Classical Sequence Match is a Competitive Few-Shot One-Class Learner

Mengting Hu1 Hang Gao2* Yinhao Bai1 Mingming Liu1
1 College of Software, Nankai University
2 Institute for Public Safety Research, Tsinghua University
mthu@nankai.edu.cn, gaohang@mail.tsinghua.edu.cn
yinhao@mail.nankai.edu.cn, liumingming@nankai.edu.cn

Abstract

Nowadays, transformer-based models gradually become the “default choice” for artificial intelligence pioneers. The models also show superiority even in the few-shot scenarios. In this paper, we revisit the classical methods and propose a new few-shot alternative. Specifically, we investigate the few-shot one-class problem, which actually takes a known sample as a reference to detect whether an unknown instance belongs to the same class. This problem can be studied from the perspective of sequence match. It is shown that with meta-learning, the classical sequence match method, i.e. Compare-Aggregate, significantly outperforms transformer ones. The classical approach requires much less training cost. Furthermore, we perform an empirical comparison between two kinds of sequence match approaches under simple fine-tuning and meta-learning. Meta-learning causes the transformer models’ features to have high-correlation dimensions. The reason is closely related to the number of layers and heads of transformer models. Experimental codes and data are available at https://github.com/hmt2014/FewOne.

1 Introduction

When the labeled data is scarce in practical application, it is struggled to learn a well-performed model using deep learning algorithms. Yet annotating data costs much labor and time. Few-shot learning (FSL) intuitively addresses this obstacle (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017; Sung et al., 2018). FSL learns at the meta-task level, where each meta-task is formulated as inferring queries with the help of a support set (Vinyals et al., 2016). Multiple meta-tasks facilitate the task-agnostic transferrable knowledge. Thus it can learn new knowledge fast after being taught only a few samples. Despite FSL has been well-studied, its one-class scenario (Frikha et al., 2021) is less investigated.

In this paper, following the one-class trait, we design each meta-task as a binary classification. It consists of a reference instance, a support set, and a query set (see Figure 1). The reference instance is one known sample of a class, which is exploited to tell whether an instance out of the support/query set belongs to the same class. Such purpose is consistent with sequence match, which also makes a decision for two sequences. Previous sequence match can mainly be categorized into two promising directions: classical methods, e.g. Siamese Network (Koch et al., 2015), Compare-Aggregate (CA) (Wang and Jiang, 2017), and transformer-based method, e.g. DistilBert (Sanh et al., 2019), BERT (Devlin et al., 2019).

In recent years, transformer models have already beaten classical ones in a wide range of tasks (Devlin et al., 2019). We wonder how two kinds of models perform under the few-shot one-class scenario. Consequently, it is presented that with meta-learning, classical sequence match method can significantly outperform transformer-based models. The classical models require much less training cost. Specifically, model-agnostic meta-learning

Reference: This was the worst hotel in Vegas. Class: hotel

<table>
<thead>
<tr>
<th>Support set</th>
<th>Query set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) It was sold out due to there was the girls’ basketball tournament taking place over the weekend.</td>
<td>1</td>
</tr>
<tr>
<td>2) The service is wonderful and the hotel is amazing.</td>
<td>1</td>
</tr>
<tr>
<td>3) Food was awesome and so was our waiter.</td>
<td>0</td>
</tr>
</tbody>
</table>

1) Might be a good place, but if you need customer support, good luck! ?
2) Basic cable and wifi. ?
3) This place was designed really well, too. ?

Figure 1: Meta-task example in few-shot one-class text classification, where 1 denotes a positive instance and 0 denotes a negative one.

* Hang Gao is the corresponding author.
(MAML) algorithm (Finn et al., 2017) is a subtle bi-level optimizing approach that aims to learn a good parameter initialization. By introducing this algorithm, classical methods act as simple but competitive few-shot one-class learners.

Furthermore, we make an empirical comparison between classical and transformer-based models under simple fine-tuning and meta-learning. Firstly, it is found that MAML has a more positive impact on the both sequence match approaches than simple fine-tuning. This suggests that a good parameter initialization is important for both of them. Secondly, MAML tends to make transformer models extract features with high-correlation dimensions. The bi-level optimization might cause the feature extraction layers of large models less trained. Yet the last classifying layer tends to be better learned relatively. We demonstrate that the high correlation is related to the number of heads and layers in the transformer.

