Inspecting Unification of Encoding and Matching with Transformer: A Case Study of Machine Reading Comprehension

Hangbo Bao[†][*]**, Li Dong**[‡]**, Furu Wei**[‡]**, Wenhui Wang**[‡]**,**
Nan Yang[‡]**, Lei Cui**[‡]**, Songhao Piao**[†] and **Ming Zhou**[‡]
[†]School of Computer Science, Harbin Institute of Technology
[‡]Microsoft Research
hangbobao@gmail.com,piaosh@hit.edu.cn
lidong1,fuwei,wenwan,nanya,lecu,mingzhou@microsoft.com

Abstract

Most machine reading comprehension (MRC) models separately handle encoding and matching with different network architectures. In contrast, pretrained language models with Transformer layers, such as GPT (Radford et al., 2018) and BERT (Devlin et al., 2018), have achieved competitive performance on MRC. A research question that naturally arises is: apart from the benefits of pre-training, how many performance gain comes from the unified network architecture. In this work, we evaluate and analyze unifying encoding and matching components with Transformer for the MRC task. Experimental results on SQuAD show that the unified model outperforms previous networks that separately treat encoding and matching. We also introduce a metric to inspect whether a Transformer layer tends to perform encoding or matching. The analysis results show that the unified model learns different modeling strategies compared with previous manually-designed models.

1 Introduction

In spite of different neural network structures, encoding and matching components are two basic building blocks for many NLP tasks like machine reading comprehension (Rajpurkar et al., 2016; Joshi et al., 2017). A widely-used paradigm is that the input texts are encoded into vectors, and then these vectors are aggregated to model interactions between them by matching layers.

Figure 1(a) shows a typical machine reading comprehension model, encoding components separately encode question and passage to vector representations. Then, we obtain context-sensitive representations for input words by considering the interactions between question and passage. Finally, an output layer is used to predict the probability of each token being the start or end position of the answer span. The encoding layers are usually built upon recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014), and self-attention networks (Yu et al., 2018). For the matching component, various model components have been developed to fuse question and passage vector representations, such as match-LSTM (Wang and Jiang, 2016), co-attention (Seo et al., 2016; Xiong et al., 2016), and self-matching (Wang et al., 2017). Recently, Devlin et al. (2018) employ Transformer networks to pretrain a bidirectional language model (called BERT), and then fine-tune the layers on specific tasks, which obtains state-of-the-art results on MRC. A research question is: apart from the benefits of pretraining, how many performance gain comes from the unified network architecture.

In this paper, we evaluate and analyze unifying encoding and matching components with Transformer layers (Vaswani et al., 2017), using MRC as a case study. As shown in Figure 1(b), compared with previous specially-designed MRC networks, we do not explicitly distinguish encoding stages and matching stages. We directly concatenate input question and passage into one sequence at first, and append segment embeddings to word vectors in order to indicate whether each token is belong to question or passage. Next, the packed sequence is fed into a multi-layer Transformer network, which utilizes the self-attention mechanism to obtain contextualized representations for both question and passage. The first advantage is that the unified architecture enables the model to automatically learn the encoding and matching strategy, rather than empirically specifying layers one by one. Second, the proposed method is conceptually simpler than previous systems, which simplifies the model implementation.

