Multi-output Machine Reading Comprehension Models for Key Entities Extraction in Finance Domain

Kunrui Zhu, Liming Zhang and Zijie Chen

School of Computer Science and Technology,
Guangdong University of Foreign Studies, Guangzhou 510006, China

Abstract. In this paper, we propose a machine reading comprehension models which support multiple outputs for the key entities extraction task in finance domain. Taking a passage and a query as its input, the model employs the BERT language model to extract interactive information between texts and borrow simple but powerful output layer from name entity recognition model. The evaluation on Chinese finance news data from CCKS 2019 shows that, the model achieves competitive results on the test set and rank 7th in the competition.

Keywords: Machine Reading Comprehension, Key Entity Extraction, Neural Network.

1 Introduction

Machine reading comprehension(MRC) tasks aims to model passage-query sentence pairs and extract key information from the passage. As a growing interest in the tasks of MRC has been seen in recent years, it is the solid foundation for applications like search engines and automatic question answering systems.

On the one hand, in the process of approaching the real-world setting, the MRC task is becoming more and more challenging. Traditionally, the task requires the model to extract an answer span from a passage-query pairs, as defined in the SQuAD [2]. More realistic problems are demonstrated in the DuReader [3] dataset, whose questions are more complex and the answers scattered in one or more passages.

On the other hand, with the population of the end-to-end neural network models, an increasing number of MRC models are proposed which keep refreshing the state-of-the-art performance. In the previous works, successful MRC models employ Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) as the encoder and apply the co-attention mechanism to extract long term interactions. To better generating the answer span from the passage a pointer network [4], which calculates the starting index as well as the ending index, would be placed at the end of the model as the output layer. Typically, models like BiDAF [5], R-Net[6] and QA-Net[7] shares the structure mentioned above. These models focus on the modification of the encoders or match-attention mechanism, but cannot take the advantages from large-scaled unsupervised corpus. Further bettering the state-of-the-art performance

on the SQuAD [2], pre-trained language model like BERT [1] which improved 11 NLP challenges, including the MRC task, was proposed in late 2018.

Models mentioned above give single answer span after processing a sentence-query pairs. A similar method in MRC models that support multiple answers spans, whose approach is modifying the loss function and compute the average loss for multiple answers, is demonstrated in a multi-answer multi-task framework [8]. However, since the approach focus on choosing the best answer span that is closest to human-generated answers, it is not suitable for handling multiple answer spans. In this study, we evaluate our model on the Chinese finance domain dataset from CCKS 2019. The model acquired good experimental results on the 3-fold cross-validation from the dataset.

2 Related Works

Various successful systems have been proposed for the Machine Reading Comprehension, which act as the driving force to boost the situation of the task. Most of the top solutions employ deep neural network to encode sentences and get interactive information from 2 sentence by applying different kinds of co-attention mechanism.

Among them, one of the most related studies, which allows MRC to form multiple answer is depicted in A Multi-answer Multi-task Framework for Real-world Machine Reading Comprehension [8]. After refining the most relevant span from several candidate documents on the DuReader dataset [3], the system converts sentences into features of word-level and character-level embedding and other extra features. The LSTM encoder as well as match attention mechanism was employed. To incorporate multiple answers in the training process, the system makes several useful modifications on the loss function, which help them greatly in improving the performance of their model.

Benefits from large scale unsupervised data, the BERT pretrained model top the leaderboards in MRC and several other NLP challenges once it was released. The pretrained model plays important role in both passage-query matching and word representation. With the pretrained weight and 12 layers of Transformer [10], models can achieve good performance with little additional parameters. The system demonstrated in BERT simply adds a linear layer after the embedding features generated by the pretrained model, mapping each word to a 2 dimensional hidden states which indicate the stating and ending position. Amazingly, the performance of their model is outstanding.