In summary, our main contributions are as follows: (1) We present a simple but competitive few-shot one-class learner, which is based on the classical sequence match approach and meta-learning. Extensive experimental results show that this learner achieves significant improvements compared with transformer-based models. This provides new insights in the transformer-dominant era. (2) Based on the testbed provided by the above approaches, an empirical study is made to further reveal their underlining natures. New observations and conclusions are derived.

2 Related Works

2.1 Few-Shot Learning

Few-shot learning (FSL) (Fei-Fei et al., 2006) deals with the practical problem of data scarcity in an intuitive way. It learns new knowledge fast with limited supervised information. An early work (Koch et al., 2015) learns to detect whether two instances belong to the same class. Later, matching network (Vinyals et al., 2016) proposes to construct multiple meta-tasks in both the training and testing procedures. This setting becomes mainstream in the subsequent works, to name a few, distance-based methods (Snell et al., 2017; Sung et al., 2018; Garcia and Bruna, 2018; Bao et al., 2020), optimization-based methods (Finn et al., 2017; Munkhdalai and Yu, 2017) or hallucination-based methods (Wang et al., 2018; Li et al., 2020). Among them, MAML (Finn et al., 2017) is special for “model-agnostic”, indicating that this algorithm can be applied in any model. Therefore, we choose to further study its effects. To the best of our knowledge, it is seldom studied in transformer-based models. We provide an interesting empirical analysis in the experiments.

Recently, prompt-based fine-tuning (Gao et al., 2021) also become popular in FSL. It classifies a template-based instance through the masked language model. This prediction manner bridges the gap between pre-training and fine-tuning. Its effectiveness in the few-shot one-class scenario is under-explored.

2.2 One-Class Few-Shot Learning

Recently some works discuss the one-class problem in FSL. Cumulative LEARning (CLEAR) (Kozerawski and Turk, 2018) uses transfer learning to model the decision boundary of SVM. One-way proto (OWP) (Kruspe, 2019) is based on the prototypical network (Snell et al., 2017). OWP computes the positive prototype by simply averaging the representations of instances. It designs a 0-vector as the negative prototype. The Euclidean distance with prototypes in the embedding space indicates that an instance is positive or negative. One-class MAML (Frikha et al., 2021) proposes a simple data sampling strategy to ensure that the class-imbalance rate of the inner-level matches the test task. Different from them, we leverage the unique direction in natural language processing, i.e. sequence match, to study the one-class FSL.

2.3 Sequence Match

Sequence match aims to make a decision for two sequences. Many tasks require to match sequences, such as text entailment (Bowman et al., 2015), machine comprehension (Tapaswi et al., 2016), recommendation (Kraus and Feuerriegel, 2019), etc. A straightforward approach is to encode each sequence as a vector and then compare the two vectors to make a decision (Bowman et al., 2015; Feng et al., 2015). However, a single vector is insufficient to match the important information between two sequences. Thus attention mechanism is adopted in this task (Rocktäschel et al., 2016).

Later, the Compare-Aggregate framework is proposed (Wang and Jiang, 2017) for matching sequences, which has been widely studied. Its extended version usually considers the bidirectional information of two inputs (Bian et al., 2017; Yoon et al., 2019). One previous work (Ye and Ling,
We choose to study its extended version, i.e. Bidi-
whether this instance belongs to the same class as
Assume the training data
Encoder
Given an input sentence with
Bidirectional Compare-Aggregate (BiCA) in detail.
In this section, we will introduce the components of
3.1 Problem Definition
Assume the training data \( D_{\text{train}} \) is composed of a
and a query set
A meta-task contains a reference sentence \( r \), a
support set \( S \) and a query set \( Q \).
\begin{align}
S &= \{(x^1_s, y^1_s), (x^2_s, y^2_s), \ldots (x^{|S|}_s, y^{|S|}_s)\} \\
Q &= \{(x^1_q, y^1_q), (x^2_q, y^2_q), \ldots (x^{|Q|}_q, y^{|Q|}_q)\} 
\end{align}
where \( x_s/x_q \) is an instance and \( y_s/y_q \) denotes
where \( W^i \) and \( W^u \) are parameter matrix, \( b^i \) and
Attention As depicted in Figure 2, the reference
This attention mechanism is non-parametric
\begin{align}
H &= \sigma(W^i E + b^i) \circ \tanh(W^u E + b^u) \\
\overline{H}_r &= \text{softmax}(H_r^T H) \\
\overline{H} &= \text{softmax}(H^T \overline{H}_r)
\end{align}
where \( \sigma \) is the sigmoid function, \( \circ \) is
This attention mechanism is non-parametric
\begin{align}
C_r &= \overline{H}_r \circ H \\
C &= \overline{H} \circ H_r
\end{align}
The comparison function is also non-parametric
Figure 2: The network architecture of BiCA. The pa-
and Jiang, 2017) to obtain contextualized informa-
tion. This gating mechanism aims at remembering
such design reduces the reliance
parameters are shared by two input instances.

