Self-Ensemble Methods with BERT on SQuAD v2.0

Thaddeus Segura *Thaddeus-segura@berkeley.edu*

Abstract

With its introduction in 2018, BERT set state of the art results on benchmark datasets such as SQuAD, even surpassing human level performance. However, today all of the top performing models on SQuAD incorporate an ensemble approach. In this paper, I explore various Self-Ensemble approaches employed on BERT against the SQuAD v2.0 dataset. The results show that Self-Ensemble methods are far more likely to "stack" their errors than they are to compensate for weaknesses inherent in the model. However, at least one self-ensemble approach leads to improvement over the baseline and may be generalizable to other models.

1 Introduction

BERT is a language model that was introduced in 2018. It was pretrained on a large corpus of unlabeled text, through two tasks, masked language modeling (MLM), and next sentence prediction (NSP). BERT has shown outstanding performance on various NLP benchmarks, delivering state-of-the-art results with its introduction (GLUE, MutliNLI, SQuAD v1.1 and v2.0). Due to its success, a number of models have been adapted from BERT such as RoBERTa and ALBERT, which have now improved upon the results set by BERT. However, the most successful models on benchmarks such as SQuAD, all feature ensemble methods where multiple models are blended together.

The SQuAD dataset is a benchmark dataset of reading comprehension questions sourced from Wikipedia. Each question offers a "context" paragraph (which is shared over many questions), a specific "question", and the model should return the text from within the context that answers said question. SQuAD v2.0 complicates this further by introducing a significant number of unanswerable questions, where the model should return only an empty string as the correct answer.

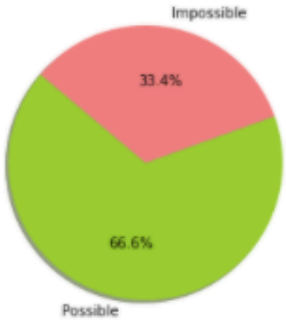

Figure 1: Question breakdown in SQuAD v2.0: The train set consists of 86,821 answerable, and 43,498 unanswerable questions.

Because BERT is pretrained only on MLM and NSP, it is critical to finetune BERT for the specific task at hand. This is still a computationally expensive task as the BERT Base model has ~110 million parameters and BERT Large has ~340 million, however, it is far less computationally expensive than training a model of BERT's scale from scratch.

In this paper, I investigate a number of Self-Ensemble approaches, in which I try to combine predictions from BERT in various ways to achieve results that surpass a baseline. In theory, insights gained from these approaches could be applied to new state-of-the-art models, improving their performance beyond the levels that could be achieved through fine-tuning alone. Specifically, I try *simple voting with discrete data presentation, a DNN for question-model matching, Multiple Choice prediction filtering, sequence classification supplement, and weighted-voting.*

1) Voting with Discrete data Presentation: In this approach, I build upon the ideas regarding weight initialization. Specifically, BERT is very sensitive to the initialization of weights, and can achieve significantly different results based only on the random initialization of its weights. I take this further to explore if fine tuning separate BERT models on separate subsets of the data and

then combining the predictions, will improve or impact overall model performance. I attempted this with a 4 way split, and 8 way split.

2) DNN for question-model matching: Here I build upon the approach in (1) by training a deep neural net to select the best potential model for answering each question.

3) Multiple Choice prediction filtering: In this approach I use the multiple models to generate their top predictions and then finetune a BERT for Multiple Choice model to select the correct choice. This is attempted with a 5 way multiple choice and a 2 way multiple choice setup.

4) Sequence Classification Supplement: This approach seeks to improve performance by focusing on the unanswerable questions. A BERT for Sequence Classification model is trained to identify unanswerable questions, and then these predictions are applied to override the predictions made by the other models.

5) Weighted Voting: This revisited the predictions made by the various models discussed in (1)-(4) and combined them in more intelligent ways. This ultimately delivered the best results.

The specifics of each model are discussed in Section 3, and the results are presented in Section 4.

