Match-Prompt: Improving Multi-task Generalization Ability for Neural Text Matching via Prompt Learning

Shicheng Xu  
Data Intelligence System Research Center, Institute of Computing Technology, CAS  
University of Chinese Academy of Sciences  
Beijing, China  
xschit@163.com

Liang Pang  
Data Intelligence System Research Center, Institute of Computing Technology, CAS  
Beijing, China  
pangliang@ict.ac.cn

Huawei Shen  
Data Intelligence System Research Center, Institute of Computing Technology, CAS  
University of Chinese Academy of Sciences  
Beijing, China  
shenhuawei@ict.ac.cn

Xueqi Cheng  
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, CAS  
University of Chinese Academy of Sciences  
Beijing, China  
cxq@ict.ac.cn

ABSTRACT
Text matching is a fundamental technique in both information retrieval and natural language processing. Text matching tasks share the same paradigm that determines the relationship between two given texts. The relationships vary from task to task, e.g., relevance in document retrieval, semantic alignment in paraphrase identification, and answerable judgment in question answering. However, the essential signals for text matching remain in a finite scope, i.e., exact matching, semantic matching, and inference matching. Ideally, a good text matching model can learn to capture and aggregate these signals for different matching tasks to achieve competitive performance, while recent state-of-the-art text matching models, e.g., Pre-trained Language Models (PLMs), are hard to generalize. It is because the end-to-end supervised learning on task-specific dataset makes model overemphasize the data sample bias and task-specific signals instead of the essential matching signals, which ruins the generalization of model to different tasks. To overcome this problem, we adopt a specialization-generalization training strategy and refer to it as Match-Prompt. In specialization stage, descriptions of different matching tasks are mapped to only a few prompt tokens. In generalization stage, text matching model explores the essential matching signals by being trained on diverse multiple matching tasks. High diverse matching tasks avoid model fitting the data sample bias on a specific task, so that model can focus on learning the essential matching signals. Meanwhile, the prompt tokens obtained in the first step are added to the corresponding tasks to help the model distinguish different task-specific matching signals, as well as to form the basis prompt tokens for a new matching task. In this paper, we consider five common text matching tasks including document retrieval, open-domain question answering, retrieval-based dialogue, paraphrase identification, and natural language inference. Experimental results on eighteen public datasets show that Match-Prompt can improve multi-task generalization capability of PLMs in text matching and yield better in-domain multi-task, out-of-domain multi-task and new task adaptation performance than multi-task and task-specific models trained by previous fine-tuning paradigm.

CCS CONCEPTS
• Information systems → Retrieval models and ranking.

KEYWORDS
Text Matching, Prompt Learning, Multi-task Learning

ACM Reference Format:
Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2022. Match-Prompt: Improving Multi-task Generalization Ability for Neural Text Matching via Prompt Learning. In Proceedings of The 31st ACM Int’l Conference on Information and Knowledge Management (CIKM ’22), Oct. 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3511808.3557388

1 INTRODUCTION
Text matching is an important technique that can find accurate information from the huge amount of resources and plays a fundamental role in many downstream tasks, such as Document Retrieval (DR) [44], Open-domain Question Answering (QA) [7, 74], Retrieval-based Dialogue (RD) [70], Paraphrase Identification (PI) [13], and Natural Language Inference (NLI) [4]. Traditional text matching methods focus on measuring the word-to-word exact matching between two texts, such as TF-IDF [23] and BM25 [53]. Such methods follow the heuristics in information retrieval [16] and can generalize to different tasks by adjusting few parameters (e.g., $k_1$ and $b$). To introduce semantic and other complex relationships into text matching models, deep learning is applied to project texts into a joint and semantic space, namely neural text matching. Especially, PLMs like BERT [11] have brought great improvement, but the multi-task generalization of them has not been well explored. In general, most deep models can only achieve good performance...
when trained and evaluated on a single specific dataset, while have poor generalization ability to different domains and tasks [14]. To deal with the above problem of text matching models, we begin with the definition and evaluation of multi-task generalization. In this paper, the multi-task generalization of a text matching model can be divided into three levels: 1) in-domain multi-task, model has access to different datasets in multiple matching tasks, e.g., QA, RD, PI, NLI, and DR, and evaluates on each given datasets. It reflects the ability to learn from diverse matching tasks and apply it to current domain of each task. 2) out-of-domain multi-task, model can access the same datasets as the in-domain multi-task level, but is evaluated on the out-of-domain unseen datasets. It reflects the ability to learn from diverse matching tasks and apply it to current domain of each task. 3) new task adaptation, making a task invisible to the model, e.g., QA, and training model on other tasks, e.g., RD, PI, NLI, DR. Given few-shot examples in unseen task to adapt the model to this new task. It reflects the ability of new task generalization and this is the hardest generalization level for matching models.

