Ensemble BERT with Data Augmentation and Linguistic Knowledge on SQuAD 2.0

Wen Zhou
zhouwen@stanford.edu

Xianzhe Zhang
xianzhez@stanford.edu

Hang Jiang
hjian42@stanford.edu

Abstract

In this project, we proposed a question answering (QA) system based on baseline BERT model and significantly improved the single baseline BERT model on SQuAD 2.0. We adopted a novel data augmentation approach and integrated linguistic knowledge to build a robust ensemble model. Our model was ranked first place on both dev and test leaderboard, and achieved 79.91(EM)/82.43(F1) and 79.87(EM)/82.54(F1) on dev and test set respectively (March 20, 2019).

1 Introduction

Question Answering (QA), as one of the most important natural language processing tasks, has attracted numerous work and challenges on building effective and robust QA systems. These QA systems make it possible asking questions and retrieve the answers using natural language queries[11]. The Standford Question Answering Dataset (SQuAD) collects 100k question/answer pairs and set up a leaderboard to attract researchers to address this challenge. Based on SQuAD 1.1, SQuAD 2.0 added more questions that could not be answered according to the provided paragraphs[9][8].

Considerable state-of-art researches proposed efficacious methods to address this problem and are improving their performance using different techniques. Bidirectional Encoder Representations from Transformers (BERT)[3] introduces a powerful language model that performs very well on multiple language tasks. Since BERT is not designed specifically for SQuAD, we designed more QA specific techniques to improve its performance on SQuAD 2.0 such as data augmentation[12][14], linguistic knowledge[5], AoA[2], etc., and ensembled those enhanced models with some other models such as BiDAF[10]. Until March 20,2019, our model achieves F1 score 82.427, EM score 79.911 on dev set which is +1.54(F1), +2.202(EM) more than best BERT baseline, and F1 score 82.54, EM score 79.865 on test set, and ranked on No.1 on on both dev and Test leaderboards.

2 Related Work

In our paper, we use BERT as our baseline model. BERT is the state-of-art model for SQuAD 2.0, which is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers of Transformers[3]. This model achieve excellent performance on various NLP tasks with two noteworthy pre-training tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Recently, Google AI Language pushed their model into a new level on SQuAD 2.0 with N-gram masking and synthetic self-training. Compared to BERT's single word masking, N-gram masking training enhanced its ability to handle more complicated problems.

However, pre-training tasks is usually extremely expensive and time-consuming. For us, data augmentation, as an effective technique to improve the model performance in many NLP tasks [12, 14], can also help break the limitation of dataset size and achieve better performance. Wei and Zou[12] proposed Easy Data Agumentation technique (EDA) with four simple operations to enhance the training dataset for text classification. In particular, they perform synonym replacement, random insertion, random swap, and random deletion. However, no one has tried this method for a more challenging task: question answering. Inspired by their method, we designed a similar technique for the SQuAD v2.0 dataset. In the QANet paper[14], researchers augmented the data by adding back-translated data and tuned the augmented data ratio to achieve great performance. We decided to combine the data augmentation technique[12] and sampling strategy[14] to improve the SQuAD 2.0 training set. Besides, researchers found linguistic patterns and knowledge base can benefit question

answering [5, 1]. Their findings inspired us to design a series of linguistic rules to help our model better predict answers.

3 Approach

3.1 Main Approaches

In our project, we first build an ensemble model with a few BERT and BiDAF[10] models. On top of the basic ensemble system, we designed a novel algorithm to augment the training data in order to enrich the diversity of the data. Besides, we added some linguistic knowledge to help the model make predictions. These techniques significantly improved the performance of our system. The overall workflow can be visualized in our appendix Figure a.f1.

