Machine Question and Answering on Chinese Judicial Reading Comprehension

Liancheng Gong & Xinyao Han & Qiaowei Li

Abstract

In this paper, we present our models on answering context span extraction questions, distinguishing questions impossible to answer based on given contexts and classifying YES/NO answer questions. Our project aims at performing Span-Extraction Machine Reading Comprehension (MRC) in the field of Chinese legal judgment documents. A Chinese Bert based binary classification model and a question answering model are applied on Chinese judicial Reading Comprehension (CJRC) data set. On the test set, our classification model gives an accuracy of 98.2946%, and our question and answering model gives a test accuracy of 73.4109% and F1-score of 0.81197.

Index Terms

Machine Reading Comprehension, Span-Extraction, Bert-Chinese, Question and Answering, Chinese judicial Reading Comprehension, MRC, CJRC

Acknowledgement

We thank our professor Wilson Tam from NYU Shanghai Computer Science Department, who provided insight and expertise that greatly assisted the project.

I. Introduction

There are thousands of cases happening on a daily basis, which form loads of legal judgments. There are thousands of cases happening on a daily basis, including traffic accidents, private loans, divorce disputes and so on. In the process of handling these cases, we form loads of legal judgments, containing summary of the entire case, involving descriptions of events, court opinions and judgement results. Moreover, compared to the number of judgments, the number of judges are relatively small. Therefore, it's challenging for judges to extract information from legal documents.

Therefore, having machines to answer questions based on given contexts in the legal field will facilitate judges and lawyers in obtaining required information efficiently and help them make better decisions.

Machine Reading Comprehension is to let computers comprehend text and answer text-related questions. There are wide applications of different level Machine Reading Comprehension, ranging from answers extraction to answers generation and from using single articles to multiple articles.

The goal of this project is to implement Span-Extraction MRC on Chinese legal judgment documents. Compared with well studied English characters data set as SQuAD[1], this project will focus on Chinese corpus data sets. We separate the MRC tasks into three stages: Firstly, determining

whether the question is a YES/NO question. Secondly, answering questions when the answers are segments of text or span from the corresponding articles. Thirdly, determining whether the answer is supported by the paragraph and abstain from answering.

II. The Dataset and Features

'**China Judgment Documents Network**' [2] is public legal judgment documents including civil and criminal judgments. Legal experts marked 4 to 5 frequently asked Question Answer pairs based on case contexts. The answer content can be a answer span of the case contexts, YES or NO, or empty if the answer is rejected. The training set contains around 40,000 questions, the development set and the test set each contains about 5,000 questions.

A. Data Demonstration

Case Name: ' 保险人代位求偿权纠纷'

Case Description: ' 经审查, 原告提供的证据 1-3、被告中华联合广东分公司提供的证据 4-5 ... 严 x3 是被告徐 11 雇请, 是从事派遣工作过程中发生案涉交通事故被告徐 11 与被告万友公司签订《车辆挂靠合同书》, 被告万友公司同意被告徐 11 就赣 C×××××× 号车辆挂靠被告万友公司名下'

QA Pairs: ['answer start':153, 'text': ' 两车不同程度损坏'], 'id': 'e139eef6-fc0c-4953-acec-a83a0095ce4e.txt 001', 'is impossible': 'false', 'question': ' 事故结果如何? ']

['answer start':180, 'text': ' 严 x3 承担事故的全部责任, 田 x17 不负事故责任'], 'id': 'e139eef6-fc0c-4953-acec-a83a0095ce4e.txt 002', 'is impossible': 'false', 'question': ' 事故由谁承担什么责任? ']

B. Data Distribution

The train dataset contains 39333 lines of information, including context, question, answer start index corresponding to context, answer, and whether it's possible to answer the questions given context.

On average, contexts contain 572 word length, with a minimum word length of 80, a maximum word length of 1000 and 75% of the data contains a context word length of 806.

On average, questions contain 14 word length, with a minimum word length of 4, maximum word length of 114 and 75% of the data contains a context word length of 17.

Answers demonstrate a right-tailed distribution. On average, answer contains 12 word length, with minimum word length of 0, which standards for questions impossible to answer based on given contexts. Answers have maximum word length of 579 and 75% of the data contains a context word length of 12.

