Neural Text Classification by Jointly Learning to Cluster and Align

Yekun Chai[1][*]   **Haidong Zhang**[1][*]   **Shuo Jin**[2]
[1]Institute of Automation, Chinese Academy of Sciences
[2]University of Pittsburgh
chaiyekun@gmail.com   haidong_zhang14@yahoo.com

Abstract

Distributional text clustering delivers semantically informative representations and captures the relevance between each word and semantic clustering centroids. We extend the neural text clustering approach to text classification tasks by inducing cluster centers via a latent variable model and interacting with distributional word embeddings, to enrich the representation of tokens and measure the relatedness between tokens and each learnable cluster centroid. The proposed method jointly learns word clustering centroids and clustering-token alignments, achieving the state of the art results on multiple benchmark datasets and proving that the proposed cluster-token alignment mechanism is indeed favorable to text classification. Notably, our qualitative analysis has conspicuously illustrated that text representations learned by the proposed model are in accord well with our intuition.

1   Introduction

Text classification, as an extensively applied fundamental cornerstone for natural language processing (NLP) applications, such as sentiment analysis [34], spam detection [13] and spoken dialogue systems [22; 8], has been widely studied for decades. In general, almost all NLP tasks can be cast into classification problems on either document, sentence, or word level. Here we are focusing on the means of it in a narrow sense, *i.e.*, given a sequence of tokens with arbitrary length, predicting the most likely categorization it belongs to.

Considerable compelling neural approaches to the text classification task have empirically demonstrated their remarkable behaviors in recent years, to whom how to orchestrate and compose the semantic and syntactic representations from texts are central. Much of the work concentrated on learning the composition of distributional word representations [25; 27; 5] for categorization, wherein plenty of deep learning methods have been adopted, such as TextCNNs [14], RCNNs [17], recurrent neural networks (RNNs) [21], FastText [12], BERT [6], *etc*. Most of them learn the word representations by firstly projecting the one-hot encoding of each token through a pretrained or randomly initialized word embedding matrices to acquire the dense real-valued vectors, and then feed them into neural models for classification.

These methods, however, have only exploited the low-dimensional semantic representations for each sample text in a supervised way. Some argued that unsupervised latent representations such as topic [10; 24; 18] or cluster modeling [4; 38; 31; 7] mined by latent variable models may be of benefit. [4] maintained that word clustering could deliver the useful semantic information by grouping all words in the corpus and can thus promote the classification accuracy. Moreover, [36] incorporated the neural topic models with Variational Autoencoder (VAE) [16] into the classification tasks so

---

[*]Equal contribution.

as to discover the latent topics in the document level and encode the co-occurrence of words with bag-of-words statistics.

Learning such corpus-level representation can administer to the enrichment of more globally informative features and is thus favorable to the task performance. There are plenty of works adopting VAE for learning these latent variables to boost the text classification performance [33; 2; 28]. Nevertheless, there remain problems that we cannot directly treat the sampled latent space of VAE for clustering centroids since there is no mechanism to modulate the representation of different samples towards different mean and variance for a better discrimination purpose under the Gaussian distribution assumption [19]. [29] and [19] alleviate these issues by minimizing the distance between the learnable latent representation from latent variable models and the clustering centers generated from statistical clustering approaches.

Grounding on this, we design an *ad hoc* **Clu**stering-**E**nchanced neural model (hereafter CluE) that jointly learns the distributional clustering and the alignment between the domain-aware clustering centroids and word representations in the Euclidean hidden semantic space for text classification, with the vector space assumption that words with similar meanings are close to each other [26]. Instead of directly treating the latent variables as the clustering centroids, we employ a co-adaptation strategy to minimize the difference between the hidden variables and trainable clustering centroids initialized by traditional clustering algorithms with soft alignments.

In the present work, we propose the cluster-token alignment mechanism by assigning relevance probability distribution of clusters to each token, indicating how likely it is that tokens are correlated with each cluster center. In which clustering centroids are co-regulated with learned latent variables and can be regarded as the domain- or task-specific feature indicators.

