EECS 595: Final Project Report
Span-based Question Answering System on SQuAD 2.0

Cheng Qian
University of Michigan
chengqia@umich.edu

Yin Yuan
University of Michigan
aoyin@umich.edu

Abstract

This is the final project report for Umich EECS 595 Natural Language Processing, where we implemented 2 innovative approaches for improving an existing span-based question answering system using BERT or BERT-related pre-trained models on SQuAD 2.0 dataset with unanswerable questions.

1 Introduction

This project works on the machine reading comprehension (MRC) problem, which aims to train the computer to answer certain questions based on the comprehension on related reading materials(Liu et al., 2019). It is a promising task which can be broadly used in the field of human-machine interaction, such as question answering and dialog system. Early-stage MRC systems are generally built with the hypothesis that it is guaranteed to be able to raise up an answer of given question from the related context, which leads to a lack of robustness and generalization for these systems(Zhang et al., 2020). Modern state-of-art MRC systems work on solving problems which contains questions that may not be answerable with given context, which raises up higher requirement of comprehending the contexts and questions.

Nowadays, MRC task can be classified into two parts: reading comprehension and answerability verification. Most models can perform well on finding the answer of answerable questions, while current approaches suffers from the accuracy of verifier, which is caused by the fact that these model fails to fully understand the contextual information and the structure of the context. For example, given a sentence "Paris is located in France" and the question "Where is France located in?", the answer should be unknown. However, modern models are highly possible to answer "Paris" because of incomplete comprehension of the context. Inspired by the human's general reading practice, we want to design a MRC model which builds an answerability verifier and employs a 2 stage reading to hopefully achieve a deeper understanding about the relation of context and question.

The benchmark that we build our MRC model on is SQuAD 2.0(Rajpurkar et al., 2016) dataset. Stanford Question Answering Dataset (SQuAD) is a span-based reading comprehension dataset, consisting of questions posed by crowd workers on a set of Wikipedia articles. Answer to every question, if any, is a segment of text, or *span*, from the corresponding reading passage. When SQuAD updates to 2.0 version, it extends the problem definition by allowing for the possibility that no short answer exists in the provided paragraph, making the problem more realistic. Thus, our challenge includes verifying whether the question is answerable given the context, and doing reading comprehension to find the accurate span of the answer in the context if it has an answer.

Our implementations include 2 methods: Answerability Verifier and 2-Stage Reading with a Reduced Range. Readers can find our code and sample on https://github.com/aoyin1106/EECS_595_Project

2 Related Works

Nowadays, pre-trained language models have achieved success in MRC tasks, such as Bidirectional Encoder Representations from Transformers (BERT). BERT is is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. Pre-trained BERT model can be applied in various specific NLP problems without further modifications.

There are efforts to improve BERT in many fields. For example, ALBERT(Lan et al., 2020) outperforms BERT in many nlp benchmarks such as GLUE, SQuAD, and RACE with less parameters. ALBERT makes it possible for more researchers

and students to fine-tune the model with their own computing devices instead of TPU.

Besides efforts to improve the BERT pre-trained model in performance and computational source constraints, there are also works aims to better utilize the word embeddings given by BERT. These approaches use BERT-related models and implement their own schematics, such as Retrospective Reader(Zhang et al., 2020), U-Net(Sun et al., 2018) and SG-Net(Zhang et al., 2019).

From the results on leaderboard, we find existing models is not good at determining whether a question is answerable. Retrospective Reader(Zhang et al., 2020) introduce a 2 stage method to verify the answerability and do question-aware matching to find more accurate range of answer.

Failure to understand sentence structure is a reason for wrong answer. There has been work that try to exploit the dependency parsing information to the BERT model, called SG-Net(Zhang et al., 2019). However its practical approach is limited and does not reaches much improvement. In practice, SG-Net directly encodes all parsed pairs into arrays of ones, whose length equals the size of each pair. This approach only determine whether there is a relation but do not tell the difference of relations, which is believed to be the key to deeper reading comprehension.

Although BERT is the current state-of-the-art solution for question answering, we find that there are useful ideas in more traditional NLP methods as well. Information retrieval based question answering is a broadly applied technique for finding a sentence segment as the answer for a question(Hirschman et al., 1999). We would use BM25 to find the most related sentence in order to shrink the range of passages with IR before using BERT to do comprehension work.