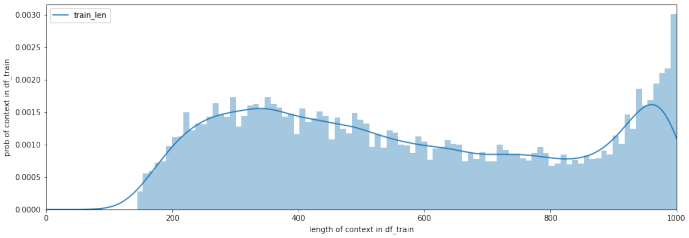

Fig. 1. Context length distribution

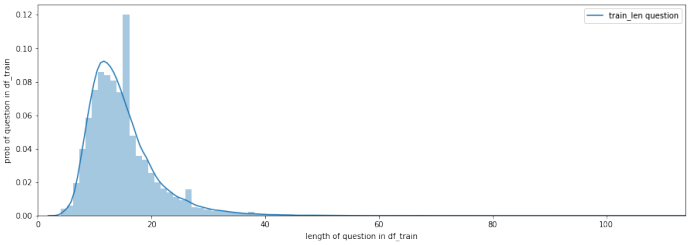

Fig. 2. Question length distribution

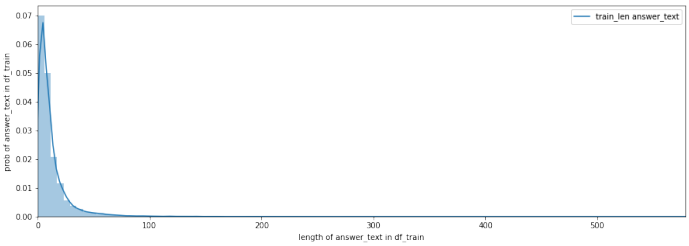

Fig. 3. Answer length distribution

C. Data Prepossessing

Append Classification Labels: Span-extraction questions and impossible to answer questions are categorized into 0, YES questions are categorized

into 1, and NO questions are categorized into 2 for multiple classification model. Additionally, span-extraction questions and impossible to answer questions are categorized into 0, YES/NO questions are categorized into 1 for binary classification model.

Add Start and End Index: The answer start indexes are computed by index searching via tokenized answers. Complicated situations are dealt with such as unmatched answer and context tokenization, multiple occurrence of answers within contexts and incomplete answer span based on given context. End indexes are computed using answer length and answer start indexes.

Text Truncation: BERT model has limitations of max input of 512 tokens, while in our CJRC dataset, contexts have maximum 1000 tokens. To process longer documents, we truncate contexts and take the most answers-related context pieces for Bert model inputs. Based on maximum question length of 117 tokens, the full questions are feed into the Bert model; and we omit answer length outliers by use the 90 percentile answer length of 65 tokens. Context length are decided based on remaining available token inputs, we take 64 tokens before the answer start index and 128 tokens after the answer start index using an approximation answer length.

We did not use full 512 Bert input tokens in consideration of computation speed. Since attention calculations have quadratic complexity by the number of tokens, processing the document of 512 tokens is usually very slow. Therefore, input token length would be question length plus 192 tokens exclusing special tokens.

III. Explanation of the Methods Used

A. *Pretrained Model*

Chinese-BERT-wwm[3]: It's a Bert based model trained on Chinese Wikipedia and Harbin Institute of Technology LTP as segmentation. This model improves on Bert-Chinese by improving the whole word masking method on English characters to fit Chinese word segmentation (CWS). It masks all Chinese characters which make up the same word instead of masking random Chinese characters, which preserves the original vocabulary meaning and sentence syntax.

We use two separate Bert models for simplification:

1. A Classification model to first classify if the question has a YES/NO answer or a SPAN answer;
2. A Question Answering model to extract answer span for the questions that are classified as having a SPAN answer by the previous classification model.

B. *BERT Classification Model*

The first step is to determine whether the question is a YES/NO question. To achieve that, we implement a Bert Classification Model. Given an input text sequence of the question, the classification model will measure the conditional probability distributions over the two possible labels $y = \{span, YES/NO\}$.