The main challenges to obtaining a good generalized text matching model lay in two folds, 1) how to learn generalized matching signals? 2) how to apply a multi-task model to different matching tasks? From the learning perspective, the end-to-end supervised learning on task-specific datasets makes the model overemphasize the data sample bias and task-specific signals instead of the essential matching signals. For example in Figure 1, learning only the PI task causes the model to over-focus on the exact match signals, such as ‘cell’, ‘cycle’, as well as question mark which make it hard to generalize. However, there are matching signals that are shared across tasks, and in this paper, we classify matching signals into three categories, namely exact matching signals, semantic matching signals and inference matching signals. Exact matching signals focus on the word overlapping. It is shown that exact matching signals are important in all text matching tasks, for example, query term importance in DR [18, 47], lexical overlap features in PI [10] and bigram counts in QA [7]. Semantic matching signals measure semantic alignment. Many studies prove that introduce semantic information into DR [60] and PI [46] can improve the performance, especially for QA that dense retrieval [73] is better than sparse retrieval, e.g., DPR [25] and ORQA [31]. Inference matching signals infer implication relationship. NLI QA and RD have to infer implication relation between two texts. For example, RD needs to confirm the reply is a logical consequence of the post and dialogue history, and QA needs to pick out the answer that the question hardly contains. From the applying perspective, the difference between text matching tasks is the different fusion and utilization of these matching signals. Thus, if one matching model can capture the matching signals shared across tasks and combine them according to a specific matching task, its multi-task generalization ability will be improved.

Inspired by prompt learning, we propose a multi-task text matching model that exhibits strong multi-task generalization ability at all three levels. Prompt learning is a new PLMs tuning strategy that transforms various NLP tasks into the form of [MASK] prediction, which is more in line with the pre-training tasks of PLMs. In our method, we adopt a specialization-generalization training strategy and refer to it as Match-Prompt. In the specialization stage, prompts of each specific text matching task are obtained by specially designed prompt engineering that maps the description of different text matching tasks into only a few prompt tokens. In the generalization stage, the PLM is trained on mixed dataset consisting of different text matching tasks, and the prompt tokens from the first stage are used as task-discriminative tokens. Diversifying text matching tasks help the model explore the essential matching signals instead of overemphasizing the data sample bias and task-specific signals, to exploit shared information from tasks and address the first challenge. The prompt tokens of each task are added to the corresponding tasks to distinguish different tasks to answer the second challenge. Besides, unlike traditional prompt learning, it is shown that the learned task-specific prompts obtain the information to combine the matching signals, in particular, we can construct a proper prompt for new task by simply combining other learned prompts.

Experimental results on eighteen public datasets show that our method yields better in-domain multi-task, out-of-domain multi-task and new task adaptation performance compared to the baseline fine-tuning paradigm. The results also indicate that Match-Prompt has a stronger ability to distinguish tasks and utilize essential matching signals shared by multiple tasks. Both of them are beneficial for improving the multi-task generalization ability.

To sum up, our contributions are as follows:

- We collect eighteen datasets from diverse text matching tasks, which can serve as a benchmark for the multi-task generalization ability of text matching models.
- We propose a specialization-generalization training strategy based on prompt learning to disentangle general matching signal learning and specific task combination.
- We divide multi-task generalization into three levels and Match-prompt can enhance the multi-task generalization ability of BERT in text matching at these three levels.
and datasets are released at https://github.com/xsc1234/Match-Prompt.

2 RELATED WORK
In this section, we review the previous studies on text matching, mixed training for multiple tasks and prompt learning.

2.1 Text Matching
Traditional text matching methods, such as TF-IDF [23] and BM25 [53], represent texts as sparse vectors, each element in vectors corresponds to a term. They pay attention to the exact matching between two texts at term-level and can get good generalization performance. However, these methods suffer from the lexical gap and cannot capture the semantic information of the text.

Neural text matching models have brought great improvement with the help of deep neural networks and the training on a lot of in-domain data. We introduce the related work of neural text matching from three aspects: exact matching, semantic matching and inference matching. In exact matching, DRMM [18] considers query item importance and is suitable for DR. DrQA [7] combines TF-IDF weighted bag-of-words vectors and bigram to represent the text. This method shows good performance in open-domain QA. In semantic matching, interaction-based text matching models such as DeepMatch [38], ARC-II [21], MatchPyramid [46], ESIM [9] and Match-SRNN [64] can describe the fine-grained semantic matching relationship between two texts. They are often used in short text matching tasks such as PI, NLI and RD. Models based on single semantic text representation such as DSSM [60], CDSSM [57], and ARC-I [58] represent text as a dense vector to get the matching score. These models ensure retrieval efficiency and are often used in DR and QA. In inference matching that needs models to infer new information from the text, asymmetric neural text matching models such as DeepRank [47] and RE2 [71] are suitable. More recently, PLMs have achieved state-of-the-art text matching performance, such as BERT [11]. In this paper, we mainly discuss the cross-relation rather than the bi-encoder form of BERT because the former can achieve better matching effect.

The previous text matching models are only suitable for one of the specific tasks according to their specifically designed structures and mechanisms. Even though PLMs can be applied in multiple text matching tasks, only task-specific models that are fine-tuned on specific tasks can achieve good performance. When using the traditional fine-tuning method to train PLMs on mixed datasets of multiple matching tasks, the performance of the model drops seriously [14]. Our approach focuses on how to allow PLMs to distinguish different tasks among multiple tasks and fuse the shared information of each task during multi-task learning to obtain stronger multi-task generalization ability.

