SCIENCE CHINA
Information Sciences

March 2021, Vol. 64 130103:1–130103:13
https://doi.org/10.1007/s11432-020-3055-1

• RESEARCH PAPER •
Special Focus on Weakly Supervised Learning

Few-shot text classification by leveraging bi-directional attention and cross-class knowledge

Ning PANG[1], Xiang ZHAO[1*], Wei WANG[2], Weidong XIAO[1] & Deke GUO[1]

[1]*Science and Technology on Information Engineering Laboratory, National University of Defense Technology, Changsha* 410000*, China;*
[2]*School of Computer Science and Engineering, The University of New South Wales, Sydney* 2052*, Australia*

Received 29 June 2020/Accepted 30 July 2020/Published online 7 February 2021

Abstract Few-shot text classification targets at the situation where a model is developed to classify newly incoming query instances after acquiring knowledge from a few support instances. In this paper, we investigate few-shot text classification under a metric-based meta-learning framework. While the representations of the query and support instances are the key to the classification, existing study handles them independently in the text encoding stage. To better describe the classification features, we propose to exploit their interaction with adapted bi-directional attention mechanism. Moreover, distinct from previous approaches that encode different classes individually, we leverage the underlying cross-class knowledge for classification. To this end, we conceive the learning target by incorporating the large margin loss, which is expected to shorten the intra-class distances while enlarging the inter-class distances. To validate the design, we conduct extensive experiments on three datasets, and the experimental results demonstrate that our solution outperforms its state-of-the-art competitors. Detailed analyses also reveal that the bi-directional attention and the cross-class knowledge both contribute to the overall performance.

Keywords text classification, meta-learning, few-shot learning, bi-directional attention, cross-class knowledge

Citation Pang N, Zhao X, Wang W, et al. Few-shot text classification by leveraging bi-directional attention and cross-class knowledge. Sci China Inf Sci, 2021, 64(3): 130103, https://doi.org/10.1007/s11432-020-3055-1

1 Introduction

Text classification (TC) is a pivotal task in natural language processing (NLP), serving a series of downstream applications, such as information retrieval and opinion mining [1, 2]. This task is defined as selecting an appropriate tag for a given unlabeled text from a candidate class set. Recent development of deep learning has led to the interest in supervised TC models via neural networks [3–5]. In practice, these approaches demand large amount of labeled training data. However, acquiring such high-quality data is labor-intensive, and the manual labeling process is time-consuming.

To alleviate the issue, few-shot learning (FSL) is proposed to train classifiers for new classes that are of only a few labeled examples [6]. In computer vision, FSL has been extensively studied [7–9], and meta-learning emerges as a promising paradigm for rapidly generalizing to new concepts in a low-resource scenario. In particular, metric-based methods [9–11] perform classification by learning the distance distributions among classes. As a representative model, the prototypical network [9] generates a prototype vector for each candidate class, and classifies the queries according to the distance between the query and every prototype measured by their vector representations.

In this paper, we look at the NLP counterpart of FSL, and concentrate on the problem of few-shot text classification (FSTC). In FSTC, a model is required to classify a newly incoming piece of text (i.e., query instance), given only few pieces, e.g., 1 piece, of text with known class labels (i.e., support instances). An example is provided as follows.

* Corresponding author (email: xiangzhao@nudt.edu.cn)

Example 1. An example of 5-way 1-shot text classification, where only 1 piece of support instance is given for each of the 5 classes. The ground truth of the query instance is Class 1. Word 'dead' in the support instance of Class 1 expresses the similar meaning to word 'killed' in query instance.

Support set:

Class 1: A person dead after shooting at nashville, tennessee, mall.

Class 2: These towns are trying out a basic-income scheme and it is already changing lives.

Class 3: Smear campaign against michigan candidate shows how hard it is for muslims to run for office.

Class 4: We ate a 7/11 feast to teach you about unhealthy eating habits.

Class 5: What firing an employee teaches you about your company's culture.

Query instance: Unarmed black man killed in mind-boggling, unjustified barrage of police gunfire: lawyer.

Previous studies reveal that metric-based learning offers elegant solutions to some FSL tasks in NLP [12, 13]. In few-shot classification, nevertheless, we observe margins for further improvement. First of all, the interdependency between the query and the support instance is not well explored. In most metric-based models, the query and the support instances are encoded individually [9, 14], and then the measurement is derived from the similarity between them. However, texts within the same class tend to include words which have the similar meaning, as shown in Example 1. The similar parts in the query and the support instance, which are informative clues to recognize the class, should be mutually emphasized by modeling their interaction.

Moreover, previous models consider different classes separately [14–16], and hence, are likely to overlook the cross-class information. The performance of metric-based models largely relies on the spacial distributions of sentences in the embedding space. If the embeddings of all support instances within the same class are far away from each other, it is hard to capture their common characteristics and generate a representative prototype; and if support instances for different classes are close to each other in the embedding space, the produced prototypes are indistinguishable. Nevertheless, this cross-class knowledge is largely neglected by existing research on FSTC.

To overcome the limits, we propose a few-shot text classification framework, which utilizes bi-directional attention mechanism and cross-class knowledge (BACK). First, unlike previous work which summarizes the query and support instances into single feature vectors independently, we calculate the bi-directional attention [17] (i.e., from support to query attention and from query to support attention) for every word to learn the interaction between the query and support instances. In this way, we can obtain the support-aware vector representation for the query and the query-aware vector representation for the support instance. Second, we exploit the underlying knowledge between classes by adding a large margin loss to supervise the distance distributions. We hold the view that instances within the same class should be placed adjacently in the embedding space, while instances with different classes had better be far away from each other. Therefore, we employ the large margin loss to shorten the distances among support instances which belong to the same class (i.e., intra-class), and enlarge the distances between different classes (i.e., inter-class).

Contributions. In summary, this paper concentrates on the task of few-shot text classification, and the main contributions include the following.

• We propose to model the interaction between query and support instances with a bi-directional attention module, which captures the interdependency between them for better recognizing the classes.

• To fully exploit the cross-class knowledge, we put forward a joint learning task by supplementing the large margin loss, which can coordinate the distance distributions of both inter- and intra-classes.

• Extensive experiments on three datasets were carried out with detailed analyses, and the results demonstrate that our proposed model BACK is superior to its competitors.

Organization. In Section 2, we discuss the related work. Afterwards, we give the formal task definition and introduce the methodology, respectively, in Sections 3 and 4. Experiment results and detailed analyses are provided in Section 5, followed by conclusion in Section 6.

2 Related work

Our work involves few-shot learning and attention mechanism. We review the related work of few-shot text classification and attention mechanism in few-shot learning respectively.

