MGIMN: Multi-Grained Interactive Matching Network for Few-shot Text Classification

Anonymous ACL submission

Abstract

Text classification struggles to generalize to unseen classes with very few labeled text instances per class. In such a few-shot learning (FSL) setting, metric-based meta-learning approaches have shown promising results. Previous studies mainly aim to derive a prototype representation for each class. However, they neglect that it is challenging-yet-unnecessary to construct a compact representation which expresses the entire meaning for each class. They also ignore the importance to capture the inter-dependency between query and the support set for few-shot text classification. To deal with these issues, we propose a meta-learning based method **MGIMN** which performs instance-wise comparison followed by aggregation to generate class-wise matching vectors instead of prototype learning. The key of instance-wise comparison is the interactive matching within the class-specific context and episode-specific context. Extensive experiments demonstrate that the proposed method significantly outperforms the existing state-of-the-art approaches, under both the standard FSL and generalized FSL settings.

1 Introduction

Few-shot text classification has attracted considerable attention in order to bridge the gap between machine intelligence and human intelligence (Gao et al., 2019; Yin, 2020; Brown et al., 2020; Bragg et al., 2021; Liu et al., 2021). Many efforts are devoted towards different goals like generalization to new classes (Gao et al., 2019; Ye and Ling, 2019; Nguyen et al., 2020; Bragg et al., 2021), adaptation to new domains and tasks (Bansal et al., 2020; Brown et al., 2020; Schick and Schütze, 2020; Gao et al., 2020; Bragg et al., 2021). The ability to generalize to unseen classes is critical to deal with the growing long-tailed categories, which is common for intention classification (Geng et al., 2019; Krone et al., 2020) and relation classification (Han et al., 2018; Gao et al., 2019).

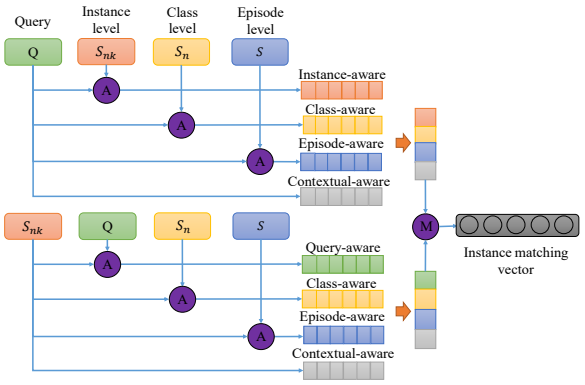

Figure 1: The example of multi-grained few-shot text classification. The "S_{nk}, S_n" and "S" represent the instance level, class level and episode level interaction respectively. While "A" in the purple circle denotes alignment between query and instances. The "'M" stands for matching operation.

To prevent overfitting to few-shot new classes and avoid retraining the model when the class space changes, metric-based meta learning has become the major framework with significant results (Yin, 2020; Bansal et al., 2020). The core idea is that episode sampling is employed in meta-training phase to learn the relationship between query and candidate classes (Bragg et al., 2021). A key challenge is inducing class-wise representations from support sets because nature language expressions are diverse (Gao et al., 2019) and highly informative lexical features are unshared across episodes (Bao et al., 2019).

Under metric-based meta learning framework, many valuable backbone networks have emerged. Snell et al. (2017) presented prototypical network that learns a metric space in which classification can be performed by computing distances to prototype representations of each class. Gao et al. (2019) proposed hybrid attention-based prototypical networks for the problem of noisy few-shot relation classfication. Geng et al. (2019) leveraged the dynamic routing algorithm in meta learning to

learn a generalized class-wise representation. Ye and Ling (2019) obtained the final class prototype for each support set by attentive aggregation over the representations of support instances, where the weights are calculated using the query instance. Bao et al. (2019) mapped distributional signatures of words into attention scores, which are then used to weight the lexical representations of words. All these previous works first obtain class-wise representations and then perform class-wise comparisons. However, it is challenging-yet-unnecessary to construct a compact representation which expresses the entire meaning for each class (Parikh et al., 2016). In text matching research, compare-aggregate methods which perform token-level comparisons followed by sentence-level aggregation has already been successful (Parikh et al., 2016; Tay et al., 2017; Yang et al., 2019). Besides backbone networks,there are still some work that can be further combined. Luo et al. (2021) utilized class-label information for extracting more discriminative feature representation of the input text. Bansal et al. (2020) introduced a self-supervised approach to generate a large, rich, meta-learning task distribution from unlabeled text.

