PCA+KM	82.0	57.7	57.2	75.1	80.3	79.0	43.7	30.2	29.8	84.6	50.2	48.8	60.3	46.7	41.7
LM+Max+N+PCA+KM	80.3	56.8	56.0	67.4	75.6	74.1	42.7	28.4	27.4	84.3	48.5	47.1	57.3	46.9	41.5
LM+Mean+I+PCA+KM	80.7	59.1	58.2	74.5	79.7	78.6	47.3	33.9	32.3	84.1	49.4	48.0	68.9	61.5	53.7
LM+Mean+LN+PCA+KM	82.2	60.5	59.7	75.0	80.0	78.9	52.0	35.5	35.0	84.6	50.3	48.9	69.5	61.9	54.1
LM+Mean+N+PCA+KM	82.1	60.3	59.6	75.2	80.1	79.0	52.6	35.6	35.2	84.6	50.3	48.9	69.5	61.9	54.1
<i>InferSent</i> +LN+PCA+KM	80.7	53.5	53.4	76.2	76.9	76.0	39.0	24.4	23.8	85.0	51.0	49.6	59.1	43.0	37.1
<i>InferSent</i> +N+PCA+KM	80.7	53.5	53.5	75.6	75.8	74.9	39.4	24.2	23.5	85.0	50.8	49.4	59.0	42.9	37.1

Section 7.1, the results acquired with the PCA-enhanced features are not substantially improved. Hence, these experimental results verify that the features extracted by the pretrained deep model are sufficient for text clustering without further processing.

7.4 Explaining the Clustering Results

We use the *TCRE* model to explain the clustering results for the *AG news* dataset. The explanations for the LM+Mean+LN+KM clustering results are shown in the first part of Table 4. For each clustering group, several indication words represent the cluster and are regarded as an explanation of the clustering results. Four clustering groups are observed in the *AG news* dataset. For our explanation model, the first row of indication words includes geographical and political terms, such as *iraq* and *president*, which are similar to the meaning of the class label *World* in the *AG news* dataset. The second row of indication words includes technological (especially

computer) terms such as *software* and *internet*. Hence, the second row represents the semantics of the class label *Sci/Tech*. The third row includes mainly sports terms, which represent the semantics of the class label *Sports*. The last row of words includes mainly economics and business terms, which represent the semantics of the class label *Business*.

As illustrated in the middle part of Table 4, the explanation of the word frequency model (*FREQ*) for the clustering results often includes noise words. For example, with the *FREQ* method, *said* is an indication word for four clusters, and *new* is an indication word for three clusters. These unrelated noise words make it difficult for the user to discern the meaning of the clustering results. Hence, the *TCRE* model is superior to the *FREQ* model.

In general, the position of a word in an article implies the importance of the word within that article. For example, news stories are organized using an inverted pyramid style, in which information is presented in descending order of importance [40]. Because *AG news* is a news corpus, each text's writing style in the corpus follows the inverted pyramid style. The indication words' positions in an article within this corpus can be used to verify the relative importance of those words. We select two indication words, i.e., *google* and *computer*, which are both indication words for the second cluster, from the *TCRE* and *FREQ* model's explanation results, respectively. We consider information about the relative position of each word in each text and plot a kernel density graph, as shown in Fig. 3. A word's relative position can be acquired by the formula $\frac{\text{word's first occurring position}}{\text{length of article}}$. The indication word *google* discovered by our *TCRE* model almost always appears in the first few sentences. However, the distribution of the indication word *computer* discovered by the *FREQ* model is highly dispersive. These results laterally validate the proposed model's ability to mine clusters' indication words.

We also employ the *TCRE* model to explain the tf-idf-based K-means results for the *AG news* dataset. As shown in the third part of Table 4, the meaning of each row is not apparent. For example, the second row includes indication words related to business and technology: we cannot directly understand the meaning of this cluster. This may occur because the tf-idf-based K-means method achieves low accuracy compared to the LM+Mean+LN+KM configuration. Hence, we can qualitatively analyze the quality of the clustering results according to the *TCRE* model.

TABLE 4
Clustering Explanation Results

LM+Mean+LN+KM's explanation from the <i>TCRE</i> model	
1st cluster:	space inc president company iraq microsoft cup corp scientists prices
2nd cluster:	wireless software internet pc google chips windows ibm product
3rd cluster:	cup league coach team football olympic athens manager formula victory
4th cluster:	prices shares bankruptcy billion quarter accounting jobs fund percent insurance
LM+Mean+LN+KM's explanation from the <i>FREQ</i> model	
1st cluster:	said 's president reuters new ap iraq two us minister
2nd cluster:	new microsoft software 's internet company quot said computer service
3rd cluster:	's new team night first – ap one game last
4th cluster:	said reuters new company inc. 's york oil percent prices
tf-idf-based K-means' explanation from the <i>TCRE</i> model	
1st cluster:	quot quote profile chart research saying 700 two yesterday ap
2nd cluster:	quot quote company microsoft chart profile companies games business billion
3rd cluster:	quot president iraq officials government minister company killed people british
4th cluster:	quot games said game company league team season night president

There are three parts in the table. In each part, each line includes the indication words for some cluster.