2.1 Few-shot text classification

Previous text classification models focus on supervised learning [4,18,19]. However, supervised approaches require large amount of labeled data for each class, and cannot find new classes in testing. To solve these inherent problems, FSL [6, 20] is proposed as a remedy, which can rapidly generalize to new classes with only a few examples per class. Meta-learning is a promising methodology for FSL problems, and generally there are three kinds of meta-learning methods so far: (1) model-based methods [8, 21, 22] utilize a designed memory unit to achieve rapid learning from a few training examples; (2) optimization-based approaches [23, 24] train the meta-learner by either generating the updating gradients for model parameters or directly predicting the model parameters; (3) metric-based meta-learning [9, 11, 25] tries to learn a generalizable metric and the corresponding matching function from the training task. Recently, it shows that FSL is adapted with some success to NLP tasks, such as relation classification [13, 14] and text classification [12, 16].

In the context of FSTC, previous work concentrates on ensemble-based methods that are not learned via the end-to-end manner. Lam et al. [26] proposed a meta-learning method for text categorization, which automatically recommends the appropriate classification algorithm based on the inherent properties of the input data. Since a single metric may be insufficient for the complex task variations, Yu et al. [12] first clustered the diverse text classification tasks, and then learned an independent metric for each cluster. Different from aforementioned literature, our work follows the episodic framework of FSL (to be introduced in Section 3), which is also adopted in [16, 27]. Bao et al. [16] designed an attentive generator to assess the class-specific importance of each word, which informs the ridge regressor [25] to quickly learn from a few examples. Our work is a metric-based meta-learning approach, which is inspired by the prototypical network [9]. The conventional prototypical network produces a prototype for each class by averaging the support sentence embeddings, and calculates the similarity between the query and each prototype by the Euclidean distance. As for our model, we use a perceptron method to calculate the dynamic weights for support instances given a query, and use the same mechanism to compute the similarity score between the query and each prototype (see Subsections 4.4 and 4.5 for details).

2.2 Attention mechanism in few-shot learning

Recently, attention mechanism is widely adopted in various tasks [17, 28, 29], and also shown useful for FSL problems [14,16]. Observing that not all support instances are equal given a query, support instance-level attention is proposed to generate prototypes by a weighted sum manner [14]. Besides, since some dimensions of the sentence embedding might be more discriminative for recognizing some special classes, Gao et al. [14] proposed to use feature-level attention to highlight important dimensions. Similarly, Wu et al. [30] used the self-attention and query-attention mechanisms to achieve dynamic prototype selection. Ye et al. [15] also proposed a matching function to calculate the attention scores over the support instances.

Considering the distributional signatures in a specific class, Bao et al. [16] designed an attentive generator to measure the importance of each word for a specific class. The attentive generator is learned based on the underlying knowledge contained in all accessible support instances. Different from only using information in the support set, Ye et al. [15] proposed a multi-level matching and aggregation network to get the matching information between the query and support instances. We also try to model the interaction between the query and support instances, but we follow the bi-directional attention in [17], which is designed for question answering. Seo et al. [17] used each word in the question to attend all context words, and do the same for the reversed direction. In this way, the bi-directional attention fully learns the interaction between the question and the context. Since this pattern is quite similar to FSL setting, we apply the bi-directional attention to FSTC, and model the interaction between the query and support instances. Following [14, 15, 30], we also use the instance-level attention to compute prototypes dynamically, but the difference is that we design a new function to calculate the attention scores.

3 Task definition

In this paper, we follow the episodic framework to train a meta-learner for FSTC, which is widely adopted in various literature [9, 10, 21]. In this paradigm, we are given two datasets, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$. These two datasets have their own label spaces, which are denoted as $\mathcal{Y}_{\text{train}}$ and $\mathcal{Y}_{\text{test}}$, respectively. It is noted that

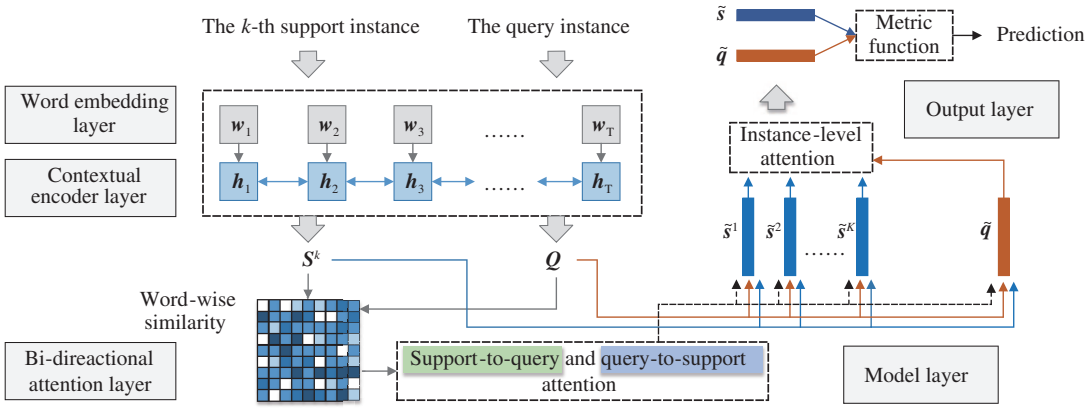

Figure 1 (Color online) The proposed model architecture of BACK.

$\mathcal{Y}_{\text{train}}$ and $\mathcal{Y}_{\text{test}}$ are disjoint with each other. Under few-shot configuration, $\mathcal{D}_{\text{test}}$ can be further split into support set $\mathcal{D}^s_{\text{test}}$ and query set $\mathcal{D}^q_{\text{test}}$. $\mathcal{D}^s_{\text{test}}$ contains K labeled examples, also referred as support instances for each class in $\mathcal{Y}_{\text{test}}$, while $\mathcal{D}^q_{\text{test}}$ is the set of unlabeled texts to be predicted. Our objective is to train a classifier using $\mathcal{D}_{\text{train}}$ that can make predictions for $\mathcal{D}^q_{\text{test}}$ with the assistance of $\mathcal{D}^s_{\text{test}}$.

Specifically, in meta-training, we sample a subset, which consists of N classes, from $\mathcal{D}_{\text{train}}$ to emulate the test scenario. The sampled data is utilized to construct the support set $\mathcal{S}$ and the query set $\mathcal{Q}$, where $\mathcal{S}$ contains K labeled examples for each of the N classes, and $\mathcal{Q}$ consists of L query instances from the same N classes. $\mathcal{S}$ and $\mathcal{Q}$ serve as the training data and testing data respectively. In meta-testing, we apply the same mechanism to make predictions for classes in $\mathcal{Y}_{\text{test}}$, which are unseen during training. The support set is sampled from $\mathcal{D}^s_{\text{test}}$, while the query set from $\mathcal{D}^q_{\text{test}}$. Here, we refer to the aforementioned prediction task as N-way K-shot classification.

4 Methodology

Our few-shot text classification model consists of five layers as shown in Figure 1.

• Word embedding layer maps discrete words into a vector space by looking up a pre-trained word embedding table.