\[ \mathbf{H} = \sigma(\mathbf{W}^{i}\mathbf{E} + \mathbf{b}^{i}) \circ \tanh(\mathbf{W}^{u}\mathbf{E} + \mathbf{b}^{u}) \]  
where \( \mathbf{W}^{i} \) and \( \mathbf{W}^{u} \) are parameter matrix, \( \mathbf{b}^{i} \) and
\( \mathbf{b}^{u} \) are biases, \( \circ \) is element-wise multiplication.
\textit{Attention} As depicted in Figure 2, the reference
and an instance from support/query set are fed into
the encoder, obtaining \( \overline{H}_r \) and \( H \). Then the inter-
action between two inputs is computed through an
attention mechanism.
\begin{align}
\overline{H}_r &= \text{softmax}(H_r^T H) \\
\overline{H} &= \text{softmax}(H^T \overline{H}_r)
\end{align}
This attention mechanism is non-parametric
since it only depends on the encoded representa-
tions \( \overline{H}_r \) and \( H \). Such design reduces the reliance
on parameters and focuses on learning the relationships
between data. Hence, this helps better adapt to
unseen classes.

\textit{Bidirectional Comparison} To compare the two
instances, we adopt a simple word-level compar-
ison function, i.e., element-wise multiplication \( \circ \).
\begin{align}
C_r &= \overline{H}_r \circ H \\
C &= \overline{H} \circ H_r
\end{align}
The comparison function is also non-parametric
for the purpose of adaptation. As shown in Figure
2, the encoded representations are applied in both
the attention and bidirectional comparison modules
to promote the mutual interaction of two inputs.
Fig. 3: Introducing meta-learning to sequence match approaches. The left part and the right part share the same architecture but employ different parameters in the meta-learning. The five steps are corresponding to the Algorithm 1. After training, the optimal parameter initialization \( \theta \) for the testing classes is obtained.

**Aggregation** Following the original work (Wang and Jiang, 2017), the comparison representations are aggregated by convolution neural network (CNN) (Kim, 2014). The convolution kernel slides over the comparison sequence to extract \( n \)-gram features, which tends to be helpful in matching.

\[
f_r = \text{CNN}(C_r) \quad f = \text{CNN}(C)
\]

where CNN(\( \cdot \)) is a convolution operation followed by max-pooling. Then the matching score is computed with the aggregation representations of two input sentences.

\[
p = \text{MLP}([f_r, f])
\]

where MLP is a single linear layer. \( p \) is a two-dimensional logits output.

**Loss** The training objective of BiCA is the cross entropy loss.

\[
L_\theta(r, S) = -\frac{1}{|S|} \sum_{|S|} \log P(y|p)
\]

where \( y \) is the ground truth. \( \theta \) represents all the parameters in the sequence match model.

### 3.3 Transformer-Based Sequence Match

Transformer-based models, e.g. BERT (Devlin et al., 2019) are pre-trained on a large-scale corpus, serving as foundational backbones for a wide range of natural language processing tasks. When aiming at sequence match, BERT utilizes two special tokens [CLS] and [SEP] to concatenate two sequences as a whole. For the two input instances shown in Figure 2, they are combined into \{[CLS], w_1, w_2, ..., w_L, [SEP], w_1, w_2, ..., w_L'\}, where the output of [CLS] is usually for classification and [SEP] is for separating two inputs. This combined sequence is then fed into BERT. The self-attention mechanism (Vaswani et al., 2017) in the transformer will compute the interaction between two inputs. Finally, the output of the first token [CLS] is adopted for inference. The training objective is also computed by Eq. (7).

