Aggregating Bidirectional Encoder Representations Using MatchLSTM for Sequence Matching

Bo Shao[†‡] **Yeyun Gong**[‡] **Weizhen Qi**[§‡] **Nan Duan**[‡] **Xiaola Lin**[†]
[†]Sun Yat-sen University
[‡]Microsoft Research Asia
[§]University of Science and Technology of China
shaobo2@mail2.sysu.edu.cn
{yegong, nanduan}@microsoft.com
weizhen@mail.ustc.edu.cn
linxl@mail.sysu.edu.cn

Abstract

In this work, we propose an aggregation method to combine the **B**idirectional **E**ncoder **R**epresentations from **T**ransformer (BERT) with a MatchLSTM layer for Sequence Matching. Given a sentence pair, we extract the output representations of it from BERT. Then we extend BERT with a MatchLSTM layer to get further interaction of the sentence pair for sequence matching tasks. Taking natural language inference as an example, we split BERT output into two parts, which is from premise sentence and hypothesis sentence. At each position of the hypothesis sentence, both the weighted representation of the premise sentence and the representation of the current token are fed into LSTM. We jointly train the aggregation layer and pre-trained layer for sequence matching. We conduct an experiment on two publicly available datasets, WikiQA and SNLI. Experiments show that our model achieves significant improvement compared with state-of-the-art methods on both datasets.

1 Introduction

Text sequence matching aims to infer the relationship of two text sequences which is a critical problem used in many NLP tasks such as answer selection (Yang et al., 2015), natural language inference (Bowman et al., 2015), and paraphrase identification (Dolan et al., 2004).

Recent years, various deep learning models have been proposed for text sequence matching tasks (Wang and Jiang, 2016; Tay et al., 2018; Tymoshenko and Moschitti, 2018; Kim et al., 2018). Most previous works are learned from task specific supervised datasets, which are always limited to the amount, due to the cost of annotation. Recently, learning a general representation of words based on language models attracts great attention, and are successfully applied to a range of NLP tasks (McCann et al., 2017; Peters et al., 2018, 2017; Alec Radford, 2018; Devlin et al., 2018). In these works, they first learn general language representations through training a language model using large scale unlabeled data. Then they use these representations to the downstream tasks. Previous works show that the representations with linguistic information perform obvious effectiveness and robustness applied to downstream tasks.

In these methods, there are two typical strategies using pre-trained models, feature-based (Peters et al., 2017) and fine-tuning approach (Devlin et al., 2018; Radford et al., 2018). Our model is based on the fine-tuning approach and use BERT (Devlin et al., 2018) as our pre-trained language model. BERT has been directly used in many tasks and achieved significantly improvement compared with other methods. However, BERT only uses a standard softmax layer for many different specific tasks. Intuitively, for more complex tasks, like sequence matching in this work, we need a better strategy for extending the BERT to infer the relationship of the sequence pair. Thus, we incorporate a MatchLSTM layer (Wang and Jiang, 2015) to BERT, and make it better to adapt to sequence matching.

In our model, we design a MatchLSTM layer to aggregate representations of words in the sequences, and then we concatenate the hidden state of first token in BERT to the output of MatchLSTM (Wang and Jiang, 2015). This method can combine the features extracted from BERT and MatchLSTM effectively, and it is much better than directly using the outputs of BERT as pre-trained word representations to MatchLSTM.

The main contributions of this work are as follows:

- We extend BERT with a MatchLSTM layer. It can be applied to a variety of sequence matching tasks.

- We compare various aggregation layers to extend BERT, with the MatchLSTM layer achieving the best performance. It gets more powerful representations than the original output from BERT for sequence matching.

- Experiments on WikiQA and SNLI datasets show that our model achieves better performance than state-of-the-art methods.