We conduct experiments on the SQuAD v1.1

[*]Contribution during internship at Microsoft Research

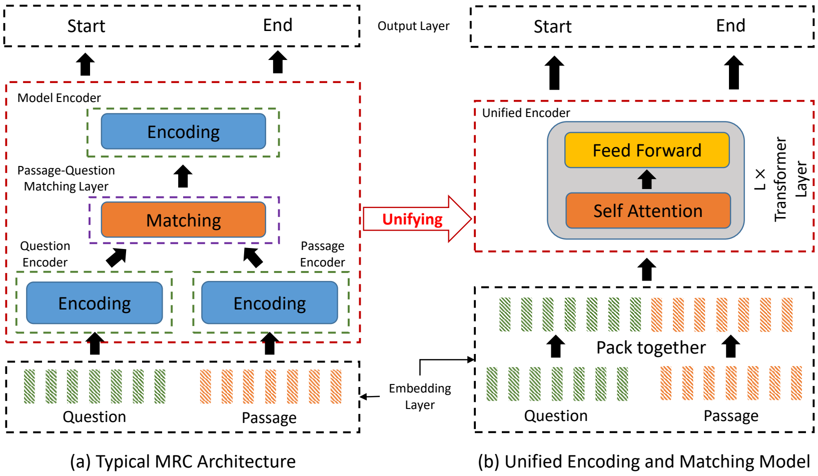


Figure 1: An illustration of a typical MRC architecture and the unified encoding and matching model.

dataset (Rajpurkar et al., 2016), which is an extractive reading comprehension benchmark. Experimental results show that the unified model outperforms previous state-of-the-art models that treat encoding and matching separately. The results indicate that part of improvements of BERT (Devlin et al., 2018) attribute to the architecture used for end tasks. Moreover, we introduce a metric to inspect the ratio of encoding and matching for each layer. The analysis illustrates that the unified model learns different strategies to handle questions and passages, which sheds lights on our future model design for MRC.

2 Unified Encoding and Matching Model

We focus on extractive reading comprehension in the work. Given input passage x^P and question x^Q, our goal is to predict the correct answer span $a = x^P_s \cdots x^P_e$ in the passage. The SQuAD v1.1 dataset assumes that the correct answer span is guaranteed to exist in the passage.

Figure 1(b) shows the overview of the unified model[1]. We first pack the question and passage into a single sequence. Then multiple Transformer (Vaswani et al., 2017) layers are employed to compute the vector representations of question and passage together. Finally, an output layer is used to predict the start and end positions of answer span. Compared with previous specially-designed networks illustrated in Figure 1(a), the model unifies encoding layers and matching layers by using multiple Transformer blocks. The self-attention mechanism is supposed to automatically learn question-to-question encoding, passage-to-passage encoding, question-to-passage matching, and passage-to-question matching.

2.1 Embedding Layer

For each word in questions and passages, the vector representation $\mathbf{x}$ is constructed by the word embedding $\mathbf{x}_w$, character embedding $\mathbf{x}_c$, and segment embedding $\mathbf{x}_s$. The character-level embeddings are computed in the similar way as (Yu et al., 2018). The segment embeddings are vectors used to indicate whether the word belongs to question or passage. The final representation is computed via $\mathbf{x} = \vartheta([\mathbf{x}_w; \mathbf{x}_c]) + \mathbf{x}_s$, where ϑ represents a Highway network (Srivastava et al., 2015).

2.2 Unified Encoder

Given question x^Q and passage x^P embeddings, we first pack them together into a single sequence $[\mathbf{x}^Q_1, \cdots, \mathbf{x}^Q_{|Q|}, \mathbf{x}^P_1, \cdots, \mathbf{x}^P_{|P|}]$, which also denoted as $\mathbf{h}_0$. Then an L-layer Transformer encoder is used to encode the packed representations:

$$\mathbf{h}_l = \text{Transformer}_l(\mathbf{h}_{l-1})$$

where $l \in [1, L]$ is the depth.

Transformer blocks use a self-attention mechanism to compute attention weights between each pair of tokens in the packed question and passage, which automatically learns the importance of encoding and matching. Specifically, for each token, the attention scores are normalized over the whole sequence. The weights between two question tokens can be regarded as question encoding. Similarly, the attention scores between

[1] The implementation and models are available at github.com/addf400/UnifiedModelForSQuAD.

two passage tokens can be viewed as passage encoding. The attention weights across the question segment and the passage segment can be considered as question-to-passage or passage-to-question matching.