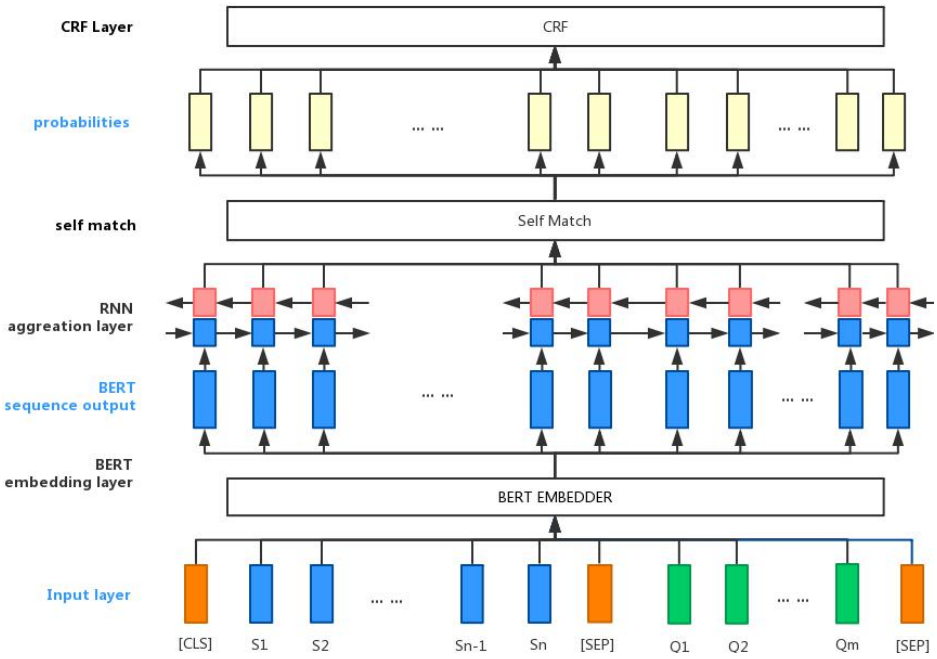

Fig. 1. The overview structure of our model for key entities extraction

3 Model Structure

The structure of our model is displayed in Fig.1. Sentence $s\{S_1, S_2, ...S_{n-1}, S_n\}$ and query $q\{Q_1, Q_2, ...Q_{m-1}, Q_m\}$ are the input of our model. To get the interaction information between sentence and query we employ a BERT embedding layer. To generate span of predictions, instead of applying a pointer net or map each character into 2 hidden states, like the previous works or the implementation in BertForQuestionAnswering [1], we borrow the output layer from the name entity recognition model which map each character to a label-size dimensional hidden states and apply the argmax function to locate the label with maximum probability.

3.1 Input Layer

Adapting sentences and queries to the input format of BERT, we concatenate tokens of two sentence with a separate mark "[SEP]". After that, a starting mark "[CLS]" and another separate mark is added to the starting and ending position of the list of token respectively.

3.2 Matching and Embedding Layer

For sentence-query matching, we employ the pre-trained BERT [1] model. This layer takes a sequence of token ids, a list of token-type which distinguish the sentence from the query and a sequence of attention mask as its input. Two functions integrated in this layer are the extraction of interactive information from the input texts and the generation of character embedding. We obtain the matching information $M \in R^{t \times d}$ from the output of the BERT model, where t is the number of time steps for the input sequence and d equals to 768 as it is the default hidden size of the BERT-base. To add more trainable parameters which serve as a helping hand to minimize the training loss, we do not freeze parameters in the BERT model.

3.3 Aggregation Layer

Following the architecture of the successful MRC model, we add a bi-directional RNN layer to fuse the result of the matching layer. The RNN pass through each position as shown in the equations:

$$\vec{h}_t = RNN(\vec{h}_{t-1}, m_t) \tag{1}$$

$$\overleftarrow{h}_t = RNN(\overleftarrow{h}_{t-1}, m_t) \tag{2}$$

The encoded feature of each time step h_t is obtained by concatenating features of both direction, see equation (3):

$$h_t = [\vec{h}_t, \overleftarrow{h}_t] \tag{3}$$

3.4 Self-Matching Layer

Inspired by the R-Net [6] model, we employ the self attention mechanism on the output of the RNN aggregation layer, which helps collect evidence from the sequence itself. The attention equations in our model comes from Attention is All You Need [10], which is commonly used. In the equation, Q,K and V are 3 inputs of the attention layer, and d_k is the size of hidden state of input K:

$$Attention(Q, K, V) = soft\max(\frac{QK^T}{\sqrt{d_k}})V \tag{4}$$

To extract features from the sequence itself we set Q, K and V as the output of the aggregation layer.

3.5 Prediction Layer

The Prediction Layer map each time step's encoding of the previous layer into a 4 hidden states layer by a weight matrix $W \in R^{d \times 4}$. For each time step, optionally, if we are not employing the CRF layer in section 3.6, we use the argmax function to select the label with highest probability from B(beginning of the entity), I(inside of the entity), E(ending of the entity) and O(other irrelevant characters).