• Contextual encoder layer refines the local representation of each word in a sentence by considering the context. These two layers are the same to the query and support instances.

• Bi-directional attention layer first couples the query with each support instance, and then generates the matching information between them.

• Model layer generates the feature vectors for both query and support instances. And the query-aware instance-level attention module calculates weights for support instances to produce the prototype dynamically.

• Output layer provides a prediction to the query instance by measuring the similarity score between the query and the prototypes.

4.1 Word embedding layer

Suppose that each instance in the support set or query set has T words. Word embedding layer maps each word w_t in the sentence into a vector space, which is denoted as

$$\boldsymbol{x}_t = f_{\text{emb}}(w_t), \tag{1}$$

where $f_{\text{emb}}(\cdot)$ is the mapping function, which transforms w_t into a pre-trained embedding. There are many word embedding learning methods, such as Glove [31] and BERT [32].

4.2 Contextual encoder layer

To refine the representation of each word, we apply a long short-term memory network (LSTM) [33] on top of the word embeddings from the previous layer. For the input sentence $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$, we can

calculate the forward and backward LSTM representations as follows:

$$\begin{aligned}\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \ldots, \overrightarrow{\boldsymbol{h}}_T &= \overrightarrow{\texttt{LSTM}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T),\\ \overleftarrow{\boldsymbol{h}}_1, \overleftarrow{\boldsymbol{h}}_2, \ldots, \overleftarrow{\boldsymbol{h}}_T &= \overleftarrow{\texttt{LSTM}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T),\end{aligned} \tag{2}$$

where $\overrightarrow{\texttt{LSTM}}$ and $\overleftarrow{\texttt{LSTM}}$ denote the forward and backward LSTM respectively. To capture the information from two directions, we concatenate the hidden representations of word w_t in both sides:

$$\boldsymbol{h}_t = \left[\overrightarrow{\boldsymbol{h}}_t : \overleftarrow{\boldsymbol{h}}_t\right], \tag{3}$$

where [:] is the concatenation operation. The output of this layer is a matrix with $T \times 2d_h$ dimensions, where d_h is hidden size of LSTM. To be clear, we use $\boldsymbol{Q} \in \mathbb{R}^{T\times 2d_h}$ to represent the output matrix of the query, and $\boldsymbol{S}^k$ is the matrix for the k-th instance in the support set.

4.3 Bi-directional attention layer

Bi-directional attention layer is in charge of fusing the information from the query and the support instances. Instead of summarizing instances into single feature vectors, this layer first computes the word-wise similarity matrix between the query and the k-th support instance, and then calculates their fused representations. The inputs to this layer are the contextual vector representations of the query $\boldsymbol{Q}$ and the k-th support instance $\boldsymbol{S}^k$. The outputs are the support-aware vector representation of the query, and the query-aware vector representation of the support instance.

Word-wise similarity matrix. Before computing the bi-directional attention, i.e., from support to query attention and from query to support attention, we first get the shared word-wise similarity matrix, $\boldsymbol{C}^k \in \mathbb{R}^{T\times T}$, between the query $\boldsymbol{Q}$ and the k-th support instance $\boldsymbol{S}^k$. The similarity matrix is computed as

$$\boldsymbol{C}^k_{ij} = \boldsymbol{v}_1^{\mathrm{T}}[\boldsymbol{Q}_{i:} : \boldsymbol{S}^k_{j:} : \boldsymbol{Q}_{i:} \circ \boldsymbol{S}^k_{j:}], \tag{4}$$

where $\boldsymbol{Q}_{i:}$ is the i-th row of $\boldsymbol{Q}$, $\boldsymbol{S}^k_{j:}$ is the j-th row of $\boldsymbol{S}^k$, $\circ$ is the element-wise multiplication operation, and $\boldsymbol{v}_1 \in \mathbb{R}^{6d_h}$ is the trainable parameter. $\boldsymbol{C}^k$ contains the attentive information between the query $\boldsymbol{Q}$ and the k-th support instance $\boldsymbol{S}^k$. We also tired other methods to calculate the similarity scores, such as dot production and cosine similarity. Among these alternatives, our method which uses the multi-layer perceptrons achieves the best results.

Support-to-query attention. Support-to-query attention signifies which words in the query are relevant to the word in the support instance. Given the t-th word in the support instance, we can compute the attention scores over words in the query:

$$\boldsymbol{\alpha}^k_t = \texttt{softmax}(\boldsymbol{C}^k_{:t}) \in \mathbb{R}^T, \tag{5}$$

where $\boldsymbol{C}^k_{:t}$ is the t-th column of $\boldsymbol{C}^k$, and $\texttt{softmax}(x_i) = \frac{\exp(x_i)}{\sum_k \exp(x_k)}$. Consequently, the query-aware vector representation of the t-th word in the support instance is

$$\widehat{\boldsymbol{S}}^k_{t:} = \boldsymbol{\alpha}^{k\mathrm{T}}_t \boldsymbol{Q}. \tag{6}$$

Hence we can obtain the query-aware vector representation of the k-th support instance $\widehat{\boldsymbol{S}}^k$, which contains the attended query vectors for the entire support instance.

Query-to-support attention. Query-to-support attention signifies which words in the support instance are relevant to the word in the query. Considering the guide of the t-th word in the query, we can calculate the attention weights over words in the support instance $\boldsymbol{S}^k$:

$$\boldsymbol{\beta}^k_t = \texttt{softmax}(\boldsymbol{C}^k_{t:}) \in \mathbb{R}^T, \tag{7}$$

where $\boldsymbol{C}^k_{t:}$ is the t-th row of $\boldsymbol{C}^k$. When the k-th support instance attends the query instance, similar to (6), we get the weighted sum of the most important words in the support instance with respect to the query:

$$\widehat{\boldsymbol{Q}}^k_{t:} = \boldsymbol{\beta}^{k\mathrm{T}}_t \boldsymbol{S}^k. \tag{8}$$

Hence $\widehat{\boldsymbol{Q}}^k$ is the support-aware vector representation of the query, which includes the attended support vectors for the entire query instance.

Feature fusion. In feature fusion, we model the interaction between the original representation and the attended representation of each query and support instance. For the query instance, two representations are fused as follows:

$$\bar{\boldsymbol{Q}} = g([\boldsymbol{Q} : \widehat{\boldsymbol{Q}} : \boldsymbol{Q} \circ \widehat{\boldsymbol{Q}}]\boldsymbol{W}_1), \tag{9}$$

where $\widehat{\boldsymbol{Q}} = \frac{1}{K}\sum_{k=1}^{K}\widehat{\boldsymbol{Q}}^k$, $\boldsymbol{W}_1 \in \mathbb{R}^{6d_h \times d_s}$ is the trainable parameter matrix, and $g(\cdot)$ is the ReLU activation function. Similarly, we fuse the original representation and the attended representation of each support instance as

$$\bar{\boldsymbol{S}}^k = g([\boldsymbol{S}^k : \widehat{\boldsymbol{S}}^k : \boldsymbol{S}^k \circ \widehat{\boldsymbol{S}}^k]\boldsymbol{W}_1), \tag{10}$$

where $\boldsymbol{W}_1$ is shared with (9) since we design a unified mapping function to reduce the dimensions of the representations of the support instance and the query instance. A unified mapping function guarantees to encode instances into the same vector space.

