Multi-Level Matching and Aggregation Network for Few-Shot Relation Classification

Zhi-Xiu Ye, Zhen-Hua Ling[*]
National Engineering Laboratory for Speech and Language Information Processing,
University of Science and Technology of China
zxye@mail.ustc.edu.cn, zhling@ustc.edu.cn

Abstract

This paper presents a multi-level matching and aggregation network (MLMAN) for few-shot relation classification. Previous studies on this topic adopt prototypical networks, which calculate the embedding vector of a query instance and the prototype vector of each support set independently. In contrast, our proposed MLMAN model encodes the query instance and each support set in an interactive way by considering their matching information at both local and instance levels. The final class prototype for each support set is obtained by attentive aggregation over the representations of its support instances, where the weights are calculated using the query instance. Experimental results demonstrate the effectiveness of our proposed methods, which achieve a new state-of-the-art performance on the FewRel dataset[1].

Support Set
class A: *mother* instance #1 The Queen Consort [Jetsun Pema]$_{e2}$ gave birth to a son on 5 February 2016 , [Jigme Namgyel Wangchuck]$_{e1}$. instance #2 He married the American actress [Cindy Robbins]$_{e2}$ and was stepfather to her daughter , [Kimberly Beck]$_{e1}$. instance #3 Edgar married actress [Moyna Macgill]$_{e2}$ and became the father of [Angela Lansbury]$_{e1}$. instance #4 In 1845 , [Cemile Sultan]$_{e1}$'s mother , Empress [Dzdidil Kadn]$_{e2}$, died. instance #5 Bo 's wife [Gu Kailai]$_{e2}$ traveled with their son [Bo Guagua]$_{e1}$ to Britain.
class B: *member_of* ...
class C: *father* ...
class D: *sport* ...
class E: *voice_type* ...
Query Instance
He was married to [Eva Funck]$_{e2}$ and they have a son [Gustav]$_{e1}$.

Table 1: A data example of 5-way-5-shot relation classification in FewRel development set. The correct relation class for the query instance is class A: *mother*. The instances for other relation classes are omitted for saving space.

1 Introduction

Relation classification (RC) is a fundamental task in natural language processing (NLP), which aims to identify the semantic relation between two entities in text. For example, the instance "[London]$_{e1}$ is the capital of [the UK]$_{e2}$" expresses the relation *capital_of* between the two entities **London** and **the UK**.

Some conventional relation classification methods (Bethard and Martin, 2007; Zelenko et al., 2002) adopted supervised training and suffered from the lack of large-scale manually labeled data. To address this issue, the distant supervision method (Mintz et al., 2009) was proposed which annotated training data by heuristically aligning knowledge bases (KBs) and texts. However, the long-tail problem in KBs (Xiong et al., 2018; Han et al., 2018) still exists and makes it hard to classify the relations with very few training samples.

This paper focuses on the few-shot relation classification task, which was designed to address the long-tail problem. In this task, only few (e.g., 1 or 5) support instances are given for each relation, as shown by an example in Table 1.

The few-shot learning problem has been studied extensively in computer vision (CV) field. Some methods adopt meta-learning architectures (Santoro et al., 2016; Ravi and Larochelle, 2016; Finn et al., 2017; Munkhdalai and Yu, 2017), which learn fast-learning abilities from previous experiences (e.g., training set) and then rapidly gen-

[*] Corresponding author: Zhen-Hua Ling.

[1] The code is available at https://github.com/ZhixiuYe/MLMAN.

eralize to new concepts (e.g., test set). Some other methods use metric learning based networks (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017), which learn the distance distributions among classes. A simple and effective metric-based few-shot learning method is prototypical network (Snell et al., 2017). In a prototype network, query and support instances are encoded into an embedding space independently. Then, a prototype vector for each class candidate is derived as the mean of its support instances in the embedding space. Finally, classification is performed by calculating the distances between the embedding vector of the query and all class prototypes. This prototype network method has also been applied to few-shot relation classification recently (Han et al., 2018).

This paper proposes a multi-level matching and aggregation network (MLMAN) for few-shot relation classification. Different from prototypical networks, which represent support sets without dependency on query instances, our proposed MLMAN model encodes each query instance and each support set in an interactive way by considering their matching information at both local and instance levels. At local level, the local context representations of a query instance and a support set are softly matched toward each other following the sentence matching framework (Chen et al., 2017). Then, the matched local representations are aggregated into an embedding vector for each query and each support instance using max and average pooling. At instance level, the matching degree between the query instance and each of the support instances is calculated via a multi-layer perceptron (MLP). Taking the matching degrees as weights, the instances in a support set are aggregated to form the class prototype for final classification. All these matching and aggregation layers in the MLMAN model are estimated jointly using training data. Since the representations of the support instances in each class are expected to be close with each other, an auxiliary loss function is further designed to measure the inconsistency among all support representations in each class.