Our work illustrates that jointly adapting the clustering centroids and learning the cluster-token alignment holds the promise of advancing the text classification performance by incorporating the clustering-aware representations. Our key contributions are:

- to empirically and visually demonstrate that the proposed model could surprisingly deliver visually-interpretable text representations for text classification (as fig. 4).
- to show that our clustering-token interaction mechanism could apparently capture semantic meanings, including the relevance alignment between clusters and input tokens.
- to confirm that our joint learning model eclipses the prevailing baseline models and achieves state-of-the-art results on classifying both short and long texts.
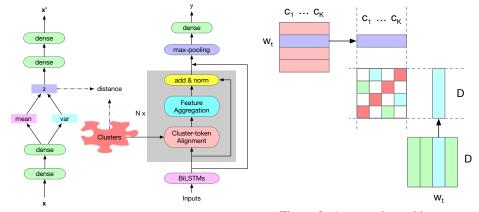

Figure 1: Illustration of CluE models.

Figure 2: Aggregation sublayers.

2 Learning to Cluster and Align

We propose a clustering-enhanced architecture CluE for text classification (see fig. 1), consisting of two main components: a cluster co-adaptation module for refining clustering centroids (Sec. 2.1) and a cluster-enhanced classifier model for categorization (Sec. 2.2).

We denote the input sequence of words as $\mathbf{w} = \{w_1, w_2, \cdots, w_T\}$ with the length of T, the corresponding one-hot representation of labels as $\mathbf{y}$, its *tf-idf* feature as $\mathbf{x}$. To project the discrete word sequences into dense word representations as inputs, we define the word embedding matrices $\mathbf{E} \in \mathbb{R}^{V \times D}$ where V is the vocabulary size of the corpus, and D is the dimension of word embeddings.

2.1 Distributional Clustering with Hidden Variables

Cluster co-adaptation components (left in fig. 1) leverage latent variable models involving autoencoder (AE) and VAE to conduct the encoding and reconstruction process, simultaneously using the latent representation to regulate the trainable clustering centroids. Typically, we take VAE for the latent variable generation, namely Clustering-VAE, (CVAE), for further explication.

CVAE encodes the *tf-idf* features $\mathbf{x} \in \mathbb{R}^V$ for each word sequence into a latent variable word representation $\mathbf{z} \in \mathbb{R}^D$, then resorts to a modulation mechanism by forcing $\mathbf{z}$ to learn together with cluster centers, concurrently reconstructing the original input as $\mathbf{x}' \in \mathbb{R}^V$ from $\mathbf{z}$.

We denote the trainable cluster centroids consisting of K cluster centers $\{\mathbf{c}_k \in \mathbb{R}^D | k = 1, 2, \cdots, K\}$, where the cluster centroid parameters are initialized with K-means or Gaussian Mixture Models (GMM) trained on the pretrained word representations.

Clustering Co-adaptation We apply two-layer feed-forward fully-connected connections with non-linearity, denoted f, to project the *tf-idf* features $\mathbf{x}$ into the mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and variance $\boldsymbol{\sigma} \in \mathbb{R}^D$ under a prior $p(\mathbf{z})$ of Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$.

$$\mu, \sigma = f(\mathbf{x}), \quad \mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \tag{1}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ represents $\boldsymbol{\epsilon} \in \mathbb{R}^D$ is sampled from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, $\odot$ denotes the element-wise product.

After sampling $\mathbf{z}$ following the Gaussian prior $p(\mathbf{z})$ with the reparameterization trick, we minimize the distance q_{ik} between the i-th sample's latent encoding $\mathbf{z}_i$ and each cluster centroid $\mathbf{c}_k$. Here we employ a Student's t-distribution kernel following [32] as the distance metric.

$$q_{ik} = \frac{(1 + \|\mathbf{z}_i - \mathbf{c}_k\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k'} (1 + \|\mathbf{z}_i - \mathbf{c}_{k'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \tag{2}$$

where α represents the freedom degree of Student's t-distribution (we set α as 1 in our experiments). q_{ik} can be considered as the probability assigning the i-th sample to the k-th cluster.