2.3 Output Layer

Inspired by Yu et al. (2018), hidden vectors of different Transformer layers $\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_k$ ($i = 6, j = 9, k = 12$ in our implementation) are used to represent the input. Moreover, we employ a self-attentive method as in Wang et al. (2017) over question vectors to obtain a question attentive vector $\mathbf{v}^q$. Finally, we predict the probability of each token being the start (p^s) or end (p^e) position of the answer span:

$$p^s = \text{softmax}(\mathbf{W}_1[\mathbf{h}_i; \mathbf{h}_i \odot \mathbf{v}^q; \mathbf{h}_j; \mathbf{h}_j \odot \mathbf{v}^q])$$
$$p^e = \text{softmax}(\mathbf{W}_2[\mathbf{h}_i; \mathbf{h}_i \odot \mathbf{v}^q; \mathbf{h}_k; \mathbf{h}_k \odot \mathbf{v}^q])$$

where $\odot$ represents elementwise multiplication, and $\mathbf{W}_1, \mathbf{W}_2$ are parameters.

To train the model, we maximize the log likelihood of ground-truth start and end positions given input passage and question. At test time, we predict answer spans approximately by greedy search.

3 Experiments

3.1 Experimental Setup

Dataset Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is composed of over 100,000 instances created by crowdworkers. Every answer is constrained to be a continuous sub-span of the passage.

Settings We employ the spaCy toolkit to preprocess data. We use 300-dimensional GloVe embeddings (Pennington et al., 2014) to initialize word vectors of both questions and passages, and keep them fixed during training. A special trainable token `<UNK>` is used to represent out-of-vocabulary words. We randomly mask some words in the passage to `<UNK>` with 0.2 probability while training. The dimension of character embedding and segment embedding is 64 and 128, respectively. The number of Transformer layers used in our model is 12. For each Transformer layer, we set the hidden size to 128, and use relative position embedding (Shaw et al., 2018) whose clipping distance is 16. The number of the attention heads is 8.

During training, the batch size is 32 and the number of the max training epochs is 80. We use Adam (Kingma and Ba, 2015) as the optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$. We use warmup over the first $4,000$ steps, and keep the learning rate fixed for the remainder of training. The learning rate is set to 6×10^{-4}. We apply the exponential moving average on all trainable variables with decay rate of 0.9999. Layer dropout (Huang et al., 2016) is used in Transformer layers with 0.95 survival probability. We also apply dropout on word, character embeddings and each layers with dropout rate of 0.1, 0.05 and 0.1 respectively.

Model	EM / F1
BiDAF (Seo et al., 2016)	68.0 / 77.3
R-Net (Wang et al., 2017)	72.3 / 80.7
QAnet (Yu et al., 2018)	73.6 / 82.7
Separate Encoding and Matching	74.6 / 83.1
Unified Encoding and Matching	**75.7 / 84.2**

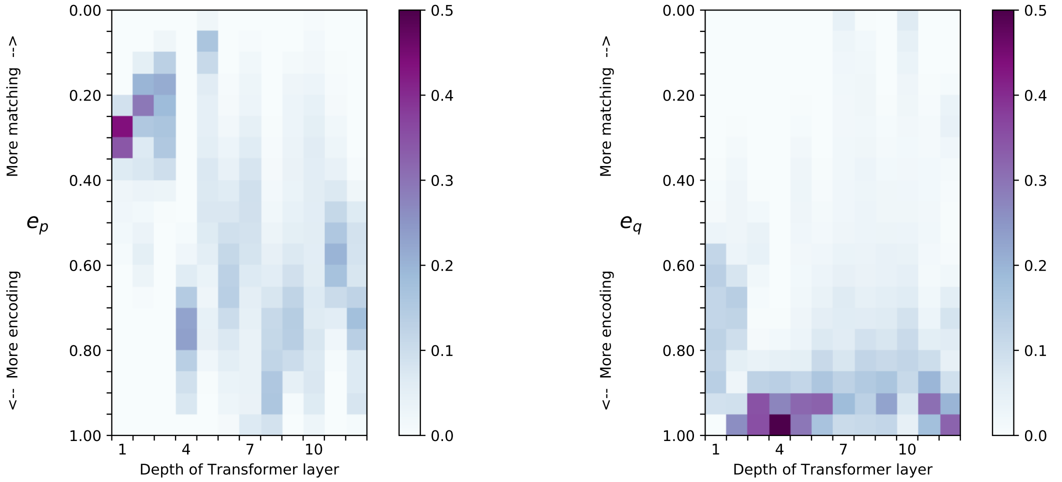
Table 1: Performance of different models on SQuAD development set.