2.2 Mixed Training for Multiple Tasks
In order to enable a single model to complete multiple tasks, a common strategy is training the model on the mixed datasets of multiple tasks. A straightforward approach is directly mixing the datasets of each task without any task-specific marks and fine-tuning PLMs on mixed datasets. For example, Alon et al. [61] propose MultiQA. They train model on multiple datasets and find it leads to robust generalization in reading comprehension task. Jean et al. [41] propose a multi-task retrieval model for knowledge-intensive tasks. They use KILT [48] as the mixed datasets without any task-specific tokens. Some approaches add additional task-specific components containing parameters for each task. Liu et al. [36] propose MT-DNN that uses shared transformer encoder to encode multiple NLU tasks and add task-specific layers for each task. There are also some methods that transform multiple tasks into a unified question answering format such as MQAN [43], UnifiedQA [26] and T5 [52]. Li et al. [32] apply this method to information extraction, Chai et al. [6] and Puri et al. [50] use it in text classification.

The main tasks of the above methods are not text matching. In addition, some of these methods cannot fully utilize the differentiating marks, some introduce additional task-specific layers, and some need super-large-scale models. Compared with previous methods, Match-Prompt can improve the multi-task generalization capability of BERT in text matching and we do not need to add any task-specific layers when using the model for prediction.

2.3 Prompt Learning
Prompt learning is a novel tuning strategy for PLMs. It converts multiple NLP tasks into the form of [MASK] prediction using LM by adding template to the input texts. How to create the template is very important for prompt learning. On the whole, it can be divided into two categories, one is manual template engineering, the other is automated template learning [35]. In manual template engineering, Petroni et al. [49] design manual template to probe knowledge in LMs. Schick et al. [55] proposes Pattern-Exploiting Training (PET), which combines few-shot learning with templates, and converts specific tasks into cloze tasks. Although creating templates manually can solve many tasks, it has some issues: 1) making templates requires a lot of knowledge and experience; 2) manually designed templates may not be globally optimal [33]. To address these problems, there are many automated template learning methods proposed such as token-based gradient searching [59], mining training corpus [22], Prefix Tuning [33], and P-tuning [37]. Recently, there are some papers on pre-training or fine-tuning on multiple tasks, such as SP6T [63], PPT [17], ExT5 [1], FLAN [65], ZeroPrompt [69] and zero-shot for task generation [54]. Different from previous studies, we do not focus on how to improve the performance of prompt learning, but how to apply it to enhance generalization ability of text matching models. We also explore the relationships of task prompt tokens and find that a new matching task prompt can be constructed by linear combining other learned task prompt tokens.

3 MATCH-PROMPT
In this section, we elaborate on the details of Match-Prompt and construct a multi-task text matching model.

3.1 Overall Framework
Match-Prompt is a two-stage training strategy including specialization and generalization. In specialization stage, the descriptions of each task are mapped to a few prompt tokens by prompt engineering. These tokens contain the information of each task and can be used as the differentiating marks of each task in multi-task learning.
The second stage is generalization. Prompt tokens obtained in specialization stage are added to the input text of the corresponding specific task and BERT is trained on mixed datasets consisting of different tasks. In this stage, high diverse matching tasks avoid the model fitting the data sample bias on a specific task, so that model can focus on learning the common and essential matching signals. Besides, prompt tokens obtained in the specialization stage help the multi-task model better distinguish different tasks, which is beneficial for improving its multi-task generalization ability.

Compared with traditional fine-tuning paradigm, Match-Prompt offers several advantages. First, same as other prompt learning methods, Match-Prompt can make full use of the knowledge of PLMs and has better generalization ability [33, 49, 55]. Second, Match-Prompt can provide better differentiation for each task in mixed training and predicting of the model. Third, the differentiating marks of each task are added to the input text in the form of prompt tokens, and the word at [MASK] is predicted by PLM, which makes the use of differentiating marks more reasonable.

### 3.2 Specialization Stage: Prompt Engineering

In prompt engineering, we design three methods to map the description of the task to prompt tokens including manual prompt, continuous prompt, and hybrid prompt. To facilitate the description of these methods, we give a formal description of the input texts. For a pair of input texts \((x_1, x_2)\), the template of text matching is:

\[
x_{input} = [CLS] P_1 x_1 [SEP] P_2 x_2 [SEP] P_q [MASK].
\]

The prompt tokens corresponding to each task are \(P_1\), \(P_2\) and \(P_q\) in \(x_{input}\), which can help the model distinguish different tasks in the mixed training stage. They control PLM to use the knowledge in pre-training to predict the word at [MASK] and complete the corresponding text matching task, which can better reflect the key factors of different tasks in the calculation process of PLM, so as to provide the strong differentiating effect.

#### 3.2.1 Manual Prompt

Inspired by LAMA [49] that provides manually created cloze templates to probe knowledge in PLMs, manual prompt is proposed to perform many NLP tasks such as text classification [55] and text generation [56]. The manual prompts designed for different text matching tasks are shown in Table 1. In manual prompt, prompt tokens are entirely composed of natural language. From the left to right, \(P_1\) and \(P_2\) are used to tell BERT what type of \(x_1\) and \(x_2\) are respectively. \(P_q\) is an interrogative sentence that asks questions about the relationship between two texts according to the specific task. And the answers to these questions can only be “yes” or “no”, which are expected to be filled in [MASK]. In addition to texts, punctuation marks such as colon and question mark also play an important role in the description of the task. Compared with continuous and hybrid prompt, this method uses natural language as prompts without additional training on prompt tokens but requires human experience and the result is sub-optimal.