The auxiliary target distribution p_i is defined to learn the high confidence assignments [32]. We denote r_{ik} as the squared q_{ik} divided by the total occurrence of each cluster k and p_{ik} as the normalized counterpart among all clusters:

$$r_{ik} = \frac{q_{ik}^2}{\sum_i q_{ik}}, \quad p_{ik} = \frac{r_{ik}}{\sum_{k'} r_{ik'}}. \tag{3}$$

The objective of clustering co-adaptation is defined as a KL divergence between the the Student's t-distribution kernel based distance q_i and the auxiliary target distribution p_i:

$$\mathcal{L}_{\text{cluster}} = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}}. \tag{4}$$

Feature Reconstruction Meanwhile, the original feature input is reconstructed as $\mathbf{x}'$ using two feed-forward layers g, *i.e.*, $\mathbf{x}' = g(\mathbf{z})$. The reconstruction component objective $\mathcal{L}_{\text{recon}}$ is defined as the mean squared error (MSE) between the original features and the reconstructed ones:

$$\mathcal{L}_{\text{recon}} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - \mathbf{x}'_i\|^2. \tag{5}$$

Besides, CVAE employs variational inference to approximate the posterior distribution $q(\mathbf{z}|\mathbf{x})$ and force it to approach the prior $p(\mathbf{z})$. The objective of KL diveregence $\mathcal{L}_{\text{kld}}$ is defined as:

$$\mathcal{L}_{\text{kld}} = \mathbb{KL}\Big(q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}) \Big) \simeq \frac{1}{2} \sum_{j=1}^D \big(1 + \log((\boldsymbol{\sigma}_j)^2) - (\boldsymbol{\mu}_j)^2 - (\boldsymbol{\sigma}_j)^2 \big), \tag{6}$$

where $\boldsymbol{\mu}_j$ and $\boldsymbol{\sigma}_j$ represent the j-th element of mean and variance vectors respectively. Refer to [16] for detailed derivations.

2.2 Clustering-token Interaction

We will further describe the process of clustering-token interaction (the right part in fig. 1). Given word sequences $\mathbf{w}$ with length T, we firstly project the discrete word indices through the embedding matrices $\mathbf{E}$ to acquire the dense representation $\mathbf{H}_e$ as the inputs to the clustering-token interaction module. Afterward, a single-layer bi-directional Long Short-Term Memory (LSTM) [11] network to encode the contextual information of tokens into $\mathbf{H}_e$, wherein the output states of forward and backward LSTMs are concatenated at each time step. The dimension of the LSTM hidden state was set to $D/2$, thus we get $\mathbf{H} \in \mathbb{R}^{T \times D}$ after concatenation.

The clustering-token interaction module is composed of a stack of N identical layers, of which each layer has two sublayers, namely clustering-token global alignment and feature aggregation sublayers. We supplement a residual connection [9] around them followed by layer normalization [3] for each stack.

Cluster-token Global Alignment The clustering-token alignment sublayer can be described as the alignment score $\mathbf{a} \in \mathbb{R}^{K \times T}$ between the k-th cluster $\mathbf{c}_k$ and the t-th word representation $\mathbf{h}_t$ in each sequence:

$$\mathbf{a}_{kt} = \text{align}(\mathbf{c}_k, \mathbf{h}_t) = \frac{\exp\left(\text{score}(\mathbf{c}_k, \mathbf{h}_t)\right)}{\sum_{k'} \exp\left(\text{score}(\mathbf{c}_{k'}, \mathbf{h}_t)\right)}. \tag{7}$$

We adopt the alignment score function as in [23], which can be interpreted as the content-based relevance function. It measures the relevance between each cluster and each input token:

$$\text{score}(\mathbf{c}_k \mathbf{h}_t) = \left\{ \begin{array}{ll} \mathbf{c}_k \mathbf{h}_t^\top & \text{dot} \\ \mathbf{c}_k \mathbf{W_a} \mathbf{h}_t^\top & \text{general} \\ \mathbf{W_a}[\mathbf{c}_k; \mathbf{h}_t] & \text{concat} \end{array} \right\}, \tag{8}$$

where $\mathbf{W_a} \in \mathbb{R}^{D \times D}$ denotes trainable parameters. Dot alignment score is used in our experiments.

Feature Aggregation The feature aggregation sublayer attends to different dimension of word representations w.r.t each cluster centroids. We refine the t-th original representation $\mathbf{h}_t$ using the corresponding cluster-word alignment score $\mathbf{a}_{kt}, k = \{1, \cdots, K\}$ via an Kronecker product.

$$\mathbf{u}_t = \left(\mathbf{a}_{kt}^\top \otimes \mathbf{h}_t\right)\mathbf{W}_d, \tag{9}$$

$$\mathbf{s}_t = \text{layer\_norm}(\mathbf{u}_t + \mathbf{h}_t), \tag{10}$$

where $\mathbf{u}_t \in \mathbb{R}^{T \times D}$ denotes the aggregated output features, $\mathbf{W}_d \in \mathbb{R}^{KD \times D}$ is the trainable weight matrix for dimension reduction, $\otimes$ is the Kronecker product.