In summary, our contributions in this paper are three-fold. First, a multi-level matching and aggregation network is proposed to encode query instances and class prototypes in an interactive fashion. Second, an auxiliary loss function measuring the consistency among support instances is designed. Third, our method achieves a new state-of-the-art performance on FewRel, a public few-shot relation classification dataset.

2 Related Work

2.1 Relation Classification

Relation classification is to identify the semantic relation between two entities in one sentence. In recently years, neural networks have been widely applied to deal with this task. Zeng et al. (2014) employed position features and convolutional neural networks (CNNs) to capture the structure and contextual information respectively. Then, a max pooling operation was adopted to determine the most useful features. Wang et al. (2016) proposed multi-level attention CNNs, which captured both entity-specific attention and relation-specific pooling attention in order to better discern patterns in heterogeneous contexts. Zhou et al. (2016) proposed attention-based bidirectional long short-term memory networks (AttBLSTMs) to capture the most important semantic information in a sentence. All of these methods require a large amount of training data and can't quickly adapt to a new class that has never been seen.

2.2 Metric Based Few-Shot Learning

In few-shot learning paradigm, a classifier is required to generalize to new classes with only a small number of training samples. The metric based approach aims to learn a set of projection functions that take support and query samples from the target problem and classify them in a feed forward manner. This approach has lower complexity and is easier for implementation than meta-learner based approach (Ravi and Larochelle, 2016; Finn et al., 2017; Santoro et al., 2016; Munkhdalai and Yu, 2017).

Some metric based few-shot learning methods have been developed for computer vision (CV) tasks, and all these methods encoded each support or query image to a vector independently for classification. Koch et al. (2015) proposed a method for learning siamese neural networks, which employed an unique structure to encode both support and query samples respectively and one more layer computing the induced distance metric between the pair. Vinyals et al. (2016) proposed to learn a matching network augmented with attention and external memories. And also, an episode-based training procedure was proposed, which was based on a principle that test and training conditions must match and has been adopted by many following studies. Snell et al. (2017) proposed prototypical networks that learn a metric space in which classification can be performed by computing distances to prototype representations of all classes, and the prototype representation of each class was the mean of all its support samples. Garcia and Bruna (2017) defined a graph neural network architecture to assimilate generic message-passing inference algorithms, which generalized above three models.

Regarding with few-shot relation classification, Han et al. (2018) adopted prototypical networks to build baseline models on the FewRel dataset. Gao et al. (2019) proposed hybrid attention-based prototypical networks to handle noisy training samples in few-shot learning. In this paper, we improve the conventional prototypical networks for few-shot relation classification by encoding the query instance and class prototype interactively through multi-level matching and aggregation.

2.3 Sentence Matching

Sentence matching is essential for many NLP tasks, such as natural language inference (NLI) (Bowman et al., 2015) and response selection (Lowe et al., 2015). Some sentence matching methods mainly rely on sentence encoding (Mueller and Thyagarajan, 2016; Conneau et al., 2017; Chen et al., 2018), which encode a pair sentences independently and then transmit their embeddings into a classifier, such as a neural network, to decide the relationship between them. Some other methods are based on joint models (Chen et al., 2017; Gong et al., 2017; Kim et al., 2018), which use cross-features to represent the local (i.e., word-level and phrase-level) alignments for better performance. In this paper, we follow the joint models to achieve the local matching between a query instance and the support set for a class. The difference between our task and the other sentence matching tasks mentioned above is that, our goal is to match a sentence to a set of sentences, instead of to another sentence (Bowman et al., 2015) or to a sequence of sentences (Lowe et al., 2015).

3 Task Definition

In few-shot relation classification, we are given two datasets, $\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$. Each dataset consists of a set of samples (x, p, r), where x is a sentence composed of T words and the t-th word is w_t, $p = (p_1, p_2)$ indicate the positions of two entities, and r is the relation label of the instance (x, p). These two datasets have their own relation label spaces that are disjoint with each other. Under few-shot configuration, $\mathscr{D}_{meta-test}$ is splited into two parts, $\mathscr{D}_{test-support}$ and $\mathscr{D}_{test-query}$. If $\mathscr{D}_{test-support}$ contains K labeled samples for each of N relation classes, this target few-shot problem is named N-way-K-shot. $\mathscr{D}_{test-query}$ contains test samples, each labeled with one of the N classes. Assuming that we only have $\mathscr{D}_{test-support}$ and $\mathscr{D}_{test-query}$, we can train a model using $\mathscr{D}_{test-support}$ and evaluate its performance on $\mathscr{D}_{test-query}$. But limited by the number of support samples (i.e,., $N \times K$), it is hard to train a good model from scratch.