#### 3.2.2 Continuous Prompt

Based on P-tuning [37], we propose a method to automatically search prompts in continuous space. In this method, prompt is a trainable continuous vector. Different from P-tuning, our method focuses on improving the multi-task generalization capability of model in text matching. Our purpose of obtaining the prompt tokens of each task is not to improve its performance in prompt learning but to let the prompt tokens give better descriptions of each specific text matching task and use these descriptions to distinguish different tasks. Therefore, different from P-tuning, we make the following improvements:
Prompt encoder: After pre-training, the word embedding in BERT has been very discrete. It means that if we just optimize the word embedding of the prompt tokens in BERT, the search space will be very small, and it is easy to fall into the local optimal point. So, a bidirectional long-short term memory network (LSTM) is used to optimize all prompt token embedding in a completely continuous space [37]. Different from this, we use LSTM for $P_1$, $P_2$, and $P_q$ respectively. It is because LSTM assumes that there is a certain dependency between various tokens of prompt. That is, in bidirectional LSTM, the tokens of $P_1$, $P_2$, and $P_q$ are interdependent. But this unnecessary dependency limits the search space of vectors. So using separate LSTM for $P_1$, $P_2$, and $P_q$ can reduce the dependency between these three prompts and expand the search space.

Fix the prompt tokens: In P-tuning and many other prompt learning methods, prompt tokens are just added to input text and alter like other tokens in self-attention. Different from this, we let the embedding of trainable prompt tokens remain fixed in layers 1 to $k$ in the self-attention of these layers, prompt tokens can affect the calculation of BERT on input text, but the input text cannot affect prompt tokens. In the layers $k$ to $n$, they are alterable. It is because if we do not impose any restrictions on prompt tokens, the embedding of the prompt tokens is affected by other input words and updates in every layer. This is not conducive to obtaining an abstract description of specific text matching task, because there is no guarantee that the embeddings of prompt tokens are only determined by the prompt encoder due to the influence of other input words, which may cause the embedding to fit the data instead of the task. While on the other hand, it is inconsistent with the mask prediction task in the pre-training process of BERT if we fix prompt tokens in each layer, which may lead to sub-optimal results when using language model to predict the word at [MASK]. Therefore, an intermediate layer needs to be determined to balance the two to achieve optimal performance. Based on our experiment, we find that the 11-th layer is an optimal boundary layer. We balance the description of the task and the interaction with other words in the text. In this way, prompt tokens can achieve better control of BERT and describe the task more comprehensively. The parameters of prompt encoders are updated through back-propagation thereby adjusting the embedding of the prompt tokens.

Let $P_{init}$ is a randomly initialized vector used as the input to prompt encoder. $MLP_1(LSTM_1)$, $MLP_2(LSTM_2)$, $MLP_q(LSTM_q)$ are the encoders of the corresponding prompt. Each encoder consists of a bidirectional LSTM and multilayer perceptron. These three encoders are optimized to obtain the continuous prompts respectively. Take $P_1$ as an example, the embedding is obtained by:

$$P_1 = MLP_1(LSTM_1(P_{init})).$$

Table 1: Design of manual prompts. Handcrafted prompts are described in natural language.

| Task                                | $P_1$         | $P_2$         | $P_q$         |
|-------------------------------------|---------------|---------------|---------------|
| Document Retrieval                  | Query:        | Passage:      | Dose the passage include the content matches the query? |
| Question and Answer                 | Question:     | Passage:      | Does the passage include the answer of the question?   |
| Retrieval-based Dialogue            | The first sentence: | The second sentence: | Can the second sentence reply to the first sentence? |
| Paraphrase Identification           | The first sentence: | The second sentence: | Do these two sentences mean the same thing? |
| Natural Language Inference          | Premise:      | Hypothesis:   | Can the hypothesis be concluded from the premise?      |

Given the $x_{input}$ and label $y$, the loss function can be expressed as $\text{Loss}(M(x_{input}, y))$ where $M$ is the per-trained language model. $\theta^*$ are the parameters of prompt encoder, the optimization of $\theta^*$ is:

$$\theta^* = \arg\min_{\theta} \text{Loss}(M(x_{input}, y)).$$

3.2.3 Hybrid Prompt. In hybrid prompt, for $P_q$, instead of using prompt encoder to generate a continuous vector, we express it as a natural language: "Do these two sentences match?". It is because in the template of continuous prompt, text matching is converted into predicting the word at [MASK]. If the result is "yes", these two texts match, otherwise they do not. In order to predict the word, the parameters of the three prompt encoders for $P_1$, $P_2$, and $P_q$ need to be updated. However, there is no prior information about the task other than training data. This leads to that although the prompt encoder can minimize loss during training, the task it describes may not be text matching. It just makes prompts control PLMs to extract features from the input texts, and output the corresponding "yes" or "no", which is not conducive to generalizing to other datasets. Expressing $P_q$ as task-relevant natural language can control the task closer to text matching and facilitate $P_1$ and $P_2$ to describe different matching tasks, which is beneficial for improving generalization ability. The template contains both manual prompt ($P_q$) and continuous prompt ($P_1$,$P_2$). We call this method hybrid prompt. The training architecture is shown in the left of Figure 2.