Comparison Models Apart from comparing with previous state-of-the-art models (Seo et al., 2016; Wang et al., 2017; Yu et al., 2018), we implement a baseline model that separately perform encoding and matching. The same settings as above are used. The first three Transformer layers are utilized to encode passage and question separately. Then we add a passage-question matching layer following Yu et al. (2018), with nine more Transformer layers used to compute the question-sensitive passage representations. To make a fair comparison, we only compare with the models that do not rely on pre-trained language models (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018).

3.2 Results

Exact match (EM) and F1 scores are two evaluation metrics for SQuAD. EM measures the percentage of the prediction that matches the ground-truth answer exactly, while F1 measures the overlap between the predicted answer answer and the ground-truth answer. The scores on the development set are evaluated by the official script.

As shown in Table 1, the unified model outperforms previous state-of-the-art models and the baseline model. We find that our unified model



(a) Density distribution of passage encoding ratio for each layer

(b) Density distribution of question encoding ratio for each layer

Figure 2: Density distribution of passage encoding ratio e_p and question encoding ratio e_q for all attention heads in Transformer layers. Vertical axis represents encoding ratio. Larger encoding ratio means that the layer performs more encoding, while smaller ratio value indicates more matching. Darker color means higher density, i.e., more attention heads' encoding ratio values are within the range. The patterns show that the unified model learns different modeling strategies compared with previous manually-designed networks (see Section 3.3 for details).

brings 1.1/1.1 absolute improvement on EM/F1 over the baseline that separately conducts encoding and matching. The results indicate the unified model not only simplifies the model architecture, but also improves performance on SQuAD.

3.3 Analysis

We introduce passage encoding ratio e_p and question encoding ratio e_q to quantify the encoding and matching strategies for each layer of the unified encoder. Let us take the question encoding ratio of an attention head in the l-th Transformer layer for example. Given the attention head's self-attention weight matrix $\mathbf{A}$, the ratio e_q is computed via:

$$\begin{aligned} s_{q|q} &= \text{avg}_{i,j\in Q}\{\mathbf{A}_{i,j}\} \\ s_{q|p} &= \text{avg}_{i\in Q,j\in P}\{\mathbf{A}_{i,j}\} \\ e_q &= s_{q|q}/(s_{q|q} + s_{q|p}) \end{aligned}$$

where $s_{q|q}$ is the average question-to-question attention weight, and $s_{q|p}$ is the average passage-to-question attention weight. To be specific, if e_q is close to 1, it means that the layer tends to perform question-to-question encoding. In contrast, if e_q is close to 0, it indicates the layer performs more passage-to-question matching. Similarly, we can compute passage encoding ratio e_p as above.

As shown in Figure 2, we compute passage encoding ratio e_p and question encoding ratio e_q for all the attention heads on the development set, and plot their density distributions for each Transformer layer. We find that the unified model learns strategies that are clearly different from manually-designed architectures:

- Figure 2(a) shows that the first three layers perform question-to-passage matching and the fourth layer conducts passage-to-passage encoding, while most previous models perform passage encoding first.
- Figure 2(a) indicates that upper layers tend to conduct more encoding than matching.
- Figure 2(b) shows that all layers tend to perform question-to-question encoding than passage-to-question matching.
- Some layers are automatically learned to perform encoding and matching at the same time instead of separate modeling.

4 Conclusion

In this work, we evaluate and analyze unifying encoding and matching components with Transformer for the MRC task. Experimental results on the SQuAD dataset illustrate that the unified model outperforms previous networks that treat encoding and matching separately. We further introduce a metric to inspect whether a layer tends to act more like encoding or matching. The analysis results show that the unified Transformer layers automatically learn strategies that are clearly different from previous specially-designed models.

References

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep networks with stochastic depth. *CoRR*, abs/1603.09382.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *In Proceedings of the International Conference on Learning Representations (ICLR)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

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 *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541.