In this paper we propose **M**ulti-**g**rained **I**nteractive **M**atching **N**etwork, a backbone network for few-shot text classification. The core difference between us with previous efforts is that our work performs instance-level comparisons followed by class-wise aggregation. Specifically, first, all text sequences including query and all support instances are encoded to contextual representations. Second, as depicted in Figure 1, we design a novel multi-grained interactive matching mechanism to perform instance-wise comparisons which capture the inter-dependency between query and all support instances. Third, class-wise aggregate layer obtains class-wise matching vector between query and each class. Finally, a prediction layer predicts final results. In contrast to standard few-shot learning setting, generalized few-shot learning setting is a more challenging-yet-realistic setting where seen classes and new classes are co-existent (Nguyen et al., 2020; Li et al., 2020a). In such a setting, we analyze the relationship between inference speed and the number of classes, and verify the necessity of retrieval, which is ignored by previous studies. Our contributions are listed as follows:

- We propose **MGIMN** which is more concerned with matching features than semantic features through multi-grained interactive matching.
- We verify the necessity of retrieval for realistic applications of few-shot text classification when the number of classes grows.
- We conduct extensive experiments and achieve state-of-the-art results under both the standard FSL and generalized FSL settings.

2 Background

2.1 Few-Shot Learning

The goal of few-shot learning is to learn a classifier $G(S, q) \rightarrow y$ which map the query q to the label y given the support set S. Few-shot learning is significantly different from traditional machine learning. In traditional machine learning tasks, a batch of training data are sampled from the training dataset in each training step. Distinct from this, few-shot learning models usually adopt episode training strategy which means we sample training data from the training dataset to construct a meta task in each training step.

Specifically, we randomly sample N unique classes, then sample R samples as query set Q and K samples for each class as support set S where $Q \bigcap S \in \emptyset$. For training, given the support set $S = \{S_n^k; i = 1, .., N, j = 1, .., K\}$, and query set $Q = \{(q_i, y_i); i = 1, 2, .., R, y_i \in 1, .., N\}$ which has R samples in each training step, the training objective is to minimize the following loss function:

$$J = -\frac{1}{R} \sum_{(q,y) \in Q} log(P(y|(S, q))). \quad (1)$$

For evaluation, as we described in introduction section, there are two settings. In standard FSL setting, we do N-way K-shot sampling on the classes for validation and test (which are unseen during training) to construct episodes for validation and test. In generalized FSL setting, we reformulate the task as C-way K-shot classification where C is the count of all classes for training, validation and test, and usually is far greater than N. In this setting, we construct episodes by sampling on all classes for test.

2.2 Matching Network

Matching Network (Vinyals et al., 2016) is a typical method for few-shot learning, which conducts

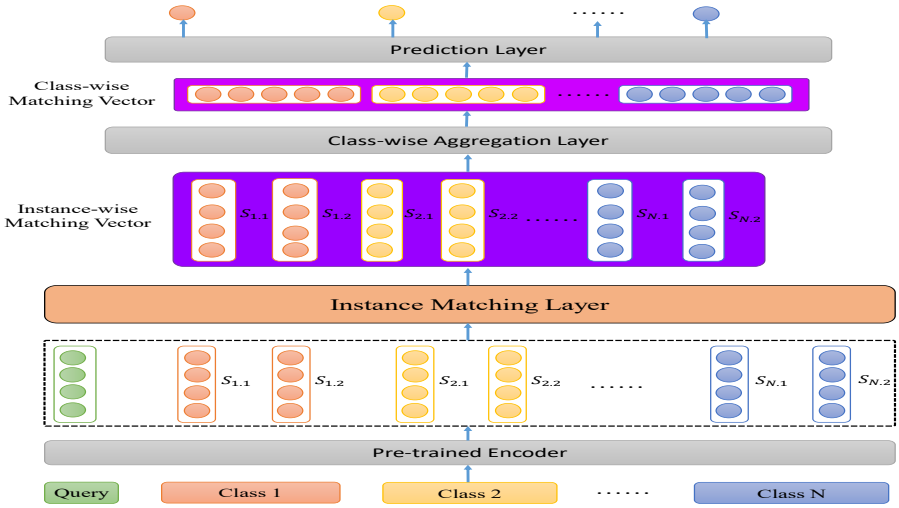

Figure 2: The main architecture of multi-grained few-shot text classification model. The details of "Multi-grained Interaction Layer" totally same as depicted as Figure 1.