3.3 Generalization Stage: Mixed Training

After prompt engineering in specialization stage, the descriptions of each specific task are mapped to prompt tokens. The second stage is generalization, in this stage, we guide BERT to learn the common and essential signals of text matching through the mixed datasets of multiple tasks, so as to improve its multi-task generalization ability. In addition, prompt tokens help multi-task models distinguish different tasks, adapt to task-specific matching signals and achieve good generalization.

We mix the datasets from the different tasks and tune BERT on these mixed datasets. The prompt tokens of each task come from specialization stage, which can be manual, continuous or hybrid prompt. We add the prompt tokens to the input text of the corresponding task according to the template shown in $x_{input}$. In order to ensure that the multi-task model is not biased towards a certain task, we sample the dataset of each task to achieve the relative balance of the amount of data for each task. Besides, we rearrange
the data for each task, so that each epoch can see roughly the same number of samples from each task. We evaluate the performance every epoch and use early stopping to stop training when the performance is lower than the best result in history for 5 consecutive epochs. In order to make the roles of \( P_1, P_2, \) and \( P_q \) in each layer of BERT consistent with the first stage, their calculations should be the same as the corresponding three prompt engineering methods. In this stage, all parameters of BERT are trainable. When using the model for predicting, we only need to follow this paradigm and add different prompts according to different tasks to achieve multi-task text matching by single model without any task-specific layers. The architecture is shown in the right of Figure 2.

3.4 Training and Evaluation

We use prompt learning to train Match-Prompt, the key of which is updating the parameters to maximize the probability that the language model correctly predicts the word at [MASK]. Let \( M \) be the pre-trained language model. The vocabulary of \( M \) is \( \mathcal{V} \). \( \mathcal{Y} \) is the label set in this task and \( y \) is each label. We can design a verbalizer to map label to the word in vocabulary and mark it \( v \). If the [MASK] is filled by “yes”, the label is 1, if it is filled by “no”, the label is 0. \( M(w|\text{xinput}) \) is used to represent the probability of filling [MASK] by \( w \in \mathcal{V} \). Relationship between \( \text{xinput} \) and \( y \) is modeled as:

\[
s(y|x_{\text{input}}) = M(v(y)|x_{\text{input}}),
\]

\[
P(y|x_{\text{input}}) = \frac{e^{s(y|x_{\text{input}})}}{\sum_{y' \in \mathcal{Y}} e^{s(y'|x_{\text{input}})}}.
\]

The loss function of our method is cross entropy:

\[
\mathcal{L} = - \sum_{i=1}^{N} [y_i \log P_i + (1 - y_i) \log (1 - P_i)].
\]

In the continuous and hybrid prompt of specialization stage, the trainable component is prompt encoder and PLM is fixed. In generalization stage, the trainable component is PLM and the prompt tokens of each task come from specialization stage. We can directly combine these prompt tokens and texts into \( x_{\text{input}} \) and input them to PLM for training and predicting without any prompt encoders.

After training, we can get the matching function \( F \), the matching process can be simplified to:

\[
y = F(x_1, x_2),
\]

\( y \in \{0, 1\} \), indicates whether \( x_1 \) and \( x_2 \) match. In ranking tasks such as DR, QA and RD, we use the probability that the word of [MASK] is predicted to be "yes" minus the probability that it is predicted to be "no" as the relevance between texts:

\[
\text{REL} = M(\text{yes}|x_{\text{input}}) - M(\text{no}|x_{\text{input}}).
\]

4 EXPERIMENTS

In this section, we introduce our experiment settings and results.

4.1 Experiment Settings

4.1.1 Research Questions. In Section 1, multi-task generalization is divided into three levels. We compare the performance of Match-Prompt with the traditional fine-tuning paradigm at these three levels in experiment. Figure 3 describes these three levels.

- **In-domain Multi-task Performance.** In order to test the ability of multi-task model to distinguish and optimize different tasks and utilize the shared information of tasks, given mixed datasets of multiple tasks, the model performs multi-task learning on the training set and test its performance on each task on the testing set.

- **Out-of-domain Multi-task Performance.** In order to investigate the out-of-domain generalization ability of multi-task models, we test the zero-shot learning ability of the multi-task model on unseen datasets.

- **New Task Adaptation.** In order to investigate the ability of multi-task model to use information from other tasks to facilitate low-resource adaptation, we use leave-one-out to perform few-shot learning on new tasks.

4.1.2 Datasets. The mixed datasets consist of six datasets from five tasks including Document Retrieval, Open-domain Question Answering, Retrieval-based Dialogue, Paraphrase Identification, and Natural Language Inference. We have sampled the datasets of each task to maintain a relative balance between each task. The details of mixed datasets are listed in Table 2. After getting the model, we observe its performance on the test dataset of each task. Besides, in order to explore the out-of-domain generalization capability of multi-task text matching model, we test the performance on the datasets that the model has not been trained on. These datasets are listed in Table 3. To ensure the same amount of data as the baseline, we keep the data used for training continuous and hybrid prompt in prompt engineering consistent with the generalization stage.

4.1.3 Training. The model used in the experiment is BERT-base (109M)\(^1\). The gradient descent method Adam [27] with learning rate \( 10^{-3} \) is used to optimize the objective and batch size is 16. We evaluate the performance of every epoch and use early stopping to avoid overfitting. In continuous and hybrid prompt, the lengths of \( P_1, P_2, \) and \( P_q \) (only in continuous prompt) are 6, 6, 5 respectively.