Although $\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$ have disjoint relation label spaces, $\mathscr{D}_{meta-train}$ can also been utilized to help the few-shot relation classification on $\mathscr{D}_{meta-test}$. One approach is the paradigm proposed by Vinyals et al. (2016), which obey an important machine learning principle that test and train conditions must match. That's to say, we also split $\mathscr{D}_{meta-train}$ into two parts, $\mathscr{D}_{train-support}$ and $\mathscr{D}_{train-query}$, and mimic the few-shot learning settings at training stage. In each training iteration, N classes are randomly selected from $\mathscr{D}_{train-support}$, and K support instances are randomly selected from each class. In this way, we construct the train-support set $S = \{s_k^i; i = 1, ..., N, k = 1, ..., K\}$, where s_k^i is the k-th instance in class i. And also, we randomly select R samples from the remaining samples of those N classes and construct the train-query set $Q = \{(q_j, l_j); j = 1, ..., R\}$, where $l_j \in \{1, ..., N\}$ is the label of instance q_j.

Just like conventional prototypical networks, we expect to minimize the following objective function at training time

$$\mathrm{J}_{match} = -\frac{1}{R} \sum_{(q,l)\in Q} \mathrm{P}(l|S, q), \qquad (1)$$

and $\mathrm{P}(l|S, q)$ is defined as

$$\mathrm{P}(l|S, q) = \frac{\exp(f(\{s_k^l\}_{k=1}^K, q))}{\sum_{i=1}^N \exp(f(\{s_k^i\}_{k=1}^K, q))}. \qquad (2)$$

The function $f(\{s_k^i\}_{k=1}^K, q)$ is to calculate the matching degree between the query instance q and

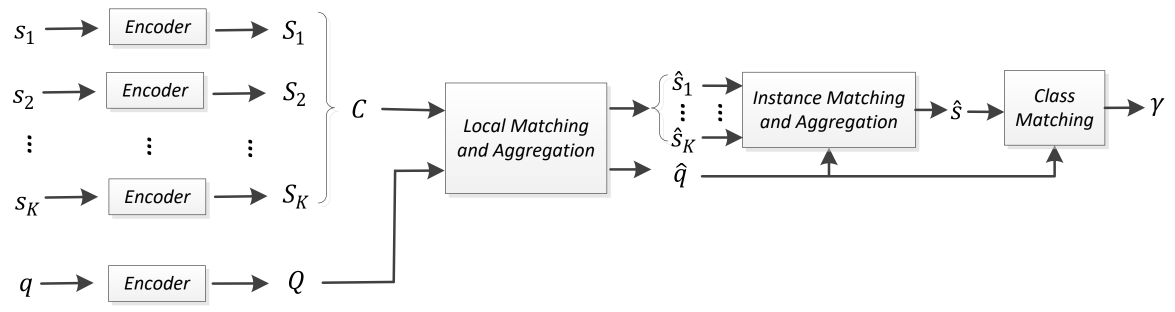

Figure 1: The framework of our proposed MLMAN model.

the set of support instances $\{s_k^i\}_{k=1}^K$. How to design this function is the focus of this paper.

4 Methodology

In this section, we will introduce our proposed multi-level matching and aggregation network (MLMAN) for modeling $f(\{s_k^i\}_{k=1}^K, q)$. For simplicity, we will discard the superscript i of s_k^i from Section 4.1 to Section 4.4. The framework of our proposed MLMAN model is shown in Fig. 1, which has four main modules.

- **Context Encoder.** Given a sentence and the positions of two entities within this sentence, CNNs (Zeng et al., 2014) are adopted to derive the local context representations of each word in the sentence.
- **Local Matching and Aggregation.** Similar to (Chen et al., 2017), given the local representation of a query instance and the local representations of K support instances, the attention method is employed to collect local matching information between them. Then, the matched local representations are aggregated to represent each instance as an embedding vector.
- **Instance Matching and Aggregation.** The matching information between a query instance and each of the K support instances are calculated using an MLP. Then, we take the matching degrees as weights to sum the representations of support instances in order to get the class prototype.
- **Class Matching.** An MLP is built to calculate the matching score between the representations of the query instance and the class prototype.

More details of these four modules will be introduced in the following subsections.