few-shot classification by calculating the similarity between the query and the support instances. Specifically, for the query instance q and each support instance $S_n^k \in S$, we calculate the cosine similarity between q and S_n^k:

$$sim(q, S_n^k) = \frac{q \cdot S_n^k}{||q||\ ||S_n^k||}. \quad (2)$$

Then, we use the attention mechanism to calculate the probability distribution of the label y of the query q:

$$P(y|S,q) = \frac{\sum_{k=1}^{K} \exp(sim(q, S_y^k))}{\sum_{n=1}^{N}\sum_{k=1}^{K} \exp(sim(q, S_n^k))}. \quad (3)$$

Finally, we calculate the label y of the query instance q by finding the class with the highest prediction score:

$$y = \arg\max_n P(y = n|S, q). \quad (4)$$

2.3 Text Classfication

Text classification is an important task in the research field of NLP. In previous works, different model architectures, including RNN (Zhou et al., 2016) and CNN (Kim, 2014) are used for text classification. With the development of pretrained language models, text classification based on BERT (Devlin et al., 2019) or other pretrained language models has become the mainstream method for text classification. Specifically, we use a Transformer encoder to learn the representation of the input sentence and then conduct text classification based on the representation. Suppose the original input sentence is $\mathbf{x}$, and we add a special token [CLS] before the sentence $\mathbf{x}$. The model calculates the final hidden state of [CLS] token to obtain the sentence representation:

$$\mathbf{h}_{\text{CLS}} = Transformer([\text{CLS}], \mathbf{x}; \theta), \quad (5)$$

where θ is the parameters of the model.

Then, the model calculates the probability distribution of the class y of the input sentence $\mathbf{x}$:

$$P(y|\mathbf{x}; \theta) = softmax(\mathbf{W}_{\text{softmax}}\mathbf{h}_{\text{CLS}}), \quad (6)$$

where $\mathbf{W}_{\text{softmax}}$ is the parameters of the softmax layer.

3 Method

As illustrated in the figure 2, MGIMN consists of four modules: Encoder Layer, Instance-wise Matching Layer, Class-wise Aggregation Layer and Prediction Layer.

3.1 Encoder Layer

We employ transformer encoder from pre-trained BERT as encoder layer. Same as original work, we add a special token [CLS] before original text. Then the encoder layer takes an input of a token sequence and outputs the sequence representation. Instead of using final hidden state of [CLS] token as sentence-wise representation, we adopt final hidden states of the rest tokens for further fine-grained instance-wise matching.

We denote $x = \{w_1, w_2, ..., w_l\}$ as a token sequence. Encoder Layer outputs the token-wise representation $h = \{h_1, h_2, ...,_l\}$, where l denotes length of the token sequence. Query and each support instance are encoded individually. We denote q as encoded result of query, s_{nk} as encoded result of the k_{th} support instance of n_{th} class.

3.2 Instance-wise Matching Layer

This is the core of our model. Instance-wise matching vectors are obtained by comparing query with each support instance.

3.2.1 Bidirectional Alignment

Following previous works(Parikh et al., 2016; Yang et al., 2019) in text matching , we use bidirectional alignment to capture inter-dependency between two text sequences.

$$\widehat{a}, \widehat{b} = BiAlign(a, b) \tag{7}$$

Where a and b are the token-wise sequence representations and $BiAlign$ denotes the bidirectional alignment function defined as follows:

$$e_{ij} = \mathbf{F}(a_i)^{\mathrm{T}}\mathbf{F}(b_j) \tag{8}$$

$$\widehat{a_i} = \sum_{j=1}^{l_b} \frac{exp(e_{ij})}{\sum_{k=1}^{l_b} exp(e_{ik})} b_j \tag{9}$$

$$\widehat{b_j} = \sum_{i=1}^{l_a} \frac{exp(e_{ij})}{\sum_{k=1}^{l_a} exp(e_{kj})} a_i \tag{10}$$

Where a_i, b_j denotes representation of i_{th} and j_{th} token of a and b. $\widehat{a_i}$ and $\widehat{b_j}$ denotes the aligned representations.