2 Related Work

Sequence Matching task has attracted lots of attention in the past decades. There are many related works on Question Answering(QA) (Yin et al., 2015; Tay et al., 2018; Wang et al., 2017; Tymoshenko and Moschitti, 2018; Min et al., 2017), Natural Language Inference(NLI) (Peters et al., 2018; Kim et al., 2018; Radford et al., 2018) and so on. (Yin et al., 2015) use attention mechanism with convolutional layer to model sentence pairs. In (Tymoshenko and Moschitti, 2018), they combine the similarity features of members within the same pair and traditional sentence pair similarity and achieve state-of-the-art results on several answer selection datasets including WikiQA (Yang et al., 2015). (Peters et al., 2018) and (Radford et al., 2018) incorporate pre-trained language models to text sequence matching task and achieve performance improvement on SNLI (Bowman et al., 2015).

Recently, pre-trained language models have been successfully used in NLP tasks. ELMo (Peters et al., 2017) is trained as a bidirectional language model. OpenAI GPT (Alec Radford, 2018) uses a basic left-to-right transformer to learn a language model. BERT, used in this paper, is based on the architecture of a bidirectional Transformer and per-trained on Masked Language Model task and Next Sentence Prediction. Using the pre-trained parameters, BERT (Devlin et al., 2018) achieves state-of-the-art performance on the GLUE benchmark (Wang et al., 2018) and SQuAD 1.1 (Rajpurkar et al., 2016) by fine-tuning in corresponding supervised data.

In this work, we design a sequence matching model based on BERT.

3 Our Model

In this section, we will introduce our sequence matching model. Our model is combined BERT encoder with the MatchLSTM layer.

3.1 BERT Encoder

Given two natural language sentences A, B, the tokens of both sentences are packed into a token sequence S as "[CLS] A [SEP] B [SEP]", where [CLS], [SEP] are special tokens. The representation of each token is the sum of three types of embedding: 1). **WordPiece embedding**(Wu et al., 2016) is used in BERT to represent the input sequence, which has 30000 token vocabulary. The tokens in S are split into several word pieces with "##". 2). **Position embedding** is used to cover position information in transformer. 3). **Segment embedding** is used to indicate whether the current token is from A or B.

Then the representation of S is fed into the BERT encoder which is a multi-layer bidirectional Transformer(Vaswani et al., 2017). The implement of its architecture can refer to (Vaswani et al., 2017; Devlin et al., 2018). BERT encodes the input token sequence S into a context aware representation $h = \{h_0, h_1, ..., h_{|S|-1}\}$. Then h is used to a match layer for downstream supervised tasks.

For downstream tasks of text sequence matching, BERT extract the final hidden state of the first token, h_0, corresponding to the special token [CLS]. The label probabilities are computed with h_0 by a standard softmax layer. Finally, all of the parameters of the model are fine-tuned to maximize the log-probability of the correct label in specific tasks.

3.2 MatchLSTM Layer

Although BERT has proved its effectiveness for text sequence matching tasks, there still exist two limitations. First, the two sentences can only interact through a general attention mechanism in the transformer, which is not enough for complex matching tasks. Secondly, BERT only uses the first token hidden state in sequence matching tasks such as QA, MNLI in (Devlin et al., 2018), ignoring the final hidden state $h_i, i > 0$ in other positions. To overcome the two limitations, we add a bidirectional MatchLSTM network (Wang and Jiang, 2015), which computes the word-by-word matching results at each position for input sentence pairs.

To get the representation of each input sentence, we first split the final hidden state h into h^A and h^B as the representations of the sentence pair, A and B, according to its position information. Then we apply a bidirectional MatchLSTM network with the input of h^A and h^B. At each position i of the tokens in sentence B, the MatchLSTM layer first uses a standard word-by-word attention mechanism to obtain attention weight $a^i = \{a^i_0, a^i_1, ..., a^i_{|A|-1}\}$ over sentence A and compute a weighted sum of sentence A as c_i, which can be formulated as,

$$e^i_j = u^T tanh(W^A h^A_j + W^B h^B_i + W^s s_{i-1} + b_e)$$
$$a^i_j = \frac{e^i_j}{\sum_{k=1}^{|p|} e^i_k}; c_i = \sum_{j=0}^{|A|} a^i_j h_j \tag{1}$$

where u, W^A,W^B,W^s and b_e are trainable parameters, s_i is the hidden state at the ith position of B. The i,j in Equation 1 represents ith word in h^A and jth word in h^B, and e^i_j is the attention score between the two words. We concatenate c_i with the current token of sentence B as:

$$\begin{aligned} r_i &= [h^B_i : c_i] \\ s_i &= LSTM(r_i, s_{i-1}) \end{aligned} \tag{2}$$