3 Approaches

This project is done by a group of 2 students. As the teaching stuff requires, each group member should has his or her own approach and contribution to the topic. For our project, Cheng Qian implemented a 2-stage answering model and Yin Yuan implemented a binary answerability verifier, both of which are built based on BERT-based question answering systems.

3.1 Binary Answerability Verifier (Yin Yuan)

I implement an an answerability verifier to improve the performance. The most weakness of existing transformer based models is that they aren't optimized for judging whether a problem is answerable by reading and understanding the context. Inspired by this fact, I decide to implement a binary classifier specialized for judging the binary classification problem.

My practical implementation is to reconstruct a statement with the question text and predicted answer text. If it is an answerable question, the reconstructed statement should agree with the meaning of context. Or the reconstructed statement should be different from what the context states out. Therefore, a difference in the meaning distance should be able to observe between answerable and unanswerable cases. To quantize this difference, I transform the statement text and context text into word embeddings with transformer based pre-trained model as well. It should be noticed that the model for fine-tuning and the model for generating word embeddings for the binary classifier can be different.

3.1.1 Baseline Model

To begin with, I need a baseline model for comparison. I apply ALBERT-base on SQuAD 2.0 datast, which is taken as the baseline of my implementation. The reason why I choose albert-base model is that it is the largest model which is possible to be fine-tuned with my graphic card which has 8G RAM while larger ones generate out-of-memory error and fails. Even the base sized original BERT fails in training on my computer. ALBERT-base model spends 115 minutes on a Nvidea RTX 3070 and gets an f1 score of 80.52%. Training result given by the github repo of albert is 82.1%, which is close to my result.

3.1.2 Statement Reconstruction

It proves to be tricky to merge a question text and an answer text to form a statement sentence because of the various syntax of the questions. There exists four typical kinds of questions, as known as Interrogative sentences: Wh-interrogatives, Yes/No interrogatives, Alternative interrogatives and Tag questions. Each type of questions should be taken into care differently.

Wh-interrogatives are questions that begin with question words. Question words include Who, where, when, why, what, which, whose, whom and how. Wh-interrogatives are the most frequently

used type of questions and is also easy to transfer into a statement. For my implementation, I would judge a question text to be a Wh-interrogatives if the first word is in the list of question word set. Then the question word is simply replaced with the predicted answer text to form a statement text. Thanks to the generosity of Wh-interrogatives in SQuAD dataset, I can get reasonable statement transformation for most cases.

There exists a special type of Wh-interrogatives that cannot be treated with the algorithm described before. These questions are combined with prepositions, such as "In what country is Normandy located?" and "From which countries did the Norse originate?". For these cases I replace the question word and the word followed by it with answer text.

Due to the feature that all answers of SQuAD dataset are picked from the contexts, there is no need to worry about cases of Yes/No interrogatives, alternative interrogatives and Tag questions. Yes/No interrogatives are answered with either yes or no, and tag questions are declarative sentences whose end has question tags. These kind of questions will not occur in the SQuAD 2.0 dataset because their answer need a logical deduction and cannot be directly derived from the context.

It has to be admitted that the statement reconstruction strategies I applied is simple and far from the practical environment. However, it is effective enough for the task implemented in SQuAD benchmark because of its simplification to the question answering task.

3.1.3 NN Based Binary Classifier

I train a binary classifier to judge whether an answer is truly related to the question. The classifier is a neural network of 4 linear layers. It takes the word embeddings of context and statement which is converted by BERT and output a binary value judging whether the input statement has same meaning with the context.

Firstly, I send the embedding of the whole context into the neural network. However, I get a poor classifier. A potential cause of this result is that there are too much irrelevant information in the context that has no relationship to the reconstructed statement sentence.

Therefore, I add a data preprocessing approach before the binary classifier. I split the context into sentences, compare the cosine similarity between each sentence in the context and the reconstructed statement in the form of word embeddings transformed by BERT. I pick up the word embedding with highest cosine similarity with statement embedding and send them to the binary classifier. By this method, I am capable of shrinking down the size of classifier and improve the output accuracy.