3.2 Baseline: Single BERT Model

3.2.1 Pre-training Tasks

The key modules of BERT are two pre-training tasks. The first task Masked LM (MLM) aims to break the limitation of traditional unidirectional model and leverage the power of bidirectional model. In this procedure, they mask 15% of all WordPiece tokens in each sequence randomly and predict these masked tokens. To alleviate the impact of the mismatch between pre-training and fine-tuning, they add more randomness in these 15% tokens: replace the word with the [MASK] token with probability of 80%, replace the word with a random word with probability of 10%, and keep the word unchanged with probability of 10%. The other pre-training task is a binarized "Next Sentence Prediction" procedure which aims to help BERT understand the sentence relationships. They choose two sentences with probability of 50% of the true "next sentence" and probability of 50% of the random sentence from the corpus. The ablation analysis shows that the majority of the improvements are coming from these pre-training tasks.

3.2.2 BERT Model Architecture

BERT[3] is a multi-layer bidirectional Transformer encoder with two novel unsupervised pre-training tasks and constructed input representation. This is the state-of-art method for pre-training contextual language representations. Compared with other contextual representations such as ELMo[6], Generative Pre-Training (GPT)[7], and ULMFit[4], BERT is capable of representing a word in a deep and bidirectional manner, which allows it to be easily adapted to a variety of NLP tasks to achieve the state-of-art results. In our project, we use the single BERT model as the baseline model. In particular, we use both pre-trained uncased base and large BERT models for the task since case information is not the most important feature for question answering. Based on original implementation of the multi-layer bidirectional Transformer, BERT provides two different model sizes:

- $BERT_{BASE}$: L=12, H=768, A=12, Total Parameters=110M
- $BERT_{LARGE}$: L=24, H=1024, A=16, Total Parameters=340M

L: the number of layers (i.e., Transformer blocks); H: the hidden size; A: the number of self-attention heads.

In terms of input, BERT constructs input representation by summing the WordPiece[13] token embeddings, segment embeddings and positional embeddings with lengths up to 512 tokens. WordPiece token embeddings allow the model to compose embeddings for OOV words. Segment embeddings are used to distinguish different sentences. Positional embeddings consider the position of words in a sentence to compose the final contextual representations. The ablation analysis shows that the majority of the improvements are coming from two pre-training tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

3.2.3 BERT Adaptation to SQuAD

We used the Google open source implementation of BERT in Tensorflow for the project. The open source github also provides its code to adapt BERT to SQuAD. The authors pack the input question and the context paragraph into a single input sequence separated by a special [sep] token. Two new learnable parameters are added: a start vector $S \in \mathbb{R}^H$, and an end vector $E \in \mathbb{R}^H$ (H denotes the hidden vector dimension). Then the start_logit and end_logit of word i can be calculated by:

$$\text{start\_logit}(i) = S \cdot T_i, \text{end\_logit}(i) = E \cdot T_i$$

, where T_i denotes the final hidden vector from BERT for the i^{th} input token. During train time, the probabilities of word i being the start and end of the answer span are calculated separately by:

$$P_i^{\text{start}} = \text{softmax}(\text{start\_logit}(i)) = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}, P_i^{\text{end}} = \text{softmax}(\text{end\_logit}(i)) = \frac{e^{E \cdot T_i}}{\sum_j e^{E \cdot T_j}}$$

3.3 Original Work

3.3.1 Data Augmentation

Easy Data Agumentation technique based on word insertion/deletion/swap/replacement operations [12] is effective to text classification. However, Question Answering (QA) is different from text classification. For text classification, the label of an augmented instance will not change after the augmentation operations are performed on the original instance [12]. Nevertheless, on our QA task, the answer span will change if we add or delete words. Random swaps can also separate a continuous answer span. Thus, it is more challenging to perform random insertions, swap, and deletions on the SQuAD dataset. To make it simple, we determine to only perform synonym and random word replacement with NLTK and WordNet[1] on the contexts of the SQuAD dataset. The questions are left unchanged. We do not have to take special care of the words inside the answer span because synonyms will do a good job at preserving the original answer span. For example, if "lead" is replaced with "leading", the original answer span remains unchanged and we can use the same answer span for the augmented data.

