Leveraging BERT and TFIDF Features for Short Text Clustering via Alignment-Promoting Co-Training

Zetong Li[1], **Qinliang Su**[1*], **Shijing Si**[2], **Jianxing Yu**[3]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]School of Economics and Finance, Shanghai International Studies University, Shanghai, China
[3]School of Artificial Intelligence, Sun Yat-sen University, Guangdong, China
{lizt9@mail2,suqliang@mail,yujx26@mail}.sysu.edu.cn,shijing.si@shisu.edu.cn

Abstract

BERT and TFIDF features excel in capturing rich semantics and important words, respectively. Since most existing clustering methods are solely based on the BERT model, they often fall short in utilizing keyword information, which, however, is very useful in clustering short texts. In this paper, we propose a **CO**-**T**raining **C**lustering (**COTC**) framework to make use of the collective strengths of BERT and TFIDF features. Specifically, we develop two modules responsible for the clustering of BERT and TFIDF features, respectively. We use the deep representations and cluster assignments from the TFIDF module outputs to guide the learning of the BERT module, seeking to align them at both the representation and cluster levels. Reversely, we also use the BERT module outputs to train the TFIDF module, thus leading to the mutual promotion. We then show that the alternating co-training framework can be placed under a unified joint training objective, which allows the two modules to be connected tightly and the training signals to be propagated efficiently. Experiments on eight benchmark datasets show that our method outperforms current SOTA methods significantly.

1 Introduction

Short text clustering aims to group short text segments with similar semantics into the same clusters without leveraging external label information. It is widely used in various real-world applications, such as topic discovery (Kim et al., 2013), news recommendation (Bouras and Tsogkas, 2017), spam detection (Wu and Liu, 2018), *etc.* Traditionally, TFIDF feature, which is computed as a kind of re-weighted word frequency, was the main text feature used for clustering, in which low-dimensional representations were first learned from the features and then fed into a clustering head like K-Means or deep embedded clustering (DEC) (Xie et al., 2016) (Yang et al., 2017; Guo et al., 2017; Xu et al., 2017; Hadifar et al., 2019). However, due to the lack of deep semantic information of texts in TFIDF features, these methods are generally not very competitive. Inspired by the great success of the BERT model (Devlin et al., 2019), some recent methods have proposed to apply a clustering head over the features output from BERT, *i.e.*, BERT features. For instance, (Huang et al., 2020; Zhang et al., 2021) proposed to use the classic DEC loss on top of the BERT features to encourage the emergence of cluster structure. Following the work (Zhang et al., 2021), (Yin et al., 2022) further proposed to use the cluster centers to reconstruct the texts, encouraging more information to be preserved in the learned centers. In (Zheng et al., 2023), instead of employing the DEC loss, it proposed to first pseudo-label the texts and then use the pseudo-labels to train the clustering model like a classifier.

Despite the substantial performance improvement observed in the BERT-based clustering methods, it does not mean that the BERT features are overwhelmingly superior to the TFIDF ones in the context of clustering. One of the limitations of BERT feature is exhibited by its weak ability in capturing the *keyword information* for texts from some professional fields (Schick and Schütze, 2020; Haley, 2020; Chen and Su, 2023). That is because BERT is trained mainly on general texts, making it not sensitive to the professional words that are rarely seen. As shown in Figure 1, for a dataset involving the domain of QT programming, words like QT, XYZ, QMAKESPEC, CURSOR can largely determine which topic or cluster a text belongs to. But since these words rarely appear in the pre-training corpus, their accurate meanings cannot be well captured by the pre-trained BERT model. Thus, if we solely use the BERT features to perform clustering, the performance could be highly restricted. By contrast, since TFIDF features are word-frequency features, keyword information can

*Corresponding author.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 14897–14913

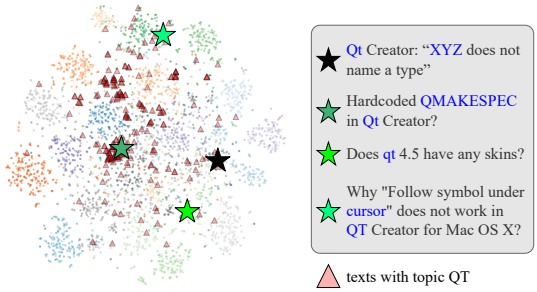

Figure 1: T-SNE of BERT features of 5000 random samples from a domain-specific dataset StackOverflow with backbone *distilbert-base-nli-stsb-mean-tokens*. The three green-star texts are the top-3 nearest neighbors of the black-star one in TFIDF features, and they all come from the same topic. But we can see that their BERT features are far away from each other.

actually be well captured by them. These keywords are closely related to the topics of the texts, thus it is helpful to connect in-cluster samples for clustering short texts (Zhao et al., 2012; Habibi and Popescu-Belis, 2015). Given that BERT and TFIDF features possess different strengths (*i.e.*, deep semantics and keyword signals), it is thereby of great value to use them together. But how to leverage their collective strengths is still under-explored. An intuitive way to make use of the two features is to fuse them into one feature, *e.g.*, concatenating them, adding them, *etc.*, and then feed the fused feature into a current SOTA clustering model. However, due to the intrinsically different natures of the two features, we observe that simply fusing them together cannot bring too much improvement.

To harvest the respective strengths of BERT and TFIDF features, in this paper, a **CO**-**T**raining **C**lustering (**COTC**) framework comprised of BERT and TFIDF modules is developed. In the BERT module, we propose to learn the semantic-rich representations and cluster assignments from BERT features with the help of contrastive learning and pseudo-labelling techniques. Furthermore, specific methods are developed to explicitly encourage the deep representations and cluster assignments to align with those learned in the TFIDF module. As for the TFIDF module, we further introduce a generative model VAE for TFIDF features and use it to output the TFIDF-induced representations and cluster assignments, with alignment with outputs from the BERT module also considered. By making use of outputs from the other module, the two modules can be trained to mutually promote each other's performance progressively. We finally show that the alternating co-training framework can be placed under a more unified joint training objective which connects the two modules more tightly and allows a more efficient propagation of the training signals between modules. Extensive experiments on eight datasets demonstrate that COTC achieves superior clustering performance than current state-of-the-art methods by a significant margin.

2 Related Work

Short text clustering is challenging due to very few words within short texts. Early studies (Banerjee et al., 2007; Hu et al., 2009) enrich texts with Wikipedia and apply K-Means or hierarchical agglomerative clustering to BoW features. Since these features are too sparse to convey enough information, some works (Yang et al., 2017; Guo et al., 2017) thus use neural networks to learn better representations, and others (Xu et al., 2017; Hadifar et al., 2019) instead use dense Word2Vec embeddings (Mikolov et al., 2013) for clustering with the DEC loss (Xie et al., 2016). Given that shallow Word2Vec embeddings are unable to capture deep semantic information, recent methods prefer BERT features (Devlin et al., 2019) due to their successes in numerous tasks. (Huang et al., 2020) is the first to fine-tune the BERT model with the masked language model loss and the DEC loss, but the subsequent ones believe that contrastive learning (Chen et al., 2020) can learn better representations. Thus, (Zhang et al., 2021) combines this idea with DEC to achieve clustering, and (Yin et al., 2022) is based on it with a topic modeling module to enhance the semantics of the representations. (Zheng et al., 2023) points out that DEC prones to assign all samples to one cluster, and instead employs pseudo-labelling (YM. et al., 2020) for clustering, leading to the best performance to date. Different from these methods using only one type of imperfect text features, we develop a co-training clustering framework to leverage the complementary strengths of BERT features and seemingly outdated but easily available TFIDF features.

3 Method

3.1 Overall Framework of COTC

Existing works mostly focus on how to leverage either BERT features $\boldsymbol{b}_i = \mathcal{B}(\boldsymbol{x}_i)$ or TFIDF features $\boldsymbol{t}_i = \mathcal{T}(\boldsymbol{x}_i)$ to cluster a text dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, where $\mathcal{B}(\cdot)$ and $\mathcal{T}(\cdot)$ denote the BERT and TFIDF transformations. Since the two features possess

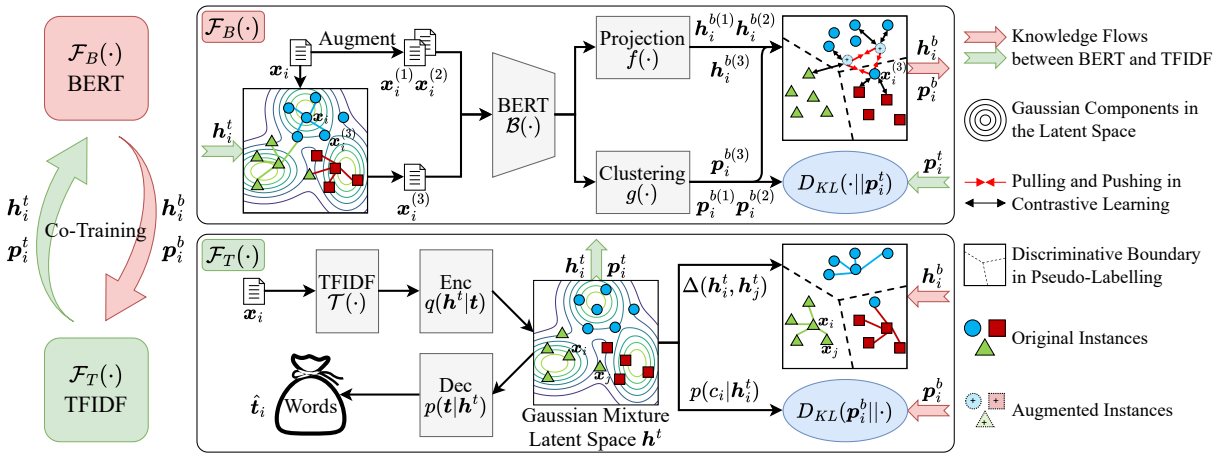

Figure 2: The overall architecture of the co-training clustering framework **COTC**.

complementary strengths, in this paper, we propose to develop a co-training framework to exploit their strengths simultaneously. Specifically, suppose a low-dimensional representation $\boldsymbol{h}_i^t$ and cluster probability $\boldsymbol{p}_i^t$ over the K clusters have been learned from the TFIDF feature $\boldsymbol{t}_i$. Then, we learn the BERT-induced representation $\boldsymbol{h}_i^b$ and cluster probability $\boldsymbol{p}_i^b$ from the BERT feature $\boldsymbol{b}_i$ as

$$\boldsymbol{h}_i^b, \boldsymbol{p}_i^b = \mathcal{F}_B\left(\boldsymbol{b}_i, \{\boldsymbol{h}_i^t\}_{i=1}^N, \{\boldsymbol{p}_i^t\}_{i=1}^N\right), \quad (1)$$

where $\mathcal{F}_B(\cdot)$ means the mapping function. Similarly, we also make use of $\boldsymbol{h}_i^b$ and $\boldsymbol{p}_i^b$ to learn the TFIDF-induced representation $\boldsymbol{h}_i^t$ and cluster probability $\boldsymbol{p}_i^t$ from the TFIDF feature $\boldsymbol{t}_i$ as follows

$$\boldsymbol{h}_i^t, \boldsymbol{p}_i^t = \mathcal{F}_T\left(\boldsymbol{t}_i, \{\boldsymbol{h}_i^b\}_{i=1}^N, \{\boldsymbol{p}_i^b\}_{i=1}^N\right). \quad (2)$$

The two modules $\mathcal{F}_B(\cdot)$ and $\mathcal{F}_T(\cdot)$ can be updated alternately to mutually promote each other. The overall architecture of the co-training clustering framework **COTC** is shown in Figure 2.