This classifier has a training accuracy of 82% and testing accuracy of 67%, which is not an impressive result. However, not all kinds of errors will affect the final result. Since it is a binary classifier, there is actually not anything for it to do with the case that a prediction answer is null while the output of answerability classifier is true. Therefore, most of the judgement errors will be masked when applying it to the ALBERT-base question answering model.

3.1.4 Filter Untrusted Answers

This verifier is connected to a normal SQuAD based model implemented with albert-base. Each time the model generates a predicted answer, this verifier takes the context text and answer text, turns them into word embeddings, finds out the sentence embedding with highest cosine similarity with the statement embedding, send it into the trained binary classifier and judge whether this answer is trusted to be a valid answer or not.

Throughout the whole process, there are some technical details needing to be emphasized. Firstly, it is tricky to the most suitable model for generating the word embedding. What's more, I scale the output of binary classifier before calculating the sigmoid function for better result. Details will be shown in the evaluation section.

3.2 2-Stage Reading with Reduced Range (Cheng Qian)

Machine Reading Comprehension (MRC) problem aims to answer questions given a related context. We focus on the span-based MRC problem, which can be expressed as a tuple (Context, Question, Answer). The answer is defined as a span of words in the context. The modern MRC problems not only need to find the accurate start and end position of the answer, but also need to distinguish whether the question is answerable to avoid wrongly getting plausible answers.(Rajpurkar et al., 2018) This means there are 2 main task MRC need to handle: 1) Verify whether the question is answerable given the context. 2) Do reading comprehension to find the accurate span of the answer in the context.

Human's practical reading habits usually include 2 steps when doing reading comprehension. The

first step is to quickly scan the whole article to get a central concept of what the article is talking about and the general idea of each paragraph.(Zhang et al., 2020) The second step is to read the question, and find the paragraph corresponding to the question and the place where the answer may locate in. Then with this reduced range of context, we can re-read that paragraph to get answer easily.

Inspired by this, I decide to design a new structural method which is a 2-stage reading with reduced range of context to improve both the accuracy of the span and the verification of answerability. The first stage is *scanning* which reads the whole context to give a score $score_{s1}$ of whether the question can be answered. The second stage is *perusing*, which aims to find a smaller range of the context where the answer may locate and do comprehension to find the span of the answer and a score of answerability $score_{s2}$ as well. The scores of answerability in 2 stages will be combined together to make a final decision.

3.2.1 Scanning stage

Word Embeddings Many modern language models such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) have showed very strong ability as a encoder to represent the input context as a dense vector. Take BERT as example, I first concatenate the context and question in the tuple as a whole text, and then tokenize it into word pieces with some additional characters such as classification token *[CLS]* and separation token *[SEP]*. Let's denote the word pieces sequence as $T = (t_1, ..., t_n)$. Each token has a representation which is composed of 3 parts: token embedding, segment embedding and position embedding.(Devlin et al., 2019) The input embedding of the input context can be represented as $E = (e_1, ..., e_n)$ consisting of the representations of tokens. After that E is passed into multiplayer transformers to learn the representation. I use the representation of the first token *[CLS]* to aggregate characterization information of the entire sequence.

Verification Module To grasp a preliminary judgment, a verifier is needed. The verification module is composed of a binary classifier which takes the representation as inputs and gives a output of whether the question can be answered. Here, I use the first output of BERT which is corresponding to *[CLS]* token as the input of this classifier. I use the (context,question,answer) tuples in the training set to train the classifier, where the samples have a lable *1* to represent having answer and lable *0* to represent no answer. The classifier is quite simple which is composed of some fully connected layers. The output of the classifier is passed into a sigmoid function to get a score that measures whether there is an answer. This score will be combined with the answerability score from stage 2 to together decide whether the question can be answered,

3.2.2 Perusing stage

This stage includes 2 main steps: 1) Find a smaller range of context. 2) Find the span of the answer and the answerability.

Reduced Range The main idea is to find a sentence in the context which is most similar to the question. Then take this sentence and another one sentence before and after it as the accurate range of the answer. It is mainly expected to improve the verification accuracy. This won't loose any important information since all the questions are simple questions whose answer can be found with 1 or 2 sentences without inference. The discarded parts play a role of noise with low information to the question.