Fig. 2 illustrates the interaction between $\mathbf{a}_{kt}^\top$ (upper-left) and $\mathbf{h}_t$ (lower right corner). So far, we maintain the idea that $\mathbf{a}_{kt}^\top$ calculated from previous sublayer measures the relevance between each cluster (*i.e.*, matrix column $\{\mathbf{c}_1, \cdots, \mathbf{c}_K\}$ in the figure) and word tokens(*i.e.*, w_t in the figure). It can be further interpreted that the resulting matrix in the middle of the figure reveals the relation between cluster and embedding space.

Classification We define the final representation of stacked clustering-token interaction layers as $\mathbf{S} = \{\mathbf{s}_t | t = \{1, 2, \cdots, T\}\}$. Firstly we add a residual connection from after the bi-LSTMs followed by a layer normalization, and pass it through a dense layer with a non-linearity. Then a max-pooling layer is employed to get the final text representation $\mathbf{o} \in \mathbb{R}^D$, with a linear projection to output the final classification logits $\hat{\mathbf{y}}$.

$$\mathbf{o} = \max_{t=1}^{T} \text{ReLU}(\text{layer\_norm}(\mathbf{S} + \mathbf{H})\mathbf{W}_O), \tag{11}$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{o}\mathbf{W}_p), \tag{12}$$

$$\mathcal{L}_{\text{cls}} = -\sum_i \mathbf{y}_i \log \hat{\mathbf{y}}_i, \tag{13}$$

where $\mathbf{W}_O \in \mathbb{R}^{D \times D'}$ and $\mathbf{W}_p \in \mathbb{R}^{D' \times |C|}$ are trainable parameters, D' denotes the hidden dimension, $|C|$ is the number of classes. $\mathcal{L}_{\text{cls}}$ represents the cross entropy loss between label $\mathbf{y}$ of one-hot encoding and the output logits $\hat{\mathbf{y}}$ for the classifier.

4

Type	Datasets	Classes	Train	Test	Vocabulary	Avg. T
short	AG's News	4	120k	7.6k	13,464	7
	DBpedia	14	560k	70k	35,165	3
	Amazon	5	3,000k	650k	47,638	5
	Yahoo	10	908,904	227,227	35,308	11
long	AG's News	4	120k	7.6k	25,537	32
	DBpedia	14	560k	70k	132,251	48
	IMDB	2	40k	10k	38,272	241
	R8	8	5,485	2,189	5,160	66

Table 1: Summary of datasets after preprocessing.

Joint Training Finally, the joint training objective of the CluE model is designed as:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_1 \mathcal{L}_{\text{cluster}} + \lambda_2 \mathcal{L}_{\text{recon}} + \lambda_3 \mathcal{L}_{\text{kld}}, \tag{14}$$

where $\{\lambda_1, \lambda_2, \lambda_3\}$ represent scalars for scaling and are selected on holdout sets.

3 Experiment Settings

Our experiments are conducted on the task of text classification using four different benchmark datasets for short and long text classification separately. For comparison, we report the performance of a bunch of strong baseline models w.r.t the prediction accuracy on the test set. Code in TensorFlow [1] will be available after the double-blind review (for reviewers, please check the attached materials).

3.1 Data

Table 1 summarizes the statistics of different datasets. Original train/test sets have remained except 80/20 train/test split on IMDB and Yahoo datasets. 10% of the training data is extracted as holdout sets. The preprocessing is composed of handling web links, digits, punctuations, and lowercasing after replacing with [UNK] placeholders tokens whose occurrences are not greater than 3.

We adopt the title texts from AG's News[§], DBpedia[§], Amazon Reviews[§] and Yahoo! Answers[‡] text data for short text classification. Meanwhile, content fields from AG's News[§] and DBpedia[§], as well as IMDB movie reviews[¶] and R8[‖] datasets are utilized as long texts[**].