\(^1\)https://huggingface.co/bert-base-uncased/tree/main
Table 2: Details of each task in mixed training datasets.

| Task | Dataset | Train(sampled) | Test |
|------|---------|----------------|------|
| DR   | MQ2007 (MQ07) [51] | 40,000 (q-d pairs) | 338 (queries) |
| QA   | TrivaQA (Triva) [24] | 40,000 (q-d pairs) | 8,837 (questions) |
| RD   | DailyDialog (DG) [34] | 40,000 (dialogue pairs) | 3,209 (dialogues) |
| PI   | QQP [2] + MSR P [12] | 35,924 + 4,076 | 50,016 + 1,752 |
| NLI  | MultiNLI (MNLIN) [66] | 40,000 | 39,296 |

Table 3: Datasets to test the out-of-domain multi-task ability.

| Task | Datasets |
|------|----------|
| DR   | ClueWeb09-B (CW) [72], Robust04 (RB04) [62], Gov2 |
| NLI  | CuratedTREC (Trec) [2], Natural Questions (NQ) [28], WebQuestions (WQ) [3] |
| QA   | PARADE [20], TwitterURL (TURL) [29] |
| RD   | Reddit, AmazonQA (AQ) [19] |

4.1.4 Specical Pre-processing. For QA, we construct q-a pairs in training set as described in DPR [25]. In order to balance the number of positive and negative samples, we sample three positive and three negative samples for each question in training. These three negative samples consist of two hard negative and one gold passage (positive passage paired with other questions). For PI, the size of public datasets is small, so we select two datasets including MSRP and QQP. For NLI, we convert this task into a binary classification task that whether hypotheses can be inferred from the premises. For DR and QA, we use BM25 to retrieve items and use our model to re-rank them. For RD, the candidate set of the sentence consists of one positive and nine negative samples.

4.1.5 Baselines and Measures. PLMs improve the performance of interaction-based text matching. For example, MonoBert [8] takes the concatenating query and document as the input and feeds the embedding of [CLS] into a feed-forward network to judge the relevance, which is the traditional fine-tuning paradigm of BERT. In experiment, we want to show that the text matching model trained with Match-Prompt can get better multi-task generalization ability than the traditional fine-tuning paradigm. Some methods [8, 15, 30, 36, 40, 45, 68] are not considered because they are just variants based on this fine-tuning paradigm for different matching tasks, which are inconsistent with our motivation. So the most important baseline in the experiment is Fine-tuning.

As for the multi-task learning method for Fine-tuning, we introduce three methods. Fine-tuning\textsubscript{multi}, which uses traditional fine-tuning paradigm to train BERT without any task-specific marks. Fine-tuning\textsubscript{mark}, which adds the task-specific marks to the input text. MT-DNN [36] adds the task-specific feed-forward networks for each task, we reproduce it on our mixed datasets. This method introduces additional parameters, and the number of parameters increases with the number of tasks, but Match-Prompt does not need any task-specific layers during inference. There is also a multi-task learning framework that converts each task into a unified question answering format [6, 43, 50, 52]. We reproduce this framework on our mixed datasets using GPT2-base (117M)\textsuperscript{4} and T5-base (220M)\textsuperscript{5} and call them MTL\textsubscript{GPT2} and MTL\textsubscript{T5} respectively. Since T5 has been pre-trained on multiple supervised downstream tasks that will be tested in our experiment, which is unfair for comparison, we choose T5 1.1 that is only pre-trained on unsupervised datasets. We also reproduce MQAN [43] on our tasks, which is a classic QA-based multi-task model. BM25 [53] also has strong multi-task generalization ability, we use it as one of the baselines. In multi-task training, there are some tricks about data sampling, loss construction and task scheduling, which can be used by both baselines and Match-Prompt, so we do not compare them in detail. In order to compare the performance of the multi-task model enhanced by Match-Prompt with the task-specific model, the models (i.e. Fine-tuning\textsubscript{sp}) are specifically trained on the dataset corresponding to each task listed in Table 2, and tested on the corresponding task.

For our method, Match-Prompt\textsubscript{hp}, Match-Prompt\textsubscript{cp}, Match-Prompt\textsubscript{mp} correspond to manual, continuous and hybrid prompt in text matching described in Section 3.

We use Accuracy and F1-score to evaluate NLI and PI [9, 67]. For QA and RD, we use P@1 and MRR [64]. For DR, we use NDCG [39].

4.2 Main Results

4.2.1 In-domain Multi-task Performance. We train multi-task model on the mixed datasets listed in Table 2 and test the performance of the model on each specific task. The results are shown in Table 4. Compared with other multi-task models, Match-Prompt shows improvement in multi-task text matching. All three prompt engineering methods exceed other multi-task models in most tasks, and hybrid prompt achieves the best results on most datasets. It is worth noting that Match-Prompt\textsubscript{hp} surpasses Fine-tuning\textsubscript{sp} in every task. We can infer from these results that Match-Prompt can improve the in-domain multi-task performance of BERT in text matching. Match-Prompt enables BERT to better utilize the matching signals shared by each task and the differentiating marks of each task are added to the input text in the form of prompts to predict the word at [MASK], which makes the use of differentiating marks more reasonable, thereby improving its multi-task performance.