3.6 CRF Layer

In order to constrain prediction from invalid entities, we put a CRF layer on the end of the model. The output probabilities of the prediction layer act as the emit score $E_t^i (i \in \{1,2,3,4\})$ of the CRF model, which represents the confidence of the observe time step *t* to be classified into the i^{th} label. And the CRF layer maintains a transition matrix where each element $T_i^j (i, j \in \{1,2,3,4\})$ represent the probabilities of transiting label *i* to label *j*. As a result the probability of time step *t* being classified into label *i* can be calculated as the following equation:

$$P_t^i = E_t^i \times T_{P_{t-1}}^i \tag{5}$$

Eventually, the layer choose a sequence of steps that maximum the probability by applying the viterbi algorithm.

4 Experiments

4.1 Dataset and Preprocessing

We evaluate our model on the CCKS 2019 datasets, which contains 30 thousand lines of Chinese sentences in finance domain, including 17 thousand for training, 3 thousand for validating and approximately 10 thousand for testing.

the task requires models to locate and extract key entities from the sentence according to the event type. For example:

Given the Chinese sentence: "受到草根投资非法集资案件影响，万家乐公司相关银行账户也被司法冻结", and an event type: "资产负面", the models are expected to extract the key entity "万家乐" from the sentence.

We see the sentence and event type as the passage and query respectively, and model it as a machine reading comprehension task. However, different from the traditional MRC tasks, some of the data samples from the training dataset have more than one answers and they are equally important as key entities. Therefore, a multiple answer supported model is needed.

The following 3 steps of data preprocess are applied to the raw data:

Combination of Key Entities. To free the model from the disturbance of different key entities corresponding with one sentence-event pair, we merge all the key entities

for the same sentence-event pair, which changes the one-to-one format of the original dataset. For example:

We merge data samples with id 104033 and 104031, whose elements are the same except for their key entities(see the following example)

Original sentence-event pairs:

"104033"," 案件一:非法吸收公众存款，最高被判7年券商中国记者了解到，作为鼎基久盛和汇融通盛的实控人，李海涛负责吸纳资金的支配使用","涉嫌违法","鼎基久盛"

"104031"," 案件一:非法吸收公众存款，最高被判7年券商中国记者了解到，作为鼎基久盛和汇融通盛的实控人，李海涛负责吸纳资金的支配使用","涉嫌违法","汇融通盛"

Merged data line:

" 案件一:非法吸收公众存款，最高被判7年券商中国记者了解到，作为鼎基久盛和汇融通盛的实控人，李海涛负责吸纳资金的支配使用","涉嫌违法","鼎基久盛,汇融通盛"

After the combination process, one line of data with a sentence-event pair could contain one or many key entities.

Modification on Stock Codes. Since most of the samples in the training dataset contain stock codes for companies, our team attempted to make use of them at the beginning. Take sample 211864 as an example, from the sentence:
"*兔宝宝*(002043)监事丁观芬短线交易 获利2.2万元险踩红线*新大新材*(300080)股东减持1200万股", the numeric codes 002043 and 300080 are stock codes for the companies named "*兔宝宝*" and "*新大新材*" respectively.

Although, stock codes can help locate the key entity "*兔宝宝*" for this data sample, the statistical result on the training dataset shows that, most of the key entities are not followed by stock codes. Furthermore, some companies in the corpus are non-listed. As a result, we remove all stock codes from sentences.

Lowercase Transformation. To reduce the vocabulary size of the model, we set all English letters to its lowercase format before training.

4.2 Prediction Optimization Strategy

To filter incomplete entities from raw prediction, we adopt several strategies to help higher the precision.

Confidence Filter. In the predicting process, we collect character that acquire B, I or E label from the predictions, with label B and E as the starting and ending position respectively. In addition, we measure how confident the model is when predicting a span as entity by calculating the average score of these 3 labels.

In order to skip some incomplete entity from being collected, we set a lowest boundary confidence for the predicted entities. That is to say, entity whose confidence score is lower than the highest score for a certain margin, would be abandoned.

Overlap Entities Filter. Generated entities that overlap with each other could contain incompleteness. In this case we pick entity with the highest score from the overlapping entity set. For instance, both "邦邦理财" and "邦邦" were included in the predicted entity set, we would remove one of them according to the confidence score.

English Letters Optimization. For the reason that, we transform sentences into its lowercase format before entering the model, entities from raw output of the model are all in lowercase format which can be different from the ground truth.