4.4 Model layer

Model layer summarizes all instances into single feature vectors, and generates the prototype for each class. We first feed $\bar{\boldsymbol{Q}}$ and each $\bar{\boldsymbol{S}}^k$ into a bi-directional LSTM with d_h hidden units along each direction. Our goal is to capture the interaction among query words conditioned on support words, and the interaction among support words conditioned on query words. We denote the outputs as $\widetilde{\boldsymbol{Q}}$ and $\widetilde{\boldsymbol{S}}^k$. Then, we aggregate them into single vectors:

$$\widetilde{\boldsymbol{q}} = [\texttt{Ave}(\widetilde{\boldsymbol{Q}}) : \texttt{Max}(\widetilde{\boldsymbol{Q}})], \tag{11}$$

$$\widetilde{\boldsymbol{s}}^k = [\texttt{Ave}(\widetilde{\boldsymbol{S}}^k) : \texttt{Max}(\widetilde{\boldsymbol{S}}^k)], \tag{12}$$

where $\texttt{Ave}(\cdot)$ and $\texttt{Max}(\cdot)$ are average and maximum pooling functions respectively. Inspired by the conventional prototypical networks [9], we calculate a prototype for each class based on the support instances $\{\widetilde{\boldsymbol{s}}^i\}_{i=1}^k$. Since the importance of each support instance varies given a query, we aggregate the support instances via an attention manner, which is defined as

$$e^k = \boldsymbol{v}_2^{\mathrm{T}} g(\boldsymbol{W}_2[\widetilde{\boldsymbol{s}}^k : \widetilde{\boldsymbol{q}} : \widetilde{\boldsymbol{s}}^k \circ \widetilde{\boldsymbol{q}}]). \tag{13}$$

e^k reflects the relevance of each support instance with the query. $\boldsymbol{W}_2 \in \mathbb{R}^{d_h \times 12d_h}$ and $\boldsymbol{v}_2 \in \mathbb{R}^{d_h}$ are trainable parameters. $g(\cdot)$ is a ReLU activation function. Then all support instances are integrated into the prototype:

$$\widetilde{\boldsymbol{s}} = \sum_{k=1}^{K} \frac{\exp(e^k)}{\sum_{i=1}^{K}\exp(e^i)}\widetilde{\boldsymbol{s}}^k. \tag{14}$$

Hence we obtain the feature vector $\widetilde{\boldsymbol{q}}$ of the query instance, and prototypes for all classes which are denoted as $\{\widetilde{\boldsymbol{s}}_{y_1}, \widetilde{\boldsymbol{s}}_{y_2}, \ldots, \widetilde{\boldsymbol{s}}_{y_N}\}$.

4.5 Output layer

This layer predicts which class the query instance belongs to based on the query feature vector and prototypes. Just like conventional prototypical networks, we determine the class of the query instance by measuring its similarity with each prototype:

$$p(y_i \mid \{\widetilde{\boldsymbol{s}}_{y_j}\}_{j=1}^N, \widetilde{\boldsymbol{q}}) = \frac{\exp(h(\widetilde{\boldsymbol{s}}_{y_i}, \widetilde{\boldsymbol{q}}))}{\sum_{j=1}^{N}\exp(h(\widetilde{\boldsymbol{s}}_{y_j}, \widetilde{\boldsymbol{q}}))}, \tag{15}$$

$$h(\widetilde{\boldsymbol{s}}_{y_i}, \widetilde{\boldsymbol{q}}) = \boldsymbol{v}_2^{\mathrm{T}} g(\boldsymbol{W}_2[\widetilde{\boldsymbol{s}}_{y_i} : \widetilde{\boldsymbol{q}} : \widetilde{\boldsymbol{s}}_{y_i} \circ \widetilde{\boldsymbol{q}}]). \tag{16}$$

Eq. (16) has the same form as (13), and shares the same parameters $\boldsymbol{v}_2$ and $\boldsymbol{W}_2$ since this design leads to less parameters and better results.

4.6 Joint training with large margin loss

Conventional prototypical networks-based models [9, 13] utilize the cross-entropy loss to optimize the parameters, which is defined as

$$\mathcal{L}_{\text{Entropy}} = -\frac{1}{L}\sum_{\widetilde{\boldsymbol{q}}\in\mathcal{Q}}\sum_{i=1}^{N} t_{y_i}\log(p(y_i \mid \{\widetilde{\boldsymbol{s}}_{y_j}\}_{j=1}^{N}, \widetilde{\boldsymbol{q}})), \tag{17}$$

where $\mathcal{Q}$ represents the set of sampled query instances in each training batch, and L is the number of query instances. t_{y_i} signifies whether y_i is the label of $\widetilde{\boldsymbol{q}}$. $t_{y_i}=1$ only if y_i is the ground truth of $\widetilde{\boldsymbol{q}}$, else $t_{y_i}=0$. Therefore, the function can be simplified into

$$\mathcal{L}_{\text{Entropy}} = -\frac{1}{L}\sum_{\widetilde{\boldsymbol{q}}\in\mathcal{Q}}\log(p(y_i \mid \{\widetilde{\boldsymbol{s}}_{y_j}\}_{j=1}^{N}, \widetilde{\boldsymbol{q}})). \tag{18}$$

During the training process, we also adopt the cross-entropy loss to optimize our model.

Besides, it is intuitive that instances of the same class should be placed closer in the spacial space than those of different classes. Therefore, we add a large margin loss function to shorten the distances within the same class, and enlarge the distances between classes. The loss is defined as

$$\mathcal{L}_{\text{Margin}} = \frac{1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K}\max(0, \gamma + ||\widetilde{\boldsymbol{s}}_{y_i} - \widetilde{\boldsymbol{s}}_{y_i}^{k}||_2^2 - ||\widetilde{\boldsymbol{s}}_{y_i} - \widetilde{\boldsymbol{s}}^{-}||_2^2), \tag{19}$$

where $\widetilde{\boldsymbol{s}}_{y_i}^{k}$ is the k-th support instance of class y_i, $\widetilde{\boldsymbol{s}}^{-}$ is a random support instance which is not for class y_i, γ is a hyper-parameter, and $||\cdot||_2$ calculates the 2-norm of a vector.

Finally, we obtain the loss function by combining (18) and (19):

$$\mathcal{L} = \mathcal{L}_{\text{Entropy}} + \lambda\mathcal{L}_{\text{Margin}}, \tag{20}$$

where λ is a hyper-parameter to balance the effects of $\mathcal{L}_{\text{Entropy}}$ and $\mathcal{L}_{\text{Margin}}$ on optimizing the whole model.