### 3.4 Meta-Learning for Sequence Match

In the few-shot one-class paradigm, a meta-task from the unseen class has a few labeled instances as the support set. To better leverage such knowledge, we introduce meta-learning to sequence match models, which is displayed in Figure 3 and Algorithm 1. Specifically, model-agnostic meta-learning (MAML) algorithm (Finn et al., 2017) is chosen to investigate its impact on the sequence match approaches. This algorithm learns a good initialization of model parameters by maximizing the sensitivity of the loss function when adapting to new tasks (Song et al., 2020).

**Construct a Meta-Task** In Algorithm 1 (line 6), given a positive class, we first sample \( N_p \) and negative classes \( C_n \) from \( D_{\text{train}} \), and obtain \( C_p \cap C_n = \emptyset \) for all \( C_p \) do

\[
\begin{align*}
&\text{Construct a meta-task} \\
&\text{Evaluate } \nabla_\theta L_\theta(r, S) \text{ in Eq. (7)} \\
&\text{Compute adapted parameters with gradient descent:} \\
&\theta' = \theta - \alpha \nabla_\theta L_\theta(r, S) \\
&\text{Update } \theta \leftarrow \theta - \beta \nabla_\theta \sum_{C_p} L_{\theta'}(r, Q) \\
&\text{until performance on the validation data set does not improve in 3 epochs.}
\end{align*}
\]

in the transformer will compute the interaction between two inputs. Finally, the output of the first token [CLS] is adopted for inference. The training objective is also computed by Eq. (7).
Meta-learning trains in a bi-level way (see Algorithm 1), including the inner-level (line 8) and the outer-level (line 9) for the support set $S$ and the query set $Q$, respectively. This way will cause the gradients for updating parameters to propagate through more layers (line 9), i.e. twice as many as the number of network layers in a sequence match model. Its special effects on the transformer models are further discussed in §4.4. When evaluating, a model is initialized from the parameters $\theta$ trained by MAML on the training classes, which is then optimized with the support set on the testing classes.

4 Experiments

4.1 Datasets

**Aspect Category Detection (ACD)** A dataset for few-shot one-class ACD is collected from YelpAspect (Bauman et al., 2017; Li et al., 2019), which is a large-scale multi-domain dataset for fine-grained sentiment analysis. The 100 aspect categories are split without intersection into 64 classes for training, 16 classes for validation, and 20 classes for testing. Table 1 displays the statistics of the dataset. The data in each class is further divided according to the number of the aspects in a sentence, into single-aspect and multi-aspect. Relatively, a multi-aspect example contains more noise when matching sequences. To explore a challenging scenario, the support/query set are both sampled from the multi-aspect set. Fixing this setup, we choose a reference instance as a single- and multi-aspect one.

<table>
<thead>
<tr>
<th>Class Num</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>64</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Num</th>
<th>single</th>
<th>multi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13677</td>
<td>26643</td>
</tr>
<tr>
<td></td>
<td>3394</td>
<td>6686</td>
</tr>
<tr>
<td></td>
<td>4671</td>
<td>7929</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Num/Class</th>
<th>single</th>
<th>multi</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>213.7</td>
<td>416.2</td>
</tr>
<tr>
<td></td>
<td>212.1</td>
<td>417.8</td>
</tr>
<tr>
<td></td>
<td>233.5</td>
<td>396.4</td>
</tr>
</tbody>
</table>

Table 1: Dataset statistics for ACD. single denotes the single-aspect sentence and multi denotes the multi-aspect one.

instances from this class, which constitutes a reference instance $r$ and $N$ positive ones. Moreover, $N$ negative examples are also sampled from the negative classes $C_n$. These positive and negative examples are mixed up randomly, which are further divided into the support set and query set.

Meta-learning trains in a bi-level way (see Algorithm 1), including the inner-level (line 8) and the outer-level (line 9) for the support set $S$ and the query set $Q$, respectively. This way will cause the gradients for updating parameters to propagate through more layers (line 9), i.e. twice as many as the number of network layers in a sequence match model. Its special effects on the transformer models are further discussed in §4.4. When evaluating, a model is initialized from the parameters $\theta$ trained by MAML on the training classes, which is then optimized with the support set on the testing classes.