3.2 Implementation of Module $\mathcal{F}_B(\cdot)$

Since BERT representations $\{\boldsymbol{h}_i^b\}_{i=1}^N$ and TFIDF representations $\{\boldsymbol{h}_i^t\}_{i=1}^N$ lie in two totally different representation spaces, directly requiring them to be globally aligned is unreasonable. But if two texts look similar in the TFIDF representation space, they should also be similar in the BERT representation space, *i.e.*, the local topological structures of the two spaces should be aligned. To exploit this alignment, we first construct a similarity graph $\mathcal{G}^t(\mathcal{V}, \mathcal{E}^t)$ with TFIDF representations $\{\boldsymbol{h}_i^t\}_{i=1}^N$, where $\mathcal{V} = \{1, 2, \cdots, N\}$ is the set of texts and $\mathcal{E}^t = \{(i,j)|i \in \mathcal{V}, j \in \mathcal{N}_i^t\}$ is the set of edges; and

$$\mathcal{N}_i^t = \{j | j \neq i \ \& \ \cos(\boldsymbol{h}_i^t, \boldsymbol{h}_j^t) \text{ is top-}L \text{ largest}\} \quad (3)$$

denotes the set of top-L nearest neighbors of the text $\boldsymbol{x}_i$. Then, we propose to learn the BERT representation $\boldsymbol{h}_i^b$ under the contrastive learning framework (Chen et al., 2020) by making use of this similarity graph. Specifically, in addition to augmenting the text $\boldsymbol{x}_i$ into two augmentations $\boldsymbol{x}_i^{(1)}$ and $\boldsymbol{x}_i^{(2)}$ by utilizing contextual augmenter (Kobayashi, 2018; Ma, 2019), we generate one more augmentation of $\boldsymbol{x}_i$ by randomly selecting a sample from its neighbor set $\mathcal{N}_i^t$, with the third augmentation denoted as $\boldsymbol{x}_i^{(3)}$. By viewing the samples $\{\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, \boldsymbol{x}_i^{(3)}\}$ as the positives, we define a contrastive loss

$$\mathcal{L}_{Contr} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=2}^{3}\log \ell_{ij}, \quad (4)$$

where $\ell_{ij} \triangleq \frac{\Delta(\boldsymbol{h}_i^{b(1)}, \boldsymbol{h}_i^{b(j)})}{\Delta(\boldsymbol{h}_i^{b(1)}, \boldsymbol{h}_i^{b(j)}) + \sum_{k\neq i, m\in\{1,j\}} \Delta(\boldsymbol{h}_i^{b(1)}, \boldsymbol{h}_k^{b(m)})}$; and

$$\boldsymbol{h}_i^{b(m)} = f(\boldsymbol{b}_i^{(m)}) = f(\mathcal{B}(\boldsymbol{x}_i^{(m)})) \quad (5)$$

denotes the representation of the m-th ($m = 1, 2, 3$) augmentation of the text $\boldsymbol{x}_i$, with $f(\cdot)$ and $\mathcal{B}(\cdot)$ representing an MLP neural network and the BERT backbone, respectively; and $\Delta(\boldsymbol{h}_i^{b(1)}, \boldsymbol{h}_i^{b(j)}) = e^{\cos(\boldsymbol{h}_i^{b(1)}, \boldsymbol{h}_i^{b(j)})/\tau}$ measures the similarity between the vectors $\boldsymbol{h}_i^{b(1)}$ and $\boldsymbol{h}_i^{b(j)}$, with τ meaning the temperature parameter. Due to viewing neighbors in $\mathcal{N}_i^t$ as positives, minimizing the loss $\mathcal{L}_{Contr}$ will encourage the similarity structures in the BERT and TFIDF representation spaces to be aligned.

Given BERT features $\{\boldsymbol{b}_i\}_{i=1}^N$ with similarity structure learned in the BERT representation space, we further compute the cluster probability $\boldsymbol{p}_i^b$ by applying a clustering head over BERT features, such as K-Means, Gaussian mixture model (GMM), deep embedded clustering (DEC), *etc.* Here, we propose to directly apply an MLP neural netwrok and a softmax function over the BERT feature $\boldsymbol{b}_i$ and then use pseudo-labelling technique to self-train the clustering head (*i.e.*, classifier). Specifically, we compute the cluster probability as

$$\boldsymbol{p}_i^b = \delta(g(\boldsymbol{b}_i)) = \delta(g(\mathcal{B}(\boldsymbol{x}_i))), \tag{6}$$

where $\delta(\cdot)$ means the softmax function; and $g(\cdot)$ and $\mathcal{B}(\cdot)$ represent an MLP neural network and the BERT backbone, respectively. To train the model, we propose to first infer the pseudo-labels from the predicted probability $\boldsymbol{p}_i^b$ and then use the pseudo-labels to train the clustering head as well as the backbone with a cross-entropy loss. There are different ways to infer the pseudo-labels, such as treating the cluster index corresponding to the maximum probability as the pseudo-label. In our experiments, we follow recent works (YM. et al., 2020; Zheng et al., 2023) to infer the pseudo-labels by solving an optimal transport (OT) problem over the predicted probabilities $\{\boldsymbol{p}_i^b\}_{i=1}^N$. Please refer to Appendix A.1 for details on the OT problem. By denoting the pseudo-label obtained for the text $\boldsymbol{x}_i$ as $\boldsymbol{q}_i$, which is a one-hot vector, we train the model by minimizing the following cross-entropy loss

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{3} \boldsymbol{q}_i^T \log \delta(g(\boldsymbol{b}_i^{(m)})), \tag{7}$$

where $\boldsymbol{b}_i^{(m)} = \mathcal{B}(\boldsymbol{x}_i^{(m)})$. Note that the model is encouraged to predict the same pseudo-label for all of the three augmented texts $\boldsymbol{x}_i^{(1)}$, $\boldsymbol{x}_i^{(2)}$ and $\boldsymbol{x}_i^{(3)}$. Moreover, to ensure the robustness of pseudo-labelling technique (Englesson and Azizpour, 2021), we also encourage the model to output consistent probability distribution for the original text $\boldsymbol{x}_i$ and its augmentions $\boldsymbol{x}_i^{(m)}$ by minimizing the loss

$$\mathcal{L}_{Consist} = \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{3} D_{KL}(\delta(g(\boldsymbol{b}_i))||\delta(g(\boldsymbol{b}_i^{(m)}))), \tag{8}$$

where $D_{KL}(\cdot)$ means the KL-divergence.

In addition to the consistency constraints among neighbors, the predicted probability inferred from the BERT feature $\mathcal{B}(\boldsymbol{x}_i)$ should also be aligned with that inferred from the TFIDF feature $\mathcal{T}(\boldsymbol{x}_i)$ to achieve the cluster-level alignment between modules. This goal can be reached by minimizing the loss

$$\mathcal{L}_{Align} = \frac{1}{N}\sum_{i=1}^{N} D_{KL}(\delta(g(\boldsymbol{b}_i))||\boldsymbol{p}_i^t), \tag{9}$$

where $\boldsymbol{p}_i^t$ denotes the cluster probability inferred from the TFIDF feature $\boldsymbol{t}_i$, and the method to obtain it will be presented in the next section. The whole module is trained by minimizing the loss

$$\mathcal{L}_B = \mathcal{L}_{Contr} + \mathcal{L}_{Cluster} + \lambda\mathcal{L}_{Align}, \tag{10}$$

where $\mathcal{L}_{Cluster} \triangleq \mathcal{L}_{CE} + \mathcal{L}_{Consist}$ is the loss responsible for clustering; and λ is the weighting parameter.

3.3 Implementation of Module $\mathcal{F}_T(\cdot)$

This module aims to learn the TFIDF representation $\boldsymbol{h}_i^t$ and cluster probability $\boldsymbol{p}_i^t$ from the TFIDF feature $\boldsymbol{t}_i = \mathcal{T}(\boldsymbol{x}_i)$. Different from $\mathcal{F}_B(\cdot)$ established on contrastive learning and pseudo-labelling techniques, we propose to use a VAE to model $\boldsymbol{t}_i$, so that keyword information can be preserved during reconstruction. We then use the VAE's encoder to output the TFIDF representation $\boldsymbol{h}_i^t$ and cluster probability $\boldsymbol{p}_i^t$. In order to align with BERT representations $\{\boldsymbol{h}_i^b\}_{i=1}^N$, similar to $\mathcal{F}_B(\cdot)$, we first use $\{\boldsymbol{h}_i^b\}_{i=1}^N$ to construct a similarity graph $\mathcal{G}^b(\mathcal{V}, \mathcal{E}^b)$, where $\mathcal{E}^b = \{(i,j)|i \in \mathcal{V}, j \in \mathcal{N}_i^b\}$ represents the set of edges with

$$\mathcal{N}_i^b = \{j | j \neq i \ \&\cos(\boldsymbol{h}_i^b, \boldsymbol{h}_j^b) \text{ is top-}L\text{ largest}\}. \tag{11}$$

Then, to achieve the alignment in the representation space, we require $\boldsymbol{h}_i^t$ responsible for the generation of the TFIDF feature $\boldsymbol{t}_i$ as well as the similarity graph $\mathcal{G}^b$. Moreover, to facilitate clustering, we encourage the TFIDF representation $\boldsymbol{h}_i^t$ to exhibit cluster structure by using a latent Gaussian mixture prior distribution. Thus, we build the generative model as the following form

$$p(\{\boldsymbol{t}_i\}_{i=1}^N, \mathcal{G}^b, \{\boldsymbol{h}_i^t\}_{i=1}^N, \{c_i\}_{i=1}^N) = \prod_{i=1}^{N} p(\boldsymbol{t}_i|\boldsymbol{h}_i^t) p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N) p(\boldsymbol{h}_i^t|c_i) p(c_i), \tag{12}$$

where c_i is randomly drawn from $\{1, 2, \cdots, K\}$ and $p(c_i) = \mathrm{Cat}(c_i; \boldsymbol{\pi})$ with a K-dimensional vector $\boldsymbol{\pi}$ denoting the class prior distribution; and $p(\boldsymbol{h}_i^t|c_i) = \mathcal{N}(\boldsymbol{h}_i^t; \boldsymbol{\mu}_{c_i}, diag(\boldsymbol{\sigma}_{c_i}^2))$ with $\boldsymbol{\mu}_{c_i}$ and $\boldsymbol{\sigma}_{c_i}^2$ denoting the mean and variance; and $p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)$ is the decoder responsible for the generation of the TFIDF feature $\boldsymbol{t}_i$ (see Appendix A.2); and $p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)$ is the decoder responsible for the generation of the similarity graph, and is defined as

$$p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N) = \prod_{j\in\mathcal{N}_i^b} \frac{\Delta(\boldsymbol{h}_i^t, \boldsymbol{h}_j^t)}{\sum_{k\neq i}\Delta(\boldsymbol{h}_i^t, \boldsymbol{h}_k^t)}, \quad (13)$$

where $\Delta(\boldsymbol{h}_i^t, \boldsymbol{h}_j^t) = e^{\cos(\boldsymbol{h}_i^t, \boldsymbol{h}_j^t)/\tau}$ measures the similarity between the vectors $\boldsymbol{h}_i^t$ and $\boldsymbol{h}_j^t$.