5 Experiments

In this section, we first introduce the datasets and competing methods. Afterwards, the implementation details are described. Finally, we present the experimental results and detailed analyses.

5.1 Datasets

In this work, we evaluate the proposed model on three datasets.

• Reuters collects Reuters articles from 1987 [34]. This dataset consists of 31 classes that have at least 20 articles.

• HuffPost is comprised of the published HuffPost news titles between 2012 and 2018 [1)]. The data can be split into 41 classes, and each class has 900 titles.

• FewRel is a dataset for few-shot relation classification [13]. The goal is to classify each example, which concerns an annotated entity pair, into a proper relation type. In the experiments, we use the publicly released 80 relation types, which have 700 instances per type.

Among them, Reuters is a document-level dataset, while HuffPost and FewRel are sentence-level datasets. These datasets have been organized by Bao et al. [16], and in our experiments, we keep the same class split of the train, valid and test sets. Table 1 presents the statistical information of the three datasets.

1) Misra R. News category dataset. 2018.

Table 1 Statistics of datasets

Dataset	# train cls.	# valid cls.	# test cls.	# inst./cls.	len (Average/Maximum)
Reuters	15	5	11	20	185.73/936
HuffPost	20	5	16	900	11.48/44
FewRel	65	5	10	700	24.95/38

5.2 Baselines

We select a variety of meta-learning approaches as our competitors.

• MetaNet [8] utilizes a high-level meta learner on top of a base learner to supervise the training process. The meta leaner generates fast weights, which helps the model to generalize to new classes with a few support instances.

• SNAIL [22] first couples each support instance with its label, and then for each class, the model concatenates all instance-label pairs and the query into a sequence. Temporal convolution neural networks and attention modules are harnessed to make predictions by learning experience from the past support instances.

• MAML [23] is a model-agnostic meta-learning algorithm, which is compatible with a variety of learning problems. The algorithm maximizes the sensitivity of the loss functions of new tasks. With the high sensitivity, small changes to parameters can improve the task loss largely.

• FT [35] trains a feature extractor and a classifier using the training examples in the pre-training stage. In the fine-tuning stage, the model fixes the feature extractor and traines a new classifier using a few support instances.

• Proto [9] learns an embedding function to map all instances into a unified vector space, and produces prototypes for classes by averaging the sentence embeddings of the support instances. The model classifies the queries by measuring the distance between the query and each prototype.

• RR [25] teaches a deep neural network to use ridge regression as part of its internal model, which enables the network to quickly adapt to the new data.

• ARR [16] consists of an attentive generator and a ridge regressor. Attentive generator module generates class-specific attention according to the distributional signatures in all accessible support instances, and ridge regressor makes a prediction for the query, after learning from the support set.

Among these competitors, model-based approaches include MetaNet [8] and SNAIL [22]; optimization-based methods consist of MAML [23] and FT [35]; and metric-based models have Proto [9], RR [25], and ARR [16]. Our proposed BACK is a metric-based meta-learning framework.

5.3 Implementation details

We used pre-trained Glove word embeddings [31] for our model and all baselines. We also experimented with pre-trained BERT embeddings [32] on sentence-level datasets. In the contextual encoder layer and model layer, we used a bi-directional LSTM with 100 hidden units and applied dropout [36] of 0.2 on the output. All parameters of the model were optimized by the stochastic gradient descent strategy with a learning rate of 0.1. The hyper-parameter γ in (19) was set to 1, and λ in (20) was set to 1.

During meta-training, we sampled 1000 batches per epoch, and in each episode, we sampled 5 query instances. Besides, we fed 10 classes for each training batch while choosing 5 classes for testing, since training the model harder might lead to better results [8]. All models were trained on the train set, and then the best epoch on the valid set was selected for meta-testing. During meta-testing, we evaluated all models under 5-way 1-shot and 5-way 5-shot settings. Specifically, we sampled 3000 batches for testing, and reported the average accuracy and the variance over 5 random seeds.

5.4 Overall results

Comparison with baselines. We selected different types of meta-learning methods as our baselines. The upper part of Table 2 shows the results of all models under 5-way 1-shot and 5-way 5-shot settings. It reads from the table that our proposed BACK achieves the best performance across almost all settings, demonstrating its effectiveness in few-shot text classification. We also note that BACK consistently outperforms all baselines under 5-way 1-shot, while performing slightly worse than ARR on Reuters under 5-way 5-shot configuration. The reason is that the bi-directional attention might find more similar parts, which are useless common words for detecting the class, in long text. However, ARR calculates

Table 2 Results of 5-way 1-shot and 5-way 5-shot classification on three datasets using Glove word embeddings[a)]

Method	Reuters		HuffPost		FewRel	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
MetaNet	51.46±0.45	62.84±0.61	30.57±0.62	41.74±0.52	58.64±0.44	79.23±0.75
SNAIL	53.16±0.53	63.57±0.88	31.55±0.71	41.59±0.62	59.23±0.53	78.67±0.56
MAML	65.51±0.19	83.67±0.89	34.19±0.79	50.56±0.67	52.36±0.71	65.52±0.84
FT	71.88±0.88	78.46±0.63	39.64±0.28	59.70±0.77	71.88±0.11	75.71±0.29
Proto	66.77±0.12	71.58±0.18	32.60±0.18	47.03±0.17	52.85±0.13	65.85±0.49
RR	72.45±0.42	83.93±0.56	36.61±0.48	50.13±0.68	57.18±0.43	70.32±0.48
ARR	68.59±0.43	**88.23±0.87**	39.98±0.53	58.75±0.42	60.03±0.37	78.83±0.51
BACK	**75.22±0.14**	88.01±0.11	**42.63±0.31**	**60.76±0.38**	**78.43±0.26**	**88.64±0.22**
w/o Bi-Att	72.13±0.23	86.42±0.38	38.48±0.28	54.79±0.22	76.41±0.19	84.56±0.23
w/o In-Att	75.22±0.14	86.88±0.37	42.63±0.31	58.62±0.42	78.43±0.26	86.77±0.31
w/o LM	74.01±0.26	85.33±0.11	41.34±0.27	56.71±0.52	76.37±0.45	86.32±0.13
w/o Shared	74.58±0.24	85.61±0.45	41.97±0.32	57.48±0.26	77.01±0.22	86.37±0.28
MLP→Dot	74.44±0.41	87.67±0.46	41.24±0.38	58.89±0.48	76.77±0.21	87.23±0.18
MLP→Cos	73.88±0.35	86.64±0.35	41.05±0.44	57.84±0.48	77.32±0.24	86.65±0.36

a) The bottom six lines report the ablation study.