I still use language models such as BERT to find the most relevant sentence. It is better than other methods such as BM25 because it can provide contextual representation of sentences which can, for example, handle synonym well. Firstly, separate the context into sentences and tokenize these sentences (including question sentence) into word pieces individually. Then feed the token sequence into BERT once a time to get the representation of the each sentence in the context ($C = (c_1, ..., c_n)$) and the representation of the question (q) respectively. Then perform cosine similarity to all of the (c_i, q) $c_i \subseteq C$ pairs. The index of most relevant sentence can be drawn:

$$i = \underset{i}{\operatorname{argmax}} \frac{c_i \cdot q}{\|c_i\| \times \|q\|}, c_i \subseteq C \qquad (1)$$

Then the sentences with index *i-1*, *i*, *i+1* are extracting from the original context and form a new reduced context. Table 1 shows an example of reduced context.

Span and Answerability Prediction For span-based question answering, I concatenate the reduced context and the question together, separated by a separation character. Then I employs the same procedure as stage 1 to get the representation

Context: The outcome was one of the most significant developments in a century of Anglo-French conflict. *France ceded its territory east of the Mississippi to Great Britain.* It ceded French Louisiana west of the Mississippi River (including New Orleans) to its ally Spain. France's colonial presence north of the Caribbean was reduced to the islands of Saint Pierre and Miquelon, confirming Britain's position as the dominant colonial power in eastern North America.
Question: What territory was ceded to Britain?
Answer: territory east of the Mississippi
Reduced Context: The outcome was one of the most significant developments in a century of Anglo-French conflict. France ceded its territory east of the Mississippi to Great Britain. It ceded French Louisiana west of the Mississippi River (including New Orleans) to its ally Spain.

Table 1: An example of the reduced context in stage 2. The sentence in italics is the most relevant to question.

p_i of each token in the reduced context. Additional linear layer is trained during fine-tuning to predict the start and end position of the span. Two special vectors: start embedding *S* and end embedding *E* are also added and trained. (Jurafsky and Martin, 2009) Thus, the probability of starting/ending position is on token p_i can be represented as the dot product of *S* or *E* and p_i. And the credible score of a span starting from *i* and ending at *j* is $score = p_i * S + p_j * E$. The probability is normalized using softmax. The span with the highest score is the final span we predict. For (context, question, answer) tuple with plausible answers, starting and ending both at *[CLS]* token means there is no answer for the question. (Devlin et al., 2019) If the value of the span at *[CLS]* is high, this means the question has lower probability to have answer. Thus, I use the score $score_{s2}$ of the span starting and ending at *[CLS]* with some variations to represent the answerability.

3.2.3 Answerability Verification

The overall score of answerability is a weighted sum of $score_{s1}$ and $score_{s2}$ from 2 stages.

$$score_{answerable} = \alpha * score_{s1} + \beta * score_{s2} \quad (2)$$

The choice of α and β is tuned based on the development set and the choice of encoding model in both stages. Then I use threshold method to make the decision given the scores from both stages (Devlin et al., 2019) (Yang et al., 2019). The threshold δ is also determined on development set. My model gives out the predicted answer if $score_{answerable} > \delta$. Otherwise, it returns an empty string.

4 Evaluation

4.1 Part of Yin Yuan

Since the binary verifier based on reconstructed statement is a schematic irrelevant to the practical models, I try different state-of-the-art pre-trained models for generating the word embeddings and find the optimal result. What's more, inspired by the combinated prediction method applied in retro-reader(Zhang et al., 2020) and SG-Net(Zhang et al., 2019), I add an additional bias factor to the output of my binary classifier which gives me more freedom to tune the verifier performance. Detailed data analysis will be shown as follows:

4.1.1 Model Selection

To generate word embeddings for analyzing semantic similarity, I utilize the api provided by Sentence-Transformer, which is able to return the word embedding generated by different pre-trained models. These models are different in many ways: Some are tuned for general purpose nlp tasks, while others are tuned specifically for semantic search. Some models are designed for best performance while others emphasize more on optimizing the speed and size for fast training speed. Due to these differences, they varies a lot in terms of performance, speed and size. To find out which model is most suitable for my answerability classifier, I apply several representative models and record their performance in table 2.