4.1 Context Encoder

For a query or support instance, each word w_t in the sentence x is first mapped into a d_w-dimensional word embedding $\mathbf{e}_t$ (Pennington et al., 2014). In order to describe the position information of the two entities in this instance, the position features (PFs) proposed by Zeng et al. (2014) are also adopted in our work. Here, PFs describe the relative distances between current word and the two entities, and are further mapped into two vectors $\mathbf{p}_{1t}$ and $\mathbf{p}_{2t}$ of d_p dimensions. Finally, these three vectors are concatenated to get the word representation $\mathbf{w}_t = [\mathbf{e}_t; \mathbf{p}_{1t}; \mathbf{p}_{2t}]$ of $d_w + 2d_p$ dimensions, and the instance can be written as $\mathbf{W} \in \mathbb{R}^{T\times(d_w+2d_p)}$.

The most popular models for local context encoding are recurrent neural networks (RNNs) with long short-term memories (LSTMs) (Hochreiter and Schmidhuber, 1997) and convolutional neural networks (CNNs) (Kim, 2014). In this paper, we employ CNNs to build the context encoder. For an input instance $\mathbf{W} \in \mathbb{R}^{T\times(d_w+2d_p)}$, we input it into a CNN with d_c filters. The output from the CNN is a matrix with $T \times d_c$ dimensions. In this way, the context representations of the query instance $\mathbf{Q} \in \mathbb{R}^{T_q\times d_c}$ and the context representations of support instances $\{\mathbf{S}_k \in \mathbb{R}^{T_k\times d_c}; k = 1, ..., K\}$ are obtained, where T_q and T_k are the sentence lengths of the query sentence and the k-th support sentence respectively.

4.2 Local Matching and Aggregation

In order to get the matching information between $\mathbf{Q}$ and $\{\mathbf{S}_k; k = 1, ..., K\}$, we first concatenate the K support instance representations into one matrix as follow

$$\mathbf{C} = \text{concat}(\{\mathbf{S}_k\}_{k=1}^K), \tag{3}$$

where $\mathbf{C} \in \mathbb{R}^{T_s\times d_c}$ with $T_s = \sum_{k=1}^K T_k$. Then, we collect the matching information between $\mathbf{Q}$

and $\mathbf{C}$ and calculate their matched representations $\widetilde{\mathbf{Q}}$ and $\widetilde{\mathbf{S}}$ as follows

$$\alpha_{mn} = \mathbf{q}_m^\top \mathbf{c}_n, \tag{4}$$

$$\widetilde{\mathbf{q}}_m = \sum_{n=1}^{T_s} \frac{\exp(\alpha_{mn})}{\sum_{n'=1}^{T_s} \exp(\alpha_{mn'})} \mathbf{c}_n, \tag{5}$$

$$\widetilde{\mathbf{c}}_n = \sum_{m=1}^{T_q} \frac{\exp(\alpha_{mn})}{\sum_{m'=1}^{T_q} \exp(\alpha_{m'n})} \mathbf{q}_m, \tag{6}$$

where $m \in \{1, ..., T_q\}$ in Eq. (5), $n \in \{1, ..., T_s\}$ in Eq. (6), $\mathbf{q}_m$ and $\widetilde{\mathbf{q}}_m$ are the m-th rows of $\mathbf{Q}$ and $\widetilde{\mathbf{Q}}$ respectively, and $\mathbf{c}_n$ and $\widetilde{\mathbf{c}}_n$ are the n-th rows of $\mathbf{C}$ and $\widetilde{\mathbf{C}}$ respectively.

Next, the original representations and the matched representations are fused utilizing a ReLU layer as follows,

$$\bar{\mathbf{Q}} = \text{ReLU}([\mathbf{Q}; \widetilde{\mathbf{Q}}; |\mathbf{Q} - \widetilde{\mathbf{Q}}|; \mathbf{Q} \odot \widetilde{\mathbf{Q}}]\mathbf{W}_1), \tag{7}$$
$$\bar{\mathbf{C}} = \text{ReLU}([\mathbf{C}; \widetilde{\mathbf{C}}; |\mathbf{C} - \widetilde{\mathbf{C}}|; \mathbf{C} \odot \widetilde{\mathbf{C}}]\mathbf{W}_1), \tag{8}$$

where $\odot$ is the element-wise product and $\mathbf{W}_1 \in \mathbb{R}^{4d_c \times d_h}$ is the weight matrix at this layer for reducing dimensionality. $\bar{\mathbf{C}}$ is further split into K representations $\{\bar{\mathbf{S}}_k\}_{k=1}^K$ corresponding to the K support instances where $\bar{\mathbf{S}}_k \in \mathbb{R}^{T_k \times d_h}$. All $\bar{\mathbf{S}}_k$ and $\bar{\mathbf{Q}}$ are fed into a single-layer Bi-directional LSTM (BLSTM) with d_h hidden units along each direction to obtain the final local matching results $\widehat{\mathbf{S}}_k \in \mathbb{R}^{T_k \times 2d_h}$ and $\widehat{\mathbf{Q}} \in \mathbb{R}^{T_q \times 2d_h}$.