4.2 Baseline Methods

**Matching sequences at the vector-level:**

**SN** (Koch et al., 2015) Siamese network can capture discriminative features to generalize the predictive power of the network. The input instances are extracted into two vectors, which are compared with cosine similarity.

**OWP** (Kruspe, 2019) One-way prototypical network designs a 0-vector as the negative prototype. It measures the Euclidean distance between an instance with the positive/negative prototypes in the embedding space.

**Matching sequences at the word-level:**

**CA** (Wang and Jiang, 2017) Compare-Aggregate is widely used to match the important units between sequences. It only compares in one direction, i.e., reference-to-candidate.

**BiCA** CA is enhanced into matching sequences bidirectionally (§3.2).

**DistilBert** (Sanh et al., 2019) It is a distilled version of BERT (§3.3).

**BERT** (Devlin et al., 2019) It is transformer-based and matching sequences at word-level (§3.3).

**BERT(p)** (Gao et al., 2021) It trains BERT with prompt-based learning. The two sequences are concatenated with “[MASK]” like Gao et al.. The representation of [MASK] is mapped into word “yes” or “no”, suggesting that two sequences belong to the same class or not.

4.2.1 Implementation Details

Baseline methods are trained with naive training, which learns the training classes in a meta-task manner, but combines the support set and query set as a whole to optimize parameters. +finetune means that the naive trained models are fine-tuned. +MAML indicates the models are optimized by MAML (Algorithm 1). During the evaluation, +finetune or +MAML exploit the support set of testing classes in the same way, with the same number of updating steps and learning rate. Thus the only difference between +finetune and +MAML is the parameters are initialized by naive training or MAML training. All testing meta-tasks share the
Table 2: Experimental results for ACD and HuffPost in terms of accuracy(%) and macro-f1(%). We report the average and standard deviation of 5 runs. Single indicates that the reference instance is single-aspect. Multi indicates setting references as multi-aspect. The marker † refers to $p$-value $<0.01$ of the T-test compared with DistilBert+MAML. The marker ‡ refers to $p$-value $<0.07$ of the T-test compared with DistilBert+MAML.

same parameter initialization without mutual interference. The implementation details are described in the Appendix.

### 4.3 Experimental Results

The experimental results on ACD and HuffPost datasets are displayed in Table 2. The first part in Table 2 shows the case that when testing, we only have the reference instance but do not use the support set. By comparing two match types, we find that a finer-granularity matching helps the few-shot one-class scenario gain significantly. This indicates that in the few-shot scenario of text tasks, learning deeper interaction between instances is a better choice. Many previous tasks gain significantly from BERT (Wang et al., 2020) or DistilBert (Wright and Augenstein, 2020). However, we surprisingly see little performing difference among the five word-level sequence match methods. The transformer-based methods do not have remarkable superiority. A possible reason is that in the unseen classes, it is difficult to discover the key words/semantics for matching only given a reference instance. Meanwhile, these models with large-scale parameters may be superior in the data-driven tasks (Gururangan et al., 2020).

Additionally, though the scale of the support set is small, exploiting it by fine-tuning or MAML can bring significant improvements. This also explains our previous guess that the support set will provide key words/semantics to match. Meanwhile, on sequence match methods, including BiCA, BERT and DistilBert, MAML outperforms fine-tuning in most situations. This indicates the importance of a good parameter initialization not only for small models but also for large pre-trained models in a few-shot problem.

It is also found that prompt-based fine-tuning, i.e. BERT(p), is also less-performed than BiCA+MAML. The possible reasons are: first, the objective of one-class sequence match is not consistent with the pre-training of language models. Thus the knowledge of transformer models might not be fully leveraged; second, prompt-based fine-tuning may achieve better results by other huge-scale pre-trained models, such as RoBERTa-large, GPT-3 (Gao et al., 2021).