There are some observations we can find from the result shown in table 2. Firstly, all models showed a good ability to tell apart reconstructed statements that has same meaning with the context from the ones that has different meaning with the context. Secondly, although mpnet-based model gives best performance for improving the score on SQuAD 2.0 benchmark, MiniLM-based model is roughly 5 times faster and also generates a pretty well score. Due to the error masking effect that we have talked before, performance difference be-

Model	Avg. Perf.	Speed	Exact Score	F1 Score
all-MiniLM-L6-v2	58.80	14200	79.77	82.43
multi-qa-MiniLM-L6-cos-v1	58.08	14200	79.80	82.46
all-mpnet-base-v2	63.30	2800	80.02	82.70
all-distilroberta-v1	59.84	4000	79.98	82.62
multi-qa-distilbert-cos-v1	59.41	4000	79.79	82.48

Table 2: Different pre-trained models with general performance specifications and score in applying binary classifiers trained by these models to the SQuAD 2.0 dev. set.

tween models is reduced. What's more, models tuned specially for semantic search tasks show a slight superiority than general models, but the advantage is pretty small. Therefore, I would choose MiniLM based model for further evaluations to save computation time.

4.1.2 Verifier Bias

As is implemented in many other schematic modification efforts such as retro reader and SG-Net, a weighted sum of results given by multiple different approaches enables authors to determine to what extent will their modification influence the final result. A too aggressive biasing may lead to great performance decrease because model modifications often pay unbalanced attention to some specific topics and lose some generality. On the other side, however, too conservative biasing would make nearly no progress than the original approach, which loses the value for model improvement.

In the case of my approach, a bias is added before calculating the sigmoid and rounding of model output so that output values near 0.5 can be recognized as either true or false based on bias value. This approach is reasonable to be applied because of the error hiding effect, which is that binary verifier does nothing to the case that the prediction is "None" while the verifier gives a true. Therefore it is safe to tune the verifier to give more positive predictions even if that hurts the prediction of verifier itself.

I evaluate the performance of binary verifier under different biases with the same model called 'multi-qa-MiniLM-L6-cos-v1' because it gives pretty good performance within much shorter time compared with other models. Effect of bias is evaluated by how good the verifier help improving the final result on SQuAD 2.0 benchmark, while other parameters like answerability prediction error rate are also collected. Results are listed in table 3.

Bias	Error	Fatal	Exact Score	F1 Score
0.30	27.24	0.59	78.94	80.73
0.20	23.36	2.24	79.80	82.46
0.10	21.51	4.24	78.89	81.39
0.00	20.34	12.12	74.77	76.75
-0.10	21.27	18.75	70.84	72.37

Table 3: Error rate, fatal error rate and scores of question answering performance on SQuAD, which are related to the bias. Fatal means errors that will influence the final result on SQuAD dataset.

4.2 Part of Cheng Qian

4.2.1 Setup

In the first stage, I use a pre-trained BERT model *all-mpnet-base-v2* from *SentenceTransformers* (Reimers and Gurevych, 2019) as encoder to get the representation of the context and question, which is tuned for general NLP tasks.

In the second stage, I build the model using the available BERT, RoBERTa and XLM-RoBERTa models, which is from Hugging Face based on Pytorch implementation. I use these pre-trained models without fine-tuning since there are fine-tuned versions for SQuAD 2.0. The threshold δ for the score and the weights α, β of 2 scores are different from models to models and will be discussed later.

4.2.2 Answerability Classifier

The answerability verification classifier has the structure as described in Table 4. It is a simple classifier with fully connected layers with parameters: *EPOCHS = 35*, *BATCH_SIZE = 64*, *LEARNING_RATE = 0.0022*, *criterion = BCEWithLogitsLoss* and *optimizer = optim.Adam*. After we employ a sigmoid function to the output score and round it to the integer to do the prediction, we get that the model has an accuracy of 88.36% on the trainning set and 81.04% on the development set.

Layer	Output Shape	Param
ReLU (Linear-1)	[1,128]	98432
Batch Norm	[1,128]	0
Sigmoid(Linear-2)	[1,32]	4128
Dropout	[1, 32]	0
Linear-3	[1, 1]	33

Table 4: Answerability Classifier

4.2.3 Evaluations on 2 Stage Structure

Here I use 2 metrics, F1 score and Exact Match (EM), to evaluate the performance of QA system. EM is a tougher metrics for goodness measure and F1 is a looser metrics which measures the overlap between the predicted answer and ground truth. I did the evaluation via the official evaluation script provided by SQuAD (Rajpurkar et al., 2018).