3.2.2 Multi-grained Interactive Matching

In few-shot text classification, judging whether query and each support instance belong to the same category cannot be separated from class-specific context and episode-specific context. There are three components including alignment, fusion and comparison.

For alignment, besides local alignment between query and each support instance, we also consider their alignments with global context information. We denote $S_n = concat(\{s_n\}_{j=1}^{K})$ as class-specific context, $S = concat(\{S_i\}_{i=1}^{N})$ as episode-specific context. The multi-grained alignments for query and each support instance are performed as follows:

$$\begin{aligned} q_{nk}^{'}, s_{nk}^{'} &= BiAlign(q, s_{nk}) \\ q_{n}^{''}, _ &= BiAlign(q, S_n) \\ s_{nk}^{''}, _ &= BiAlign(s_{nk}, S_n) \\ q^{'''}, _ &= BiAlign(q, S) \\ s_{nk}^{'''}, _ &= BiAlign(s_{nk}, S) \end{aligned} \tag{11}$$

where $q_{nk}^{'}$, $q_{n}^{''}$ and $q^{'''}$ denote instance-aware, class-aware and episode-aware query representations respectively, $s_{nk}^{'}$, $s_{nk}^{''}$ and $s_{nk}^{'''}$ denote query-aware, class-aware and episode-aware support instance representations respectively.

For fusion, we fuse original representation and three kinds of aligned representations together as follows:

$$\begin{aligned} \boldsymbol{q_{nk}^{'}} &= H_1(q; q_{nk}^{'}; \left|q - q_{nk}^{'}\right|; q \odot q_{nk}^{'}) \\ \boldsymbol{q_{nk}^{''}} &= H_2(q; q_{n}^{''}; \left|q - q_{n}^{''}\right|; q \odot q_{n}^{''}) \\ \boldsymbol{q_{nk}^{'''}} &= H_3(q; q^{'''}; \left|q - q^{'''}\right|; q \odot q^{'''}) \\ \boldsymbol{s_{nk}^{'}} &= H_1(s_{nk}; s_{nk}^{'}; \left|s_{nk} - s_{nk}^{'}\right|; s_{nk} \odot s_{nk}^{'}) \\ \boldsymbol{s_{nk}^{''}} &= H_2(s_{nk}; s_{nk}^{''}; \left|s_{nk} - s_{nk}^{''}\right|; s_{nk} \odot s_{nk}^{''}) \\ \boldsymbol{s_{nk}^{'''}} &= H_3(s_{nk}; s_{nk}^{'''}; \left|s_{nk} - s_{nk}^{'''}\right|; s_{nk} \odot s_{nk}^{'''}) \end{aligned} \tag{12}$$

$$\begin{aligned} \boldsymbol{q_{nk}} &= H(\boldsymbol{q_{nk}^{'}}; \boldsymbol{q_{nk}^{''}}; \boldsymbol{q_{nk}^{'''}}) \\ \boldsymbol{s_{nk}} &= H(\boldsymbol{s_{nk}^{'}}; \boldsymbol{s_{nk}^{''}}; \boldsymbol{s_{nk}^{'''}}) \end{aligned} \tag{13}$$

Where H_1, H_2, H_3, H are single-layer feed forward networks with independent parameters, $\odot$ denotes the element-wise multiplication operation, ; denotes concatenation operation.

For comparison, the instance-wise matching vector is computed as follows:

$$\begin{aligned} q_{nk} &= [max(\{\boldsymbol{q_{nk}}\}); avg(\{\boldsymbol{q_{nk}}\})] \\ s_{nk} &= [max(\{\boldsymbol{s_{nk}}\}); avg(\{\boldsymbol{s_{nk}}\})] \\ m_{nk} &= G(\vec{q_{nk}}; \vec{s_{nk}}; |\vec{q_{nk}} - \vec{s_{nk}}|; \vec{q_{nk}} \odot \vec{s_{nk}}) \end{aligned} \tag{14}$$

where $\boldsymbol{q_{nk}}$ is a $l_q \times D$ matrix, $\boldsymbol{s_{nk}}$ is a $l_s \times D$ matrix, $\vec{q_{nk}}$ and $\vec{s_{nk}}$ are vectors with shape $(1 \times D)$, G is single-layer feed forward networks.