Local aggregation aims to convert the results of local matching into a single vector for each query and each support instance. In this paper, we employ a max pooling together with an average pooling, and concatenate their results into one vector$\widehat{\mathbf{s}}_k$ or $\widehat{\mathbf{q}}$. The calculations are as follows,

$$\widehat{\mathbf{s}}_k = [\max(\widehat{\mathbf{S}}_k); \text{ave}(\widehat{\mathbf{S}}_k)], \forall k \in \{1, ..., K\}, \tag{9}$$
$$\widehat{\mathbf{q}} = [\max(\widehat{\mathbf{Q}}); \text{ave}(\widehat{\mathbf{Q}})], \tag{10}$$

where $\{\widehat{\mathbf{s}}_k, \widehat{\mathbf{q}}\} \in \mathbb{R}^{4d_h}$.

4.3 Instance Matching and Aggregation

Similar to conventional prototypical networks (Snell et al., 2017), our proposed method calculates class prototype $\widehat{\mathbf{s}}$ via the representations of all support instances in this class, i.e., $\{\widehat{\mathbf{s}}_k\}_{k=1}^K$. However, instead of using a naive mean operation, we aggregate instance-level representations via attention over $\{\widehat{\mathbf{s}}_k\}_{k=1}^K$, where each weight is derived from the instance matching score between $\widehat{\mathbf{s}}_k$ and $\widehat{\mathbf{q}}$. The matching function is as follow,

$$\beta_k = \mathbf{v}^\top(\text{ReLU}(\mathbf{W}_2[\widehat{\mathbf{s}}_k; \widehat{\mathbf{q}}])), \tag{11}$$

where $\mathbf{W}_2 \in \mathbb{R}^{d_h \times 8d_h}$ and $\mathbf{v} \in \mathbb{R}^{d_h}$. β_k describes the instance-level matching degree between the query instance q and the support instance s_k. Then, all $\{\widehat{\mathbf{s}}_k\}_{k=1}^K$ are aggregated into one vector $\widehat{\mathbf{s}}$ as

$$\widehat{\mathbf{s}} = \sum_{k=1}^{K} \frac{\exp(\beta_k)}{\sum_{k'=1}^{K} \exp(\beta'_k)} \widehat{\mathbf{s}}_k, \tag{12}$$

and $\widehat{\mathbf{s}}$ is the class prototype.

4.4 Class Matching

After the class prototype $\widehat{\mathbf{s}}$ and the embedding vector of the query instance $\widehat{\mathbf{q}}$ have been determined, the class-level matching function $f(\{s_k\}_{k=1}^K, q)$ in Eq. (2) is defined as

$$f(\{s_k\}_{k=1}^K, q) = \mathbf{v}^\top(\text{ReLU}(\mathbf{W}_2[\widehat{\mathbf{s}}; \widehat{\mathbf{q}}])). \tag{13}$$

Eq. (11) and (13) have the same form. In our experiments, sharing the weights $\mathbf{W}_2$ and $\mathbf{v}$ in these two equations, i.e., employing the exactly same function for both instance-level and class-level matching in each training iteration, lead to better performance.

4.5 Joint Training with Inconsistency Measurement

If the representations of all support instances in a class are far away from each other, it could become difficult for the derived class prototype to capture the common characteristics of all support instances. Therefore, a function which measures the inconsistency among the set of support instances is designed. In order to avoid the high complexity of directly comparing every two support instances in a class, we calculate the inconsistency measurement as the average Euclidean distance between the support instances and the class prototype as

$$\text{J}_{incon} = \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} ||\widehat{\mathbf{s}}_k^i - \widehat{\mathbf{s}}^i||_2^2, \tag{14}$$

where i is the class index and $||\cdot||_2$ calculates the 2-norm of a vector.

By combining Eqs. (1) and (14), the final objective function for training the whole model is defined as

$$\text{J} = \text{J}_{match} + \lambda \text{J}_{incon}, \tag{15}$$

where λ is a hyper-parameter and was set as 1 in our experiments without any tuning.