- **Question**: What was Beyoncé's role in Destiny's Child?
- **Context**: Beyoncé Giselle Knowles-Carter is an American singer, songwriter, record producer and actress. ..... rose to fame in the late 1990s as **lead singer** of R&B girl-group Destiny's Child.....
- **Answer Span**: 44-45 **before**: lead singer vs. **after**: leading singer

The following explains how our first data augmentation algorithm works. In our later work, we also experimented with different varations of the algorithm by removing the random word replacement part and adding a condition to check whether a word is a stop word before replacement. For each word in a paragraph context, we will:

- 20% of the time: call the function to replace a word with a synonym
  - if synonyms exist: randomly choose a synonym from its synonyms using wordnet
  - otherwise: replace the word with a random word from the large BERT vocabulary
- 80% of the time: leave the word unchanged

3.3.2 Post-processing with Liguistic Knowledge

During prediction time, the probability for a text span from word i to word j being the answer is:

$$P(i, j) = softmax(\text{start\_logit}(i) + \text{end\_logit}(j))$$

The authors also make a constraint that the end token must be after the start token ($i \leq j$).

On top of that, we incorporated linguistic knowledge to the predictions:

- for "When" questions, if Text(i) $\in$ ['before', 'after', 'about', 'around', 'from', 'during'], which are common prepositional words that human beings usually use when answering "when" questions, for example: "before World War I": $P(i, j)$ += 0.2;
- for "Where" questions, if Text(i) $\in$ ['in','at','on','behind','from','through','between','throughout'], which are common prepositional words that human beings usually use when answering "where" questions, for example: "at Konwiktorska Street": $P(i, j)$ += 0.2;
- for "Whose" questions, if " 's " is in Text(i, j):$P(i, j)$ += 0.2, as a typical answer format to "whose" question would be "SOMEONE's";
- for "Which" questions, if Text(i) = "the", $P(i, j)$ += 0.2, as "the" is usually used when people are trying to explain which thing they mean.

Note that after adding 0.2, $P(i, j)$ may be greater than 1, but this does not really matter because we only need to get the text spans with top probabilities.

[1] https://github.com/nltk/nltk

3.3.3 Ensemble

In our project, we also ensemble several different BERT models (cased/uncased, base/large): for each question, we output the n-best predictions made by each model along with the probability (after post-processing), then for each proposed prediction, we add up the probability from each model together, and finally we output the prediction with the highest probability as the answer to that question. Since large BERT models significantly outperforms base BERT models, we assigned a B$\times$ weight to large BERT models (B = number of base models in the ensemble, as we want the B base models together to have the same voting power as 1 large model).

We also incorporated BiDAF into our ensemble model: for each proposed prediction to a question, if that prediction is the final answer to that question predicted by BiDAF, then add 0.2 probability:

$$\text{ensemble\_prob}(i,j) = \left(\sum_{m \in \text{models}} w_m * P_m(i,j)\right) + 0.2 \text{ \{if BiDAF predicts i-j as the answer span\}}$$

$$w_m = \begin{cases} 1 & \text{(if m is a base BERT model)} \\ \text{B} & \text{(if m is a large BERT model)} \end{cases}$$

4 Experiments

4.1 Data

We applied our model on the Stanford Question Answering Dataset (SQuAD) V2.0 [9]. The input to the model is a question with a context paragraph, and the output should be the the span of text in the paragraph that can answer the question. There are about half of the questions in the dataset that cannot be answered given the provided paragraph [8]. We used 129,941 examples as the training set, 6078 examples as the dev set, and 5915 examples as the test set.

4.2 Evaluation Method

We used the same evaluation metrics as used by the SQuAD leader board: Exact Match (EM) score and F1 score. EM measures whether the model prediction matches the true answer exactly:

$$EM = \begin{cases} 1 & \text{(if prediction exactly matches the true answer)} \\ 0 & \text{(otherwise)} \end{cases}$$

F1 is the harmonic average of precision and recall: F1-score $= 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$, where precision $= \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$, recall $= \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$

4.3 Experimental details

As a starting point, we first downloaded the starting code for BiDAF model [2] and trained it with the default setting, it took around 6 hours to train 30 epochs on a NV12 GPU. Then we focused on the BERT model. We downloaded the TensorFlow implementation of BERT model and pretrained weights released by Google[3]. On GPU, We could only train the base BERT model with a very small batch size due to the memory limit and it took about 3.5 hours to train one epoch. The evaluations of each trial are summarized in Table a.t1 in Appendix. We turned to TPUs on Google Cloud Platform to train BERT models more efficiently. With a TPU, we were able to train the base BERT model in about 9 min per epoch, and the large BERT model in about 15 min per epoch. The evaluations of each trial are summarized in Table 1.

