NEURAL TEXT CLASSIFICATION BY JOINTLY LEARNING TO CLUSTER AND ALIGN

Anonymous authors
Paper under double-blind review

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 promising results on multiple benchmark datasets and proving that the proposed cluster-token alignment mechanism is 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 a fundamental cornerstone for natural language processing (NLP) applications, such as sentiment analysis (Xue & Li, 2018), spam detection (Kennedy et al., 2019) and spoken dialogue systems (Lowe et al., 2016; Gupta et al., 2019), 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 (Mikolov et al., 2013a; Pennington et al., 2014; Bojanowski et al., 2017) for categorization, wherein plenty of deep learning methods have been adopted, such as TextCNNs (Kim, 2014), RCNNs (Lai et al., 2015), recurrent neural networks (RNNs) (Liu et al., 2016), FastText (Joulin et al., 2016), BERT (Devlin et al., 2018), 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 (Hingmire & Chakraborti, 2014; Ma et al., 2015; Li et al., 2016) or cluster modeling (Baker & McCallum, 1998; Zhang et al., 2009; Wang et al., 2016; Gowda et al., 2016) mined by latent variable models may be of benefit. (Baker & McCallum, 1998) 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, (Zeng et al., 2018) incorporated the neural topic models with Variational Autoencoder (VAE) (Kingma & Welling, 2013) into the classification tasks so 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 (Xu et al., 2017; Ayinde & Zurada, 2017; Soares, 2018). 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 (Lim et al., 2020). (Song et al., 2013) and (Lim et al., 2020) 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 Clustering-Enhanced 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 (Mikolov et al., 2013b). 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 the compromising results on classifying both short and long texts.

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 \( w = \{w_1, w_2, \cdots, w_T\} \) with the length of \( T \), the corresponding one-hot representation of labels as \( y \), its tf-idf feature as \( x \). To project the discrete word sequences into dense word representations as inputs, we define the word embedding matrices \( E \in \mathbb{R}^{V \times D} \) 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 \( x \in \mathbb{R}^V \) for each word sequence into a latent variable word representation \( z \in \mathbb{R}^D \), then resorts to a modulation mechanism by forcing \( z \) to learn together with cluster centers, concurrently reconstructing the original input as \( x' \in \mathbb{R}^V \) from \( z \).

We denote the trainable cluster centroids consisting of \( K \) cluster centers \( \{ c_k \in \mathbb{R}^D | k = 1, 2, \cdots, K \} \), 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 \( x \) into the mean \( \mu \in \mathbb{R}^D \) and variance \( \sigma \in \mathbb{R}^D \) under a prior \( p(z) \) of Gaussian distribution \( \mathcal{N}(\mu, \sigma) \).

\[
\mu, \sigma = f(x), \quad z = \mu + \sigma \odot \epsilon, \quad (1)
\]

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

After sampling \( z \) following the Gaussian prior \( p(z) \) with the reparameterization trick, we minimize the distance \( q_{ik} \) between the \( i \)-th sample’s latent encoding \( z_i \) and each cluster centroid \( c_k \). Here we employ a Student’s \( t \)-distribution kernel following (Xie et al., 2016) as the distance metric.

\[
q_{ik} = \frac{(1 + \|z_i - c_k\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_k (1 + \|z_i - c_k\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \quad (2)
\]

where \( \alpha \) represents the freedom degree of Student’s \( t \)-distribution (we set \( \alpha \) 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 (Xie et al., 2016). 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:

\[
r_{ik} = \frac{q_{ik}^2}{\sum_i q_{ik}}, \quad p_{ik} = \frac{r_{ik}}{\sum_k r_{ik}}. \quad (3)
\]

The objective of clustering co-adaptation is defined as a KL divergence between the the Student’s \( t \)-distribution kernel based distance \( q_t \) and the auxiliary target distribution \( p_i \):

\[
L_{\text{cluster}} = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}}. \quad (4)
\]

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

\[
L_{\text{recon}} = \frac{1}{N} \sum_{i=1}^N \|x_i - x'_i\|^2. \quad (5)
\]

Besides, CVAE employs variational inference to approximate the posterior distribution \( q(z|x) \) and force it to approach the prior \( p(z) \). The objective of KL divergence \( L_{\text{kl}} \) is defined as:

\[
L_{\text{kl}} = \mathbb{KL} \left( q(z|x) \| p(z) \right) \approx \frac{1}{2} \sum_{j=1}^D \left( 1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2 \right), \quad (6)
\]

where \( \mu_j \) and \( \sigma_j \) represent the \( j \)-th element of mean and variance vectors respectively. Refer to (Kingma & Welling, 2013) 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 \( w \) with length \( T \), we firstly project the discrete word indices through the embedding matrices \( E \) to acquire the dense representation \( H_w \) as the inputs to the clustering-token interaction module. Afterward, a single-layer bi-directional Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) network to encode the contextual information of tokens into \( H_w \), 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 \( H \in \mathbb{R}^{T \times D} \) 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 ([He et al., 2016]) around them followed by layer normalization ([Ba et al., 2016]) for each stack.