Model	5 Way 1 Shot	5 Way 5 Shot	10 Way 1 Shot	10 Way 5 Shot
Meta Network (Han et al., 2018)	64.46 ± 0.54	80.57 ± 0.48	53.96 ± 0.56	69.23 ± 0.52
GNN (Han et al., 2018)	66.23 ± 0.75	81.28 ± 0.62	46.27 ± 0.80	64.02 ± 0.77
SNAIL (Han et al., 2018)	67.29 ± 0.26	79.40 ± 0.22	53.28 ± 0.27	68.33 ± 0.25
Prototypical Network (Han et al., 2018)	69.20 ± 0.20	84.79 ± 0.16	56.44 ± 0.22	75.55 ± 0.19
Proto-HATT (Gao et al., 2019)	- -	90.12 ± 0.04	- -	83.05 ± 0.05
MLMAN	82.98 ± 0.20	92.66 ± 0.09	75.59 ± 0.27	87.29 ± 0.15

Table 2: Accuracies (%) of different models on FewRel test set.

5 Experiments

5.1 Dataset and Evaluation Metrics

The few-shot relation classification dataset FewRel[2] was adopted in our experiments. This dataset was first generated by distant supervision and then filtered by crowdsourcing to remove noisy annotations. The final FewRel dataset consists of 100 relations, each has 700 instances. The average number of tokens in each sentence is 24.99, and there are 124,577 unique tokens in total. The 100 relations are split into 64, 16 and 20 for training, validation and test respectively.

Our experiments investigated four few-shot learning configurations, 5 way 1 shot, 5 way 5 shot, 10 way 1 shot, and 10 way 5 shot, which were the same as Han et al. (2018). According to the official evaluation scripts[3], all results given by our experiments were the mean and standard deviation values of 10 training repetitions, and were tested using 20,000 independent samples.

5.2 Training Details and Hyperparameters

All of the hyperparameters used in our experiments are listed in Table 3. The 50-dimensional Glove word embeddings released by Pennington et al. (2014) [4] were adopted in the context encoder and were fixed during training. For the unknown words, we just replaced them with an unique special token <UNK> and fixed its embedding as a zero vector. Previous study (Munkhdalai and Yu, 2017) found that the models trained on harder tasks may achieve better performances than using the same configurations at both training and test stages. Therefore, we set $N = 20$ to construct the train-support sets for 5-way and 10-way tasks. In our experiments, grid searches among $d_c \in \{100, 150, 200, 250\}$, $d_h \in \{100, 150, 200, 250\}$

[2] https://thunlp.github.io/fewrel.html.

[3] https://thunlp.github.io/fewrel.html.

[4] https://nlp.stanford.edu/projects/glove/.

Component	Parameter	Value
word embedding	dimension	50
position feature	max relative distance	± 40
	dimension	5
CNN	window size	3
	filter number d_c	200
dropout	dropout rate	0.2
unidirectional LSTM	hidden size d_h	100
optimization	strategy	SGD
	learning rate	0.1
	size of query set R	5
	N_{train}	20
	λ	1

Table 3: Hyper-parameters of the models built in our experiments.

and $R \in \{5, 10, 15\}$ were conducted to determine their optimal values. For optimization, we employed mini-batch stochastic gradient descent (SGD) with the initial learning rate of 0.1. The learning rate was decayed to one tenth every 20,000 steps. And also, dropout layers (Hinton et al., 2012) were inserted before CNN and LSTM layers and the drop rate was set as 0.2.

5.3 Comparison with Previous Work

Table 2 shows the results of different models tested on FewRel test set. The results of the first four models, Meta Network (Munkhdalai and Yu, 2017), GNN (Garcia and Bruna, 2017), SNAIL (Mishra et al., 2018), Prorotypical Network (Snell et al., 2017), were reported by Han et al. (2018). These models were initially proposed for image classification. Han et al. (2018) just replaced their image encoding module with an instance encoding module and kept other modules unchanged. Proto-HATT (Gao et al., 2019) added hybrid attention mechanism to prototypical networks, mainly focusing on improving the performance on few-shot relation classification with $N > 1$. From Table 2, we can see that our proposed MLMAN model outperforms all other models by a large margin, which shows the effectiveness of considering the