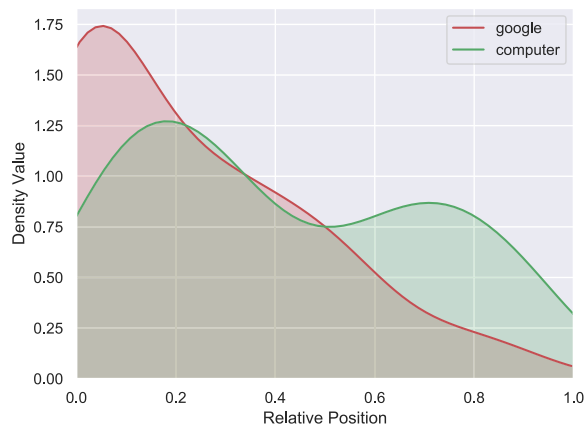


Fig. 3. Kernel density plot of the relative position.

8 CONCLUSION

In this paper, we have proposed a deep feature-based text clustering (DFTC) framework that integrates sequence information and natural language inference semantics. The experimental results show that our DFTC model framework outperforms classic text clustering models and the state-of-the-art pretrained language model, i.e., BERT. The performance of most existing data clustering algorithms relies heavily on the quality of features, and these algorithms are vulnerable to high-dimensional features. Among text clustering algorithms, the bag-of-words model is the most common. Some corpora, such as a social media corpus, will contain some slang words and misspelled words that will induce a high-dimensional feature space. In addition, these models cannot process gaps in word meaning such as synonyms and polysemy. Our proposed text clustering model is based on a deep pretrained model that can construct the meaning of words by contextual information. When processing texts, our model will map the texts into a dense, low-dimensional space, which directly avoids the processing of high-dimensional sparse features. Hence, our model is not vulnerable to high-dimensional data. The DFTC framework can substantially contribute to document organization, corpus summarization, and content-based recommender systems from the perspective of deep semantics. In this paper, we visualize deep text features and investigate the latent mechanisms of DFTC.

Moreover, a text clustering results explanation (TCRE) model is proposed to describe the semantics of the clustering results and provide a qualitative method to help the user analyze the quality of the clustering results. The TCRE model not only demonstrates why DFTC framework models outperform the best-compared methods but also sheds light on why a deep learning-based deep feature extractor can lead to performance improvements. We reveal evidence for why BiLSTMs work well for the extraction of text semantics: the reasoning is based on an inverted pyramid style of writing. However, our current text clustering model is not an end-to-end approach; hence, in the future, we will explore an end-to-end deep text clustering model.

ACKNOWLEDGMENT

The authors are grateful for the support of the National Natural Science Foundation of China (61602207 and 61972174),

the Science Technology Development Project of Jilin Province (20190302107GX), the Special Research and Development of Industrial Technology of Jilin Province under Grant 2019C053-7, the Guangdong Key Project for Applied Fundamental Research (2018KZDXM076) and the Guangdong Premier Key-Discipline Enhancement Scheme under Grant 2016GDYSZDXK036. Fausto Giunchiglia's work was funded by National Project PRIN 2017 "Delphi."

REFERENCES

- [1] T. Kohonen et al., "Self organization of a massive document collection," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 574–585, May 2000.
- [2] C. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds. Boston, MA, USA: Springer, 2012, pp. 77–128.
- [3] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowl.-Based Syst.*, vol. 157, pp. 1–9, 2018.
- [4] R. Guan, X. Shi, M. Marchese, C. Yang, and Y. Liang, "Text clustering with seeds affinity propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 627–637, Apr. 2011.
- [5] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, "Short text conceptualization using a probabilistic knowledgebase," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 2330–2336.
- [6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.
- [7] J. Yin and J. Wang, "A Dirichlet multinomial mixture model-based approach for short text clustering," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 233–242.
- [8] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.
- [9] W. Huang et al., "Hierarchical multi-label text classification: An attention-based recurrent network approach," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 1051–1060.
- [10] A. Lozano-Diez, R. Zazo, D. T. Toledano, and J. Gonzalez-Rodriguez, "An analysis of the influence of deep neural network (DNN) topology in bottleneck feature based language recognition," *PLoS One*, vol. 12, pp. 1–22, Aug. 2017.
- [11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–14.
- [12] J. Xu et al., "Self-Taught convolutional neural networks for short text clustering," *Neural Netw.*, vol. 88, pp. 22–31, 2017.
- [13] Z. Wang, H. Mi, and A. Ittycheriah, "Semi-supervised clustering for short text via deep representation learning," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 31–39.
- [14] D. Snchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *Expert Syst. Appl.*, vol. 39, no. 9, pp. 7718–7728, 2012.
- [15] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 670–680.
- [16] M. E. Peters et al., "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 2227–2237, doi: 10.18653/v1/N18-1202.
- [17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
- [18] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
- [19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.
- [20] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 328–339.
- [21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186.
- [22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, pp. 1–24, Feb. 2019.