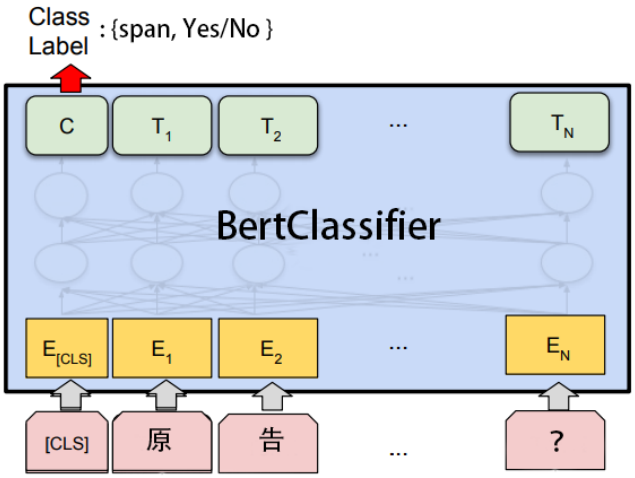

Fig. 4. BertClassifier

Model Input For the classification model, we only take question tokens as input for the BERT model.

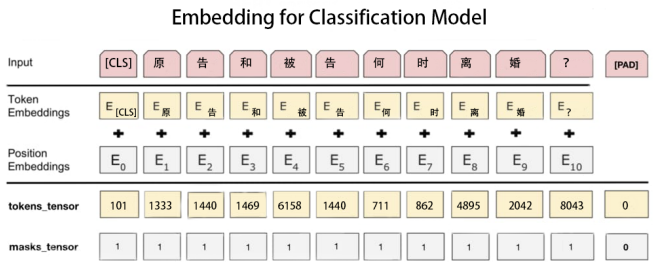

Fig. 5. Input embeddings for BERT classification model

As displayed in Figure 5, the input has 2 layers of embeddings:

Token Embeddings: A [CLS] token is added at the very beginning of the input question tokens to signal the start of the input. 0 paddings [PAD] are appended after the end of the context if the input didn't reach question's max length.

Token embeddings will be put into the model in the form of a token tensor. Each token has their own token id, and the token id for [CLS] is 101, zero paddings [PAD] is 0.

Position Embeddings: Position embedding is implemented here for the model to recognize which part of the input sequence it should pay attention to.

Position embeddings will be put into the model in the form of a mask tensor, with 1 indicating the position from the start token [CLS] to the end of the question tokens, and 0 indicating the remaining zero paddings.

Output and Fine Tune: For the classification task, the BERT model takes the final hidden state h of the first token [CLS] as the representation of the whole input question token sequence. Then to the top of BERT a softmax classifier is added to predict the probability of the class labels c:

$$p(c|h) = softmax(Wh)$$

where W is the task-specific parameter matrix. All parameters are fine-tuned to maximizing the log-probability of the correct class label.

Loss Function: For this question type classification task, cross-entropy loss is chosen as the loss function. For each input sequence x, denote $p(x)$ as the probability distribution of the expected output, $q(x)$ as the probability distribution of the actual output, and the cross-entropy loss is:

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

Implementation Detail

Dataset Input: The dataset for classification task is very unbalanced, with only 3565 questions have an answer of YES, 1543 questions having an answer of NO, while 34225 questions having answers in the class of span answers. The ratio of the two classes is: YES/NO : span = 1 : 6.7. Such unbalanced dataset will lead to biased training result.

Therefore, the method of undersampling is applied to create a more balanced dataset input to the classification model. For the YES/NO answer questions, questions with answer of YES and questions with answer of NO are grouped together, and by merging the YES/NO set the class has a size of 5108 pieces of data. For the span class, 6500 data are randomly selected from the entire span question set.

Final result of the input dataset for the classification model contains 11608 pieces of data in total, 5108 of them belongs to the class YES/NO and 6500 belongs to the other class.

Model Configuration: During training, we used Adam Optimizer with a batch size of 32, and the initial learning rate is set to 6.25e-5.

The input size of the data is (11608, 115), 11608 sequences of question tokens with a padded max length of 115.

Training epoch is set to 8.