Model	No.	5 Way 1 Shot	5 Way 5 Shot	10 Way 1 Shot	10 Way 5 Shot
MLMAN	1	79.01 ± 0.20	88.86 ± 0.20	67.37 ± 0.19	80.07 ± 0.18
-J$_{incon}$	2	79.01 ± 0.20	88.33 ± 0.15	67.37 ± 0.19	79.38 ± 0.22
IM(shared $\rightarrow$ untied)	3	79.01 ± 0.20	86.77 ± 0.19	67.37 ± 0.19	77.66 ± 0.09
IA(att. $\rightarrow$ max.)	4	79.01 ± 0.20	87.84 ± 0.13	67.37 ± 0.19	78.86 ± 0.15
IA(att. $\rightarrow$ ave.)	5	79.01 ± 0.20	87.48 ± 0.17	67.37 ± 0.19	78.58 ± 0.23
-J$_{incon}$	6	79.01 ± 0.20	86.23 ± 0.22	67.37 ± 0.19	77.36 ± 0.26
LM(-concatenation)	7	79.01 ± 0.20	85.48 ± 0.28	67.37 ± 0.19	74.56 ± 0.36
CM(MLP $\rightarrow$ ED)	8	76.52 ± 0.23	81.91 ± 0.13	62.89 ± 0.13	69.41 ± 0.15
-LM	9	74.13 ± 0.16	82.73 ± 0.16	59.71 ± 0.22	70.23 ± 0.23
CM(MLP $\rightarrow$ ED)	10	75.42 ± 0.23	82.36 ± 0.07	62.54 ± 0.26	70.45 ± 0.11

Table 4: Accuracies (%) of different models on FewRel development set. Here, IM stands for instance matching, IA stands for instance aggregation, LM stands for the local matching, CM stands for the class matching, MLP stands for multi-layer perceptrons and ED stands for Euclidean distance.

interactions between query instance and support set at multiple levels.

5.4 Ablation Study

In order to evaluate the contributions of individual model components, ablation studies were conducted. Table 4 shows the performance of our model and its ablations on the development set of FewRel. Considering that the first 6 ablations only affected the few-shot learning tasks with $N > 1$, model 2 to model 7 achieved exactly the same performance as the complete model (i.e., model 1) under 5 way 1 shot and 10 way 1 shot configurations.

5.4.1 Instance Matching and Aggregation

First, the attention-based instance aggregation introduced in Section 4.3 was replaced with a max pooling (model 4) or an average pooling (model 5). We can see that the model with instance-level attentive aggregation (model 1) outperformed the ones using a max pooling (model 4) or an average pooling (model 5) on 5-shot tasks. Their difference were significantly at 1% significance level in t-test. The advantage of attentive pooling is that the weights of integrating all support instances can be determined dynamically according to the query. For example, when conducting instance matching and aggregation between the query instance and the support set in Table 1, the weights of the 5 instances in class A were 0.03, 0.46, 0.25, 0.08 and 0.18 respectively. Instance #2 achieved the highest weight because it had the best similarity with the query instance and was considered as the most helpful one when matching the query instance with class A.

Then, the effectiveness of sharing the weight parameters in Eqs. (11) and (13) was evaluated by untying them (model 3). The performance of model 3 was much worse than the complete model (model 1) as shown in Table 4, which demonstrates the need of sharing the weights for calculating matching scores at both instance and class levels.

5.4.2 Inconsistency Measurement

As introduced in Section 4.5, J$_{incon}$ is designed to measure the inconsistency among the representations of all support instances in a class. After removing J$_{incon}$, model 2 was optimized only using the objective function J$_{match}$. We can see that it performed much worse than the complete model. Furthermore, we calculated the mean of the Euclidean distances between every support instance pair $(\widehat{\mathbf{s}}_k^i, \widehat{\mathbf{s}}_{k'}^i)$ in the same class using model 1 and model 2 respectively. For each support set, the calculation can be written as

$$D = \frac{2}{NK(K-1)} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{k'=k+1}^{K} ||\widehat{\mathbf{s}}_k^i - \widehat{\mathbf{s}}_{k'}^i||_2^2. \tag{16}$$

We sampled 20,000 support sets under the 5-way 5-shot configuration and calculated the mean of them. The results were 0.0199 and 0.0346 for model 1 and model 2 respectively, which means that J$_{incon}$ was effective at forcing the representations of the support instances in the same class to be close with each other.

J$_{incon}$ was further removed from model 5 and model 6 was obtained. It can be found that the accuracy degradation from model 5 to model 6 was larger than the one from model 1 to model 2. This implies that the J$_{incon}$ objective function also benefited from the attentive aggregation over support instances.

5.4.3 Local Matching

First, the concatenation operation in local matching was removed from model 6 in this ablation study. That's to say, instead of concatenating the representations of all support instances $\{\mathbf{S}_k\}_{k=1}^{K}$ into one single matrix as Eq. (3), local matching was conducted between the query instance and each support instance separately to get their vector representations $\{(\widehat{\mathbf{s}}_k, \widehat{\mathbf{q}}_k); k = 1, ..., K\}$ (model 7). It should be noticed that this led to K different representations of a query instance according to each support class. Then, the mean over k for $\widehat{\mathbf{s}}_k$ and $\widehat{\mathbf{q}}_k$ were calculated to get the representations of the support set $\widehat{\mathbf{s}}$ and the query instance $\widehat{\mathbf{q}}$. Comparing model 6 and model 7, we can see that the concatenation operation plays an important role in our model. One possible reason is that the concatenation operation can help local matching to restrain the support instances with low similarity to the query.