Finally, it is worth noting that BiCA+MAML consistently outperforms BERT+MAML and DistilBert+MAML. Compared with transformer models, BiCA+MAML has fewer parameters, suggesting that the classical methods are still worth revisiting in the large pre-trained models’ dominant era. We further see another interesting phenomenon. BiCA gains significantly from MAML but slightly improves by using fine-tuning. Contrarily, the transformer-based method gains much from fine-tuning. It is possible that the pre-trained BERT already contains abundant knowledge, suggesting a good initialization for fine-tuning. Meanwhile, BiCA is a classical model with much fewer parameters, which is easier for MAML to learn a good initialization. Hence, MAML has a more significant contribution to BiCA.
4.4 Discussion

In this section, an empirical comparison, between two kinds of sequence match approaches, including classical and transformer-based ones, is presented. Because DistilBert+MAML and BERT+MAML are comparable, as shown in Table 2. Meanwhile, the parameter scale of DistilBert is smaller, which is chosen in the following study. We randomly sample 12 batches from the testing classes, and obtain the extracted features of the query set. In each batch, we have 5 meta-tasks, each of them has 10 support instances and 10 query instances. Thus, the total number of features is $5 \times 10 \times 12 = 600$. The features are visualized by t-SNE (Maaten and Hinton, 2008).

4.4.1 Comparison of Features

We compare the effects of MAML with simple fine-tuning in Figure 4 and Figure 5, respectively. In Figure 4, it can be seen that fine-tuning can help the features learned by BiCA and DistilBert both become more separable (plot a-d). This is also reflected by the 2-dimensional logits output (e-g).

In Figure 5, it can be observed that in MAML training, exploiting the support set can also make the features more discriminative in BiCA (a-b), and so does the logits output (e-f). The features (plot b in Figure 4 and plot b in Figure 5) validates that MAML is more effective than fine-tuning in BiCA. Interestingly, the phenomenon of DistilBert+MAML is completely different. It is found that the features show less separability (c-d), while the logits output is well distinguished (g-h). This indicates the features are linearly separable in high dimensions, i.e. 768. Recalling the purpose of PCA (Principal Component Analysis) (Abdi...
and Williams, 2010), it defines an orthogonal linear transformation that transforms the data into a new coordinate system and preserves the greatest variance. We guess that the unique phenomena (c-d) are caused by less-orthogonal feature dimensions. To verify this assumption, we compute the covariance matrix based on extracted features in various models. Each element in the matrix indicates the correlation between two dimensions in feature, where a larger score means a higher correlation. We define the \textit{Cov}_\text{Score} as below, which is the average absolute value of the covariance matrix.

\[
\text{Cov}_\text{Score} = \text{avg} |\text{Cov}(F-\text{rowavg}(F))| 
\]

where \( F \) is the extracted features.

In Table 3, the \textit{Cov}_\text{Score} of three sequence match approaches are presented separately. Firstly, compared with BiCA+finetune, BiCA+MAML extracts features with slightly higher \textit{Cov}_\text{Score}. However, for DistilBert and BERT, MAML dramatically increases the score, which is more significant in BERT. As the scores indicate, DistilBert+MAML really extracts features with less-orthogonal dimensions. A possible reason is that MAML trains the model in a bi-level manner (see Algorithm 1). DistilBert has deeper layers and large-scale parameters. The gradients are propagated through deeper layers, causing the feature extraction learned insufficiently while the last linear layer is trained adequately. Thus the logits are separable while features are not, as plots (c, d, g, h) shown in Figure 5. The above phenomenon becomes also serious in BERT+MAML, because BERT has more layers compared with DistilBert, leading to larger \textit{Cov}_\text{Score}.

The plots of other datasets are displayed in the Appendix. The empirical observations are similar.

4.4.2 DistilBert+MAML: Layers and Heads?
To further investigate why the features learned by DistilBert+MAML have high-correlation dimensions, we plot the \textit{Cov}_\text{Score} of multiple variants of DistilBert+MAML in Figure 6. The horizontal ordinate indicates the number of heads in self-attention, which should divide 768 exactly (768 is the dimension size of the hidden states in DistilBert and BERT models).

It is first seen that the score shows an increasing trend as the number of heads grows. Meanwhile, by comparing the curves between \#Layer=1 and \#Layer=6, we observe that the \textit{Cov}_\text{Score} also becomes larger as the layers of DistilBert increase. We draw an empirical conclusion that the high-correlation of feature dimensions in DistilBert+MAML is caused by the multiple layers and heads of DistilBert.