We tuned the model with the data described in section 4.1 and searched for the optimal combination of hyper-parameters including **batch size**, **max sequence length**, **learning rate**, **number of train epochs**, **base/large pre-trained model**, **uncased/cased model**. In detail, to search for one hyper-parameter, we fixed other parameters to observe the result of changing this one. For example, we fixed on base uncased pre-trained model, max sequence length of 384, learning rate 3E-5, 3 epochs, then tried 16, 32, 64 for batch size.

After that, we implemented Attention-over-Attention at the last hidden layer of BERT because we thought it might help if we treat question and context individually and extract Context-to-Question

[2] https://github.com/chrischute/squad

[3] https://github.com/google-research/bert

attention and Question-to-Context separately. The evaluations of each experiment trial are summarized in Table a.t2 in Appendix. However, we found that adding AoA did not improve the performance of BERT. We thought this is because the BERT Transformer already uses bidirectional self-attention, so no additional attention is needed for BERT. Therefore, we terminated our exploration on AoA, and in order to further improve the performance of BERT, we turned to data augmentation and incorporating linguistic knowledge into BERT, as discussed in section 3.3.1 and 3.3.2.

By adopting our novel approach for data augmentation in 3.3.1, we did two major experiments to explore how data augmentation can improve our BERT model. We hope to add diversity to the existing training data and prevent our model from overfitting the original training set. We conducted our first experiments by manipulating the settings of data augmentation algorithm to see whether removing random word replacement and adding stop word check can help create a dataset with higher quality. Removing random word replacement means not to replace a selected word without synonyms with a random word; instead, we will simply leave the word unchanged. The other change is to add a stop word check so that only non-stop words can be selected for word replacement. Meanwhile, we conducted the second round of experiments to explore how much data should be added to the original training set. We did experiments by adding 33%, 1, 2, and 3 augmented data. Due to time constraint, we did not have a chance of applying the best word replacement setting found in the previous step to this experiment. We used the original synonym replacement setting with random word replacement. We then fined-tuned models for the best performing augmented datasets and used them for ensemble.

After fine-tuning single models, we picked the ones with decent performance on the dev set and post-processed their predictions using linguistic knowledge, and then combine them into several ensemble models using the approach described in section 3.3.3. The evaluations of ensemble models are also summarized in Table 1.

4.4 Results

Trial #	model	batch_size	max_seq_length	learning_rate	#epochs	dev_f1	dev_em
1	base_uncased	16	384	3.00E-05	3	75.355	72.162
2	base_uncased	32	384	3.00E-05	3	**75.841**	**72.557**
3	base_uncased	64	384	3.00E-05	3	74.931	71.685
4	base_uncased	64	384	3.00E-05	4	72.917	69.464
5	base_uncased	32	384	2.00E-05	3	75.225	72.244
6	base_uncased	32	384	5.00E-05	3	75.173	71.932
7	base_uncased	32	384	3.00E-05	5	75.106	71.389
8	base_uncased	32	256	3.00E-05	3	73.755	70.665
9	base_uncased	32	128	3.00E-05	3	74.591	71.833
10	base_uncased	24	384	3.00E-05	2	75.693	72.54
11	base_uncased	24	384	5.00E-05	4	75.207	71.619
12	base_cased	32	384	3.00E-05	3	**75.361**	**72.113**
13	large_uncased	24	384	3.00E-05	2	**80.167**	**77.081**
14	large_uncased	24	480	3.00E-05	2	78.667	75.699
15	large_uncased	32	384	3.00E-05	2	78.71	75.712
16	large_cased	24	384	3.00E-05	2	**80.56**	**77.345**
Ens1	1 +2 + 5 + 6 + 10 + 11					77.352	74.515
Ens2	1 +2 + 5 + 6 + 10 + 11 + 0.2 BiDAF					77.362	74.58
Ens3	1 +2 + 5 + 6 + 10 + 11 + 12 + 13 + 0.2 BiDAF					78.479	75.699
Ens4	1 +2 + 5 + 6 + 10 + 11 + 12 + 13 + Weighting + 0.2 BiDAF					80.771	78.019
Ens5	1 +2 + 5 + 6 + 10 + 11 + 12 + 13 + 15 + 16 + Weighting + 0.2 BiDAF					81.666	79.154
Ens6	1 +2 + 5 + 6 + 10 + 11 + 12 + 13 + 15 + 16 + Weighting + 0.2 BiDAF + linguistic knowledge					**81.764**	**79.434**