3.2 Models

Comparison Models We test mainstream baseline models including TextCNN[†], BiLSTM, RCNN [17] and AttBiLSTM [20]. 6 and 12 Transformer [30] encoder layers followed by a max-pooling layer are implemented. Topic Memory Network (TMN)[†] [36] that combines neural topic model and memory network to tackle topic modeling and classification jointly were experimented. To further compare with the recently prevailing Text Graph Convolutional Networks [†] (TextGCN) [35], which builds corpus-level graph using word co-occurrences and document word relations, we conduct experiments on the source code. Additionally, we reproduce a text-level GCN based on the dependency parsing graph for each input sample, similar to [37].

Proposed Models We run our model with three experimental settings: **CluE-baseline**, which directly use clusters initialized with K-means method for CluE training by removing the latent variable components like CVAE; **CluE-CAE**, which applies Autoencoder (AE) Networks based on CluE-baseline to generate hidden states, namely Clustering-AE (CAE); **CluE-CVAE** supplemented on CluE-baseline as described in sec. 2.

[§] These datasets are obtained from [39]

[‡] Available online at `https://webscope.sandbox.yahoo.com/catalog.php?datatype=l&did=11`

[¶] `https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews`.

[‖] Available online at `https://www.cs.umb.edu/~smimarog/textmining/datasets/`

[**] Long texts from Amazon and Yahoo datasets are not adopted due to the infeasibility to train models like TMN and TextGCN on such large volumes of data.

[†] indicates we obtain the source code from the author and use its default settings for experiments.

		short text				long text			
		AG's News	DBpedia	Yahoo	Amazon	AG's News	DBpedia	R8	IMDB
TextCNN [14]		0.8696	0.7049	0.6369	0.4762	0.9122	0.9864	0.9571*	0.9023
BiLSTM		0.8622	0.6983	0.6364	0.4683	0.9064	0.9839	0.9631*	0.8976
RCNN [17]		0.8636	0.7020	0.6397	0.4730	0.9109	0.9867	0.9719	0.9067
AttBiLSTM [20]		0.8682	0.6997	0.6366	0.4775	0.9082	0.9858	0.9604	0.8954
Transformer [30]	L6	0.8649	0.6816	0.6229	0.4630	0.8904	0.9794	0.9357	0.8151
	L12	0.8661	0.6710	0.6178	0.4654	0.8913	0.9693	0.9301	0.8242
TMN [36]		0.8537	0.5632	0.5854	-	-	-	-	-
TextGCN [35]		0.8572	-	-	-	-	-	0.9707*	-
Text-level GCN		0.8652	0.6980	0.6357	0.4689	0.9051	0.9847	0.9554	0.8761
CluE-baseline		0.8821	0.7053	0.6410	0.4318	0.9150	0.9867	0.966	0.8709
CluE-CAE		0.8832	0.7039	0.6446	0.4782	0.9146	0.9873	0.9641	0.8996
CluE-CVAE		**0.8846**	**0.7055**	**0.6475**	**0.4884**	**0.9198**	**0.9887**	**0.9738**	**0.9096**

Table 2: Summary of model performance (accuracy on the test set) on classification benchmark datasets, where '*' indicates the result is from [35].

Hyperparameter settings Hyper-parameters are tuned with grid search on both short and long texts. We use the 300-dimensional pretrained Glove embeddings[‡] to initialize the word embedding matrix and clustering centroids. The maximum sequence length is set to 20 for all short texts except Yahoo dataset with 28 and the counterpart for long texts are 120 for AG's News and DBpedia and 200 for Yahoo! Answers and Amazon Reviews dataset. For the optimization, we use Adam optimizer [15] with the learning rate 1e-3, batch size of 512 and 64 for long and short inputs, respectively. The training process is set as a maximum of 10,000 steps with early stopping patience of 30 steps. Gradients are clipped when the L2 norm is more than 5. To avoid model over-fitting, we set the dropout rate as 0.2 during training.

4 Results

4.1 Quantitative Results

Table 2 exhibits the classification performances of described models (sec. 3.2) measured in test accuracy. It is obvious from the table that proposed CluE models outperform various dominant baseline models by a clear margin in different benchmark datasets involving both short and long texts.

As shown in the table, CNN-dominant models, *i.e.*, TextCNNs, perform steadily well on all datasets while LSTM-based models including Bi-LSTM, RCNN, and AttBiLSTM achieve better results on long texts. This is consistent with our initial intuition.