Number of class labels: The number of class labels is initially set to three, having {0:span, 1:YES, 2:NO}. However, having YES and NO as separate class makes it hard to have a balanced dataset, and

it is hard for machine to learn if the answer is yes or no, so the two classes are merged together and we finally have 2 classes for the classification model.

Input sequence: The original input sequence of the classification model is:

[CLS] Question [SEP] Context [SEP].

However, there are some problem with this input format and the outcome performance is not good:

- To distinguish whether a question has a YES/NO answer or span we only need question. Contexts are not necessary and too long for the input, which might disturb the classification process.

- There are some questions that are unable to answer, so the answer is neither yes/no, nor span, but none, which also influence the classification process and worsen the model performance.

Therefore, to improve model performance, the final input is in the format as follows:

[CLS] Question.

By doing so we have a much shorter and clearer input for the classification model.

C. BERT Question and Answering Model

From the last section, Bert Classifier will help us filter out questions that has an answer type of span, and those span questions will be further input into this Question and Answering Model with its context, and the model will extract the answer span from the case contexts.

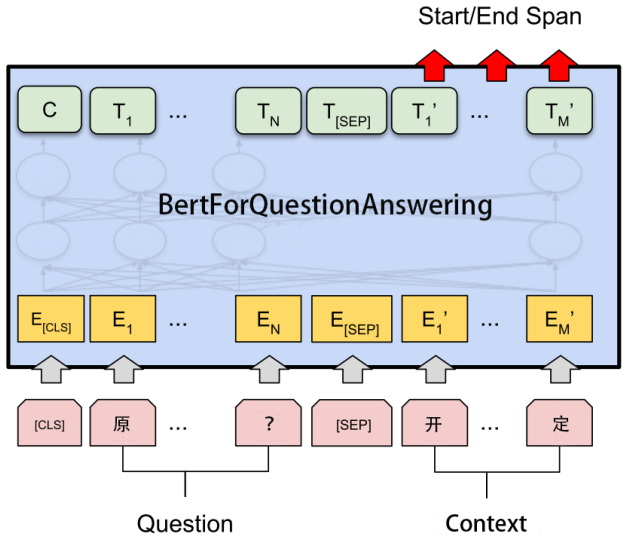

Fig. 6. BERT for Question Answering

Model Input

For the Question Answering task model, we packed question and context into a single sequence as shown above.

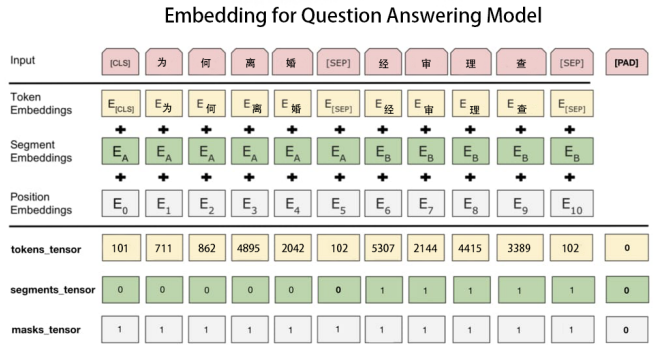

Fig. 7. Input embeddings for BERT QA model

As displayed in Figure 7, the input has 3 layers of embeddings:

Token Embeddings: A [CLS] token is added at the very beginning of the input question tokens and a [SEP] token is inserted at the end of both the question and the context. 0 paddings [PAD] are appended after the end of the context if the input didn't match the max length.

Token embeddings will be put into the model in the form of a token tensor. Each token has their own token id, and the token id for [CLS] is 101, [SEP] will be represented as 102.

Segment Embeddings: Each input token will be marked as Segment A or Segment B. In this case, question tokens together with its start token [CLS] and end token [SEP] will be marked as Segment A, while the rest of the context tokens will be Segment B. This allows the model to distinguish between question and context.

Segment embeddings will be put into the model in the form of a segment tensor, with 0 indicating Segment A and 1 for Segment B.

Position Embeddings: Position embedding is implemented here for the model to recognize which part of the input sequence it should pay attention to.

Position embeddings will be put into the model in the form of a mask tensor, with 1 indicating the position from the start token [CLS] to the end token [SEP] after the context tokens, and 0 indicating the remaining zero paddings.