Table 1: Summary of evaluations on BERT models as well as the ensemble models(Ens).

As shown in Table 1, we made 16 trails on single BERT models on TPU. In these experiments, the best-performed base BERT model is #2, which achieved a 75.841 F1 and a 72.557 EM on dev set. The best-performed large BERT model is #16, which achieved a 80.56 F1 and a 77.345 EM on dev set. With data augmentation, the best single model is #Aug8, which achieved a 80.056 F1 and a 76.933 EM on dev set.

As for ensemble models, we found that incorporating BiDAF and models trained with augmented data, using proper weighting scheme, and post-processing with linguistic knowledge can all help to

Trial #	model	batch_size	dev_f1	dev_em
Aug1	Large_uncased + aug x 3 + rdn	24	77.710	74.301
Aug2	Large_uncased + aug x 2 + rdn	24	**79.220**	**76.061**
Aug3	Large_uncased + aug x 1 + rdn	24	78.979	75.337
Aug4	Large_uncased + aug x 0.33 + rdn	24	78.734	75.568
Aug5	Large_uncased + aug x 0.33 + rdn	32	79.095	75.897
Aug6	Large_uncased + aug x 0.33 + stp	24	77.919	74.548
Aug7	Large_uncased + aug x 0.33 + stp	32	**79.586**	**76.167**
Aug8	Large_cased + aug x 0.33 + stp	32	**80.056**	**76.933**
Ens7	Ens6 + Aug2 + Aug7 + Aug8		**82.427 (+0.663)**	**79.911 (+0.477)**

Table 2: Experiments with augmented data. **rdn**: random word replacement; **stp**: not replacing stop words. Experiments all have epoch number 2, learning rate 3.00e-5, and max sequence length 384.

Trial #	model	dev_f1	dev_em	test_f1	test_em
Ens5	BERTs with Weighting + 0.2 BiDAF	81.666	79.154	81.518	78.664
Ens7	Ens5 + linguistic knowledge + data augmentation	82.427	79.911	**82.540**	**79.865**

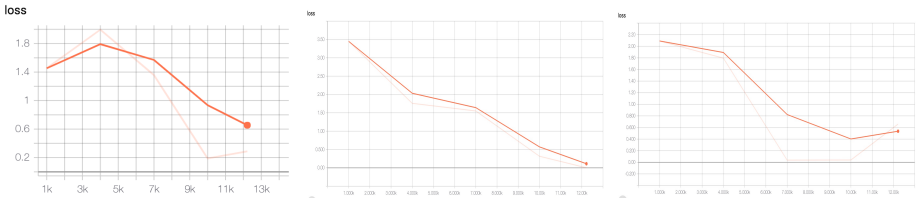
Table 3: F1 and EM scores on test set.

enhance the performance. The best-performed ensemble model is #Ens7, which achieved a 82.427 F1 and a 79.911 EM on dev set, and a 82.540 F1 and a 79.865 EM on test set. Up to now (March 20, 2019), we are ranked at top 1 on both the test and the dev class PCE leaderboards.