The generative model can be trained by minimizing the negative evidence lower bound (ELBO) $\mathcal{L}_{ELBO} = \frac{1}{N}\sum_{i=1}^N \ell_i^{elbo}\left(q(\boldsymbol{h}_i^t, c_i|\boldsymbol{t}_i)\right)$, where

$$\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t, c_i|\boldsymbol{t}_i)\right) = -\mathbb{E}_q\left[\log\frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t|c_i)p(c_i)}{q(\boldsymbol{h}_i^t, c_i|\boldsymbol{t}_i)}\right], \quad (14)$$

and $\mathbb{E}_q[\cdot]$ means the expectation w.r.t. the variational posterior $q(\boldsymbol{h}_i^t, c_i|\boldsymbol{t}_i)$, which can be defined in different forms. By restricting $q(\boldsymbol{h}_i^t, c_i|\boldsymbol{t}_i)$ to the form $q(\boldsymbol{h}_i^t, c_i|\boldsymbol{t}_i) = q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)p(c_i|\boldsymbol{h}_i^t)$, we obtain

$$\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)p(c_i|\boldsymbol{h}_i^t)\right) = -\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\left[\log\frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t)}{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\right], \quad (15)$$

where $q(\boldsymbol{h}_i^t|\boldsymbol{t}_i) = \mathcal{N}(\boldsymbol{h}_i^t; \boldsymbol{\mu}(\boldsymbol{t}_i), diag(\boldsymbol{\sigma}^2(\boldsymbol{t}_i)))$, with $\boldsymbol{\mu}(\cdot)$ and $\boldsymbol{\sigma}^2(\cdot)$ denoting the outputs of two neural networks; and $p(c_i|\boldsymbol{h}_i^t) = \frac{p(\boldsymbol{h}_i^t|c_i)p(c_i)}{\sum_c p(\boldsymbol{h}_i^t|c)p(c)} = \frac{\pi_{c_i}\mathcal{N}(\boldsymbol{h}_i^t; \boldsymbol{\mu}_{c_i}, diag(\boldsymbol{\sigma}_{c_i}^2))}{\sum_c \pi_c\mathcal{N}(\boldsymbol{h}_i^t; \boldsymbol{\mu}_c, diag(\boldsymbol{\sigma}_c^2))}$, which is the posterior distribution of the joint prior $p(\boldsymbol{h}_i^t|c_i)p(c_i)$. After the VAE is trained, we can use its encoder to output the TFIDF representation $\boldsymbol{h}_i^t$ and cluster probability $\boldsymbol{p}_i^t$ as

$$\boldsymbol{h}_i^t = \mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}[\boldsymbol{h}] = \boldsymbol{\mu}(\boldsymbol{t}_i), \quad (16)$$

$$\boldsymbol{p}_i^t[c] = \mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\left[\frac{\pi_c\mathcal{N}(\boldsymbol{h}_i^t; \boldsymbol{\mu}_c, diag(\boldsymbol{\sigma}_c^2))}{\sum_c \pi_c\mathcal{N}(\boldsymbol{h}_i^t; \boldsymbol{\mu}_c, diag(\boldsymbol{\sigma}_c^2))}\right], \quad (17)$$

where $\boldsymbol{\mu}(\boldsymbol{t}_i)$ is the mean defined in $q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)$; and $\boldsymbol{p}_i^t[c]$ represents the c-th element of $\boldsymbol{p}_i^t$. In practice, the expectation in $\boldsymbol{p}_i^t[c]$ can be approximated by a sample drawn from $q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)$.

Moreover, since the two predicted probabilities inferred from BERT and TFIDF features of the text $\boldsymbol{x}_i$ should be largely identical, similar to that done in the module $\mathcal{F}_B(\cdot)$, we further encourage the alignment between the two predicted probabilities by minimizing the alignment loss $\mathcal{L}_{Align} = \frac{1}{N}\sum_{i=1}^N D_{KL}(\boldsymbol{p}_i^b||\boldsymbol{p}_i^t)$. Thus, the whole module can be trained by minimizing the loss

$$\mathcal{L}_T = \mathcal{L}_{ELBO} + \lambda'\mathcal{L}_{Align}, \quad (18)$$

where λ' is the weighting parameter.

3.4 A Unified Training Objective

The entire model can be trained by optimizing the losses $\mathcal{L}_B = \mathcal{L}_{Contr} + \mathcal{L}_{Cluster} + \lambda\mathcal{L}_{Align}$ and $\mathcal{L}_T = \mathcal{L}_{ELBO} + \lambda'\mathcal{L}_{Align}$ in an alternating manner. But in practice, we can directly minimize the joint loss $\mathcal{L}_{Joint} = \mathcal{L}_B + \lambda_1\mathcal{L}_T$, that is,

$$\mathcal{L}_{Joint} = \mathcal{L}_{Contr} + \mathcal{L}_{Cluster} + \lambda_1(\mathcal{L}_{ELBO} + \lambda_2\mathcal{L}_{Align}), \quad (19)$$

where λ_1 and λ_2 are the two weighting parameters. We can show that

$$\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)p(c_i|\boldsymbol{h}_i^t)\right) + \ell_i^{align} \leq \ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c]\right), \quad (20)$$

where $\ell_i^{align} = D_{KL}(\boldsymbol{p}_i^b||\boldsymbol{p}_i^t)$; and $\boldsymbol{p}_i^b[c]$ denotes the c-th element of $\boldsymbol{p}_i^b$. Detailed derivation is provided in Appendix A.2. Comparing with the ELBO $\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)p(c_i|\boldsymbol{h}_i^t)\right)$, we can see that in the ELBO $\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c]\right)$, the cluster probability $\boldsymbol{p}_i^b[c]$ from the BERT module is used as the variational posterior in the TFIDF module. From the inequality, we have

$$\mathcal{L}_{ELBO} + \mathcal{L}_{Align} \leq \mathcal{L}'_{ELBO}, \quad (21)$$

where $\mathcal{L}'_{ELBO} \triangleq \frac{1}{N}\sum_{i=1}^N \ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c]\right)$. By setting $\lambda_2 = 1$ in (19), with the inequality, we can obtain a new joint training loss

$$\mathcal{L}'_{Joint} = \mathcal{L}_{Contr} + \mathcal{L}_{Cluster} + \lambda_1\mathcal{L}'_{ELBO}. \quad (22)$$

By using the loss $\mathcal{L}'_{Joint}$, the cluster probability $\boldsymbol{p}_i^b[c]$ output from the BERT module $\mathcal{F}_B(\cdot)$ is directly used as the variational posterior to train the TFIDF module $\mathcal{F}_T(\cdot)$. Hence, the gradient in the module $\mathcal{F}_T(\cdot)$ can be directly backpropagated into the module $\mathcal{F}_B(\cdot)$ via the variational posterior $\boldsymbol{p}_i^b[c]$, enabling the training signal to be propagated between the two modules more efficiently. Moreover, according to the property of the KL-divergence, directly optimizing the alignment

term $\mathcal{L}_{Align} = \frac{1}{N}\sum_{i=1}^{N} D_{KL}(\boldsymbol{p}_i^b||\boldsymbol{p}_i^t)$ tends to encourage the distribution $\boldsymbol{p}_i^t$ to allocate some probabilities over all of the K clusters, otherwise it is risky to incur a very large loss. Thus, if the loss $\mathcal{L}_{Joint}$ is used, it is difficult to yield sharp predicted probabilities $\boldsymbol{p}_i^t$ and $\boldsymbol{p}_i^b$. But by using the loss $\mathcal{L}'_{Joint}$, the problem can be circumvented since it does not involve the term explicitly. In addition, when the loss $\mathcal{L}'_{Joint}$ is employed, the Gumbel-Softmax trick (Jang et al., 2017) can be employed to approximate the expectation $\mathbb{E}_{\boldsymbol{p}_i^b[c_i]}[\cdot]$ over the categorical variable c_i in $\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]\right) = -\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}\Big[\log \frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t|c_i)p(c_i)}{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}\Big]$. Together with the Gaussian re-parameterization trick that is used to approximate the expectation $\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}[\cdot]$ over the variable $\boldsymbol{h}_i^t$, the loss $\mathcal{L}'_{Joint}$ can be optimized efficiently. After the model is trained, the predicted cluster probability $\boldsymbol{p}_i^b$ is used to obtain the final clustering result.

4 Experiment

4.1 Experimental Setup

Datasets We assess our method on eight benchmark datasets for short text clustering including **AgNews** (Rakib et al., 2020), **SearchSnippets** (Phan et al., 2008), **StackOverflow**, **Biomedical** (Xu et al., 2017), **GoogleNews-TS**, **GoogleNews-T**, **GoogleNews-S**, **Tweet** (Yin and Wang, 2014). The details are described in Appendix B.

Baselines We compare our method with the following baselines, similar to (Zheng et al., 2023). **TFIDF-K-Means** and **BERT-K-Means** apply K-Means to TFIDF and BERT features respectively. **K-Means_IC** (Rakib et al., 2020) applies K-Means to TFIDF features with iterative classification algorithm. **STC^2-LPI** (Xu et al., 2017) uses Word2Vec to fit codes pre-obtained with LPI (He and Niyogi, 2003) through a CNN. Then, the deep representations are clustered with K-Means. **Self-Train** (Hadifar et al., 2019) uses an autoencoder to model Word2Vec enhanced with SIF (Arora et al., 2017). Then, the encoder network is fine-tuned with DEC loss. **SCCL** (Zhang et al., 2021) performs contrastive learning on BERT features and achieves clustering with DEC loss. **RSTC** (Zheng et al., 2023) performs pseudo-labelling on BERT features, where pseudo-labels are obtained by solving the OT problem (YM. et al., 2020). We also include **GMVAE** to model TFIDF features via VAE with Gaussian mixture prior (Jiang et al., 2017) as a comparison to the TFIDF module $\mathcal{F}_T(\cdot)$ in our method. Following previous works, the BERT model is set as *distilbert-base-nli-stsb-mean-tokens* by default.