Table 3 Results of 5-way 1-shot and 5-way 5-shot classification on sentence-level datasets using BERT embeddings

Method	HuffPost		FewRel	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
MAML	33.19±0.22	49.87±0.41	47.60±0.82	70.56±1.77
FT	38.33±0.41	58.62±0.28	61.22±0.45	80.34±0.64
Proto	33.49±0.25	46.70±0.23	62.54±0.59	76.53±0.98
RR	37.48±0.48	50.72±0.62	66.41±0.71	78.56±0.71
ARR	39.36±0.14	58.24±0.17	70.38±0.35	87.58±0.24
BACK	**43.21±0.19**	**60.77±0.27**	**80.85±0.25**	**90.67±0.42**

the class-specific word-level attention based on all accessible support instances, which is unaffected about the text length.

Ablation study. To understand the effect of each component, we conducted an ablation study by removing a particular part of the model at a time to show its impact. Specifically, we studied the effect of bi-directional attention module (Eqs. (4)–(10)), instance-level attention module (Eq. (13)), large margin loss (Eq. (19)), and shared parameters, which are denoted as w/o Bi-Att, w/o In-Att, w/o LM, and w/o Shared respectively. Besides, we tried other alternatives to calculate the word-wise similarity scores, and investigated the performance of two variants by replacing our multi-layer perceptron (MLP) method (Eq. (4)) with dot production (Dot) and cosine similarity (Cos). These two variants are denoted as MLP→Dot and MLP→Cos. From the lower part of Table 2, we can see that each of the four components contributes to the overall performance of BACK, and that bi-directional attention and large margin loss are more important than instance-level attention in improving the accuracy. It is also observed that sharing parameters, such as (13) and (16), can lead to better performance. Moreover, in comparison with the two variants which use other methods to compute the word-wise similarity matrix, our model achieves the best results on three datasets, because the MLP method has adaptive parameters which can be updated according to the data distribution.

Effect of word embeddings. To show the effect of pre-trained word embeddings, we also experimented with the word representations given by BERT [32] for sentence-level datasets, i.e., HuffPost and FewRel. In this set of experiments, we selected MAML, FT, Proto, RR, and ARR as our baselines, and reported the results in Table 3. From the results, we can observe that the performance of all models gains from BERT embeddings on FewRel, but fails to get boost on HuffPost dataset. The reason is that although BERT provides high-quality pre-trained word embeddings to represent semantic information, news title classification on HuffPost dataset is more dependent on key words in the title than contextual understanding.

Number of testing classes. In this part, we set the range of the number of testing classes from 5 to 10 and evaluated the performance of different few-shot models. We selected MAML, FT, Proto, RR, and

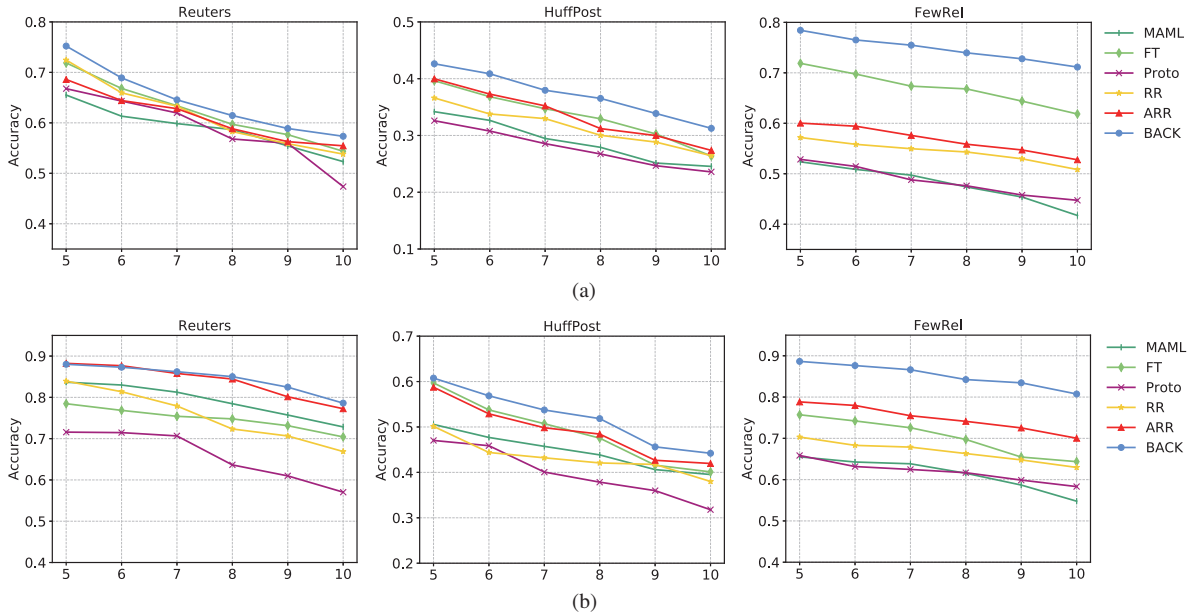

Figure 2 (Color online) The performance trend of few-shot models along the increase of testing classes, where we set the range of the number of classes from 5 to 10. (a) The results under 1-shot setting on three datasets; (b) the results under 5-shot setting on three datasets.

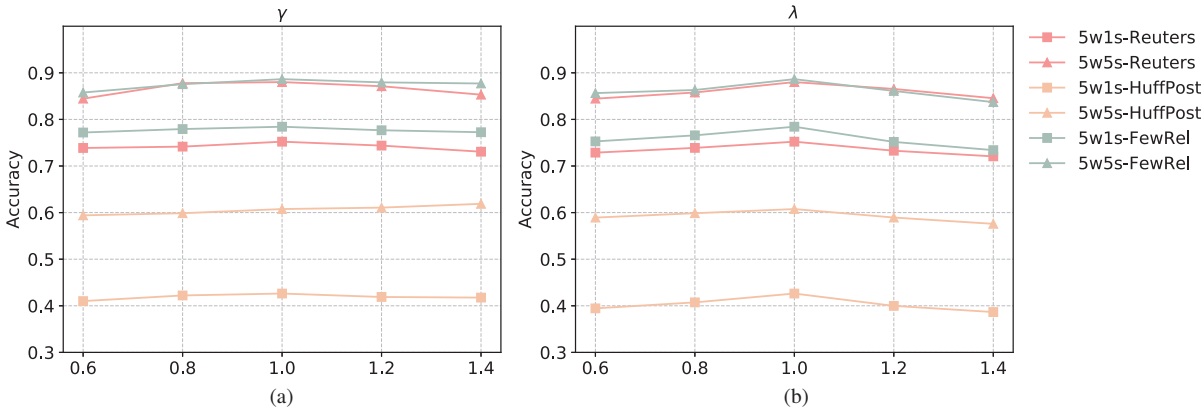

Figure 3 (Color online) The performance of BACK given different values of (a) γ and (b) λ. 5w1s stands for the 5-way 1-shot setting, and 5w5s stands for the 5-way 5-shot setting. The performance of BACK on three datasets is presented.