5 Analysis

5.1 Hyper-parameter Search

We first searched for the best learning rate (2E-5:trail #5, 3E-5:trial #2, 5E-5:trial #6), and their loss vs. iteration plots are shown in Fig.1. We found that 3E-5 is the best learning rate and continued to use it in the later experiments.



(1)#5 LR too small - not converge (2)#2 Good LR (3)#6 LR too large - miss the optimal

Figure 1: Plots of loss vs. iterations on tuning learning rate(LR).

For base BERT model, by comparing trials #1, #2, #3, we found that the best batch size is 32; by comparing trials #3 vs. #4 and #2 vs. #7, we found that the best number of training epochs is 3; and by comparing trials #2, #8, #9, we found that the best sequence length is 384.

For large BERT model, by comparing trials #13, #14, we found that the best sequence length is 384; by comparing trials #13, #15, we found that larger batch size will make both F1 and EM scores lower, which should be due to over-fitting, as larger batch size tends to decrease the model's generalization ability and the large model has so many learnable parameters that it is more inclined to overfit.

5.2 Ensemble of BERTs

We first assembled 6 base uncased BERT models that have relatively high F1/EM scores together (#Ens1), and we observed that the ensemble model achieved both higher F1 and higher EM scores than the best single base model (#2). Then we incorporated 0.2 BiDAF into this ensemble model (as described in section 1.3) and observed a small improvement (#Ens2). We gave BiDAF such a low voting power because it performs significantly worse than BERT, but we still want to incorporate it into our model in order to mitigate some systematic error of the BERT model.

Then we continued to incorporate one base cased BERT model and one large uncased BERT model into the ensemble model (#Ens3), however, we found that this new model outperformed all the single base models but not the single large model (#13), which performed significantly better than all single base models. Considering this issue, we decided to incorporating a weighting scheme described in section 1.3 to give a higher voting power to the large model. With this weighting, the ensemble model (#Ens4) outperformed all single models. We continued to add 2 more large BERT models and get a well-performed ensemble model (#Ens5). To make further improvements, we applied data augmentation and linguistic knowledge.

5.3 Data Augmentation

The first experiment was to find the best parameters for data augmentation algorithm. As shown in Table 2, we applied different parameters (#4, #5, #6, #7) on 33% augmented data and fine-tuned with large uncased BERT to see whether these changes could better a dataset. It turned out that replacing non-stop words and stop using random word replacement can improve the data quality. Therefore, we also used this setting on a large cased BERT in Table 2's #8 and achieved great results. This small change of settings works well, probably because stop words are not important to the sentence meaning and using random words can introduce too much noise to the new dataset.

The second round of experiments was to explore how much data should be added to the original training set. In the Table 2, trials #1, #2, #3, #4 experiments showed that injecting 2 augmented data works the best. This is similar to the findings in the previous work [14], where they found adding too much or too little data could lead to worse performance. It works better because adding x 2 augmented data introduces a decent amount of diversity to the training set. Due to time constraint, we did not have a chance of applying the best word replacement setting from the previous step to this experiment. We used the original synonym replacement setting with random word replacement.

5.4 Post-processing with linguistic knowledge

By comparing #Ens5 and #Ens6 in Table 1, we can see that post-processing the predictions with linguistic knowledge can improve EM by 0.28 and improve F1 by 0.098. Here is an example showing how linguistic knowledge can improve the predictions:

- **Question**: **Where** does heat rejection occur in the Rankine cycle?
- **Answer**: in the condenser
- **#Ens5 Prediction (without post-processing)**: condenser
- **#Ens6 Prediction (with post-processing)**: in the condenser

As illustrated in this example, #Ens5 (ensemble of weighted BERTs + BiDAF) cannot learn grammar rules and just predicts "condenser" without the proper prepositional word in the front. With our post-processing described in section 3.3.3, since we added extra probability for answer start with a prepositional word commonly used to describe "where", #Ens6 outputs a better answer. Such improvement can significantly boost the EM score, while can only slightly boost the F1 score since the prediction "condenser" partially matches the true answer and thus already has a high F1.