The performance of prevailing GCN models, *i.e.*, TextGCN and Text-level GCN, matches that of LSTM- or CNN- dominant methods. However, TextGCN suffers from the lack of transferability and memory-efficiency, making it difficult to apply corpus-level TextGCNs on large datasets.

It is empirically shown that combing topic networks and memory networks to construct TMN models dramatically slows down the training process and can not be afforded for large short text datasets, even not to mention long texts. Notably, '-' symbols in table 2 represent the inacquirability of experimental results. Specifically, the overlong time costs (*e.g.*, ≈4600 seconds per training epoch out of maximum 800 epochs on single NVIDIA Titan RTX GPU for Amazon short datasets) for TMN and memory/running errors in the corpus-level graph construction for TextGCN when handling large volumes of data.

Impact of CluE Architectures Table 2 illustrates the comparison between our CluE models with different settings for classifying AG's News short texts. Our CluE-baseline, CluE-CAE and CluE-CVAE models achieve 2.31%, 2.44% and 2.6% improvement on the test accuracy compared with CluE-baseline without clustering-token interaction layers, *i.e.*, bi-LSTM (see fig. 1 for schematic intuition).

Further, we evaluate the impact of the number of layers and clusters ranging from 1 to 8 on CluE-CVAE models and plot a line graph in fig 3 using AG's News dataset.

[‡]Available at `http://nlp.stanford.edu/data/glove.6B.zip`

Impact of Cluster Numbers   It can be seen from fig. 3a that the model performance remains stable when the range of layer number is within 1 to 5, and then decreases afterward when classifying long texts. In contrast, the curve of short texts initially shows an increasing trend and reach a plateau from 4 to 8. It is observed that the model performance is superior when the number of clusters is approaching to that of classes. In our optimal settings of CluE-CAE and CluE-CVAE, the optimal cluster counts are both 4, which is exactly the class number of AG's News. Such curves could match our intuition that clustering mechanisms might learn the semantic meanings beneficial for classification to some extent.

Impact of Layer Numbers   Fig. 3b witnesses the influence trend of layer numbers. We find that models with three layers reach their acme on test performance for short texts whilst the test accuracy with layer numbers range from 3 to 7 remains a steady stage for long texts. It shows that models with layer number 3 can get superior results. We recommend readers not to increase the depth of layers too much according to the law of Occam's razor.

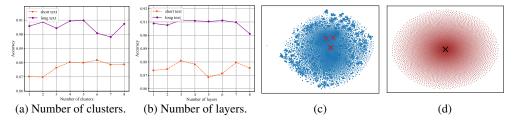| (a) Number of clusters. | (b) Number of layers. | (c) | (d) |

Figure 3: (a) and (b) reports the accuracy on test set with the increase of the number of clusters (a) or that of layers (b); (c) and (d) are scatter visualization of learned cluster centroids ('x' marks) and trained word embeddings (drawn as blue points in (c)) or latent hidden variables $\mathbf{z}$ (drawn as brown points in (d)) using t-SNE on AG's News dataset.
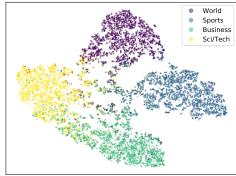
4.2 Qualitative Analysis

To illustrate the impact of CluE-CVAE models, we will display visualizations on AG's News test set unless mentioned otherwise. Main model settings used for visualization are as follows: clustering number 4, clustering-token interaction layer number 4, scaling factors $\lambda_1 = \lambda_2 = \lambda_3 = 1$.
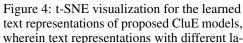
Text Representation Visualization   In the proposed CluE-CVAE model, sentence embeddings are acquired after the max-pooling layer (upper-right in fig. 1). We plot the sentence embedding together with its corresponding classes into a scatter graph in fig. 4. It is dramatically vivid that sentence representations from different label classes are apparently partitioned into four distinct clusters and well separated from each other, hugely validating the ideas that our CluE models deliver the better discrimination between different classes. Meanwhile, such orthogonal clustering groups could provide more informative meanings to classifiers and thus boost the classification performance. Notably, we can observe from the plot that the data points near the intersection between the region of "Business" (in yellow) and "Sci/Tech" (in green) have plenty of mixed points. We make an extrapolation that this is due to the disambiguity of samples from these two classes. We will testify this at the end of this section.