For comprehensive comparison, we include some simple methods that combine BERT and TFIDF features intuitively as additional baselines. **RSTC$_{\text{BERT-TFIDF-Linear}}$** uses an autoencoder to reduce the TFIDF feature from $\boldsymbol{t}$ to $\hat{\boldsymbol{t}}$ with the same dimensionality as the BERT feature $\boldsymbol{b}$. Then, $\boldsymbol{b}$ and $\hat{\boldsymbol{t}}$ are linearly combined as $\omega\boldsymbol{b} + (1-\omega)\hat{\boldsymbol{t}}$ with $\omega = 0.5$ and fed into the previous SOTA clustering method RSTC. **RSTC$_{\text{BERT-TFIDF-Concat-1}}$** concatenates the BERT feature and the reduced TFIDF feature as $[\boldsymbol{b}; \hat{\boldsymbol{t}}]$ and feeds the combined one into RSTC. **RSTC$_{\text{BERT-TFIDF-Concat-2}}$** is a similar method to RSTC$_{\text{BERT-TFIDF-Concat-1}}$, except that the original TFIDF feature is chosen to concatenate as $[\boldsymbol{b}; \boldsymbol{t}]$.

Evaluation Metrics We evaluate our method with the clustering accuracy (ACC) and normalized mutual information (NMI) metrics. We average results over 5 different random runs. Each run stops when the change rate of the clustering assignments between two consecutive epochs is less than 0.01 or the number of epochs reaches 25. We left training details in Appendix C.

4.2 Experimental Result

4.2.1 Performance Comparison

The clustering performance of the baselines and our method **COTC** on eight benchmark datasets is presented in Table 1. Compared with the baselines, COTC obtains promising performance across all datasets in terms of the ACC and NMI metrics. Furthermore, several observations are noteworthy. Shallow clustering methods applying K-Means to raw features, *i.e.* TFIDF-K-Means and BERT-K-Means, do not produce satisfactory performance. Deep clustering methods using TFIDF features, *e.g.* K-Means_IC, seem to be underperforming, and those using Word2Vec like STC^2-LPI and Self-Train perform slightly better but not very well either. In contrast, methods using BERT features such as SCCL and RSTC outperform previous methods by a large margin, validating the power of BERT features. However, although TFIDF features do not show any potential when clustered with K-Means, they show decent performance beating most baselines when modeled with GMVAE, validating that seemingly outdated TFIDF features are still valuable. This can be further verified in methods combining BERT and TFIDF such

Table 1: The clustering performance of the baselines and our method **COTC** on eight benchmark datasets. The results of the baselines are quoted from (Zheng et al., 2023). The best results are bolded and the second ones are underlined.

Method	AgNews		SearchSnippets		StackOverflow		Biomedical	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
TFIDF-K-Means	34.39	12.19	30.85	18.67	58.52	59.02	29.13	25.12
BERT-K-Means	65.95	31.55	55.83	32.07	60.55	51.79	39.50	32.63
K-Means_IC	66.30	42.03	63.84	42.77	74.96	70.27	40.44	32.16
STC^2-LPI	-	-	76.98	62.56	51.14	49.10	43.37	38.02
Self-Train	-	-	72.69	56.74	59.38	52.81	40.06	34.46
SCCL	83.10	61.96	79.90	63.78	70.83	69.21	42.49	39.16
RSTC	84.24	62.45	80.10	69.74	<u>83.30</u>	74.11	48.40	40.12
GMVAE	82.62	55.76	80.11	58.96	82.90	71.44	48.17	40.57
$RSTC_{\text{BERT-TFIDF-Linear}}$	84.45	60.86	<u>83.21</u>	<u>71.17</u>	78.79	76.14	<u>50.17</u>	<u>45.18</u>
$RSTC_{\text{BERT-TFIDF-Concat-1}}$	85.79	<u>63.26</u>	80.90	69.99	82.41	<u>78.45</u>	49.34	45.00
$RSTC_{\text{BERT-TFIDF-Concat-2}}$	<u>85.80</u>	63.11	82.54	70.74	78.55	73.95	49.24	43.15
COTC	**87.56**	**67.09**	**90.32**	**77.09**	**87.78**	**79.19**	**53.20**	**46.09**

Method	GoogleNews-TS		GoogleNews-T		GoogleNews-S		Tweet	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
TFIDF-K-Means	69.00	87.78	58.36	79.14	62.30	83.00	54.34	78.47
BERT-K-Means	65.71	86.60	55.53	78.38	56.62	80.50	53.44	78.99
K-Means_IC	79.81	92.91	68.88	83.55	74.48	88.53	66.54	84.84
SCCL	82.51	93.01	69.01	85.10	73.44	87.98	73.10	86.66
RSTC	83.27	93.15	72.27	87.39	79.32	89.40	75.20	87.35
GMVAE	83.37	93.48	<u>79.98</u>	90.25	80.65	90.04	73.23	88.86
$RSTC_{\text{BERT-TFIDF-Linear}}$	83.72	93.26	74.29	88.67	81.57	91.17	78.20	89.42
$RSTC_{\text{BERT-TFIDF-Concat-1}}$	83.74	<u>93.79</u>	79.31	<u>91.06</u>	<u>82.91</u>	<u>91.55</u>	75.61	88.50
$RSTC_{\text{BERT-TFIDF-Concat-2}}$	<u>84.03</u>	93.55	74.46	87.70	81.23	90.60	<u>83.62</u>	<u>90.30</u>
COTC	**90.50**	**96.33**	**83.53**	**92.07**	**86.10**	**93.49**	**91.33**	**95.09**

as $RSTC_{\text{BERT-TFIDF-Linear}}$, $RSTC_{\text{BERT-TFIDF-Concat-1}}$ and $RSTC_{\text{BERT-TFIDF-Concat-2}}$. That is to say, by introducing TFIDF features to complement BERT features via simple linear combination or direct concatenation, the performance can be improved in some datasets. However, these methods cannot always fully exploit TFIDF features, as some are even behind RSTC in some cases. This is possibly because simple fusion do not consider the characteristics of TFIDF features when modeling them with the same techiques as BERT features.

Instead, with proper techniques to model BERT and TFIDF respectively and co-train them to learn from each other, our method leverages the collective strengths of deep semantics from BERT and keyword signals from TFIDF. Compared with other methods using only one type of imperfect text features or intuitively combining different features, COTC obtains the best performance, demonstrating the effectiveness of leveraging complementary strengths of BERT and TFIDF features.

4.2.2 Ablation Study

To study the effects of several components in our method, we compare the ACC results of the basic variants for BERT features (*i.e.* results from $\boldsymbol{p}_i^b$) in Table 2. (i) Basis ($\mathcal{M}$) is a basic method performing contrastive learning and pseudo-labelling on BERT features with consistency constraint $\mathcal{L}_{Consist}$. (ii) w/ $\boldsymbol{h}_i^t$ ($\mathcal{M}_{Graph}$) utilizes the TFIDF representations $\{\boldsymbol{h}_i^t\}_{i=1}^N$ to construct the similarity graph $\mathcal{G}^t$ for the learning of the BERT module, *i.e.* using them as neighborhood augmentations to support contrastive learning and pseudo-labelling. (iii) w/ $\boldsymbol{p}_i^t$ ($\mathcal{M}_{Align}$) further utilizes the cluster probabilities $\{\boldsymbol{p}_i^t\}_{i=1}^N$ from TFIDF features and requires the cluster-level alignment between $\boldsymbol{p}_i^b$ and $\boldsymbol{p}_i^t$ with $\mathcal{L}_{Align}$. This model is trained by minimizing $\mathcal{L}_{Joint}$ with the KL-divergence. (iv) COTC ($\mathcal{M}_{Joint}$) is our final method, which connects the BERT and TFIDF modules tightly

Table 2: The ACC results of the basic variants for BERT features. vs Last means the average improvement comparing the current row with the last one.

Variant		AN	SS	SO	Bio	GN-TS	GN-T	GN-S	Tw	vs Last
Basis	($\mathcal{M}$)	85.55	80.78	83.23	50.97	83.25	74.79	79.98	83.32	-
w/ $\boldsymbol{h}_i^t$	($\mathcal{M}_{Graph}$)	86.06	88.30	86.35	52.16	87.08	81.76	82.13	87.27	**+3.66**
w/ $\boldsymbol{p}_i^t$	($\mathcal{M}_{Align}$)	86.56	89.03	87.22	52.55	89.85	82.73	85.23	90.64	**+1.59**
COTC	($\mathcal{M}_{Joint}$)	**87.56**	**90.32**	**87.78**	**53.20**	**90.50**	**83.53**	**86.10**	**91.33**	**+0.81**

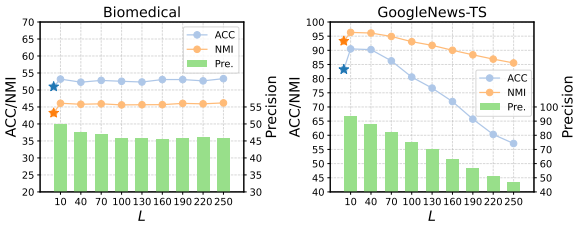

Figure 3: The sensitivity of the number of neighbors L. Pre. means precision, which is the ratio of the neighbors in the same class as the anchor.

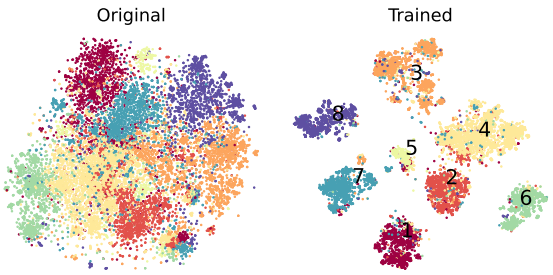

Figure 4: The visualization on SearchSnippets.

Table 3: Different keywords revealed by the cluster centers in the TFIDF module on SearchSnippets.

clusters	keywords	topics
#1	business, market, services, financial, finance	Business
#2	computer, software, programming, linux, web	Computers
#3	movie, music, com, movies, film	Culture-Arts-Entertainment
#4	edu, research, science, university, theory	Education-Science
#5	electrical, car, motor, engine, products	Engineering
#6	health, medical, information, disease, gov	Health
#7	political, party, democracy, government, democratic	Politics-Society
#8	sports, football, news, games, com	Sports

and trains the two modules with the final joint training loss $\mathcal{L}'_{Joint}$. Viewing the results from the top down, the improvement is obvious. Comparing $\mathcal{M}_{Graph}$ with $\mathcal{M}$, the performance gap validates the value of learning the similarity structure exhibited by the TFIDF representations. Furthermore, $\mathcal{M}_{Align}$ performs better than $\mathcal{M}_{Graph}$, which may result from the cluster-level alignment between $\boldsymbol{p}_i^b$ and $\boldsymbol{p}_i^t$. Our method (COTC) $\mathcal{M}_{Joint}$ outperforms $\mathcal{M}_{Align}$, demonstrating the superiority of the unified joint training objective $\mathcal{L}'_{Joint}$ that enables the BERT and TFIDF modules to be connected tightly. More relevant results are described in Appendix D.