The total improvements on the EM and F1 scores are limited because we only performed post-processing for the "when", "Where", "Whose", "Which" questions. Only about 14% of the questions in the dataset are these types as shown in Fig.a.f2 in Appendix. For the other types of questions such as "What" or "How", it is hard to think of a good linguistic rule to apply on the predicted answers since the answers to these questions can have varied forms.

5.5 Error Analysis

We examined the predictions by our best-performed model on dev set (#Ens7) and found that it suffered from the following types of errors:

1. Context: ...there were rich and well socially standing Chinese while there were less rich Mongol and Semu than there were Mongol and Semu who lived in poverty and were ill treated.

- **Question:** There were many Mongols with what unexpected status?
- **Answer:** lived in poverty and were ill treated vs. **Prediction:** less rich
- **Analysis:** This error is caused by model's lack of knowledge in linguistic structure. Human never makes this mistake because the predicted answer and the human answer are indeed

semantically similar at the first glance. However, the head of "less" is not "rich" but "Mongols", meaning "less" and "rich" is not forming a phrase here. Thus, a human will never choose "less rich" as an answer, but BERT does not understand the underlying linguistic structure and lead it to make the mistake.

- **Fix:** There is an easy fix to this error. We can parse the sentence using dependency parse and check whether the predicted words are in the same clause. If not, we can directly abandon the possible answer as a candidate, and search an answer from other predicted possible answers.

2. Context: ...the Commission has a monopoly on initiating the legislative procedure, although the Council is the d̈e facto catalyst of many legislative initiatives.. The Parliament can also formally request the Commission to submit a legislative proposal but the Commission can reject such a suggestion...

- **Question:** Who is the sole governing authority capable of initiating legislative proposals?
- **Answer:** the Commission vs. **Prediction:** the Council
- **Analysis:** the model is sort of weak at distinguishing two close entities in the paragraph, even though the difference between "the Commission" and "the Council" is quite obvious to human readers. Both "the Commission" and "the Council" are predicted candidates by the model. The model maybe does not understand the transition word "although", which signals a contrast relationship.
- **Fix:** We can fix this by checking which candidate answer is the actor of "initiating the legislative procedure" in the context. We can obtain the original linguistic structure of the context and pick the one that can be the actor of action in the query.

3. Context: ...In turn, imposing restrictions on the available resources is what distinguishes computational complexity from computability theory: the latter theory asks what kind of problems can, in principle, be solved algorithmically

- **Question:** What field of computer science is primarily concerned with determining the likelihood of whether or not a problem can ultimately be solved using algorithms?
- **Answer:** computability theory vs. **Prediction:** computational complexity theory
- **Analysis:** For some domain specific problems, BERT performs not very well. It's hard to tell the difference between some terminologies even for a human not specialized in this area. On one hand, not enough semantic analysis might yield this type of error. On the other hand, the generic word embeddings tend to give similar result for derivatives of same words which might be quite different in a specific context, e.g. computational complexity theory and computability theory in this example.
- **Fix:** Enhanced semantic analysis might help tell similar word embeddings but with different meanings. Relation extractor could help address this type of problems. We can also train domain specific embeddings to help correct the error in a domain specific context.

6 Conclusion

In our project, we significantly improved the BERT model using a few novel NLP techniques. First of all, we designed a data augmentation algorithm to generate additional training data for question answering task, which enriched the diversity of the original training set and improved the the ensemble model by around 1%. Secondly, we designed a few linguistic heuristics to help the model choose the best possible answer, which also improves the performance of BERT models consistently. At last, we are able to ensemble BERT models, BiDAF, and linguistic knowledge together to build a robust model that places #1 on both dev and test set in the leaderboard. Overall, our model improves around 2% in F1 and EM on the dev set compared to a single large uncased BERT.