3.3 Class Aggregation Layer

This layer aggregate instance-wise matching vectors into class-wise matching vectors for final pre-

diction.

$$\vec{c_n} = [max(\{\vec{m_{nk}}\}_{k=1}^{K}); avg(\{\vec{m_{nk}}\}_{k=1}^{K})] \quad (15)$$

Where $\vec{c_n}$ denotes the final matching vector of n_{th} class, ";" denotes the concatenation operation.

3.4 Prediction Layer

Finally, prediction layer, which is two-layer fully connected network with the output size is 1, is applied to the matching vector $\vec{c_n}$ and output final predicted result.

$$logit_n = MLP(\vec{c_n}), n = 1, .., N \quad (16)$$

4 Experiments

4.1 Setup

4.1.1 Data Preparation

We evaluate our approach on five diverse datasets: OOS (Larson et al., 2019), Liu (Liu et al., 2019), FaqIr (Karan and Šnajder, 2016), Amzn (Yury., 2020) and Huffpost (Misra., 2018). Among them, OOS, Liu and FaqIr datasets are all intent classification datasets. Amzn dataset is designed for fine-grained classification of product reviews. Huffpost dataset is constructed to identify the types of news based on headlines and short descriptions. The detailed dataset statistics is list in Table 1. In standard FSL setting, we sample N unique classes and K samples per class as support set, and sample R samples per class as query set in each training episode. We conduct two groups of experiments using $N = [5, 10], K = 5, R = 5$. In evaluation phase, we sample 500 episodes and report the averaged accuracies. In generalized FSL setting(GFSL for short), we train the model with episode sampling of 5-way 5-shot. And then we evaluate the model performance with C-way K-shot.

For all experiments, we divide all datasets for 5 times using different random seeds, just like the way of cross validation, to remove the impact of dataset division. And we conduct 3 runs experiments for each model by using different random seeds for model initialization. The final results are reported by averaging $5 \times 3 = 15$ runs. For fair comparison, all models implemented in this paper adopt BERT-Tiny as encoder layer which is a 2-layer 128-hidden 2-heads version of BERT. Its parameters are initialized by the pretrained model published by Google and finetuned during training time. Besides encoder layer, all parameters of other layers are randomly initialized. The hidden size is set to 128 in all experiments. We use Adam optimizer. The learning rate is tuned from $1e^{-5}$ to $1e^{-4}$ on validation dataset. Dropout with a rate of 0.1 is applied before each fully-connected layer. The feed-forward networks described in section 3 , such as F, H_1, H_2, H_3, H, G are all single fully-connected layers. The prediction layer is a two-layer fully-connected layer.

4.1.2 Baselines

We compare MGIMN model with previous approaches: Prototypical network, Matching network, Induction network, Proto-HATT and MLMAN with two evaluation metrics mentioned above. Note that we re-implement all models with the same pretrained encoder for fairly comparison.

- **Prototypical Network (Proto)** (Snell et al., 2017) was initially designed for image classification and has been used to solve the few-shot text classification problem in recent studies.

- **Matching Network (Matching)** (Vinyals et al., 2016) computes the similarity between query and each support instance, then averages them as final prediction score.

- **Induction network (Induction)** (Geng et al., 2019) proposed an induction module to induce the prototype by using dynamic routing.

- **Proto-HATT** (Gao et al., 2019) was designed to address the problem of noisy and diverse by leveraging instance-level attention and feature-level attention.

- **MLMAN** (Ye and Ling, 2019), a variant of prototypical network, encodes query and support in an interactive way.

4.2 Main results

Overall Performance The main results are displayed in Table 2, 3 and 4. We report the averaged scores over 15 runs (different seen-unseen class splits and random seeds as introduced in section 4.1.1) for each dataset and model. Our approach consistently outperforms all baselines on the five diverse datasets, especially in more challenging generalized FSL setting: the improvements on Huffpost and Amzn datasets are 2.83% and 2.75% respectively.

Datasets	Standard FSL Setting		Generalized FSL Setting						
	#sentences	$C_{tr}/C_{val}/C_{test}$	C	K	#sc	#uc	$\#D_{tr}$	$\#D_{val}$	$\#D_{test}$
OOS	22500	50/50/50	150	5	50	100	7000	1250	1250
Liu	25478	18/18/18	54	5	18	36	8312	450	450
Amzn	3057	106/106/106	318	5	106	212	1043	530	530
Huffpost	41000	14/13/14	41	5	14	27	13860	340	340
FaqIr	1233	17/16/17	50	5	17	33	309	381	381

Table 1: The detailed dataset statistics. In standard FSL setting, we split all classes into train/valid/test according to the ratio of 1:1:1. In generalized FSL setting, we reformulate task as a C-way K-shot classification in which only subset of classes are seen in training phase.