4.2.3 Hyperparameter Sensitivity

To investigate how the number of neighbors L affects the performance, we test it with different values. Figure 3 shows that a proper number of neighbors can benefit BERT features in the clustering performance compared with the case $L = 0$ flagged as stars in the figure. However, for some datasets, *e.g.* GoogleNews-TS, with the increase of L, the precision decreases, which introduces much noise and thus hurts the performance. To balance the performance, we set L to 10 in all datasets. Sensitivities of weighting parameter λ_1 and temperature parameter τ are shown in Appendix D.

4.2.4 Case Study

To understand what keywords are learned, we map the cluster centers in the TFIDF module to the vocabulary space with the embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{W}\times 128|}$ in the decoder of VAE (A.2), where $\mathcal{W}$ is the vocabulary and 128 is the dimensionality of the TFIDF representations. A center $\boldsymbol{\mu}_i \in \mathbb{R}^{128}$ can be mapped to $\boldsymbol{E}\boldsymbol{\mu}_i \in \mathbb{R}^{|\mathcal{W}|}$, with each element meaning the relevance between the center and the word. We list the top-5 relevant keywords of each cluster on SearchSnippets in Table 3. The corresponding visualization is also shown in Figure 4. Due to the space limitation, we show the case study on StackOverflow in Appendix D.

4.2.5 Investigation of Other Features and Base Models

We investigate whether TFIDF features can be replaced by other features reflecting keywords. We choose BoW and Word2Vec as alternatives. The experimental results are shown in Table 4. It can

Table 4: The ACC results using BoW or Word2Vec instead of TFIDF features. $\text{COTC}_{\text{BERT-TFIDF}}$ is our final method.

dataset	AN	SO	Bio	GN-TS	Tw
$\text{RSTC}_{\text{BERT}}$	84.24	83.30	48.40	83.27	75.20
$\text{COTC}_{\text{BERT-W2V}}$	34.24	28.06	27.50	74.06	14.36
$\text{COTC}_{\text{BERT-BoW}}$	87.41	84.91	52.68	89.15	88.43
$\text{COTC}_{\text{BERT-TFIDF}}$	**87.56**	**87.78**	**53.20**	**90.50**	**91.33**

Table 5: The ACC results using different base models instead of the default sentence-distilbert.

dataset	AN	GN-TS	Tw
$\text{RSTC}_{\text{XLNet-base-uncased}}$	71.75	34.47	10.07
$\text{COTC}_{\text{XLNet-base-uncased}}$	**84.60**	**80.21**	**71.97**
$\text{RSTC}_{\text{BERT-base-uncased}}$	82.23	77.45	73.87
$\text{COTC}_{\text{BERT-base-uncased}}$	**87.83**	**89.10**	**89.81**
$\text{RSTC}_{\text{RoBERTa-base}}$	85.76	75.63	71.08
$\text{COTC}_{\text{RoBERTa-base}}$	**87.44**	**88.32**	**90.45**

be seen that $\text{COTC}_{\text{BERT-BoW}}$ performs on par with $\text{COTC}_{\text{BERT-TFIDF}}$, showing that BoW features can also work very well in our proposed co-training framework. Differently, $\text{COTC}_{\text{BERT-W2V}}$ performs badly in our method. We conjecture that Word2Vec is more like BERT features than frequency-based features, and it strongly relies on its pre-training corpus.

Moreover, we also investigate different base models rather than merely the default sentence-distilbert in our method. We test XLNet-base-cased, BERT-base-uncased and RoBERTa-base. The experimental results are shown in Table 5. By comparing every two lines, it is obvious that the performance of RSTC is strongly dependent on the initial quality of the backbone representations, especially on Tweet with XLNet-base-uncased. In contrast, with the help of TFIDF features, our method is relatively stable and still works well when using different base models.

4.2.6 Clustering with Noisy Data

To observe the stability of COTC, we conduct some experiments with noisy data as an exploration. We use the StackOverflow dataset as the base, and add random samples from the Biomedical dataset as the noise. We then perform clustering with our method on the contaminated data. The experimental results are listed in Table 6. As the percentage of the noisy samples increases, the performance gradually declines, which is predictable and reasonable. However, it can be observed that our method still maintains a certain level stability even when clustering data contaminated by noise from a completely different domain, which validates the robustness of our method.

Table 6: The clustering results when our method performs clustering under noisy data condition, *i.e.*, StackOverflow contaminated by Biomedical.

percentage of noisy samples	0%	1%	2%	3%	4%
ACC	87.78	87.13	84.32	83.59	81.72
NMI	79.19	78.54	77.29	77.20	76.60

5 Conclusion

In this paper, we rediscover the value of seemingly outdated TFIDF features and use them to complement BERT features for short text clustering. To leverage BERT and TFIDF, we develop two modules to model them and propose a co-training framework to enable them to learn from each other, where their deep representations and cluster assignments are fully communicated and aligned. We show that the framework can be achieved by a unified joint training objective, which empirically benefits the clustering performance. Extensive experiments show the superiority of our method over previous methods.

Limitations

As previous works, our method requires that the number of clusters is known. Furthermore, due to the huge difference between BERT and TFIDF features (the former are low-dimensional and dense while the latter are high-dimensional and sparse), the neural networks of the related modules require different learning rates, which requires some efforts to tune them to make the training work. Also, compared with other works using only BERT features, the introduction of TFIDF features requires extra time and space to train the TFIDF module. In the future, we are going to explore a more compact and efficient way to learn the collective strengths of different text features without incurring much extra cost.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 62276280, 62276279, U1811264), Guangzhou Science and Technology Planning Project (No. 2024A04J9967), Guangdong Basic and Applied Basic Research Foundation (2024B1515020032).

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. 2007. Clustering short texts using wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 787–788.

Christos Bouras and Vassilis Tsogkas. 2017. Improving news articles recommendations via user clustering. *International Journal of Machine Learning and Cybernetics*, 8(1):223–237.

Jiayang Chen and Qinliang Su. 2023. Exploiting multiple features for hash codes learning with semantic-alignment-promoting variational auto-encoder. In *Natural Language Processing and Chinese Computing*, pages 563–575.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Erik Englesson and Hossein Azizpour. 2021. Consistency regularization can improve robustness to label noise.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved deep embedded clustering with local structure preservation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1753–1759.

Maryam Habibi and Andrei Popescu-Belis. 2015. Keyword extraction and clustering for document recommendation in conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(4):746–759.

Amir Hadifar, Lucas Sterckx, Thomas Demeester, and Chris Develder. 2019. A self-training approach for short text clustering. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 194–199.

Coleman Haley. 2020. This is a BERT. now there are several of them. can they generalize to novel words? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 333–341.

Xiaofei He and Partha Niyogi. 2003. Locality preserving projections. In *Advances in Neural Information Processing Systems*, volume 16.

Xiaohua Hu, Xiaodan Zhang, Caimei Lu, E. K. Park, and Xiaohua Zhou. 2009. Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 389–396.

Shaohan Huang, Furu Wei, Lei Cui, Xingxing Zhang, and Ming Zhou. 2020. Unsupervised fine-tuning for text clustering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5530–5534.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1965–1972.

Hwi-Gang Kim, Seongjoo Lee, and Sunghyon Kyeong. 2013. Discovering hot topics using twitter streaming data: Social topic detection and geographic clustering. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, page 1215–1220.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel,

Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web*, page 91–100.

Md Rashadul Hasan Rakib, Norbert Zeh, Magdalena Jankowska, and Evangelos Milios. 2020. Enhancement of short text clustering by iterative classification. In *Natural Language Processing and Information Systems*, pages 105–117.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Timo Schick and Hinrich Schütze. 2020. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8766–8774.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Liang Wu and Huan Liu. 2018. Tracing fake-news footprints: Characterizing social media messages by how they propagate. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, page 637–645.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 478–487.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, Jun Zhao, and Bo Xu. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31.

Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3861–3870.

Jianhua Yin and Jianyong Wang. 2014. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 233–242.

Qing Yin, Zhihua Wang, Yunya Song, Yida Xu, Shuai Niu, Liang Bai, Yike Guo, and Xian Yang. 2022. Improving deep embedded clustering via learning cluster-level representations. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2226–2236.

Asano YM., Rupprecht C., and Vedaldi A. 2020. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*.

Dejiao Zhang, Feng Nan, Xiaokai Wei, Shang-Wen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew O. Arnold, and Bing Xiang. 2021. Supporting clustering with contrastive learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5419–5430.

Qinpei Zhao, Mohammad Rezaei, Hao Chen, and Pasi Fränti. 2012. Keyword clustering for automatic categorization. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2845–2848.

Xiaolin Zheng, Mengling Hu, Weiming Liu, Chaochao Chen, and Xinting Liao. 2023. Robust representation learning with reliable pseudo-labels generation via self-adaptive optimal transport for short text clustering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10493–10507.

A Derivation Details

A.1 Optimal Transport

Following the recent works (YM. et al., 2020; Zheng et al., 2023), we infer the pseudo-labels $\{\boldsymbol{q}_i\}_{i=1}^N$ by solving the optimal transport problem over the predicted probabilities $\{\boldsymbol{p}_i^b\}_{i=1}^N$. Denoting $\boldsymbol{P} = [\boldsymbol{p}_1^b, .., \boldsymbol{p}_N^b]^T \in [0,1]^{N\times K}$, we use $\boldsymbol{C} = -\log \boldsymbol{P}$ as the cost matrix and optimize

$$\begin{aligned} &\min_{\boldsymbol{r},\boldsymbol{b}} \langle \boldsymbol{r}, \boldsymbol{C}\rangle - \epsilon_1 H(\boldsymbol{r}) + \epsilon_2 U(\boldsymbol{b}), \\ &\text{s.t. } \boldsymbol{r}\mathbf{1} = \boldsymbol{a}, \boldsymbol{r}^T\mathbf{1} = \boldsymbol{b}, \boldsymbol{b}^T\mathbf{1} = 1, \boldsymbol{r} \geq 0, \boldsymbol{b} \geq 0, \end{aligned} \tag{23}$$

where $\boldsymbol{r} \in [0,1]^{N\times K}$ is the transport matrix between samples and classes, $\boldsymbol{a} = \frac{1}{N}\mathbf{1}$ is the uniform marginal distribution over samples, $\boldsymbol{b}$ is the adaptive marginal distribution over classes, $H(\boldsymbol{r}) = -\sum_{i=1}^N \sum_{j=1}^K r_{ij}(\log r_{ij} - 1)$ is the entropy regularization, $U(\boldsymbol{b}) = -\sum_{j=1}^K (\log b_j + \log(1-b_j))$

is the penalty function and ϵ_1, ϵ_2 are two weighting parameters. To be more specific, the element r_{ij} in $\boldsymbol{r}$ means the probability of transporting sample i to class j, $\boldsymbol{b}$ is set to be adaptive over classes due to the class imbalance problem and $U(\boldsymbol{b})$ encourages $\boldsymbol{b}$ to be uniform to avoid collapsing. With Lagrange Multiplier Method, it is equivalent to optimize

$$\begin{aligned}\min_{\boldsymbol{r},\boldsymbol{b}} L = &\langle \boldsymbol{r}, \boldsymbol{C}\rangle - \epsilon_1 H(\boldsymbol{r}) + \epsilon_2 U(\boldsymbol{b}) \\ &-\boldsymbol{f}^T(\boldsymbol{r}\mathbf{1}-\boldsymbol{a})-\boldsymbol{g}^T(\boldsymbol{r}^T\mathbf{1}-\boldsymbol{b})-h(\boldsymbol{b}^T\mathbf{1}-1),\end{aligned} \tag{24}$$

where $\boldsymbol{f}$, $\boldsymbol{g}$ and h are multipliers.