In the future, there are a few directions that researchers can follow to improve the model. First of all, it is possible to compare the existing data augmentation algorithm with a NMT-based data augmentation[14]. NMT-based method can probably make the augmented data more diverse and potentially improve the model performance. Secondly, more linguistic knowledge can be integrated into the model. For instance, it is plausible to use POS and NER tagger to detect certain patterns and help the classifier make decisions. At last, ensembling different models can overcome the systematic errors of BERT.

7 Appendix

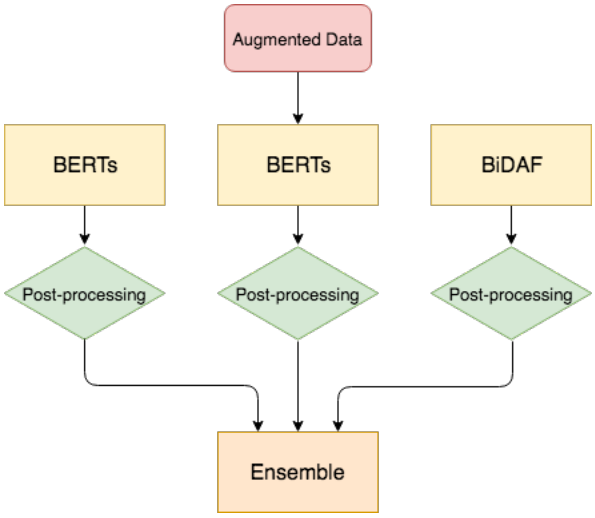


Figure a.f1: Diagram of our model architecture.

Trial #	model	batch_size	max_seq_length	learning_rate	#epochs	dev_f1	dev_em
1	base_uncased	4	100	3.00E-05	2	74.441	72.08
2	base_uncased	4	256	5.00E-05	3	73.444	70.303
3	base_uncased	4	128	3.00E-05	4	74.256	71.52
4	BiDAF	64	400	0.5	30	60.489	57.184

Table a.t1: Summary of evaluations on baseline BERT models and BiDAF model trained on GPU.

Trial #	model	batch_size	max_seq_length	learning_rate	#epochs	dev_f1	dev_em
AoA1	base_uncased + AoA	32	384	3.00E-05	3	75.619	71.882
AoA2	base_uncased + AoA	32	384	3.00E-05	5	74.529	70.895
AoA3	base_uncased + AoA	32	512	3.00E-05	3	74.276	70.879
AoA4	base_uncased + AoA	24	512	2.00E-05	2	75.174	72.047
AoA5	large_uncased + AoA	24	440	3.00E-05	2	78.158	75.008
AoA6	large_uncased + AoA	24	440	3.00E-05	3	79.803	76.39
AoA7	large_cased + AoA	24	440	3.00E-05	3	79.047	75.88

Table a.t2: Summary of evaluations on BERT + AoA models (trials marked as AoA).

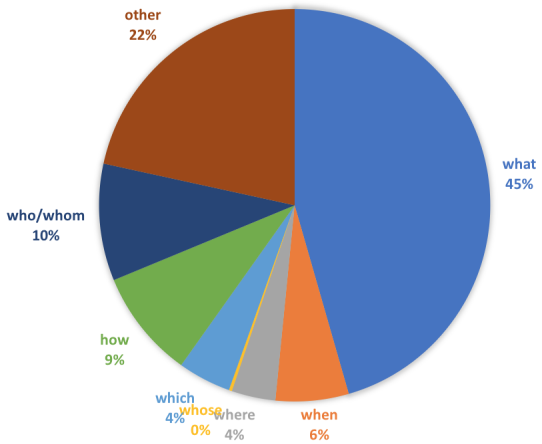


Figure a.f2: Statistics of question types in SQuAD 2.0.

References

[1] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[2] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*, 2016.

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

[4] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[5] Diego Mollá and Mary Gardiner. Answerfinder: question answering by combining lexical, syntactic and semantic information. In *Proceedings of the Australasian Language Technology Workshop 2004*, pages 9–16, 2004.

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

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

[8] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

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

[10] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[11] Marco Antonio Calijorne Soares and Fernando Silva Parreiras. A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University-Computer and Information Sciences*, 2018.

[12] Jason W Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

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

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