Second, the whole local matching module together with the concatenation and attentive aggregation operation were removed from model 6, which led to model 9. Model 9 is similar to the one proposed by Snell et al. (2017) that encoded the support and query instances independently. The difference was that model 9 was equipped with more components, including an LSTM layer, two pooling operations, and a learnable class matching function. Comparing the performance of model 6 and model 9 in Table 4, we can see that the local matching operation significantly improves the performance in few-shot relation classification. Fig. 2 shows the attention weight matrix calculated between the query instance and the support instance #2 of class A in Table 1. From this figure, we can see that the attention-based local matching is able to capture some matching relations of local contexts, such as the head entities *Eva Funck* and *Cindy Robbins*, the tail entities *Gustav* and *Kimberly Beck*, the key phrases *son* and *daughter*, the same keyword "*married*", and so on.

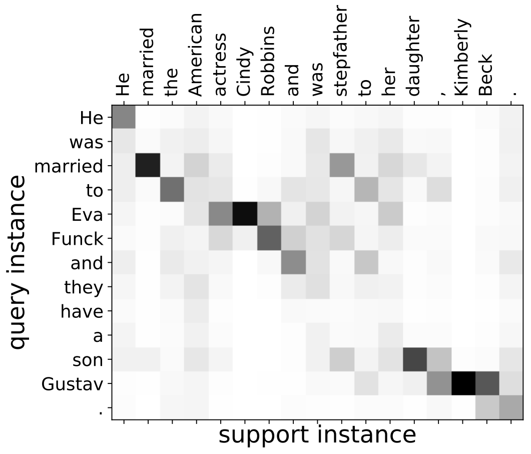

Figure 2: The attention weight matrix calculated between the query instance and the support instance #2 of class A in Table 1. The darker units have larger value. The summation of one column in the matrix is one.

5.4.4 Class Matching

In this experiment, we compared two class matching functions, (1) Euclidean distance (ED) (Snell et al., 2017) and (2) a learnable MLP function as shown by Eq. (13). In order to ignore the influence of the instance-level attentive aggregation, these two matching functions were compared based on model 6 and model 9. After converting the MLP function in model 6 and model 9 to Euclidean distance, model 8 and model 10 were obtained. Comparing the performance of these models in Table 4, we have two findings. (1) When local matching was adopted, the learnable MLP for class matching (model 6) outperformed the ED metric (model 8) by a large margin. (2) After removing local matching, the learnable MLP for class matching (model 9) performed not as good as the ED metric (model 10). One possible reason is that the local matching process enhances the interaction between a query instance and a support set when calculating $\widehat{\mathbf{s}}$ and $\widehat{\mathbf{q}}$. Thus, simple Euclidean distance between them may not be able to describe the complex correlation and dependency between them. On the other hand, MLP mapping is more powerful than calculating Euclidean distance, and can be more appropriate for class matching when local matching is also adopted.

6 Conclusions

In this paper, a neural network with multi-level matching and aggregation has been proposed for few-shot relation classification. First, the query and support instances are encoded interactively via local matching and aggregation. Then, the support instances in a class are further aggregated to form the class prototype and the weights are calculated by attention-based instance matching. Finally, a learnable MLP matching function is employed to calculate the class matching score between the query instance and each candidate class. Furthermore, an additional objective function is designed to improve the consistency among the vector representations of all support instances in a class. Experiments have demonstrated the effectiveness of our proposed model, which achieves state-of-the-art performance on the FewRel dataset. Studying few-shot relation classification with data generated by distant supervision and extending our ML-MAN model to zero-shot learning will be the tasks of our future work.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Nature Science Foundation of China (Grant No. U1636201, 61871358).

References

Steven Bethard and James H. Martin. 2007. Cu-tmp: Temporal relation classification using syntactic and semantic features. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 129–132. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics.

Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. 2018. Enhancing sentence embedding with generalized pooling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1815–1826. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification.

Victor Garcia and Joan Bruna. 2017. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809. Association for Computational Linguistics.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Seonhoon Kim, Jin-Hyuk Hong, Inho Kang, and Nojun Kwak. 2018. Semantic sentence matching with densely-connected recurrent and co-attentive information. *arXiv preprint arXiv:1805.11360*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Eprint Arxiv*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. *arXiv preprint arXiv:1703.00837*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307. Association for Computational Linguistics.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1980–1990. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344. Dublin City University and Association for Computational Linguistics.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212. Association for Computational Linguistics.