Then, we concatenate the last hidden state of MatchLSTM layer $s_{|B|}$ with the first hidden state h_0 of BERT encoder and feed it into the output softmax layer,

$$\begin{aligned} f &= [s_{|B|} : h_0] \\ P &= softmax(W^f f + b_f) \end{aligned} \tag{3}$$

where W^f and b_f are trainable parameters, P is the label probability distribution.

3.3 Loss Function

Finally we compute the loss function to maximize the log-probability of corresponding label $y \in \{0, 1, .., m-1\}$, m is the number of target label types. It can be formulated as,

$$\mathcal{L} = -log(P_i)[i = y] \tag{4}$$

3.4 Other Aggregation Layers

In this section, we introduce two other general methods to utilize all the final hidden states of BERT encoder.

3.4.1 Average Layer

To incorporate all the hidden states of BERT, a basic idea is to sum all final hidden states h directly as the feature vector f_{avg}. It will be used to calculate the probability of final label by Eq.3.

3.4.2 Convolutional Layer

We also evaluate the performance using the convolutional layer with max pooling. We set several fixed window sizes to extract feature vectors. For each windows size l, we use a kernel matrix $W^l \in R^{l \times d}$ and a non-linear function to operate on the hidden vector h from BERT encoder. The output of one operation is a local feature which can be computed as follows:

$$z_i = g(W^l \cdot v[i : i + l] + b^l) \tag{5}$$

where W^l and b^l are trainable parameters, g is a non-linear function. We conduct this operation on different hidden vectors $v_{1:l}, v_{2:l+1}, ..., v_{n-l+1:n}$ and get a set of local features $z = \{z_1, z_2, ..., z_{n-l+1}\}$. We then extract a maximum feature from the local features z generated by one kernel through the max pooling layer and get feature vector f_{cnn}. We concatenate f_{cnn} with the first hidden state h_0 as above to get the final feature vector and calculate the probability distribution of labels.

4 Experiment

We conduct experiments on the WikiQA and SNLI corpuses. We use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as the evaluation metrics on WikiQA. On SNLI, we use accuracy(ACC) as our evaluation metric.

4.1 Datasets

WikiQA dataset (Yang et al., 2015) is a sentence-level QA dataset, containing 2.0k/0.3k/0.6k train/dev/test examples. Each sample contains a query in Bing's log and a snippet of a Wikipedia retrieved by Bing. Each query contains 18.6 candidate sentences on average. The task is to select an answer from candidate sentences for each query.

The Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) is a large-scale entailment classification task, containing 570K human annotated sentence pairs. Each sample consists of a sentence pair and a label from relationship class (entailment, neutral, contradiction, and -). We remove samples in ”-” class same as previous works, since they do not have a consensus label among the annotators. For training, development and test sets, we use the same split with (Bowman et al., 2015) to report our results.

4.2 Settings

We use the pre-trained model of BERT$_{BASE}$ (Our goal is to prove the effectiveness of our aggregation method, so we do not use the BERT$_{LARGE}$ which is time and resource consuming), the number of Transformer blocks is 12, and dimension of all hidden state is 768 in our model. The batch size of the model is 32 and the initial learning rate is 2e-5. We set the dropout rate to 0.5.

Except for using state-of-the-art methods as our baseline, we also list variants of aggregated layers with BERT in the experiments. "BERT+AVG" is the model described in 3.4.1. "BERT+CNN" is the model proposed in 3.4.2. "BERT" represents the BERT with pre-trained parameters as initialization. "BERT$_{no\ init}$" denotes BERT using random parameters as initialization. The "-FHS" means the model does not use the hidden state of first token in BERT encoder.