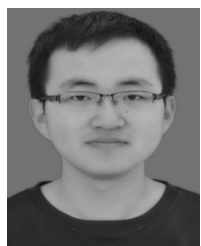
- [23] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
- [24] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
- [25] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1753–1759.
- [26] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards K-means-friendly spaces: Simultaneous deep learning and clustering," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3861–3870.
- [27] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *Proc. Int. Joint Conf. Artif. Intell. Org.*, 2017, pp. 1965–1972.
- [28] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. H. Cernocky, "RNNLM - Recurrent neural network language modeling toolkit," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2011.
- [29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [30] A. Graves, "Generating sequences with recurrent neural networks," 2014, *arXiv:1308.0850*.
- [31] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.
- [32] S. Merity, N. S. Keskar, and R. Socher, "An analysis of neural language modeling at multiple scales," 2018, *arXiv:1803.08240*.
- [33] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
- [34] X. Zhang and Y. LeCun, "Text understanding from scratch," 2016, *arXiv:1502.01710*.
- [35] C. Deng, X. He, and J. Han, "Locally consistent concept factorization for document clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 902–913, Jun. 2011.
- [36] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2003, Art. no. 267.
- [37] K. Y. Yeung and W. L. Ruzzo, "Principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001.
- [38] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc. Nat. Acad. Sci. USA*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.
- [39] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.
- [40] R. Kevelson, *The Inverted Pyramid: An Introduction to a Semiotics of Media Language*. Bloomington, IN, USA: Indiana Univ., 1977.



research interests include machine learning, bioinformatics, and knowledge engineering.

Renchu Guan (Member, IEEE) received the PhD degree from Jilin University, Changchun, China, in 2010. He is currently a professor with the College of Computer Science and Technology, Jilin University, China. He has published more than 50 papers. His research was featured in the *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Geoscience and Remote Sensing*, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, etc. He was the recipient of several grants from NSFC. His

Hao Zhang received the BSc degree in computer science and technology from Jilin University, Changchun, China, in 2017. He is currently working toward the graduate degree in the College of Computer Science and Technology, Jilin University, Changchun, China. He is supervised by Professor Renchu Guan. His major research interests include information extraction and machine learning.



Yanchun Liang received the PhD degree in applied mathematics from Jilin University, Changchun, China, in 1997. He is currently a professor in the College of Computer Science and Technology, Jilin University, China. He was a visiting scholar with Manchester University, U.K. from 1990 to 1991, a visiting professor with the National University of Singapore from 2000 to 2001, a guest professor with the Institute of High Performance Computing of Singapore from 2002 to 2004, a guest professor with Trento University, Italy, from 2006 to 2010, and a guest professor with the University of Missouri, Missouri, from 2011 to 2017. He has published more than 400 papers. His research has been featured in the *Bioinformatics*, the *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Knowledge and Data Engineering*, *Journal of Micromechanics and Micro-engineering*, *Physical Review E*, *Neural Computing & Applications*, *Smart Materials and Structures*, *Artificial Intelligence in Medicine*, *Applied Artificial Intelligence*, etc. He was the recipient of several grants from NSFC and the EU, among others. His research interests include computational intelligence, machine learning methods, text mining, MEMS modeling, and bioinformatics.



Fausto Giunchiglia received the PhD degree in computer engineering from the Faculty of Engineering, University of Genoa, Genoa, Italy. He is currently a full professor with the Faculty of Science, University of Trento, Italy. He is a member of the Academia Europaea. His research has been featured in IJCAI, AAAI, AIJ, etc. He was the recipient of several grants from the EU. His research interests concentrate on artificial intelligence, formal methods, knowledge management, and agent-oriented software engineering.



Lan Huang received the PhD degree from the College of Computer Science and Technology, Jilin University, Changchun, China, in 2003. She is currently a professor and a supervisor for PhD candidates. She is engaged mainly in business intelligence theory and application research. She was one of the winners of the Outstanding Youth Project Fund of Jilin Province in 2005 and the person in charge of the Young and Middle-aged Leader and Innovation Team of Jilin Province in 2012. She was invited to Italy's Trento University as a senior visitor in 2010. As PI and Co-PI, she has undertaken more than 10 teaching and scientific research projects granted by the National 863 Hi-tech Research and Development Program, the National Natural Science Foundation of China, provincial/ministerial foundations, and other sources. The works that she participated in as the main investigator were awarded first prize in 2017 for the Jilin Province Scientific and Technological Progress Award and first prizes in 2010 and 2014 for the National Commercial Science and Technology Award bestowed by the China General Chamber of Commerce. In recent years, her research interests have focused on business intelligence applications and social network mining algorithms. She has published 64 academic papers and obtained fourteen software copyrights. The software results researched and developed by her team have brought considerable economic benefits for cooperative enterprises and application enterprises.



Xiaoyue Feng received the PhD degree in computer science from Jilin University, Changchun, China, in 2008. She is currently an associate professor with the College of Computer Science and Technology, Jilin University. Her main interests include bioinformatics, machine learning, and text mining.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.