Table 5 shows some leading models with their performance from *Hugging Face* on SQuAD 2.0. I will use these 3 models as the baseline. Then I will test my implementation's performance by employ the 2 stage structure on these models. I will show the effect of stage 1 and stage 2 separately.

Model	F1	Exact Match
BERT-large	93.15	86.91
RoBERTa-base	81.00	77.97
XLM-RoBERTa	83.79	79.45

Table 5: The results from pre-trained models from Hugging Face

Table 6 shows the performance of the combination of stage 1 and stage 2 with whole context. In this setting, I used these models respectively in stage 2 when doing final judgement and predicting. I didn't perform the reduced range in stage to see the effect of stage 1 alone. As we can see, both F1 and EM score drop a little, which is not expected. This may due to the inaccuracy of the classifier and the untuned model in stage 2.

Model	F1	Exact Match
BERT-large	92.68	86.66
RoBERTa-base	80.47	78.32
XLM-RoBERTa	82.91	79.02

Table 6: The results from the combination of stage 1 and stage 2 with whole context

Table 7 shows the results from the combination of 2 stages. Here I reduced the range of context to get the more accurate range. We can compare the results in previous experiment to see the effects of employing reduced range. Again, both scores drop a little compared to previous experiments, which is not expected as well. It seems that the reduced range method has few effects towards the result. This will be discussed in the next section.

Model	F1	Exact Match
BERT-large	91.38	86.47
RoBERTa-base	78.17	76.70
XLM-RoBERTa	80.71	78.91

Table 7: The results from the combination of 2 stages with reduced range

5 Discussion

5.1 Part of Yin Yuan

5.1.1 Reconstructed Statement

From the evaluation data, we are confident to conclude that the reconstructed statement can be used to check the answerability because it shows obvious difference between the cases that this statement has semantic similarity with context or not. As is listed in table 3, the neural network shows an accuracy of around 80% without any biasing. Therefore, it can be concluded that there exists semantic difference between right and wrong answers and we can utilize that difference to double check the output of transformer.

However, the current approach too simple to generate statements with suitable and commonly-used syntax structure. For example, in the case that the question text is "In what country is Normandy located?" and the answer text is "France", the statement I generate is "In France is Normandy located", while a more reasonable version should be "Normandy is located in France". Although transformer is able to catch semantic similarity between sentences with different structures, it is still problematic because it may introduce additional noise to the system and thus decrease the answerability prediction accuracy.

5.1.2 Binary Answerability Verifier

It can be concluded that the bias of answerability is useful. A positive bias will change some predicted false with low confidence into predicted true. As is talked before, the mispredicted true will not affect the final result. Therefore, this approach succeeds in preventing the verifier from masking out valid predictions. As is shown in table 3, when bias

increases, the fatal error rate decreases although total prediction error increases. When bias goes from 0.2 to 0.3 the final performance decreases because the verifier takes less effect on the SQuAD prediction.

We can also conclude that the performance difference for models generating sentence embedding does not greatly affect final results. As is shown in table 2, f1 score from verifiers based on different models are quite similar to each other. However, there exists a large speed difference between models. Therefore, simpler models are preferred for this case because of the huge computational resource required for learning the verifier.

5.1.3 Improvement, Limitation and Future Work

This approach does not achieve a great performance improvement than the baseline model, which is to some extent reasonable because it is specialized in improving the case that an unanswerable question is answered. Although I don't have enough computational resource and time to try this approach on larger pre-trained models, I would expect that the improvement for larger models will be less obvious because there will be less cases that can be corrected by this method.

The current verifier model takes the embeddings of reconstructed statement and context sentence that is most similar to the statement as input. However, there may be more complex semantic information lying between the interconnections of sentences, which is not utilized by this model. So one of the future works of this model is to raise up a method that can find out and utilize these more complicated semantic connections.

5.2 Part of Cheng Qian

The overall effect and performance of my design is not good compared to existing model. Both F1 and EM drop a little. This may due to many reasons. We can do some future work to improve the performance based on the following issues of the method.