2 Related Work

The inspiration for the experiments described above were taken from a number of sources. The foundations of this project can be traced back to the work of Devlin et al., 2018 with their introduction of BERT and its application on SQuAD.

The idea of discrete data presentation came from observations made on weight initializations by Dodge et al., 2020 in which they only changed the weight initializations and data order on the final layer of the model (representing only 0.0006% of the total number of parameters). My belief was that if this small tweak could lead to such significant variance in results, then training separate models on separate subsets of data could allow each to learn various features which together could offer greater generalization and accuracy.

The Self-Ensemble approach was an adaptation of an approach applied by Xu et al., 2020 in which they explored both self-ensemble and self-distillation. Specifically, they identified that they could achieve ensemble-like results through using various check points in the training process on a single model which is an approach I applied in my experiments. Additionally, they made key observations regarding voting and averaging of BERT models for question answering.

For the multiple choice approaches, much of the work here was adapted from work done with BERT on datasets such as RACE, which also features a context paragraph and then specific prompts, but with multiple choice answers. Zhu et al., 2018 lay the ground work for this approach and subsequent work such as that done by Si et al., 2019 digs deeper into the approach to determine specific what is being learned in this process.

Finally, the application of BERT for sequence classification was inspired by work done by Sun et al., 2019 and sentiment classification approaches such as those employed by Alaparthi et al., 2020 where they employed multiple techniques on the IMDB dataset. Here I reworked their approaches to classify sentences as answerable/unanswerable, rather than identifying sentiment.

3 Methodology

The experiments were all done through the fine-tuning a BERT model with the goal of reducing cross entropy loss through stochastic gradient descent. Experiments included both BERT base and BERT Large model architectures, but all models used the "cased" versions as the EDA revealed that there was a significant increase in cased words that appeared as answers compared to the overall dataset. Hyper-parameters were tuned for each specific experiment, with some models building upon previous ones. The overall goal was to explore if variations on the training strategy could improve the results over the baseline. The hope is that if this is the case, then as new state-of-the-art models emerge, results can be improved beyond fine-tuning through a self-ensemble process.

3.1 Baseline

A fine-tuned baseline model was created first using BERT Large Cased trained on trained through TensorFlow on GCP TPU's. Optimal results were achieved using a learning rate of 3e-5, a batch size of 24, and training for 3 epochs. The full training set was used in its normal form, and predictions were taken "as-is" for evaluation.

3.2 Voting with Discrete Data Presentation

In this first experiment, I split the dataset into discrete groups and trained a model on each one. I attempted both a 4-way split, and an 8-way split, resulting in 4 and 8 BERT models respectively. Each of these were also BERT Large trained with TensorFlow on GCP TPU's. The potential advantage with this approach is that the overall training time was very similar to that of the baseline as the total number of training examples were the same. If successful, this means that a similar approach could be applied to modeling in the context of "continual learning" as a new model could be trained on the newest batch of data, and then be factored into overall predictions. However, the risk is that the model may not converge or generalize well enough given the limited samples. The results were combined through both simple voting and weighted voting based on the model confidence for their top predictions.

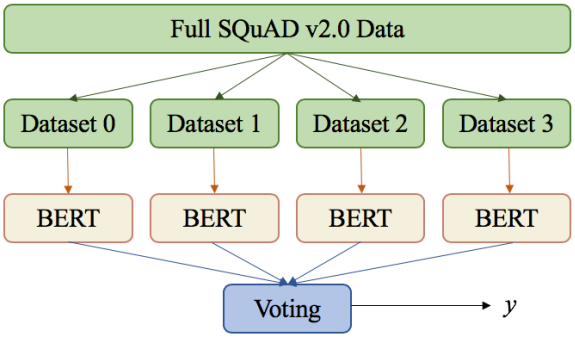

Figure 2: Architecture of the discrete data presentation model.

3.3 Model combination with DNN

This set of experiments attempted to use a deep neural network to look at a specific question and then predict the model that would be most likely to provide the correct answer. Ultimately, this did not work in practice as the DNN lacked understanding of language, but it did give rise to the multiple-choice models discussed below.