**Cluster-token Global Alignment** The clustering-token alignment sublayer can be described as the alignment score \( a \in \mathbb{R}^{K \times T} \) between the \( k \)-th cluster \( c_k \) and the \( t \)-th word representation \( h_t \) in each sequence:

\[
a_{kt} = \text{align}(c_k, h_t) = \frac{\exp \left( \text{score}(c_k, h_t) \right)}{\sum_{k'} \exp \left( \text{score}(c_{k'}, h_t) \right)},
\]

We adopt the alignment score function as in ([Luong et al., 2015]), which can be interpreted as the content-based relevance function. It measures the relevance between each cluster and each input token:

\[
\text{score}(c_k, h_t) = \begin{cases} 
  c_k h_t^\top & \text{dot} \\
  c_k W_a h_t^\top & \text{general} \\
  W_a [c_k; h_t] & \text{concat}
\end{cases},
\]

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

**Feature Aggregation** The feature aggregation sublayer attends to different dimension of word representatives \( w.r.t \) each cluster centroids. We refine the corresponding cluster-word alignment score \( a_{kt} \) by calculating from previous sublayer measures the relevance between each cluster and each input token:

\[
\begin{align*}
  u_t &= \left( a_{kt} \otimes h_t \right) W_d, \\
  s_t &= \text{layer_norm}(u_t + h_t),
\end{align*}
\]

where \( u_t \in \mathbb{R}^{T \times D} \) denotes the aggregated output features, \( W_d \in \mathbb{R}^{KD \times D} \) is the trainable weight matrix for dimension reduction, \( \otimes \) is the Kronecker product.

**Classification** We define the final representation of stacked clustering-token interaction layers as \( S = \{ s_t | t = \{1, 2, \cdots, T\} \} \). 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 \( \hat{o} \in \mathbb{R}^{D'} \), with a linear projection to output the final classification logits \( \hat{y} \).

\[
\hat{o} = \max_{t=1}^{T} \text{ReLU}(\text{layer_norm}(S + H) W_o),
\]

\[
\hat{y} = \text{softmax}(\hat{o} W_p),
\]

\[
L_{\text{cls}} = - \sum_{i} y_i \log \hat{y}_i,
\]

where \( W_o \in \mathbb{R}^{D \times D'} \) and \( W_p \in \mathbb{R}^{D' \times |C|} \) are trainable parameters, \( D' \) denotes the hidden dimension, \( |C| \) is the number of classes. \( L_{\text{cls}} \) represents the cross entropy loss between label \( y \) of one-hot encoding and the output logits \( \hat{y} \) for the classifier.
Table 1: Summary of datasets after preprocessing.

<table>
<thead>
<tr>
<th>Type</th>
<th>Datasets</th>
<th>Classes</th>
<th>Train</th>
<th>Test</th>
<th>Vocabulary</th>
<th>Avg. J</th>
</tr>
</thead>
<tbody>
<tr>
<td>short</td>
<td>AG's News</td>
<td>4</td>
<td>120k</td>
<td>7.6k</td>
<td>13,464</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>DBpedia</td>
<td>14</td>
<td>560k</td>
<td>70k</td>
<td>35,165</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Amazon</td>
<td>5</td>
<td>3,000k</td>
<td>650k</td>
<td>47,638</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Yahoo</td>
<td>10</td>
<td>908,904</td>
<td>227,227</td>
<td>35,308</td>
<td>11</td>
</tr>
<tr>
<td>long</td>
<td>AG's News</td>
<td>4</td>
<td>120k</td>
<td>7.6k</td>
<td>25,537</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>DBpedia</td>
<td>14</td>
<td>560k</td>
<td>70k</td>
<td>132,251</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>IMDB</td>
<td>2</td>
<td>40k</td>
<td>10k</td>
<td>38,272</td>
<td>241</td>
</tr>
<tr>
<td></td>
<td>R8</td>
<td>8</td>
<td>5,485</td>
<td>2,189</td>
<td>5,160</td>
<td>66</td>
</tr>
</tbody>
</table>

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

\[
L = L_{\text{cls}} + \lambda_1 L_{\text{cluster}} + \lambda_2 L_{\text{recon}} + \lambda_3 L_{\text{kld}},
\]

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 [Abadi et al., 2016] 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. We randomly split the train/test with 80/20 ratio on IMDB and Yahoo datasets, and remained the original train/test sets for other 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.