ARR as our competing methods in this set of experiments. Figure 2 shows the results on three datasets. It reads from the results that our proposed BACK consistently outperforms the baselines under almost all settings. Moreover, the performance of all models drops along the increase of testing classes because tasks with more classes become more difficult.

5.5 Qualitative analyses

Influence of hyper-parameters. As mentioned in Subsection 5.3, we set γ and λ to 1 empirically. Here, we aim to prove that minor changes of these hyper-parameters do not have a significant influence on the final results. Specifically, we keep λ as 1, and choose γ from $[0.6, 0.8, 1.0, 1.2, 1.4]$; then we keep γ as 1, and choose λ from $[0.6, 0.8, 1.0, 1.2, 1.4]$. It can be concluded from Figure 3 that the results fluctuate slightly given different values of γ and λ.

Text embedding visualization. In this set of experiments, we give an illustrative visualization of support instance embeddings and query embeddings. We visualize the learned text embeddings using the t-SNE tool [37]. Figure 4 presents the support instance embeddings of a 5-way 5-shot classification scenario. From the results, we can see the following. (1) With the addition of bi-directional attention (Figures 4(a) and(b)), the support instance embeddings of the target class, i.e., cls. 5, become distin-

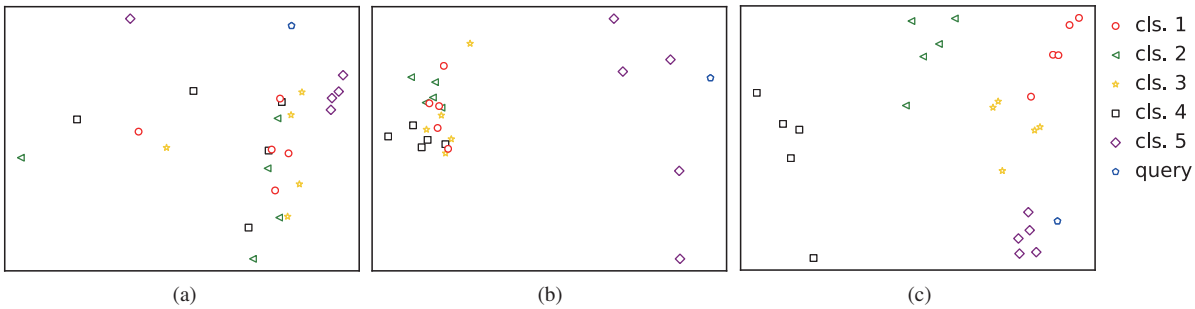

Figure 4 (Color online) Comparison between support instance embeddings trained by three models: (a) BACK without Bi-Att and LM; (b) BACK without LM; (c) BACK. A 5-way 5-shot testing batch is sampled from Reuters dataset, and the ground truth of the given query is cls. 5, where cls. is the abbreviation of class.

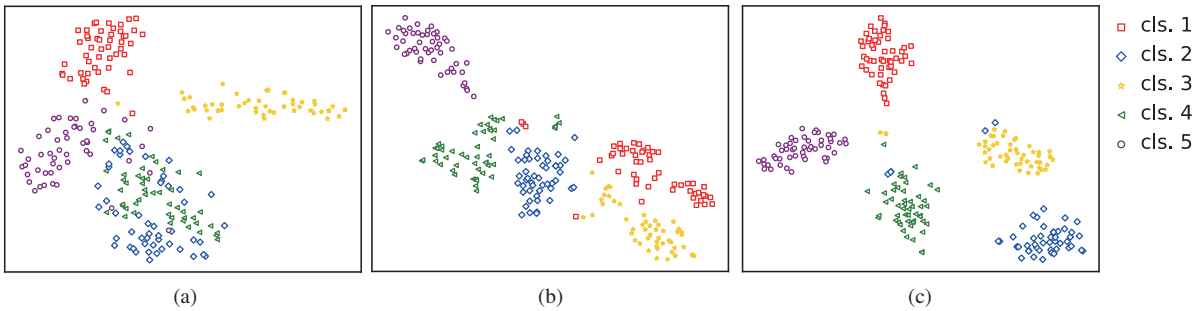

Figure 5 (Color online) Comparison between query instance embeddings trained by three models: (a) BACK without Bi-Att and LM; (b) BACK without LM; (c) BACK. We sampled 50 queries for each class from a 5-way 5-shot classification scenario on FewRel dataset.

guishing from other classes, since the interaction between the query and support instances enhances the representation of the target support instances. We also find that other classes are placed adjacently. The reason is that the bi-directional attention module fuses the query vector representation into each support instance (see (10) for details), which integrates the similar query information into cls. 1–4. (2) With the supervision of large margin loss (Figures 4(b) and (c)), BACK generates more differentiated representations for instances, and different classes become more separable.

Figure 5 shows the query embeddings of a 5-way 5-shot scenario. We read from the comparison that (1) the bi-directional attention helps query instance embeddings of different classes more separable (Figures 5(a) and (b)), since the fusion of support-aware vector representation improves the quality of final query embeddings (see (9) for details). (2) Although the employment of large margin loss aims at supervising the spacial distribution of support instances, its addition also enables the query embeddings more separable in the embedding space. It is because the query instance and the prototype, where each prototype is generated by its corresponding support instances, are interdependent in the metric function (see (16) for details).

Bi-directional attention visualization. To validate whether the bi-directional attention module can find the similar parts in the query and the support instance, we selected two cases from the 5-way 1-shot classification on HuffPost dataset. We used the heat map to visualize the similarity matrix between the query and the support instance (see (4) for details). The example on the left side of Figure 6 shows the similarity matrix between two news titles, which belongs to the same class 'Education'. From the heat map, we can see that the same part 'protesting gun violence' in the query and the support instance is mutually emphasized. The case on the right side of Figure 6 shows the similarity matrix of two 'College' news titles. From the results, we can observe that the related words 'students', 'college', and 'graduation' get higher similarity scores. These clues are quite important for our model, since the final predication is largely based on the similarity between the query and the support instances.