Methods	OOS			Liu			FaqIr		
	5-way	10-way	GFSL	5-way	10-way	GFSL	5-way	10-way	GFSL
Proto	92.20	87.91	61.94	82.46	73.23	47.66	89.83	81.56	60.78
Matching	89.78	84.41	58.34	78.25	67.45	41.95	86.74	78.77	53.85
Induction	80.44	70.92	34.00	65.58	51.56	24.73	71.62	56.99	20.10
Proto-HATT	92.84	89.11	65.52	82.38	75.29	51.27	85.01	76.17	62.62
MLMAN	95.99	93.41	74.39	87.39	79.82	57.24	94.77	89.49	74.42
MGIMN(ours)	**96.36**	**94.00**	76.23	**87.84**	**80.60**	57.66	**95.14**	**90.69**	**75.81**

Table 2: Experiment results of standard FSL(5-way 5-shot and 10-way 5-shot) and generalized FSL settings on intent classification datasets(OOS,Liu and FaqIr datasets)

Methods	Amzn		
	5-way	10-way	GFSL
Proto	78.40	69.02	41.03
Matching	75.73	64.17	38.34
Induction	64.02	50.12	20.09
Proto-HATT	78.05	69.00	41.81
MLMAN	85.64	79.39	46.71
MGIMN(ours)	**85.96**	**80.07**	**49.46**

Table 3: Experiment results of standard FSL (5-way 5-shot and 10-way 5-shot) and generalized FSL settings on Amzn datasets

Methods	Huffpost		
	5-way	10-way	GFSL
Proto	51.57	36.74	16.47
Matching	49.77	34.28	14.18
Induction	44.69	29.35	10.40
Proto-HATT	51.23	36.65	16.06
MLMAN	52.76	38.22	16.78
MGIMN(ours)	**54.98**	**40.12**	**19.61**

Table 4: Experiment results of standard FSL (5-way 5-shot and 10-way 5-shot) and generalized FSL settings on Huffpost datasets

Generalized FSL In most studies of few-shot text classification (Bao et al., 2019; Gao et al., 2019), N-way K-shot accuracy is the standard evaluation metric. There are two problems:(1)The metric is not challenging, usually $N = 5$ or $N = 10$, much smaller than C. We also can see that high scores are often reported in some work (Bao et al., 2019; Gao et al., 2019). (2)It cannot reflect the real application scenarios where we usually face the fully classes space(both seen classes and unseen classes). Consequently, the more challenging generalized FSL evaluation metric is employed to focus on the problems. As shown in Table 2, 3 and 4, the performance of generalized FSL evaluation is worse and challenging than standard FSL. It's very meaningful in realistic scenario and can contribute to the further research. It's noteworthy that, comparing with Proto, our proposed approach makes bigger improvement in the challenging generalized FSL metric (GFSL) than the improvements in standard FSL metric(FSL), e.g. OOS dataset: 14.29% of GFSL vs 6.09% of FSL and FaqIr dataset: 15.03% of GFSL vs 9.13% of FSL. It proves that our method is more effective in more challenging scenarios.

Huffpost Dataset Samples of the same class are more diverse and scattered on Huffpost. For instance, "green streets are healthy streets", "the real

heroes of pakistan" and "what next for kurdistan ?" are from the same class:"WORLD NEWS". In this scenario, a single class-wise prototype is difficult to represent the entire class semantic. Interestingly, our approach improves more significantly than other datasets, 2.22% of 5-way 5-shot standard FSL metric,1.9% of 10-way 5-shot standard FSL metric and 2.83% of generalized FSL metric. In our approach, richer matching features gained through interacting from low level with multi-grained interaction, is effective for the dataset with diverse expressed samples.

4.3 Ablation Study

We perform the ablation studies on Liu and Huffpost datasets to verify the effectiveness of different interaction levels and the effectiveness of instance matching vector.