Clustering Centroid Visualization   We plot the scatter graph of learned clustering centroids with trained word embeddings (fig. 3c), and those with all hidden variables of each input texts (fig. 3d). We find that word embeddings after training circumfuse cluster centroids and have a high density around clustering centers. This underpins the proposed clustering-token interaction mechanism, showing that word representations move towards clustering centers to be more classifier-discriminative and class-informative in the compact cluster-token high-dimensional vector space, though no direct minimization mechanism in between were conducted.

Fig 3d witnesses that cluster centers are located at near the center of the multi-variance Gaussian distribution that latent variables $\mathbf{z}$ obey, indicating that our KL divergence loss (eq. 4) for minimizing the distance between hidden variables and cluster centers works well.

Cluster-token Alignment Visualization   It is evident from the heatmap visualization (fig. 6) that the relevance for clusters progressively increases or decreases with the depth increase of layers, finally focusing on a single cluster to form the final sentence representations (fig. 4).

Figure 4: t-SNE visualization for the learned text representations of proposed CluE models, wherein text representations with different la-
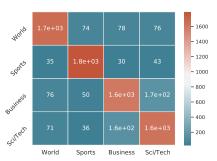

Figure 5: Plot of confusion matrix to evaluate the classification accuracy, where rows and columns denote the actual and predicted labels, the annotation
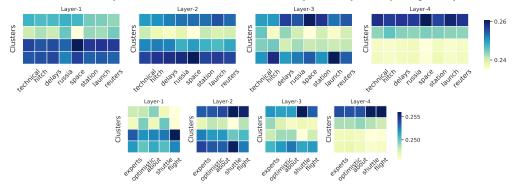

Figure 6: Visualization of outputs from each clustering-token alignment sublayer for example sentences "technical hitch delays russia space station launch reuters" and "experts optimistic about shuttle flight". The x-axis and y-axis of each plot correspond to tokens and clusters respectively. Each pixel shows the relevance weight α_{kt} between the k-th cluster (row) and t-th token (column) (see eq. 7).

Error Analysis   Fig. 5 displays the confusion matrix between the golden and predicted labels, in which samples of label "Business" (hereafter 'B') and "Sci/Tech"(hereafter 'S') are mis-predicted in the most cases.

To unveil this, we go through the wrongly predicted samples and attribute this to three reasons: true mispredictions, mislabels, or ambiguities between these two classes.

Firstly, we observe that samples involving tech company or product names could be predicted as 'S' by mistake, such as "google puts desktop search privacy up front", "amazon sues spammers for misleading consumers", "ibm quitting computer business".

Some are labeled by mistake but predicted correctly, such as:

• labeled 'B' but predicted as 'S': "fcc mulls airborne mobile phone use", "web based kidney match raises ethics questions".
• labeled 'S' but predicted as 'B': "google founders selling off stock", "ibm buys two danish services firms", "'microsoft signs two indian deals'".

However, there also exist plenty of ambiguous samples, like "google targets software giant", "google puts desktop search privacy up front", "red hat replaces cfo" which are labeled as 'B' in the dataset.

This exactly matches our previous assumption, according to fig. 4. It can be further inferred that text representations learned by our models could surprisingly be consistent with our intuition in terms of the semantic vector space, validating the impact of the proposed models.

5   Conclusion

The proposed CluE architecture enjoys both the advantage of latent variable models and clustering-token interaction mechanisms without introducing additional knowledge from outside, which allows

for attending to semantic meanings from both tokens and their corresponding embedding elements. It overshadows the performance of prevailing baseline models and empirically proves that the proposed clustering-token interaction mechanism could be of benefit for achieving informative sentence embeddings. In the future, we will expand our experiments on the other NLP tasks such as natural language inference and neural machine translation.

Broader Impact

This work has the following potential positive impact on society:

- Leveraging the proposed clustering enhanced method, user-generated text data can be analyzed and categorized to interpret social behaviors of users, such as rumor detection, spam detection, hate-speech detection, and sentiment analysis. In addition, these cluster-based analyses can also be embedded into recommender systems for the purpose of user-profiles and recommendations.
- Latent variable models and clustering mechanisms have been leveraged to boost the prediction accuracy on text classifiers, including intent detection, Twitter hashtag prediction, sentiment analysis, *etc*. Latent variable models are shown to be effective in enriching representations in an unsupervised way.
- Proposed cluster-token interaction mechanism could learn more informative and meaningful text representations as visualized in the paper, which can be adopted in other relevant applications such as natural language inference, topic modeling, *etc*.
- Our methods connect between the neural clustering approaches and text classification tasks, which may be widely applied in the industry. Besides, other prevailing methods could also borrow this idea, like enriching BERT families with similar methods.
- This clustering-enhanced method can be transferred into semi-supervised learning in practice by providing pseudo labels by assigning learned text representations to the closest cluster of which label is defined as the mode. This may reduce the requirement of amounts of golden labeled data and thereby curtail the cost.