4.4.3 Initialization for Loss Sensitivity or a Good Performance Start
In Figure 7, we depict the average batch loss and accuracy in the 3 update steps for ACD with single-aspect references. Step 0 indicates the model is trained by naive training or MAML without using the testing support set. Concretely, it can be seen that for both BiCA and DistilBert, MAML leads to faster loss degradation than fine-tuning (plot a). The loss in MAML
declines to the bottom even within one updating step. We also observe a rapid accuracy increase (c). Overall, MAML outperforms fine-tuning for both BiCA and DistilBert. However, is MAML always a good choice? The answer is negative. In step 0 (b and d), we see a good parameter initialization does not lead to a good performance start. BiCA and DistilBert learned by naive training achieve lower losses and better accuracy scores without updating parameters. A possible explanation is that they have different training objectives. Naive training aims to facilitate task-agnostic sequence matching. While the bi-level optimization in MAML focuses on promoting the generalization ability of models after seeing a few support examples.

4.4.4 Various Numbers of Support Instances

We further explore the performances of BiCA+MAML and DistilBert+MAML under various support set scales. As displayed in Figure 8, the number of support instances ranges from 1 to 10. For a fair comparison, the number of query instances is all set to 10. Firstly, it can be seen that as the support set scale grows, the performances on the query set present an increasing trend. Secondly, we also observe that BiCA+MAML outperforms DistilBert+MAML in most experimental settings. This indicates that classical sequence match is a competitive few-shot one-class learner.

5 Conclusion

In this work, we revisit the classical sequence match approaches and find that with meta-learning, the classical method can significantly outperform transformer models in the few-shot one-class scenario. The training cost is greatly reduced. Furthermore, an empirical study is made to explore the effects of simple fine-tuning and meta-learning. Interestingly, although meta-learning is more effective than simple fine-tuning on both sequence match approaches, it makes the transformer features have high correlation dimensions. The correlation is closely related to the number of layers and heads in the transformer models. We hope this work could provide insights for future research on few-shot problems and transformer models.

Acknowledgements

We sincerely thank all the anonymous reviewers for providing valuable feedback. This work is supported by the key program of National Science Fund of Tianjin, China (Grant No. 21JCZDJC00130) and the Basic Scientific Research Fund, China (Grant No. 63221028).

References


Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), page 717–725.


A Additional Experimental Results

The visualizations of aspect category detection with multi-aspect reference instance are displayed in Figure 9 and Figure 10.

For HuffPost dataset, the visualizations are displayed in Figure 11 and 12.

B Reproducibility

B.1 Computing Infrastructure

All experiments are conducted on the same hardware and software. We use a single NVIDIA A6000 GPU with 48GB of RAM.

B.2 Average Running time

The average running time of each model is shown in Table 5.

B.3 Number of Parameters

The number of parameters of each model is shown in Table 4.

B.4 Datasets

The datasets are available at https://github.com/hmt2014/FewOne.

B.5 Implementation Details

B.5.1 Classical Methods

All baselines and our model are implemented by Pytorch. We initialize word embeddings with 50-dimension GloVE vectors (Pennington et al., 2014).

In the aggregation module, the channels of the CNNs for input and output are both 50. The kernel sizes of five CNNs are [1, 2, 3, 4, 5], respectively. Relu is the activation function for CNN. We adopt a dropout of 0.1 after both the comparison and CNN in aggregation. MLP is a single linear layer.

The batch size is \(|C_p|=5\), indicating a batch comprises 5 meta-tasks. The instance number in the support set \(|S|\) and query set \(|Q|\) are both set to 10. Every epoch we randomly sample 400 batches for training, 300 batches for validation and 300 batches for testing. The average results of the testing batches are reported. We exploit an early stop strategy during training if the macro-f1 score on the validation set does not improve in 3 epochs, and the best model is chosen for evaluation.

We describe the training details as the format of (optimizer, learning rate, other information):

Naive Training: Adam, 1e-3, early stop.

+finetune SGD, 0.1, 3 updating steps.

+MAML

The inner-level: \(\alpha=0.1\).

The outer-level: Adam, \(\beta=1e-3\).