In addition, the WordPiece [9] segmentation system of BERT skips the space of some short English phrases. Taking company "Whole Food" as an example, after the WordPiece convert, the company name becomes "wholefoods".

To address the English related issues, we break the raw prediction into several parts, and locate the starting, ending index of the phrase from the lower-cased raw sentence with the help of Regex expression. Finally, we extract the valid entity from the original sentence with the stating and ending index.

Unknown Chinese Character Optimization. With the default BERT vocabulary dictionary, some Chinese characters are not included. Therefore, entities form raw predictions might contain the "[UNK]" mark. To solve the problem we locate the starting and ending position of entities containing the "[UNK]" with the Regex expression.

4.3 Experiments Results

We evaluate our models on the training dataset from CCKS 2019, where 17 thousand Chinese sentences are included. For the reason that, labeled data is available only in training dataset, we run an 3-fold cross validation to evaluate our model, and the results is shown in table 1.

Table 1. Performance of different models on 3 fold cross-validation.

Model	Precision	Recall	F1
Bert_QA	0.8720	0.8593	0.8631
Bert_QA(optimized)	**0.9327**	0.9198	0.9236
Bert_LSTM_LINEAR(optimized)	0.9261	0.9317	0.9259
Bert_CNN_LINEA(optimized)	0.9289	0.9308	0.9271
Bert_CNN_Attn_LINEAR(optimized)	0.9287	0.9287	0.9275
Bert_LSTM_CRF(optimized)	0.9266	0.9278	0.9247
Bert_LSTM_Attn_LINEAR(optimized)	0.9288	**0.9328**	0.92807
Bert_LSTM_Attn_CRF(optimized)	0.9293	0.9322	**0.92809**

The implementation is based on Pytorch deep learning platform, and most of the codes for BERT pretrained model are adapted from the open source github pytorch-transformers [1].

[1] https://github.com/huggingface/pytorch-transformers

Note that the Bert_QA model refers to the BertForQuestionAnswering implemented in the pytorch-transformers repository. And details for the optimized version is discussed in section 4.2.

Models whose name end with LINEAR has skipped the CRF layer and the Attn marks the attention layer discussed in the self-matching layer section.

5 Conclusion

From the results, we can see that the multiple output supported models denoted by LINEAR or CRF are slightly better than the original MRC model as it is improving the Recall. On the other hand, however, more efforts are needed to keep the precision from dropping when using the multiple output structure.

In addition, thanks to the LSTM encoder and Self Attention mechanism, the performance of our model also improves a little. Integrating the advantages of all the additional layers after the BERT embedding and list of output optimization tools, the performance of our system improves greatly from our baseline system.

6 References

1. Devlin, J. , et al.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 4171-4186 (2019).
2. Rajpurkar, P. , et al.: SQuAD: 100,000+ Questions for Machine Comprehension of Text. In: Conference on Empirical Methods in Natural Language Processing. pp. 2383--2392 (2016).
3. He, W. , Liu, K. , et al.: DuReader: a Chinese Machine Reading Comprehension Dataset from Real-world Applications. Proceedings of the Workshop on Machine Reading for Question Answering. pp. 37-46 (2018).
4. Vinyals, O. , Fortunato, M. , Jaitly, N.: Pointer networks. In: Conference on Neural Information Processing Systems. pp. 2692-2700 (2015).
5. Seo, M. , Kembhavi, A. , et al.: Bidirectional Attention Flow for Machine Comprehension. In: International Conference on Learning Representations. (2017).
6. Natural Language Computing Group. , Microsoft Research Asia.: R-NET: Machine Reading Comprehension with Self-matching Networks. (2017).
7. Yu, W. , Dohan, D. , et al.: QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In: International Conference on Learning Representations. (2018).
8. Liu, J. , Wei, W. , Sun, M. , Chen, H. , et al.:A Multi-answer Multi-task Framework for Real-world Machine Reading Comprehension. In: Conference on Empirical Methods in Natural Language Processing. pp. 2109–2118 (2018).
9. Wu, Y. , Schuster, M. , et al.: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. CoRR. abs/1609.08144 (2016).
10. Vaswani, A. , Shazeer, N. , Parmar, N. , Uszkoreit, J. , Jones, L. , Gomez, A. N. , Kaiser, L. , Polosukhin, I.: Attention Is All You Need. In: Conference on Neural Information Processing Systems. pp. 5990-6008 (2017)