For variables $\boldsymbol{r}$, $\boldsymbol{f}$ and $\boldsymbol{g}$, by fixing $\boldsymbol{b}$, h and computing $\frac{\partial L}{\partial r_{ij}} = 0$, we have

$$r_{ij} = \exp\left(\frac{f_i + g_j - C_{ij}}{\epsilon_1}\right). \tag{25}$$

Combined with the constraints $\boldsymbol{r}\mathbf{1} = \boldsymbol{a}$ and $\boldsymbol{r}^T\mathbf{1} = \boldsymbol{b}$, we have

$$\exp\left(\frac{f_i}{\epsilon_1}\right) = \frac{a_i}{\sum_{j=1}^K \exp\left(\frac{g_j - C_{ij}}{\epsilon_1}\right)}, \tag{26}$$

$$\exp\left(\frac{g_j}{\epsilon_1}\right) = \frac{b_j}{\sum_{i=1}^N \exp\left(\frac{f_i - C_{ij}}{\epsilon_1}\right)}. \tag{27}$$

For variables $\boldsymbol{b}$ and h, by fixing $\boldsymbol{r}$, $\boldsymbol{f}$, $\boldsymbol{g}$ and computing $\frac{\partial L}{\partial b_j} = 0$, we have

$$(g_j - h)b_j^2 + (-g_j + h - 2\epsilon_2)b_j + \epsilon_2 = 0, \tag{28}$$

$$\Delta_j = (g_j - h)^2 + 4\epsilon_2^2 > 0, \tag{29}$$

$$b_j = \frac{g_j - h + 2\epsilon_2 - \sqrt{\Delta_j}}{2(g_j - h)}, \tag{30}$$

where Δ_j is the discriminant of the equation (28) and $\tilde{b}_j = \frac{g_j - h + 2\epsilon_2 + \sqrt{\Delta_j}}{2(g_j - h)}$ is discarded because $\tilde{b}_j > 1$ when $g_j - h > 0$ and $\tilde{b}_j < 0$ when $g_j - h < 0$, which are beyond the domain of the definition of b_j. Combined with the constraint $\boldsymbol{b}^T\mathbf{1} = 1$, we need to solve $f(h) = 0$ with

$$f(h) = \boldsymbol{b}^T\mathbf{1} - 1 = \sum_{j=1}^K \frac{g_j - h + 2\epsilon_2 - \sqrt{\Delta_j}}{2(g_j - h)} - 1. \tag{31}$$

We use Newton's Method to compute the root h of $f(h) = 0$, *i.e.*,

$$h \leftarrow h - \frac{f(h)}{f'(h)}, \tag{32}$$

with

$$f'(h) = \sum_{j=1}^K \left(\frac{2\epsilon_2 - \sqrt{\Delta_j}}{2(g_j - h)^2} + \frac{1}{2\sqrt{\Delta_j}}\right). \tag{33}$$

In practice, we set $\boldsymbol{u} = \exp(\boldsymbol{f}/\epsilon_1)$, $\boldsymbol{v} = \exp(\boldsymbol{g}/\epsilon_1)$ and $\boldsymbol{W} = \exp(-\boldsymbol{C}/\epsilon_1)$. Thus, initializing $\boldsymbol{u}^{(0)} = \mathbf{1}$, $\boldsymbol{v}^{(0)} = \mathbf{1}$ and $\boldsymbol{b}^{(0)} = \frac{1}{K}\mathbf{1}$, the update process is

$$\boldsymbol{u}^{(t+1)} \leftarrow \frac{\boldsymbol{a}}{\boldsymbol{W}\boldsymbol{v}^{(t)}}, \tag{34}$$

$$\boldsymbol{v}^{(t+1)} \leftarrow \frac{\boldsymbol{b}^{(t)}}{\boldsymbol{W}^T\boldsymbol{u}^{(t)}}, \tag{35}$$

$$\boldsymbol{b}^{(t+1)} \leftarrow \frac{\boldsymbol{g} - h + 2\epsilon_2 - \sqrt{\boldsymbol{\Delta}}}{2(\boldsymbol{g} - h)}, \tag{36}$$

where $\boldsymbol{\Delta} = [\Delta_1, .., \Delta_K]$ (see (29)) and h is computed by the above Newton's Method (see (32)). After several optimization iterations, we can obtain the transport matrix

$$\boldsymbol{r} = \boldsymbol{u}^T\boldsymbol{W}\boldsymbol{v}. \tag{37}$$

Finally, the pseudo-labels $\{\boldsymbol{q}_i\}_{i=1}^N$ are obtained by

$$q_{ij} = 1 \text{ if } \arg\max_{j'} r_{ij'} = j \text{ else } 0. \tag{38}$$

A.2 Variational Autoencoder

We build the generative model for the TFIDF features $\{\boldsymbol{t}_i\}_{i=1}^N$ and the similarity graph $\mathcal{G}^b$ as

$$\begin{aligned}&p(\{\boldsymbol{t}_i\}_{i=1}^N, \mathcal{G}^b, \{\boldsymbol{h}_i^t\}_{i=1}^N, \{c_i\}_{i=1}^N) \\ =&\prod_{i=1}^N p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t|c_i)p(c_i).\end{aligned} \tag{39}$$

The negative evidence lower bound (ELBO) $\mathcal{L}_{ELBO} = \sum_{i=1}^N \ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)p(c_i|\boldsymbol{h}_i^t))\right)$ is minimized to train the model, where

$$\begin{aligned}&\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)p(c_i|\boldsymbol{h}_i^t)\right) \\ =&-\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\left[\log\frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t)}{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\right].\end{aligned} \tag{40}$$

We also minimize the alignment loss $\mathcal{L}_{Align} = \frac{1}{N}D_{KL}(\boldsymbol{p}_i^b[c_i]||\boldsymbol{p}_i^t[c_i])$ to encourage the consistency between the predictions from the BERT and TFIDF features, where

$$\boldsymbol{p}_i^b[c_i] = \delta(g(\boldsymbol{b}_i)), \tag{41}$$

$$\boldsymbol{p}_i^t[c_i] = \mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}[p(c_i|\boldsymbol{h}_i^t)]. \tag{42}$$

Here $\boldsymbol{p}_i^b[c_i]$, $\boldsymbol{p}_i^t[c_i]$ denote the predictions from the BERT and TFIDF features respectively, $g(\cdot)$ means the clustering head and $\delta(\cdot)$ means the softmax function.

To see the derivation of (20), we first compute the term in the alignment loss as

$$\begin{aligned}
&D_{KL}(\boldsymbol{p}_i^b[c_i]||\boldsymbol{p}_i^t[c_i]) \\
=&\sum_{c'} \boldsymbol{p}_i^b[c'] \log \frac{\boldsymbol{p}_i^b[c']}{\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}[p(c'|\boldsymbol{h}_i^t)]} \\
\leq&\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\left[\sum_{c'} \boldsymbol{p}_i^b[c'] \log \frac{\boldsymbol{p}_i^b[c']}{p(c'|\boldsymbol{h}_i^t)}\right] \\
=&\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}\left[\log \frac{\boldsymbol{p}_i^b[c_i]}{p(c_i|\boldsymbol{h}_i^t)}\right],
\end{aligned} \tag{43}$$

where Jensen's inequality is used. Combined with the term in the negative ELBO, we then have

$$\begin{aligned}
&\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)p(c_i|\boldsymbol{h}_i^t)\right) + D_{KL}(\boldsymbol{p}_i^b[c_i]||\boldsymbol{p}_i^t[c_i]) \\
\leq&-\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\left[\log \frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t)}{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}\right] \\
&+\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}\left[\log \frac{\boldsymbol{p}_i^b[c_i]}{p(c_i|\boldsymbol{h}_i^t)}\right] \\
=&-\mathbb{E}_q\left[\log \frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t)p(c_i|\boldsymbol{h}_i^t)}{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}\right] \\
=&-\mathbb{E}_q\left[\log \frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t|c_i)p(c_i)}{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}\right] \\
\triangleq&\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]\right),
\end{aligned} \tag{44}$$

where the expectation $\mathbb{E}_q[\cdot]$ is taken w.r.t the posterior $q(\boldsymbol{h}_i^t, c_i|\boldsymbol{t}_i) = q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)q(c_i|\boldsymbol{t}_i)$ with $q(c_i|\boldsymbol{t}_i) = \boldsymbol{p}_i^b[c_i]$. Thus, we have the inequality

$$\begin{aligned}
\mathcal{L}'_{ELBO} =& \sum_{i=1}^N \ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]\right) \\
\geq& \mathcal{L}_{ELBO} + \mathcal{L}_{Align}.
\end{aligned} \tag{45}$$

Decoder $p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)$ We have described most components of the VAE in the main text. Here, we clarify the definition of the decoder $p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)$. We regard $\boldsymbol{t}_i$ as a set of words $\{\boldsymbol{w}_j \in \boldsymbol{t}_i\}$, with $\boldsymbol{w}_j$ a one-hot representation in vocabulary $\mathcal{W}$. To be more specific, if a text contains the 1-st, 3-rd and 5-th word in $\mathcal{W}$, it is represented as $\{\boldsymbol{w}_1, \boldsymbol{w}_3, \boldsymbol{w}_5\}$. Thus, denoting $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{W}|\times 128}$ as the embedding matrix, *i.e.* the decoder network (128 is the dimension of $\boldsymbol{h}_i^t$), we define

$$\begin{aligned}
&p(\boldsymbol{t}_i|\boldsymbol{h}_i^t) \\
=&\prod_{\boldsymbol{w}_j\in\boldsymbol{t}_i} p(\boldsymbol{w}_j|\boldsymbol{h}_i^t) = \prod_{\boldsymbol{w}_j\in\boldsymbol{t}_i} \frac{\exp(\boldsymbol{h}_i^{tT}\boldsymbol{E}\boldsymbol{w}_j)}{\sum_{k=1}^{|\mathcal{W}|}\exp(\boldsymbol{h}_i^{tT}\boldsymbol{E}\boldsymbol{w}_k)}.
\end{aligned} \tag{46}$$