6 Conclusion

In this paper, we propose a meta-learning model for few-shot text classification, that aims to (1) extract

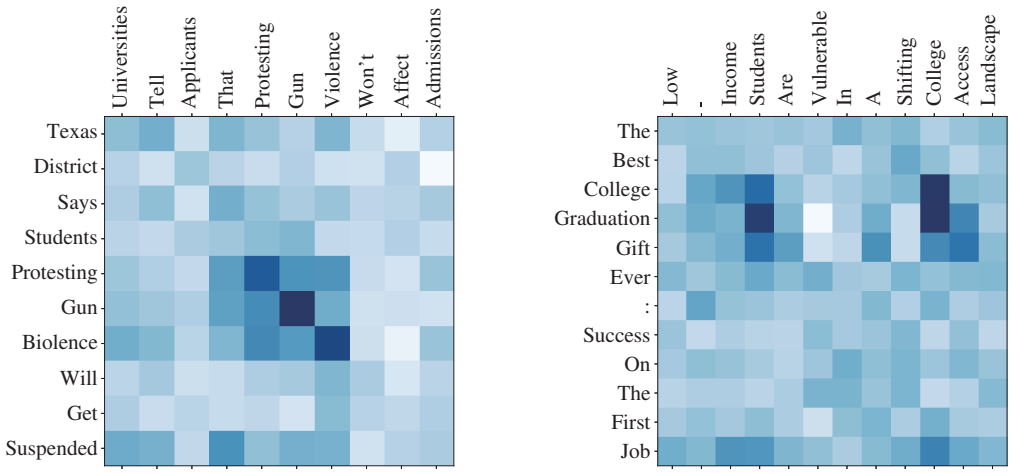

Figure 6 (Color online) Two visualized examples of the similarity matrix between the query and the support instance. Owing to the limited space, these two cases are selected from HuffPost dataset, where the text is relatively short. In the heat maps, words in the support instance are located in the horizontal direction, while words in the query are placed vertically. Deeper color signifies the higher similarity score.

the intra-class knowledge by modeling the interaction between the query and support instances; (2) exploit the cross-class knowledge by learning the distance distributions of different classes. Our framework can effectively find the similar parts in the query and support instances, which are good clues for recognizing the class. Besides, the addition of large margin loss leads to more separable sentence embeddings between different classes. The experimental results validate the effectiveness of our designs, and demonstrate the superiority of our proposal over the state-of-the-art competitors. In the future, we will try to extend our framework to zero-shot learning, which is more challenging.

Acknowledgements This work was partially supported by National Natural Science Foundation of China (Grant Nos. 61872446, U19B2024), Natural Science Foundation of Hunan Province (Grant No. 2019JJ20024), and the Science and Technology Innovation Program of Hunan Province (Grant No. 2020RC4046).

References

1 Pang B, Lee L. Opinion mining and sentiment analysis. FNT Inf Retrieval, 2008, 2: 1–135
2 Aggarwal C C, Zhai C. A survey of text classification algorithms. In: Proceedings of Mining Text Data, 2012. 163–222
3 Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification. In: Proceedings of Advances in Neural Information Processing Systems, 2015. 649–657
4 Kim Y. Convolutional neural networks for sentence classification. 2014. ArXiv: 1408.5882
5 Yao L, Mao C, Luo Y. Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2019. 7370–7377
6 Li F-F, Fergus R, Perona P. One-shot learning of object categories. IEEE Trans Pattern Anal Machine Intell, 2006, 28: 594–611
7 Sung F, Yang Y, Zhang L, et al. Learning to compare: relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 1199–1208
8 Munkhdalai T, Yu H. Meta networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017. 2554–2563
9 Snell J, Swersky K, Zemel R S. Prototypical networks for few-shot learning. In: Proceedings of Advances in Neural Information Processing Systems, Long Beach, 2017. 4077–4087
10 Vinyals O, Blundell C, Lillicrap T, et al. Matching networks for one shot learning. In: Proceedings of Advances in Neural Information Processing Systems, Barcelona, 2016. 3630–3638
11 Koch G, Zemel R, Salakhutdinov R. Siamese neural networks for one-shot image recognition. In: Proceedings of ICML Deep Learning Workshop, 2015
12 Yu M, Guo X, Yi J, et al. Diverse few-shot text classification with multiple metrics. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, 2018. 1206–1215
13 Han X, Zhu H, Yu P, et al. Fewrel: a large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, 2018. 4803–4809
14 Gao T, Han X, Liu Z, et al. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019. 6407–6414
15 Ye Z, Ling Z. Multi-level matching and aggregation network for few-shot relation classification. In: Proceedings of the 57th Conference of the Association for Computational Linguistics, Florence, 2019. 2872–2881
16 Bao Y, Wu M, Chang S, et al. Few-shot text classification with distributional signatures. In: Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, 2020
17 Seo M, Kembhavi A, Farhadi A, et al. Bidirectional attention flow for machine comprehension. 2016. ArXiv: 1611.01603
18 Yang Z, Yang D, Dyer C, et al. Hierarchical attention networks for document classification. In: Proceedings of the 2016

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016. 1480–1489
19 Tao H, Tong S, Zhao H, et al. A radical-aware attention-based model for chinese text classification. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019
20 Miller E G, Matsakis N E, Viola P A. Learning from one example through shared densities on transforms. In: Proceedings of Conference on Computer Vision and Pattern Recognition, Hilton Head, 2000. 1464–1471
21 Santoro A, Bartunov S, Botvinick M, et al. Meta-learning with memory-augmented neural networks. In: Proceedings of the 33rd International Conference on Machine Learning, New York City, 2016. 1842–1850
22 Mishra N, Rohaninejad M, Chen X, et al. A simple neural attentive meta-learner. In: Proceedings of the 6th International Conference on Learning Representations, Vancouver, 2018
23 Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning, Sydney, 2017. 1126–1135
24 Al-Shedivat M, Bansal T, Burda Y, et al. Continuous adaptation via meta-learning in nonstationary and competitive environments. In: Proceedings of the 6th International Conference on Learning Representations, Vancouver, 2018
25 Bertinetto L, Henriques J F, Torr P H S, et al. Meta-learning with differentiable closed-form solvers. In: Proceedings of the 7th International Conference on Learning Representations, New Orleans, 2019
26 Lam W, Lai K Y. A meta-learning approach for text categorization. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001. 303–309
27 Jiang X, Havaei M, Chartrand G, et al. Attentive task-agnostic meta-learning for few-shot text classification. In: Proceedings of International Conference on Learning Representations, 2019
28 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Advances in Neural Information Processing Systems, Long Beach, 2017. 5998–6008
29 Ji G, Liu K, He S, et al. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, 2017. 3060–3066
30 Wu L, Zhang H, Yang Y, et al. Dynamic prototype selection by fusing attention mechanism for few-shot relation classification. In: Proceedings of the 12th Asian Conference Intelligent Information and Database Systems, Phuket, 2020. 431–441
31 Pennington J, Socher R, Manning C D. Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, 2014. 1532–1543
32 Devlin J, Chang M W, Lee K, et al. Bert: pre-training of deep bidirectional transformers for language understanding. 2018. ArXiv: 1810.04805
33 Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput, 1997, 9: 1735–1780
34 Lewis D. Reuters-21578 text categorization test collection, distribution 1.0. 1997. http://www.research/.att.com
35 Chen W, Liu Y, Kira Z, et al. A closer look at few-shot classification. In: Proceedings of the 7th International Conference on Learning Representations, New Orleans, 2019
36 Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res, 2014, 15: 1929–1958
37 Maaten L V D, Hinton G. Visualizing data using t-SNE. J Mach Learn Res, 2008, 9: 2579–2605