3.2 MODELS

Comparison Models We test mainstream baseline models including TextCNN [2], BiLSTM, RCNN [Lai et al., 2015] and AttBiLSTM [Lin et al., 2017]. 6 and 12 Transformer [Vaswani et al., 2017] encoder layers followed by a max-pooling layer are implemented. Topic Memory Network (TMN) [2] (Zeng et al., 2018) 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) [Yao et al., 2019], 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 [Zhang et al., 2019]. It is worth to note that models using outside data such as semi-supervised learning or pretraining are not included here for the fair comparison because the baseline models do not leverage outside data source.

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 vari-

---

1 These datasets are obtained from [Zhang et al., 2015].
4 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.
5 indicates we obtain the source code from the author and use its default settings for experiments.
Table 3: Summary of model performance (accuracy on the test set) on classification benchmark datasets, where '*' indicates the result is from (Yao et al., 2019).

<table>
<thead>
<tr>
<th>Model</th>
<th>short text</th>
<th>long text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AG’s News</td>
<td>DBpedia</td>
</tr>
<tr>
<td>TextCNN (Kim, 2014)</td>
<td>0.8696</td>
<td>0.7049</td>
</tr>
<tr>
<td>BiLSTM (Lai et al., 2015)</td>
<td>0.8622</td>
<td>0.6983</td>
</tr>
<tr>
<td>RCNN (Lai et al., 2015)</td>
<td>0.8636</td>
<td>0.7020</td>
</tr>
<tr>
<td>AttBiLSTM (Kim et al., 2017)</td>
<td>0.8682</td>
<td>0.6997</td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017)</td>
<td>0.8649</td>
<td>0.6816</td>
</tr>
<tr>
<td>Transformer (Lai et al., 2015)</td>
<td>0.8661</td>
<td>0.6710</td>
</tr>
<tr>
<td>TMN (Zeng et al., 2018)</td>
<td>0.8537</td>
<td>0.5632</td>
</tr>
<tr>
<td>TextGCN (Yao et al., 2019)</td>
<td>0.8572</td>
<td>-</td>
</tr>
<tr>
<td>Text-level GCN</td>
<td>0.8652</td>
<td>0.6980</td>
</tr>
<tr>
<td>CluE-baseline</td>
<td>0.8821</td>
<td>0.7053</td>
</tr>
<tr>
<td>CluE-CAE</td>
<td>0.8832</td>
<td>0.7039</td>
</tr>
<tr>
<td>CluE-CVAE</td>
<td>0.8846</td>
<td>0.7055</td>
</tr>
</tbody>
</table>

Impact of CluE Architectures Table 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. for schematic intuition).

Impact of Cluster Numbers It can be seen from 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.

4 RESULTS

4.1 QUANTITATIVE RESULTS

Table 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 short and long text.

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 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.

Hyperparameter settings See Appendix A for details.
**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.

![Impact of Layer Numbers](image)

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 $z$ (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.

![Clustering Centroid Visualization](image)

Figure 4: t-SNE visualization for the learned text representations of proposed CluE models, wherein text representations with different labels are drawn in different colors.

![Clustering Centroid Visualization](image)

Figure 5: Plot of confusion matrix to evaluate the classification accuracy, where rows and columns denote the actual and predicted labels, the annotation values are the number of samples.
Fig 3d witnesses that cluster centers are located at near the center of the multi-variance Gaussian distribution that latent variables $z$ 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).

![Heatmap Visualization](image)

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 $\alpha_{kt}$ 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”(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, it is a good direction to extend our work using more advanced pretrained models such as BERT and GPT.
REFERENCES


A HYPERPARAMETER DETAILS

The hyperparameters of proposed models are tuned with grid search on both short and long texts. We use the 300-dimensional pretrained Glove embeddings\(^3\) 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 (Kingma & Ba, 2014) 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.

The hyperparameters are fine-tuned on AG’s News dataset and remains the same on the others. The tunable hyperparameter sets are: cluster numbers \(\{1, 2, 3, 4, 5, 6, 7, 8\}\), CluE layer numbers \(\{1, 2, 3, 4, 5, 6, 7, 8\}\), learning rate \(\{1e^{-2}, 1e^{-3}, 2e^{-3}, 3, 1e^{-4}\}\), L2 norm of gradient clipping \(\{5, 20, 40\}\), the discount factors (\(i.e., \lambda_1, \lambda_2, \lambda_3\) \(\{0, 0.01, 0.1, 0.2, 0.5, 1\}\), etc. The number of layers and clusters is selected on the validation set. As to the baseline hyper-parameter tuning, we utilize the optimal hyperparameter settings reported in the reference or source code obtained from authors.

\(^3\)Available at http://nlp.stanford.edu/data/glove.6B.zip