Approximation $\mathcal{L}'_{ELBO}$ We have obtained the new negative ELBO $\mathcal{L}'_{ELBO}$ for the joint training loss. Here, we clarify how to compute it. We factorize the term in the negative ELBO as

$$\begin{aligned}
&\ell_i^{elbo}\left(q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]\right) \\
=&-\mathbb{E}_q\left[\log \frac{p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)p(\boldsymbol{h}_i^t|c_i)p(c_i)}{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}\right] \\
=&-\mathbb{E}_q[\log p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)] - \mathbb{E}_q[\log p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)] \\
&+\mathbb{E}_q[\log q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)] - \mathbb{E}_q\left[\log \frac{p(c_i)}{\boldsymbol{p}_i^b[c_i]}\right] - \mathbb{E}_q[\log p(\boldsymbol{h}_i^t|c_i)].
\end{aligned} \tag{47}$$

The five factorized subterms can be computed separately. We use the Gaussian re-parameterization trick to approximate $-\mathbb{E}_q[\log p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)]$ as

$$\begin{aligned}
&-\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}[\log p(\boldsymbol{t}_i|\boldsymbol{h}_i^t)] \\
\approx&-\sum_{\boldsymbol{w}_j\in\boldsymbol{t}_i} \log \frac{\exp(\tilde{\boldsymbol{h}}_i^{tT}\boldsymbol{E}\boldsymbol{w}_j)}{\sum_{k=1}^{|\mathcal{W}|}\exp(\tilde{\boldsymbol{h}}_i^{tT}\boldsymbol{E}\boldsymbol{w}_k)},
\end{aligned} \tag{48}$$

where $\tilde{\boldsymbol{h}}_i^t = \boldsymbol{\mu}(\boldsymbol{t}_i) + \boldsymbol{\epsilon}^T\boldsymbol{\sigma}(\boldsymbol{t}_i)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{1})$. As for $-\mathbb{E}_q[\log p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)]$, we approximate it as

$$\begin{aligned}
&-\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}[\log p(\mathcal{G}_i^b|\{\boldsymbol{h}_i^t\}_{i=1}^N)] \\
\approx&-\sum_{j\in\mathcal{N}_i^b} \log \frac{\Delta(\tilde{\boldsymbol{h}}_i^t, \tilde{\boldsymbol{h}}_j^t)}{\sum_{k\neq i}\Delta(\tilde{\boldsymbol{h}}_i^t, \tilde{\boldsymbol{h}}_k^t)}.
\end{aligned} \tag{49}$$

Differently, $\mathbb{E}_q[\log q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)]$ can be analytically computed as

$$\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)}[\log q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)] \\
=&-\frac{1}{2}\sum_{j=1}^{128}(\log 2\pi + 1 + \log \boldsymbol{\sigma}^2(\boldsymbol{t}_i)|_j),
\end{aligned} \tag{50}$$

where 128 is the dimension of $\boldsymbol{h}_i^t$ and $\boldsymbol{\sigma}^2(\boldsymbol{t}_i)|_j$ is the j-th element of $\boldsymbol{\sigma}^2(\boldsymbol{t}_i)$. With the Gumbel-Softmax trick, $c_i \sim \boldsymbol{p}_i^b[c_i]$ can be achieved by first sampling $g_{ik} \sim \text{Gumbel}(g_{ik}; 0, 1)$ and then computing $\tilde{c} \sim \arg\max_k(g_{ik} + \log \boldsymbol{p}_i^b[k])$. By using softmax to approximate argmax, we obtain a probability vector $\tilde{\boldsymbol{c}}_i$ and the k-th element is

$$\tilde{c}_{ik} = \frac{\exp((g_{ik} + \log \boldsymbol{p}_i^b[k])/\tau)}{\sum_{j=1}^K \exp((g_{ij} + \log \boldsymbol{p}_i^b[j])/\tau)}, \tag{51}$$

where τ is the temperature parameter controlling the sharpness of $\boldsymbol{c}_i$. Thus, we can approximate $-\mathbb{E}_q\left[\log \frac{p(c_i)}{\boldsymbol{p}_i^b[c_i]}\right]$ and $-\mathbb{E}_q[\log p(\boldsymbol{h}_i^t|c_i)]$ as

$$-\mathbb{E}_{\boldsymbol{p}_i^b[c_i]}\left[\log \frac{p(c_i)}{\boldsymbol{p}_i^b[c_i]}\right] \approx -\sum_{k=1}^{K} \tilde{c}_{ik} \log \frac{p(k)}{\boldsymbol{p}_i^b[k]}, \quad (52)$$

$$-\mathbb{E}_{q(\boldsymbol{h}_i^t|\boldsymbol{t}_i)\boldsymbol{p}_i^b[c_i]}[\log p(\boldsymbol{h}_i^t|c_i)]$$

$$\approx \frac{1}{2}\sum_{k=1}^{K} \tilde{c}_{ik}\left(\sum_{j=1}^{128} \log 2\pi + \log \boldsymbol{\sigma}_k^2|_j \right. \quad (53)$$

$$\left. + \frac{(\boldsymbol{\mu}(\boldsymbol{t}_i)|_j - \boldsymbol{\mu}_k|_j)^2 + \boldsymbol{\sigma}^2(\boldsymbol{t}_i)|_j}{\boldsymbol{\sigma}_k^2|_j}\right).$$

B Dataset Details

The statistics of the datasets are summarized in Table 7 and the details of the datasets are described as follows.

- **AgNews**[1] is a subset of news articles collected from more than 2000 news sources[2]. 8000 news titles from 4 topic categories are randomly selected by (Rakib et al., 2020).
- **SearchSnippets**[3] is a subset of snippets extracted from the web search results. 12340 snippets from 8 domains are carefully selected by (Phan et al., 2008) from search transactions using domain-specific phrases.
- **StackOverflow**[4] is a subset of data published in Kaggle[5]. 20000 question titles from 20 tags are randomly selected by (Xu et al., 2017).
- **Biomedical**[4] is a subset of data published in BioASQ[6]. 20000 paper titles from 20 categories are randomly selected by (Xu et al., 2017).
- **GoogleNews**[7] contains titles and snippets of 11109 news articles corresponding to 152 events, constructed by (Yin and Wang, 2014). **GoogleNews-TS**, **GoogleNews-T** and **GoogleNews-S** are three subsets with both titles and snippets, titles only and snippets only respectively.
- **Tweet**[7] contains 2472 tweets from 89 queries, constructed by (Yin and Wang, 2014).

Table 7: The statistics of the datasets. N: the number of texts; Len: the average length of texts; K: the number of classes; L/S: the size ratio of the largest class versus the smallest one.

Dataset	N	Len	K	L/S
AgNews	8000	23	4	1
SearchSnippets	12340	18	8	7
StackOverflow	20000	9	20	1
Biomedical	20000	13	20	1
GoogleNews-TS	11109	28	152	143
GoogleNews-T	11109	6	152	143
GoogleNews-S	11109	22	152	143
Tweet	2472	9	89	249

C Training Details

We implement our method with *PyTorch* (Paszke et al., 2019), and obtain 2048-dimensional TFIDF features using *TfidfVectorizer* from scikit-learn (Pedregosa et al., 2011) and 768-dimensional BERT features using *distilbert-base-nli-stsb-mean-tokens* (Reimers and Gurevych, 2019) from HuggingFace (Wolf et al., 2020). We use *ContextualAugmenter* (Kobayashi, 2018; Ma, 2019) to generate data augmentations with 10% word substitution and solve the optimal transport problem (YM. et al., 2020; Zheng et al., 2023) to generate pseudo-labels. In our experiments, we use Adam optimizer and set the batch size to 128. We set the learning rate to 5e-6 for BERT backbone, 5e-4 for BERT projection and clustering heads, 1e-5 for TFIDF Gaussian mixture parameters and 1e-3 for TFIDF encoder decoder parameters. For all datasets, the number of neighbors for each sample L is fixed to 10, the temperature in both contrastive learning and Gumbel-Softmax trick τ is fixed to 0.5, and weighting parameter λ_1 is fixed to 0.1.

The data flows and network architectures are described in Table 10 and 11. In practice, we first pre-train VAE with static graph constructed by raw TFIDF features for 50 epochs. We then perform K-Means (for AgNews, SearchSnippets, StackOverflow and Biomedical) or hierarchical agglomerative clustering (for GoogleNews-TS, GoogleNews-T,

[1] https://github.com/rashadulrakib/short-text-clustering-enhancement
[2] http://groups.di.unipi.it/~gulli/AG-corpus-of-news-articles.html
[3] https://jwebpro.sourceforge.net/data-web-snippets.tar.gz
[4] https://github.com/jacoxu/STC2
[5] https://www.kaggle.com/competitions/predict-closed-questions-on-stack-overflow
[6] http://participants-area.bioasq.org
[7] https://github.com/jackyin12/GSDMM

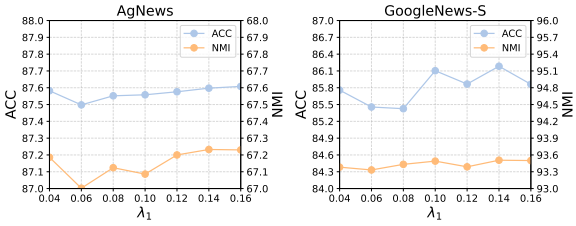

Figure 5: The sensitivity of the weighting parameter λ_1.

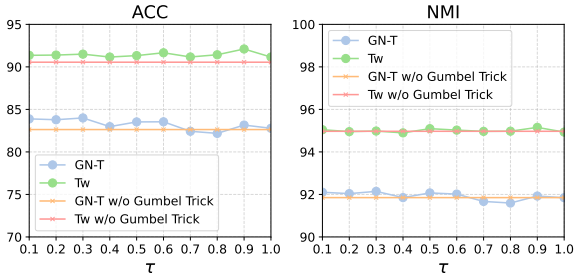

Figure 6: The sensitivity of the temperature parameter of Gumbel trick τ.

GoogleNews-S and Tweet) to obtain the clustering assignments as the initialized labels to pre-train the clustering head $g(\cdot)$ for 100 iterations. After pre-training, the two modules can be trained with our final joint training loss $\mathcal{L}'_{Joint}$. Following the recent works (YM. et al., 2020; Zheng et al., 2023), we spread out the update process of the nearest neighbors and pseudo-labels throughout the whole training process in a logarithmic distribution.

D Result Details

Metric Definition The clustering accuracy, *i.e.* ACC, is defined as

$$ACC = \frac{\sum_{i=1}^{N} \mathbb{1}_{y_i = map(\hat{y}_i)}}{N}, \tag{54}$$

where y_i and $\hat{y}_i$ are the ground-truth and predicted labels of the i-th sample respectively, and $map(\cdot)$ is obtained by Hungarian algorithm and maps predicted labels to the corresponding ground-truth labels. The normalized mutual information, *i.e.* NMI, is defined as

$$NMI = \frac{2I(Y;\hat{Y})}{H(Y) + H(\hat{Y})}, \tag{55}$$

where Y and $\hat{Y}$ are the ground-truth and predicted labels respectively, and $I(\cdot;\cdot)$ is mutual information and $H(\cdot)$ is entropy.