4.3 Experiments on WikiQA

Methods	**MAP**	**MRR**
ABCNN (Yin et al., 2015)	69.21	71.08
MULT (Wang and Jiang, 2016)	74.33	75.45
HyperQA (Tay et al., 2018)	72.70	71.20
BiMPM (Wang et al., 2017)	71.80	73.10
PTK (Tymoshenko and Moschitti, 2018)	73.34	74.68
SQuAD* (Min et al., 2017)	83.20	84.58
BERT$_{no\ init}$	58.91	60.09
BERT (Devlin et al., 2018)	80.15	82.31
BERT+AVG	81.95	83.26
BERT+CNN	81.23	83.16
BERT+MatchLSTM	**83.53**	**85.13**
-FHS	70.46	72.44

Table 1: Performance for answer sentence selection on WikiQA dataset. Results of PTK are ensemble results and SQuAD* is trained with extra supervised corpus, SQuAD.

We first study the effectiveness of our method for answering on the WikiQA dataset. Our model achieves new state-of-the-art performance compared with previous methods. First, comparing "BERT+MatchLSTM" with "BERT", we find that "BERT+MatchLSTM" achieves significant improvement, which illustrates the effectiveness of our method using a MatchLSTM layer to extend "BERT". Moreover, from the results of "BERT+MatchLSTM", "BERT+AVG" and "BERT+CNN", we find that the MatchLSTM layer utilizes the output representations from the BERT encoder more efficiently than "AVG" and "CNN". We also see that "BERT$_{no\ init}$" does not perform well on this task, which shows that the architecture of BERT is not superior to existing architectures, thus using other architectures to boost the pre-trained BERT is meaningful and important. From the result of "-FHS" we find that the performance drops a lot, which illustrates the method we used to concatenate the hidden state of first token in BERT to the output of MatchLSTM is very important.

4.4 Experiments on SNLI

SNLI	**train**	**test**
SLRC (Zhang et al., 2018)	89.1	89.1
LMPT (Radford et al., 2018)	96.6	89.9
ESIM+ELMo (Peters et al., 2018)	91.6	88.7
+Ensemble	92.1	89.3
DRCN (Kim et al., 2018)	93.1	88.9
+Ensemble	95.0	90.1
BERT$_{no\ init}$	90.7	82.5
BERT (Devlin et al., 2018)	95.9	89.6
BERT+AVG	96.2	90.1
BERT+CNN	94.7	90.2
BERT+MatchLSTM	**96.8**	**90.9**
-FHS	91.4	88.3

Table 2: Accuracy on SNLI dataset. All compared results are from the SNLI Leaderboard.

Table 2 shows the comparisons of our method with the state-of-the-art methods on the SNLI dataset. The trends observed in this experiment are consistent with the former one. Compared with baseline methods, our "BERT+MatchLSTM" achieves the best performance on both training and test sets. Furthermore, compared with "BERT", "BERT+AVG" and "BERT+CNN", our model shows obvious superiority. Comparing the results of "BERT" and "BERT$_{no\ init}$", we also observe that the pre-training process is significantly helpful in this task.

5 Conclusion

In this work, we design an aggregation method to combine BERT with a MatchLSTM layer for sequence matching. We show that our model is more effective in computing the interaction features with the BERT encoder than the original BERT model. Experiments show that our method achieves new state-of-the-art performance on WikiQA and SNLI datasets.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grants No.U1711263.

References

Tim Salimans andIlya Sutskever Alec Radford, Karthik Narasimhan. 2018. Improving language understanding with unsupervised learning. *Technical report, OpenAI.*

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

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.*

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. *arXiv preprint arXiv:1702.02171.*

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108.*

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365.*

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf.*

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250.*

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591. ACM.

Kateryna Tymoshenko and Alessandro Moschitti. 2018. Cross-pair text representations for answer sentence selection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2162–2173.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461.*

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849.*

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747.*

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

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144.*

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193.*

Zhuosheng Zhang, Yuwei Wu, Zuchao Li, Shexia He, Hai Zhao, Xi Zhou, and Xiang Zhou. 2018. I know what you want: Semantic learning for text comprehension. *arXiv preprint arXiv:1809.02794.*