When testing: SGD, 0.1, 3 updating steps.

B.5.2 Transformer-based Methods

The output hidden state of \([CLS]\) in DistilBert (distilbert-base-uncased) and BERT (bert-base-uncased) is exploited for classification.

Naive Training Adam, 2e-5, 5 epochs.
(a) BiCA  (b) BiCA+finetune  (c) DistilBert  (d) DistilBert+finetune

Figure 11: Effects of fine-tuning on HuffPost.

(a) BiCA (b) BiCA+MAML(init) (c) DistilBert+MAML(init) (d) DistilBert+MAML

Figure 12: Effects of MAML on HuffPost.

<table>
<thead>
<tr>
<th>Method</th>
<th>ACD</th>
<th>Huffpost</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN</td>
<td>1,437,950</td>
<td>1,393,050</td>
</tr>
<tr>
<td>OWP</td>
<td>1,437,950</td>
<td>1,393,050</td>
</tr>
<tr>
<td>CA</td>
<td>1,476,202</td>
<td>1,431,302</td>
</tr>
<tr>
<td>BiCA</td>
<td>1,476,702</td>
<td>1,431,802</td>
</tr>
<tr>
<td>BiCA+finetune</td>
<td>1,476,702</td>
<td>1,431,802</td>
</tr>
<tr>
<td>BiCA+MAML</td>
<td>1,476,702</td>
<td>1,431,802</td>
</tr>
<tr>
<td>DistilBert</td>
<td>66,364,418</td>
<td>66,364,418</td>
</tr>
<tr>
<td>DistilBert+finetune</td>
<td>109,483,778</td>
<td>109,483,778</td>
</tr>
<tr>
<td>DistilBert+MAML</td>
<td>109,483,778</td>
<td>109,483,778</td>
</tr>
<tr>
<td>BERT</td>
<td>133,545,786</td>
<td>133,545,786</td>
</tr>
<tr>
<td>BERT(p)</td>
<td>133,545,786</td>
<td>133,545,786</td>
</tr>
<tr>
<td>BERT(p)+finetune</td>
<td>133,545,786</td>
<td>133,545,786</td>
</tr>
</tbody>
</table>

Table 4: Number of parameters in each model.

+finetune Adam, 2e-5, 3 updating steps.

+MAML
The inner level: $\alpha=2e-3$.
The outer-level: Adam, $\beta=2e-5$.
When testing: Adam, 2e-5, 3 updating steps.
We use early stop for DistilBert+MAML since the model does not learn optimally within 5 epochs.

<table>
<thead>
<tr>
<th>Method</th>
<th>single</th>
<th>ACD multi</th>
<th>Huffpost</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN</td>
<td>6m47s</td>
<td>11m44s</td>
<td>20m5s</td>
</tr>
<tr>
<td>OWP</td>
<td>13m57s</td>
<td>17m49s</td>
<td>11m45s</td>
</tr>
<tr>
<td>CA</td>
<td>21m36s</td>
<td>35m17s</td>
<td>18m2s</td>
</tr>
<tr>
<td>BiCA</td>
<td>22m26s</td>
<td>18m5s</td>
<td>11m11s</td>
</tr>
<tr>
<td>BiCA+finetune</td>
<td>20s</td>
<td>20s</td>
<td>23s</td>
</tr>
<tr>
<td>BiCA+MAML</td>
<td>39m8s</td>
<td>1h11m57s</td>
<td>40m53s</td>
</tr>
<tr>
<td>DistilBert</td>
<td>21m2s</td>
<td>21m26s</td>
<td>21m56s</td>
</tr>
<tr>
<td>DistilBert+finetune</td>
<td>4m45s</td>
<td>4m43s</td>
<td>4m45s</td>
</tr>
<tr>
<td>DistilBert+MAML</td>
<td>4h32m2s</td>
<td>4h3m38s</td>
<td>4h46m4s</td>
</tr>
<tr>
<td>BERT</td>
<td>39m33s</td>
<td>40m18s</td>
<td>41m27s</td>
</tr>
<tr>
<td>BERT+finetune</td>
<td>8m40s</td>
<td>8m41s</td>
<td>9m14s</td>
</tr>
</tbody>
</table>

Table 5: Average running time of each model.