Table 8: Different keywords revealed by the cluster centers in the TFIDF module on StackOverflow.

clusters	keywords	topics
#1	excel, vba, cell, macro, data	excel
#2	haskell, type, function, scala, list	haskell
#3	mac, os, osx, application, app	osx
#4	linq, sql, query, using, join	linq
#5	ajax, jquery, javascript, request, php	ajax
#6	visual, studio, 2008, 2005, project	visual-studio
#7	cocoa, using, file, use, text	cocoa
#8	hibernate, mapping, criteria, query, hql	hibernate
#9	sharepoint, web, site, 2007, list	sharepoint
#10	bash, script, command, shell, file	bash
#11	apache, rewrite, mod, htaccess, redirect	apache
#12	wordpress, posts, post, page, blog	wordpress
#13	svn, subversion, repository, files, commit	svn
#14	drupal, node, views, module, content	drupal
#15	qt, widget, window, creator, application	qt
#16	scala, java, class, type, actors	scala
#17	magento, product, products, page, admin	magento
#18	matlab, matrix, plot, array, function	matlab
#19	oracle, sql, table, pl, database	oracle
#20	spring, bean, hibernate, security, using	spring

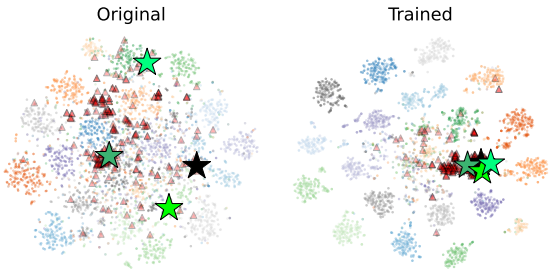

Figure 7: The visualization on StackOverflow.

Ablation Study The remaining results of the ablation study are described in Table 12, 13 and 14. In Table 12, $\mathcal{M}$ is a basic method performing contrastive learning and pseudo-labelling on BERT features with additional consistency constraint $\mathcal{L}_{Align}$; $\mathcal{M}_{Graph}$ introduces the TFIDF representation $\boldsymbol{h}_i^t$ for the learning of the BERT module and $\mathcal{M}_{Align}$ further aligns $\boldsymbol{p}_i^b$ with the cluster probability $\boldsymbol{p}_i^t$. In Table 13 and 14, $\mathcal{M}$ is basic method modeling TFIDF features via VAE with a Gaussian mixture prior, *i.e.* GMVAE; $\mathcal{M}_{Graph}$ introduces the BERT representation $\boldsymbol{h}_i^b$ for the learning of the TFIDF module and $\mathcal{M}_{Align}$ further aligns $\boldsymbol{p}_i^t$ with the cluster probability $\boldsymbol{p}_i^b$. $\mathcal{M}_{Joint}$ is our final method (COTC) which connects the two modules tightly, and the results in Table 12 are from $\boldsymbol{p}_i^b$ and those in Table 13 and 14 are from $\boldsymbol{p}_i^t$. Generally, comparing $\mathcal{M}_{Graph}$ with $\mathcal{M}$, it can be seen that the introduction of the similarity structure from another module greatly improves the clustering performance, validating the importance of learning the complementary strengths from each other. Furthermore,

Table 9: The clustering results of LLM for zero-shot short text clustering on AgNews.

	ACC	NMI
Qwen2-7B-Instruct-zero-shot	75.28	48.27
COTC	87.56	67.09

from $\mathcal{M}_{Graph}$ to $\mathcal{M}_{Align}$, the further alignment at the cluster level can further boost both the BERT and TFIDF modules. Our method (COTC) $\mathcal{M}_{Joint}$ outperforms other variants, demonstrating the benefit from the unified joint training objective $\mathcal{L}'_{Joint}$ that trains the BERT and TFIDF modules jointly and connects the two modules tightly.

Hyperparameter Sensitivity We investigate how the weighting parameter λ_1 and the temperature parameter of the Gumbel-Softmax trick τ affects the clustering performance. Figure 5 shows that it is necessary to search a proper weighting parameter to balance the losses $\mathcal{L}_B$ and $\mathcal{L}'_{ELBO}$ for the BERT and TFIDF modules respectively since they possess different network architectures. However, it can be seen that λ_1 in the range from 0.04 to 0.16 does not incur huge fluctuation. We set λ_1 to 0.1 for all datasets to avoid excessive tuning. Figure 6 shows that utilizing Gumbel-Softmax trick with the temperature parameter τ can generally enhance the clustering performance, which may be due to the exploration nature of the trick. We simply set τ to 0.5 for all datasets.

Case Study There are 20 classes in StackOverflow. Due to the limited space in the main text, we show a similar case study on StackOverflow as in Section 4.2.4. The keywords revealed by the cluster centers in the TFIDF module are listed in Table 8. We also visualize the corresponding result of the example illustrated in Figure 1. As shown in Figure 7, the four star texts are finally clustered together and the decision boundaries within different clusters are more clear than before.

Comparison with LLM Zero-Shot Clustering Witnessing the rapid development of the large language models, we are also interesting in whether LLM can be used for short text clustering. As a naive attempt, the following is a toy experiment that uses Qwen2-7B-Instruct for zero-shot clustering on the AgNews dataset with a designed prompt, where 4 category names (World, Sports, Business and Sci/Tech) are required for helping LLM answer. The prompt is shown below.

> You are very good at judge the category of a text. Below, I will give you a short text, and you should think carefully and answer me its category.
> An example is:
> —
> INPUT TEXT: The Lakers wins the finals.
> OUTPUT CATEGORY: **Sports**
> —
> You should remember that you can choose only from **World**, **Sports**, **Business**, and **Sci/Tech**.
> Now, start:
> —
> INPUT TEXT: {input_text}

The result is listed in Table 9. Although the exact category names are required for clustering, Qwen2-7B-Instruct has obtained a very strong performance given its zero-shot setting. We think this is quite compelling since the size 7B is acceptable to cluster a large dataset. However, currently a specialized model for clustering like our method is still competitive.

Computation Budget The number of parameters in our model is 77M. On average for all datasets, our method takes about 30 minutes to complete the training process with a GeForce RTX 3090 GPU.

Table 10: The data flows and network architectures for BERT features. K is the number of clusters, 768 is the dimension of BERT features and 128 is the dimension of BERT representations.

Data Flow	-	Network Architecture
Raw Text $\boldsymbol{x}$	-	-
BERT Transformation $\boldsymbol{b} = \mathcal{B}(\boldsymbol{x})$	$\mathcal{B}(\cdot)$	BERT Backbone
Projection Head $\boldsymbol{h}^b = f(\boldsymbol{b})$	$f(\cdot)$	Linear(768, 768); ReLU(); Linear(768, 128); Normalize().
Clustering Head $\boldsymbol{p}^b = g(\boldsymbol{b})$	$g(\cdot)$	Dropout(); Linear(768, 768); ReLU(); Dropout(); Linear(768, 768); ReLU(); Linear(768, K); Softmax().

Table 11: The data flows and network architectures for TFIDF features. K is the number of clusters, 2048 is the dimension of TFIDF features and 128 is the dimension of TFIDF representations.

Data Flow	-	Network Architecture
Raw Text $\boldsymbol{x}$	-	-
TFIDF Transformation $\boldsymbol{t} = \mathcal{T}(\boldsymbol{x})$	$\mathcal{T}(\cdot)$	TFIDF Vectorizer
Encoder Network	Enc-$\boldsymbol{\mu}(\cdot)$	Linear(2048, 2048); ReLU(); Linear(2048, 128); Tanh().
$\boldsymbol{\mu}$=Enc-$\boldsymbol{\mu}(\boldsymbol{t})$, $\boldsymbol{\sigma}$=Enc-$\boldsymbol{\sigma}(\boldsymbol{t})$	Enc-$\boldsymbol{\sigma}(\cdot)$	Linear(2048, 2048); ReLU(); Linear(2048, 128); Exp().
Sample Process $\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{1})$, $\boldsymbol{h}^t$=$\boldsymbol{\mu}$+$\boldsymbol{\epsilon}^T\boldsymbol{\sigma}$	-	-
Decoder Network $\tilde{\boldsymbol{t}} = \mathrm{Dec}(\boldsymbol{h}^t)$	$\mathrm{Dec}(\cdot)$	Linear(128, 2048); Softmax().
Class Distribution	-	$\boldsymbol{\pi} \in [0,1]^K, \sum_{i=1}^K \pi_i = 1$
Gaussian Components	-	$\{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i\}_{i=1}^K$

Table 12: The NMI results of the basic variants for BERT features. vs Last means the average improvement comparing the current row with the last one.

Variant		AN	SS	SO	Bio	GN-TS	GN-T	GN-S	Tw	vs Last
Basis	($\mathcal{M}$)	62.97	68.95	76.12	43.23	93.28	87.65	90.18	89.94	-
w/ $\boldsymbol{h}_i^t$	($\mathcal{M}_{Graph}$)	63.88	73.57	78.89	45.64	95.22	91.42	92.04	92.89	**+2.65**
w/ $\boldsymbol{p}_i^t$	($\mathcal{M}_{Align}$)	66.13	75.68	78.33	44.90	96.22	91.62	93.14	94.27	**+0.84**
COTC	($\mathcal{M}_{Joint}$)	**67.09**	**77.09**	**79.19**	**46.09**	**96.33**	**92.07**	**93.49**	**95.09**	**+0.77**

Table 13: The ACC results of the basic variants for TFIDF features. vs Last means the average improvement comparing the current row with the last one.

Variant		AN	SS	SO	Bio	GN-TS	GN-T	GN-S	Tw	vs Last
Basis	($\mathcal{M}$)	82.62	80.11	82.90	48.17	83.37	79.98	80.65	73.23	-
w/ $\boldsymbol{h}_i^b$	($\mathcal{M}_{Graph}$)	84.89	87.41	84.12	49.68	85.07	80.82	82.11	74.99	**+2.26**
w/ $\boldsymbol{p}_i^b$	($\mathcal{M}_{Align}$)	85.83	88.95	85.04	51.55	87.86	81.68	84.40	87.65	**+2.98**
COTC	($\mathcal{M}_{Joint}$)	**87.26**	**90.00**	**86.87**	**52.41**	**90.35**	**83.36**	**86.03**	**91.05**	**+1.80**

Table 14: The NMI results of the basic variants for TFIDF features. vs Last means the average improvement comparing the current row with the last one.

Variant		AN	SS	SO	Bio	GN-TS	GN-T	GN-S	Tw	vs Last
Basis	($\mathcal{M}$)	55.76	58.96	71.44	40.57	93.48	90.25	90.04	88.86	-
w/ $\boldsymbol{h}_i^b$	($\mathcal{M}_{Graph}$)	60.57	71.91	78.77	44.25	94.44	91.13	91.17	89.70	**+4.07**
w/ $\boldsymbol{p}_i^b$	($\mathcal{M}_{Align}$)	63.94	74.80	75.07	43.39	94.33	90.83	92.29	92.85	**+0.70**
COTC	($\mathcal{M}_{Joint}$)	**66.16**	**76.53**	**78.97**	**45.69**	**96.19**	**91.91**	**93.41**	**94.72**	**+2.01**