Output and Fine Tune

The Question Answering model generates answer by predicting the start and end position of the answer in the context. The QA model has two vectors, one containing the start weight of each word in the context and the other containing the end weight.

For each word in the context, the probability of being the start-word is calculated by taking a dot product between its final embedding and the start weight vector, and then implementing the softmax activation, as following:

$$P_{start} = softmax(E_{start} \cdot w_{start}) \in \mathbb{R}^T$$

Accordingly, we get a probability distribution over all context tokens, and the token that has the highest probability will be picked as the start token of the answer span.

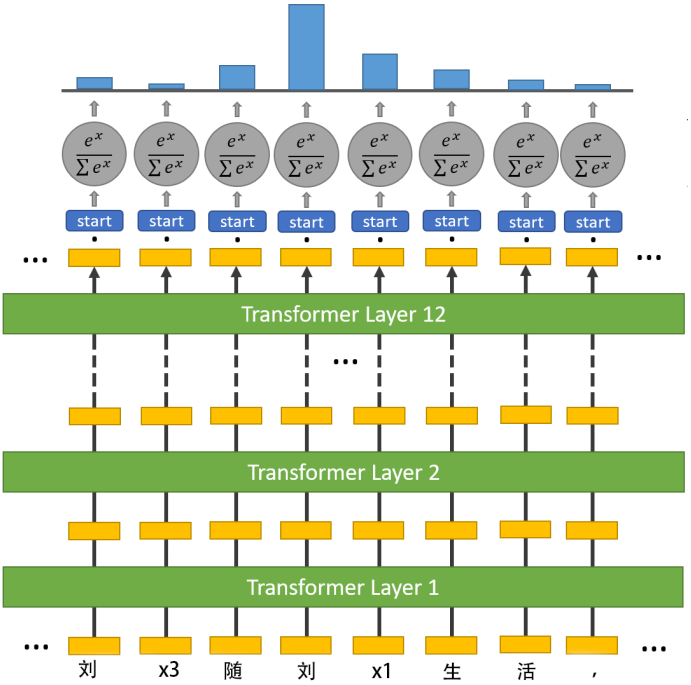

Fig. 8. start token calculation

Similarly, we can calculate the end logits by:

$$P_{end} = softmax(E_{end} \cdot w_{end}) \in \mathbb{R}^T$$

and get the end token of the answer span.

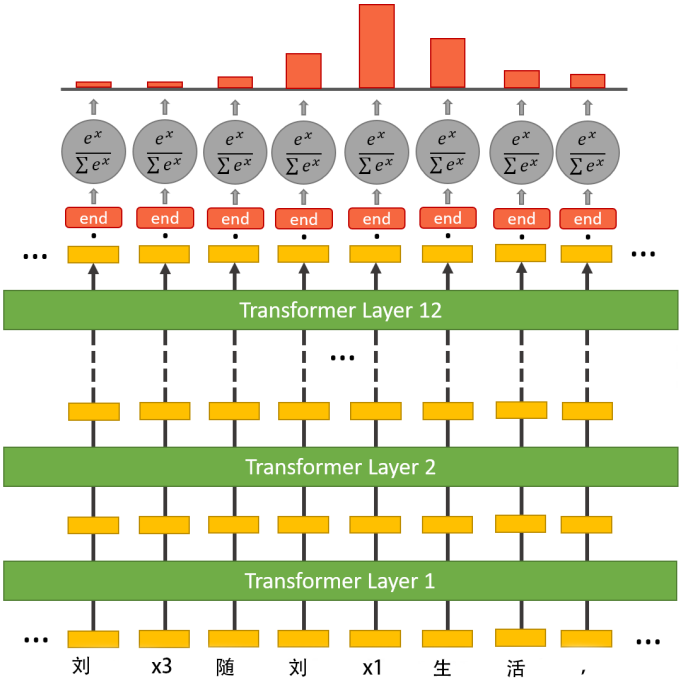

Fig. 9. End token calculation

Loss Function

For QA model, still cross-entropy loss is chosen as the loss function. For each token x, the task could be viewed as a start/end classification, each token could belong to one of the two class: {is the start/end token, not the start/end token}. Denote $p(x)$ as the probability distribution of the expected output, $q(x)$ as the probability distribution of the actual output, and the cross-entropy loss is:

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

Total loss of the answer span extraction takes the average of start token loss and end token loss.

start_loss = CrossEntropyLoss(start_logits, true_start_positions)

end_loss = CrossEntropyLoss(end_logits, true_end_positions)

$total\_loss = \frac{start\_loss + end\_loss}{2}$

Implementation Detail

Dataset Input:

After filtering out the question answer pairs that has a YES/NO answer, we have 34,108 pieces of QA pairs left for answer span extraction.