5.2.1 Answerability Classifier

Table 6 shows utilizing the classifier with the whole context lowers the score. This shows that there is some room for improvement. The neural network for the classifier's accuracy on development set is only about 80%. This value is too low for us to believe its judgement. It hurts the overall score much. The simplicity of the network may cause it not to learn the features well. In addition, I use a language model for general NLP tasks to encode the sentences without fine-tuning it to fit SQuAD 2.0. This may cause the representations of sentences do not show their contextual meanings well. This is a drawback of stage 1. Training or fine-tuning a model for sentences in SQuAD is expected to improve the accuracy of classifier.

5.2.2 Reduced Range

From Table 7 we can see the score drops with reduced range. This is not expected since the deleted sentences contain no information about the question if we locate the most relevant sentence correctly.

Unsatisfactory results may come from the following factors. First, the encoding model for the sentences are not tuned for SQuAD. This may cause the system to find the wrong sentence in the context and then predict the wrong answer. In addition, although the abandoned sentences has low information, the models I use on the 2 stage systems are fine-tuned for SQuAD datasets with complete context. I use them to predict the answer from a narrower span, which may not be appropriate for a smaller context. The models for the prediction need to be fine-tuned for smaller context as well. Also, if let encoders in both stages be the same, the score is expected to grow because of the unity and inheritance.

5.2.3 Answerability Verification

The most important thing in this part is tuning the hyperparameters such as the weight and threshold. We don't have a theoretical method to set the parameters, we can only guess how to adjust the parameters based on the evaluation results. This is not good, because we don't know whether our parameters are optimal. Also, the set of parameters need to adjust to fit different models. It's necessary to develop a set of methods to fine-tune the hyperparameters, or use a new method instead of setting weights and threshold.

6 Conclusion

In this project, we aim at improving scores of the state-of-art transformer-based models on SQuAD 2.0 dataset. Based on our understanding on the process of human reading, we propose two architectural designs which convert the understanding

mechanism of human brain into mathematical models. One design is an answerability verifier, which double check the predicted answer by understanding the meaning of answer and question and compare it with the semantic meaning of context. When a human do similar question answering work, we think it is a common approach to verify his or her potential answer by assuming it is true and see if there is any contradictions. The second approach is a 2-stage reader, which shrinks the range of context and merge the result of two round of predictions. This approach is also based on the human reading process that after a rough reading of the whole context, human would have a brief idea of place of answer and pay more attention to them in the second round of reading.

However, the evaluation result of both methods are not satisfying enough. The answerability verifier reaches little improvement and the 2-stage reader gets little decrease of results. It seems that the architectural designs do not dominate the accuracy of model. What really matters is the pre-trained model that our designs are based on. The lesson we learn form this experience is that we may take it too simple to correctly and effectively model how human brain works when reading and convert it into math models. It is too easy to think we can correctly understand the way of thinking of ourselves. For now, there has not been enough resource and evidence for judging whether these architectural designs meet with the mechanism of human brain or not. To sum up, simple architecture applied to massive data may be the most effective and hopeful choice.

There is still works we can try in the future work. Due to the limit of time and hardware resource, we do not use large size models that have state-of-art results. Instead we use smaller-sized pre-trained models with lower accuracy. We would like to apply our architectural design on more models with larger sizes

7 Division of Work

In this project, Cheng Qian and Yin Yuan contributed equally. For the selection of project, we discussed together to decide which task we aimed to do, and how to design new methods different from previous works. We have read some paper together to determine the direction of the research at the beginning. We all put forward some ideas and discussed their feasibility.

During implementing the specific methods, Yin Yuan is responsible for implementing the Answerability Verifier. And Cheng Qian takes charge of 2 stage reading with reduced range method. We did our methods individually but we often discussed together and gave some suggestions to each other.

For writing the final report, we contributed evenly as well. We jointly completed introduction, related works, conclusion and work division together. For the remaining sections: approaches, evaluation and discussion, there are 2 sub-parts in each of these sections. Cheng Qian and Yin Yuan wrote our own approach, discussion and evaluation corresponding to our implementation.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332, College Park, Maryland, USA. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang. 2019. Neural machine reading comprehension: Methods and trends. *Applied Sciences*, 9(18).

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. 2018. U-net: Machine reading comprehension with unanswerable questions.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2019. Sg-net: Syntax-guided machine reading comprehension.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2020. Retrospective reader for machine reading comprehension.