4.3.1 Different Interaction Levels

We respectively take out the single-level interaction layer and see how the specific alignment feature affects the performance. As shown in the Table 5, when taking out the specific interaction layer, the performance decreases in varying degrees, which explains that each alignment layer has positive effect for the performance and can complement for each other. It's noted that class-level interaction layer has greatest impact. The model can pay attention to the whole class context through class-level interaction, which make the model encode more precise class semantic information. It's the key to judge the relationship between query and class.

4.3.2 Instance-wise Matching Vector

We remove all interaction layers in our model, named 'w/o instance & class & episode' in Table 5. Then it's the same with matching network except that the scalar matching score is replaced by instance-wise matching vector. We compare with matching network to verify the effectiveness. As shown in the Table 5, 2 and 4, it performs better than matching network, e.g. Liu dataset:improves 3.49% of 10-way 5-shot FSL score and Huffpost dataset improves 2.30% of GFSL score. Different with scalar comparison in matching network, our approach can make fine-grained instance vector comparisons in dimensional level.

4.4 Number of Classes and Inference Speed

As shown in Table 6, the inference speed increases linearly with the increase of the number of classes, from $315ms/query$ to $1630ms/query$ when c increases from 50 to 318. It's challenged for deploying the model to the online application. To address the problem of inference speed, motivated by the idea of retrieval in traditional search system, we propose the retrieval-then-classify method (RTC-mgimnbert). We construct the class-wise representation by averaging the vector of each support instance, produced by MGIMN encoder and then calculate the similarity between query and class-wise representation. In our experiments, we retrieve top $N = 10$ classes with $K = 5$ instances per class. (2)Stage2-classify: Retrieved support instances by stage1 are classified by MGIMN proposed in this paper. The C-way 5-shot task is reduced to a 10-way 5-shot which can greatly save the inference time and computation cost.

As shown in Table 6, in addition to the generalized FSL metric score, we also report the inference speed (processing time per query) to verify the effectiveness of retrieval-then-classify. We regard the inference speed of retrieval-then-classify is greatly improved, increasing by $5\times$ to $23\times$, with a small amount of loss. Comparing with other retrieval methods (e.g. BM25 and original bert), our approach can further improve the performance.

5 Related Work

5.1 Few-shot Learning

Few-shot learning aims to learn a classifier with only a few labeled training examples, which is similar to human intelligence. Since the aim of few-shot learning is highly attractive, researchers have proposed various few-shot learning methods.

Matching network (Vinyals et al., 2016) encodes query text and all support instances independently, then compute the cosine similarity score between query vector and each support instance vector, finally compute average of all scores for each class. Prototypical network (Snell et al., 2017) encodes query text and support instances independently, then compute class-wise vector by averaging vectors of all support instances in each class, finally compute euclidean distances as final scores. Relation network (Sung et al., 2018) uses a relation module to model relationship between query vector and class-wise vectors instead of using a predefined distance metric like euclidean or the cosine one. Induction network (Geng et al., 2019) uses an induction module that leverages a dynamic routing algorithm to induce and generalize class-wise rep-

Methods	Liu			Huffpost		
	5-way	10-way	GFSL	5-way	10-way	GFSL
MGIMN(ours)	**87.84**	**80.60**	**57.66**	**54.98**	**40.12**	**19.61**
w/o episode	86.22	78.99	56.67	54.14	39.53	18.69
w/o class	84.56	76.89	54.62	54.09	39.10	17.53
w/o instance	87.74	79.93	57.39	53.65	38.86	18.67
w/o instance&class&episode	80.53	70.94	42.54	51.81	37.10	16.48

Table 5: Ablation study results on Liu and Huffpost datasets

Methods	Liu(c=50)		OOS(c=150)		Amzn(c=318)	
	score	speed	score	speed	score	speed
MGIMN-overall	57.66	315	76.23	757	49.46	1630
RTC-BM25	54.97	55	74.80	56	44.76	58
RTC-oribert	52.93	60	70.55	65	31.09	70
RTC-mgimnbert	56.21	60	75.58	65	46.80	70

Table 6: The generalized FSL accuracies(%) and inference speed. Speed is reported by averaging for processing 100 queries and the value is the processing time per query($ms/query$)

resentations. To extend the usage of few-shot learning, Finn et al. (2017) propose a model-agnostic few-shot learning method called MAML, which can be used in different tasks, including machine translation (Gu et al., 2018; Li et al., 2020b) and dialog generation (Qian and Yu, 2019; Huang et al., 2020).