However, for some pieces of data, the answer span extracted using start and end index does not match the ground truth answer. After filtering out those data, we finally have 32,570 pieces of data left for the QA model.

Model Configuration:

During training, we use Adam Optimizer with a very small batch size of 6 to avoiding running out of cuda memory, and we set an initial learning rate of $6.25e-5$.

The input size of the data is (32570, 259), 32570 sequences of question and context tokens with a padded length of 259.

Training epoch is set to 10.

IV. Results and Evaluation

A. *Classification Model*

Evalutation method for the classification model contains 2 parts: Test Accuracy and Test Loss.

Test Accuracy: 98.2946%

Test accuracy of the classification model is computed by calculating the ratio of correct prediction of class label to overall number of predictions.

$$test\ accuracy = \frac{\#\ correct\ class\ label\ prediction}{\#\ predictions}$$

Test Loss: 0.015154

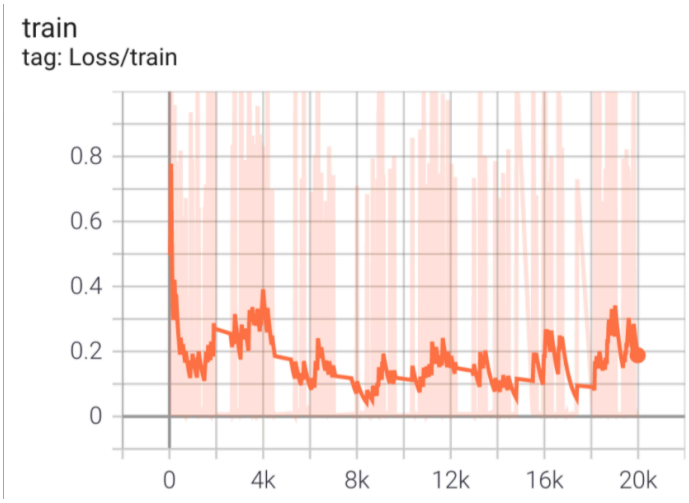

Fig. 10. Train Loss for Classification Model

Detailed result data see table I in appendix.

B. *Question Answering Model*

Evaluation for Question Answering model contains 3 parts: Test Accuracy, Test Loss, and F1-score.

Test Accuracy/ Exact Match: 73.4109%

For the QA model, the test accuracy is computed using the metric of Exact Match(EM). For each question and answer pair, if and only if all tokens in the prediction matches the ground truth answer $EM = 1$, otherwise $EM = 0$. Overall EM is computed by averaging over the all individual EM scores of each question and answer pair in the train set.

Test Loss: 1.863349

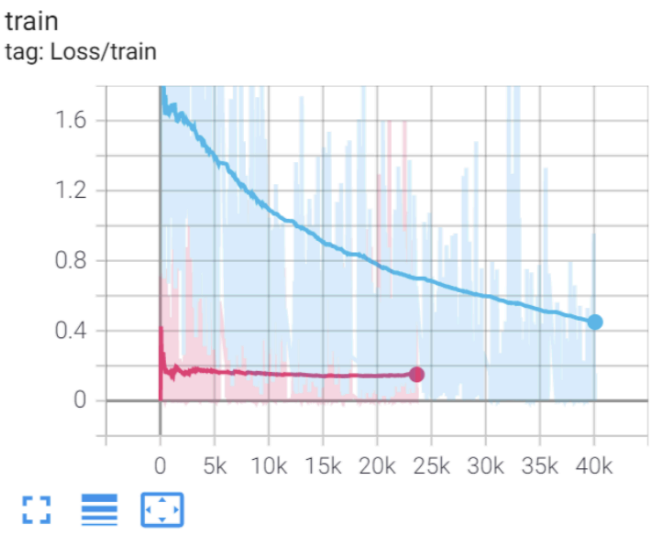

Fig. 11. Train Loss for QA Model

In the above Figure 11, due to gpu limit, the curve is divided into two sections.