4.2.2 Out-of-domain Multi-task Performance. We test the multi-task model trained on the mixed datasets (listed in Table 2) on unseen datasets (listed in Table 3). The experimental results in Table 5 indicate that Match-Prompt\textsubscript{hp} gets the best performance on most datasets. This experiment further demonstrates that Match-Prompt greatly improves the generalization ability of text matching models to different tasks and datasets. Prompt learning is closer to the pre-training process, so it preserves the knowledge of the BERT. In addition, prompt tokens have the description for each specific task, which facilitates migration on different datasets and training on mixed datasets consisting of multiple tasks can improve the generalization robustness of the model [61], so Match-Prompt can improve the generalization ability of the model. The performance of other multi-task models drops seriously than Fine-tuningsp but Match-Prompt\textsubscript{hp} exceeds it. This shows that our method is conducive to fusing and exploiting the knowledge of each task and

\textsuperscript{4}https://huggingface.co/gpt2
\textsuperscript{5}https://huggingface.co/google/t5-v1_1-base

\textsuperscript{3}https://www.kaggle.com/c/quora-question-pairs
\textsuperscript{2}https://ir-datasets.com/gov2.html
Table 5: Multi-task generalization ability of the model on unseen datasets listed in Table 3. Boldface indicates the best results of multi-task models and the results over Fine-tuning are denoted as ‘*’. Results with significant performance improvement with p-value ≤ 0.05 compared with all multi-task models of baselines are denoted as ‘+’.

| Method            | TriviQA_p | DAILYDialog p | MSR+QQP p | MultiNLIE p | MQ2007DR |
|-------------------|-----------|---------------|-----------|-------------|----------|
|                  | P@1 (%)   | MRR(%)        | Acc(%)    | F1(%)       | NDCG@10% |
| Fine-tuning_p     | 47.17     | 55.62         | 78.59     | 84.86       | 83.42    | 79.66    | 84.40    | 77.35    | 50.60    |
| Multi-task model  |           |               |           |             |          |
| for all tasks     |           |               |           |             |          |
| BM25              | 46.88     | 54.67         | 35.81     | 50.39       | 70.31    | 77.62    | 66.37    | 57.23    | 41.40    |
| MQAN              | 7.33      | 16.72         | 68.06     | 74.92       | 60.00    | 64.66    | 61.72    | 50.31    | 30.35    |
| MTL_GPT2          | 34.02     | 43.29         | 58.58     | 70.12       | 65.20    | 62.15    | 67.59    | 58.37    | 45.72    |
| MTL_T5            | 36.24     | 46.63         | 62.49     | 74.79       | 71.54    | 74.60    | 77.28    | 64.73    | 48.63    |
| MT-DNN            | 46.01     | 54.46         | 75.10     | 82.05       | 81.50    | 77.10    | 82.33    | 73.92    | 47.94    |
| Fine-tuning_multi | 46.19     | 54.95         | 74.29     | 80.85       | 77.13    | 81.45    | 72.77    | 47.59    |          |
| Fine-tuning_sp    | 46.05     | 54.72         | 74.80     | 81.95       | 81.47    | 75.44    | 82.55    | 74.01    | 49.94    |
| Match-Prompt      | 47.97     | 55.98         | 78.01     | 84.58       | 83.36    | 79.42    | 84.56    | 75.78    | 51.52    |
| Match-Prompt_sp   | 47.97     | 55.98         | 78.01     | 84.58       | 83.36    | 79.42    | 84.56    | 75.78    | 51.52    |

Table 4: Performance of multi-task model trained on the mixed datasets listed in Table 2. Boldface indicates the best results of multi-task models and the results over Fine-tuning are denoted as ‘*’. Results with significant performance improvement with p-value ≤ 0.05 compared with all multi-task models of baselines are denoted as ‘+’.

| Method                | TriviQA_p | DAILYDialog_p | MSR+QQP_p | MultiNLIE_p | MQ2007DR |
|-----------------------|-----------|---------------|-----------|-------------|----------|
|                       | P@1(%)    | MRR(%)        | Acc(%)    | F1(%)       | NDCG@10% |
| Fine-tuning_p         | 47.17     | 55.62         | 78.59     | 84.86       | 83.42    | 79.66    | 84.40    | 77.35    | 50.60    |
| Multi-task model      |           |               |           |             |          |
| for all tasks         |           |               |           |             |          |
| BM25                  | 46.88     | 54.67         | 35.81     | 50.39       | 70.31    | 77.62    | 66.37    | 57.23    | 41.40    |
| MQAN                  | 7.33      | 16.72         | 68.06     | 74.92       | 60.00    | 64.66    | 61.72    | 50.31    | 30.35    |
| MTL_GPT2              | 34.02     | 43.29         | 58.58     | 70.12       | 65.20    | 62.15    | 67.59    | 58.37    | 45.72    |