For few-shot text classification, researchers have also proposed various techniques to improve the existing few-shot learning methods. For example, Gao et al. (2019) propose hybrid attention-based prototypical networks that leverage instance-level and feature-level attention to highlight the crucial support point and features. Ye and Ling (2019) encodes the query instance and each support set in an interactive way by considering their matching information at word level.

5.2 Text Matching

Text matching model aims to predict the score of text pair dependent on massive labeled data. In previous of BERT, relate works focus on deriving the matching information between two text sequences based on the matching aggregation framework. It performs matching in low-level and aggregate matching results based on attention mechanism. Many studies are proposed to improve performance. Parikh et al. (2016) proposed a simple attend-compare-aggregate method which performs alignment and aggregate aligned feature with feed-forward networks. Wang et al. (2017) extract fine-grained matching feature with bilateral multi-perspective matching operation. Tay et al. (2017) adopt factorization layers for efficient and expressive compression of alignment vectors into scalar features, which are then used to augment the base word representations. Yang et al. (2019) propose a simple and effective text matching model using richer alignment features. After BERT (Devlin et al., 2019) is proposed, pretrained language models are widely used for various NLP tasks. Thus, various text matching methods using pretrained language models like BERT have been proposed. For example, Reimers and Gurevych (2019) use the sentence embeddings of BERT to conduct text matching, and Gao et al. (2021) use contrastive learning to train text matching models. In this paper, we employ the similar idea of comparing and aggregation and propose a multi-grained interactive matching network to address the problem of few-shot learning problem.

6 Conclusion

We propose **M**ulti-**g**rained **I**nteractive **M**atching **N**etwork(**MGIMN**) for few-shot text classification task. We also present a two-stage method retrieval-then-classify (RTC) to solve the inference performance in realistic scenery. Experiment results demonstrate our approach achieves the state-of-the-art result on all five different kinds of datasets under two evaluation metrics. Moreover, RTC method obviously accelerate the inference speed. In future, we will further extend our method for the task of domain adaptation problem.

References

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020. Self-supervised meta-learning for few-shot natural language classification tasks. *arXiv preprint arXiv:2009.08445*.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2019. Few-shot text classification with distributional signatures. *arXiv preprint arXiv:1908.06039*.

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. Flex: Unifying evaluation for few-shot nlp. *arXiv preprint arXiv:2107.07170*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6407–6414.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. *arXiv preprint arXiv:1902.10482*.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. *arXiv preprint arXiv:1810.10147*.

Yi Huang, Junlan Feng, Shuo Ma, Xiaoyu Du, and Xiaoting Wu. 2020. Towards low-resource semi-supervised dialogue generation with meta-learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4123–4128.

Mladen Karan and Jan Šnajder. 2016. Faqir–a frequently asked questions retrieval test collection. In *International Conference on Text, Speech, and Dialogue*, pages 74–81. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Jason Krone, Yi Zhang, and Mona Diab. 2020. Learning to classify intents and slot labels given a handful of examples. *arXiv preprint arXiv:2004.10793*.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. 2020a. Boosting few-shot learning with adaptive margin loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12576–12584.

Rumeng Li, Xun Wang, and Hong Yu. 2020b. Metamt, a meta learning method leveraging multiple domain data for low resource machine translation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566*.

Qiaoyang Luo, Lingqiao Liu, Yuhao Lin, and Wei Zhang. 2021. Don't miss the labels: Label-semantic augmented meta-learner for few-shot text classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2773–2782.

Rishabh Misra. 2018. Huffpost news category dataset.

Hoang Nguyen, Chenwei Zhang, Congying Xia, and Philip S Yu. 2020. Dynamic semantic matching and aggregation network for few-shot intent detection. *arXiv preprint arXiv:2010.02481*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 2639–2649.

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

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen. 2019. Simple and effective text matching with richer alignment features. *arXiv preprint arXiv:1908.00300*.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Multi-level matching and aggregation network for few-shot relation classification. *arXiv preprint arXiv:1906.06678*.

Wenpeng Yin. 2020. Meta-learning for few-shot natural language processing: A survey. *arXiv preprint arXiv:2007.09604*.

Kashnitsky Yury. 2020. Hierarchical text classification of amazon product reviews.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495.