F1-score: 0.81197

F1 score considers equally about precision and recall and is computed over the individual words in the prediction against those in the ground truth answers. Computation of F1 score is based on the number of shared tokens between the prediction and the ground truth. To calculate F1 score, values of precision and recall are needed.

`Precision` is the ratio of the number of shared words to the total number of words in the prediction.

$$precision = \frac{\#\ shared\ words}{total\ \#\ of\ tokens\ in\ the\ prediction}$$

`Recall` is the ratio of the number of shared words to the total number of words in the ground truth.

$$recall = \frac{\#\ shared\ words}{total\ \#\ of\ tokens\ in\ the\ ground\ truth}$$

And then F1 score could be calculated based on precision and recall:

$$F1 = \frac{2}{recall^{-1} + precision^{-1}}$$

Overall F1 score is computed by averaging over the all individual F1 scores of each question and answer pair in the train set.

V. Conclusion and Future Work

In this section we will list potential improvements for future work.

Slide window for input context into the QA model: Currently when training the Question Answering model, context that exceeds the maximum input length are truncated with a fixed window, taking 64 tokens before and 128 after the start index position.

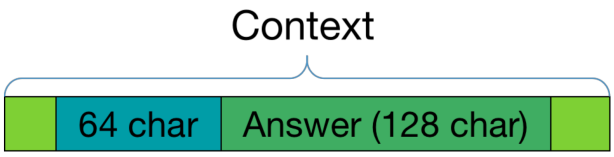

Fig. 12. Context truncation with a fixed window

To improve the training process,a slide window might be implemented. The slide window will start from the very beginning of the context, take the max size of the context input length, then slides. For example, we first take [0,300],then [280,580] having a little overlap between each interval. The slide window will allow to train on the complete context to improve accuracy.

Multitask Model: We currently divide our task into two separate models, one for classification and one for Question Answering. Questions first go through the classification model, and if the question does not belong to the SPAN class, then it will be fed into the QA model with its context for answer span extraction. The two models are fine-tuned separately, which will lead a problematic situation when the classification model returns a wrong class label. Feeding a YES/NO answer question into the QA model due to mis-classification will result in a large loss value. However, little could be done to correct the classification model.

For improvement, the two separated models could be combined into one multitask model, so that if the classification gets mistake and leads to a large loss

after the QA model, we could come back and fine tune the classification model. The fine tune process will be more cohesive.

Better tokenizer for answer and context matching: As mentioned in the dataset input of QA model Implementation section, we have some problematic data that has a mismatch between answer and extracted context. We could further improve the tokenizer to fix this problem.

Answer YES/NO questions: Current classification model can only classify whether the question has an answer of YES/NO, but the model cannot answer if the answer is exactly yes or no. For future work, the classification model could be improved and larger dataset should be fed into the model so that it can not only distinguish whether its a yes/no question but also answer the yes /no questions.

VI. Appendix

表 I

Binary Bert Question Type Classification Results

	Average training loss	**Validation Accuracy**	**Validation Loss**
0	0.195548	0.998191	0.018934
1	0.099253	0.989922	0.041423
2	0.121034	0.998966	0.030197
3	0.097436	0.993798	0.023742
4	0.097659	0.998708	0.032481
5	0.117786	0.998966	0.024009
6	0.105527	0.993023	0.037650
7	0.071998	0.985271	0.041762

References

[1] "SQuAD: The Stanford Question Answering Dataset". In: (). URL: https://rajpurkar.github.io/SQuAD-explorer/.

[2] "china-ai-law-challenge/CAIL2019". In: (). URL: https://github.com/china-ai-law-challenge/CAIL2019/tree/master/%E9%98%85%E8%AF%BB%E7%90%86%E8%A7%A3/data.

[3] "Pre-Training with Whole Word Masking for Chinese BERT". In: (). URL: https://github.com/ymcui/Chinese-BERT-wwm.