At the same time, this work may have some negative consequences because there is no mechanism to enforce clustering centroids to be differential and far apart with each other, which can be seen from our plots. The performance may be further promoted by handling this problem.

It should be noted that the failure of the system could result in the wrong prediction of classifications, which may further lead to accumulative errors in downstream tasks, such as spoken dialogue systems.

The current study may suffer from implicit biases in NLP systems:

- the demographic/representative bias that might be learned implicitly from a certain domain. Take our experimental AG's News dataset, for example, texts containing technical company names may be possibly predicted into "Sci/Tech" categories.
- the implicit bias from word embeddings pretrained on Wikipedia. For example, family- or gender-related topics occurred more about women. Although Wikipedia is a major dataset used for pretraining embeddings, it has inevitable biases with crowdsourcing.

As for the ethical aspect, our initialized cluster centroids might leverage some implicit gender- and ethnicity-relevant bias learned from pretrained embeddings, which should be checked when applying our methods into industrial applications.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Babajide O Ayinde and Jacek M Zurada. Deep learning of constrained autoencoders for enhanced understanding of data. *IEEE transactions on neural networks and learning systems*, 29(9):3969–3979, 2017.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] L Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103, 1998.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Harsha S Gowda, Mahamad Suhil, DS Guru, and Lavanya Narayana Raju. Semi-supervised text categorization using recursive k-means clustering. In *International Conference on Recent Trends in Image Processing and Pattern Recognition*, pages 217–227. Springer, 2016.

[8] Arshit Gupta, John Hewitt, and Katrin Kirchhoff. Simple, fast, accurate intent classification and slot labeling for goal-oriented dialogue systems. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 46–55, 2019.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Swapnil Hingmire and Sutanu Chakraborti. Sprinkling topics for weakly supervised text classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 55–60, 2014.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[13] Stefan Kennedy, Niall Walsh, Kirils Sloka, Andrew McCarren, and Jennifer Foster. Fact or factitious? contextualized opinion spam detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 344–350, 2019.

[14] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[18] Zhenzhong Li, Wenqian Shang, and Menghan Yan. News text classification model based on topic model. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–5. IEEE, 2016.

[19] Kart-Leong Lim, Xudong Jiang, and Chenyu Yi. Deep clustering with variational autoencoder. *IEEE Signal Processing Letters*, 27:231–235, 2020.

[20] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[21] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

[22] Ryan Lowe, Iulian V Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. On the evaluation of dialogue systems with next utterance classification. *arXiv preprint arXiv:1605.05414*, 2016.

[23] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[24] Chenglong Ma, Weiqun Xu, Peijia Li, and Yonghong Yan. Distributional representations of words for short text classification. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 33–38, 2015.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[28] Rodrigo GF Soares. Effort estimation via text classification and autoencoders. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2018.

[29] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican Congress on Pattern Recognition*, pages 117–124. Springer, 2013.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[31] Peng Wang, Bo Xu, Jiaming Xu, Guanhua Tian, Cheng-Lin Liu, and Hongwei Hao. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174:806–814, 2016.

[32] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[33] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[34] Wei Xue and Tao Li. Aspect based sentiment analysis with gated convolutional networks. *arXiv preprint arXiv:1805.07043*, 2018.

[35] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377, 2019.

[36] Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R Lyu, and Irwin King. Topic memory networks for short text classification. *arXiv preprint arXiv:1809.03664*, 2018.

[37] Chen Zhang, Qiuchi Li, and Dawei Song. Aspect-based sentiment classification with aspect-specific graph convolutional networks. *arXiv preprint arXiv:1909.03477*, 2019.

[38] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. Text classification using semi-supervised clustering. In *2009 International Conference on Business Intelligence and Financial Engineering*, pages 197–200. IEEE, 2009.

[39] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.