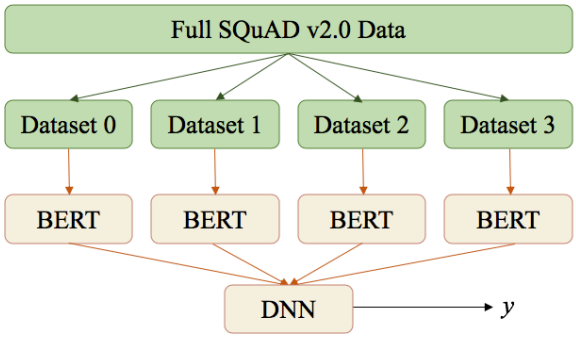

Figure 3: Architecture of the deep neural net layered on the discrete data presentation model. Here, the solution is selected from the predications made by the other BERT Models.

3.4 Multiple Choice Models

In an attempt to overcome the limitations of the DNN approach, I used another BERT model to select the top answer from a list of predictions generated by the other models, as this would capture the understanding of the language in the context/questions.

I tried two different approaches, a 5-way selection, and a 2-way selection. In both approaches, I had to first apply used a fine-tuned model back on the training set to generate predictions for the training set. Then I took the top predictions from each model and used them to generate an answer set for the model to choose from. For the 5-way selection model I also appended the null option ("") as the first choice in every answer set.

I then used the Hugging Face Transformer implementation of BERTforMultipleChoice to train the model on Tesla P100 GPU's. However, this ran into major training limitations, particularly on the 5-way model as the training embeddings were so large that the GPU could only handle a batch size of 4. The 2-way model came after identifying that most of the time the correct answer was one of the top two choices. This also reduced the demands on the GPU RAM, allowing for a larger batch size and thus smoother SGD updates. These models were built on the 4-way splits, 8-way splits, and an unsplit model using intermittent checkpoints as described by Xu et al., 2020. Of these, the unsplit model had the best overall performance.

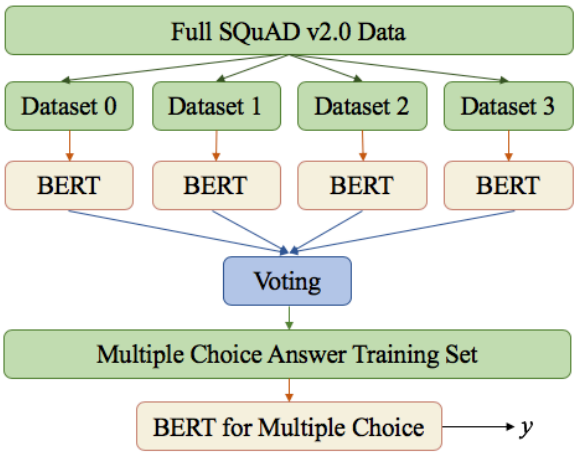

Figure 4: Architecture of the multiple choice model.

3.5 Sequence Classification Supplement

The next approach employed was aimed at improving the model performance on the questions that could not be answered. This was done by labeling the full training dataset as 0 for answerable, and 1 for unanswerable. This was then used to train a BERTforSequenceClassification model from Hugging Face again on Tesla P100 GPU's.

Once training was completed, predictions were run on the dev set, and the results were combined by comparing the predictions from the initial predictions for the dev set, to the sequence classification results. For any question that was identified as unanswerable by the sequence classification model, the prediction was changed to the null answer. This approach also ran into memory issues, as the question and context had to be concatenated, resulting in very long input strings that used the full 512 token embedding.

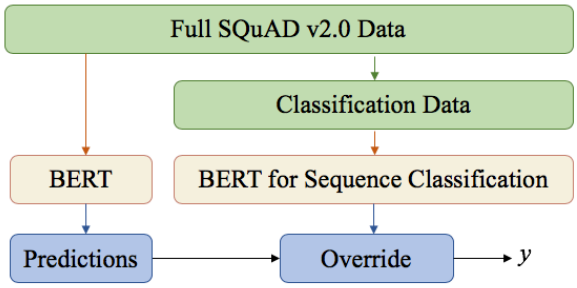

Figure 5: Architecture of the classification model. Here the predictions made by a single BERT model are passed through the predicted classifications and anything deemed "unanswerable" is automatically changed to the null answer.