4.2.3 New Task Adaptation. We also explore the few-shot learning capability of the multi-task model in new tasks. For the five specific text matching tasks, we select four of them for mixed training and call this model multi-task_sp and the remaining one is used as the new task of the multi-task_sp (leave-one-out). In order to ensure the number of samples used by each method is consistent, in Match-Prompt_sp, the prompt tokens of the new task are obtained by the weighted sum of the prompt tokens of the other four tasks, and the weight of each task is learnable variable in few-shot learning, reflecting the similarity of the matching signals between two tasks. The w/o fusion is do not fuse the prompts of other tasks and train directly from scratch under few-shot settings. In order to better demonstrate the promotion effect of other tasks on low-resource learning, we also use Fine-tuning_sp to directly perform few-shot learning on each task. For the new task, we select 32 positive and 32 negative samples for training and observe the performance of the model on testing set. The experimental results are shown in Table 6. The models based on multi-task_sp get better few-shot learning performance than Fine-tuning_sp. Besides, Multi-task_sp based on Match-Prompt has better few-shot learning ability in new tasks than that based on other methods. This indicates that the information of other tasks can improve the adaptability of the model to new tasks and Match-Prompt performs best. In addition, the prompts of other tasks contain descriptions of essential matching signals, and the integration of them can further promote few-shot learning.

4.3 Model Analysis. The quality of learned prompt tokens is very important for the multi-task generalization of text matching models. We first measure the specificity of the information stored in the prompt tokens by avoiding interference between different tasks in multi-task learning. Match-Prompt_sp performs better than Match-Prompt_p, which proves that it is effective to use natural language to control continuous prompt tokens learning to describe the task.
Table 6: Few-shot learning performance of the models. Each task in this table is a new task not included in mixed datasets for multi-task adaptation (leave-one-out). Results with significant performance improvement with p-value $\leq 0.05$ compared with all multi-task models of baselines are denoted as "$+$".

| Method          | Trivia MRR | DG MRR | MSRP Acc | NLI Acc | MNLI Acc | MQ07 Acc | NDCG@10 |
|-----------------|------------|--------|----------|---------|----------|----------|---------|
| Fine-tuning      | 24.13      | 38.97  | 64.93    | 35.76   | 33.87    |          |         |
| Few-shot learning based on multi-task adaptation on each task |           |        |          |         |          |          |         |
| MTL$GPT_2       | 35.12      | 41.26  | 69.21    | 48.53   | 34.12    |          |         |
| MTL$T_5         | 35.85      | 45.92  | 71.13    | 48.78   | 35.34    |          |         |
| MT-DNN          | 37.39      | 50.97  | 68.17    | 59.16   | 35.12    |          |         |
| Fine-tuning$multi$ | 40.11  | 51.03  | 67.01    | 52.50   | 36.11    |          |         |
| Fine-tuning$mix$ | 37.49      | 50.06  | 70.90    | 59.98   | 36.02    |          |         |
| Match-Prompt$hg$ | 45.13      | 52.59  | 74.15    | 62.02   | 37.18    |          |         |
| - w/o fusion    | 42.37      | 51.09  | 71.01    | 60.92   | 36.28    |          |         |

Table 7: Illustrate the ability to distinguish tasks.

| Method          | As QA Task |  | As PI Task |  |
|-----------------|------------|  |------------|  |
|                 | Trec $p_{a\rightarrow q}$ |  | WQ $p_{a\rightarrow q}$ |  |
|                 | NQ $p_{a\rightarrow q}$ |  | Trec $p_{q\rightarrow a}$ |  |
|                 | NQ $p_{q\rightarrow a}$ |  |   |  |
| MTL$GPT_2$      | 35.89      | 10.52 | 14.82     | 78.25   |
|                 | 90.72      | 94.53 |          |          |
| MTL$T_5$        | 38.35      | 6.39  | 9.03      | 76.07   |
|                 | 97.09      | 95.45 |          |          |
| MT-DNN          | 27.76      | 29.11 | 19.94     | 98.67   |
|                 | 99.56      | 97.90 |          |          |
| Fine-tuning$multi$ | 20.91  | 13.47 | 4.86      | 79.09   |
|                 | 86.53      | 95.14 |          |          |
| Fine-tuning$mix$ | 40.75      | 34.82 | 10.49     | 63.70   |
|                 | 72.06      | 97.37 |          |          |
| Match-Prompt$hg$ | 71.98      | 76.68 | 35.92     | 100.00  |
|                 | 100.00     | 100.00 |          |          |

Figure 4: Task relationships using prompt tokens. The sum of each row is 1, and the value (keep two decimal places) represents the proportion of the cosine similarity or weight of other tasks to the task of each row.

5 CONCLUSION

We introduce prompt learning into multi-task text matching and propose a specialization-generalization training strategy, namely Match-Prompt, to improve the multi-task generalization capability of BERT in text matching tasks. In specialization stage, the descriptions of different matching tasks are mapped to only a few prompt tokens. In generalization stage, text matching model explores the essential matching signals by being trained on diverse multiple matching tasks. In the inference, the prompt tokens help the model distinguish different tasks and can be combined to adapt to new tasks based on the essential matching signals. Experimental results show that our method improves multi-task generalization ability of BERT in text matching and the performance of our multi-task model on each task can surpass the task-specific model. Our work indicates that there are not only different but also essential and common matching signals in different text matching tasks, Match-Prompt can help PLMs leverage these signals and distinguish different tasks to get stronger in-domain multi-task, out-of-domain multi-task and new task adaptation performance than traditional fine-tuning paradigm.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants No.61906180, U21B2046, and Beijing Academy of Artificial Intelligence (BAAI).