3.6 Weighted Voting

This final approach revisited the predictions made by the previous models, in an attempt to weight the predictions based on model confidence and manual weighting of models such as the classification model. Ultimately, this was the only approach that exceeded the baseline, but it offers insights into how these models could be combined in more successful ways.

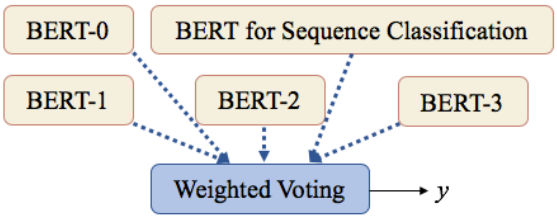

Figure 6: Architecture of the weighted voting model. This combines the predictions with the model confidence level and then includes the sequence classification as a vote which is manually weighted.

4 Results

Overall, each approach on its own underperformed the baseline model, revealing that ensemble methods are not a "magic bullet" that will automatically improve results. However, the weighted voting that included the classification predictions provided the best results with an F1 of 81.99, beating the baseline by 1.61. The top results from each approach are listed in table 1.

1) Voting with Discrete Data Presentation: This approach did achieve the goal of producing a broader range of predictions for consideration on the dev set. However, as the number of splits grew, the number of questions for which no model found the correct answer increased dramatically as well. Overall, the 4-way split performed the best of this subset, with an F1 of 75.3.

2) DNN for question-model matching: Because the neural net lacked any attention mechanism or understanding of language, all attempts to train the model were unsuccessful. The results are not included in table 1, as they were no better than random in all cases.

3) Multiple Choice Models: These models were conceptually the most promising but in practice these also significantly reduced overall model performance. For the 5 way model, during training it was only able to learn to correctly predict the answer around 60% of the time. While this was much better than chance, it still was far below the baseline. When applied on the dev set it resulted in a

Ensemble Type	Total		HasAns		NoAns
	EM	F1	EM	F1	EM/F1
Baseline: BERT Large	77.09	80.39	77.41	83.99	76.77
Simple Voting: 4 Way Discrete Data	71.91	75.30	75.69	82.48	68.14
Multiple Choice: A/B on unsplit data	63.77	67.42	75.67	82.99	51.89
Classification: BERT Base	75.36	78.37	70.58	76.61	**80.13**
Intelligent Weighting: BERT Large + BERT Large unsplit + Classification	**79.04**	**81.99**	**78.19**	**84.10**	79.88

Table 1: Summary of the top results achieved by each unique ensemble method. *Total:* All questions in the dev set. *HasAns:* These are the questions that were answerable. *NoAns:* Questions that were not answerable. *EM:* Exact match. *F1:* F1 score. The top results for each category are in bold.

F1 of 62.9. The two-option split did slightly better with an F1 67.42, with its performance on questions with no answers causing nearly all of the decline in accuracy.

4) Sequence Classification Supplement: During training, the model was able to accurately predict the questions without answers 83% of the time, but when applied to the dev set through the methods discussed above, it too hurt the overall model performance, however, it did achieve the highest performance of all models on questions without answers. The final F1 for this model was 78.37.

5) Weighted Voting: In applying confidence weighting to a subset of models trained throughout this process and the classification model predictions, this model was able to achieve an 81.99 F1 score, beating the baseline by 1.61.

5 Conclusion

Self-ensemble approaches are promising because, if successful, they could be applied to cutting edge models to achieve results that surpass fine-tuning. However, in practice, most self-ensemble approaches tend to take more time, more memory, and deliver lower accuracy. The exception seems to be if an additional task can be identified that is easier for the model to perform, such as identifying unanswerable questions in SQuAD v2.0.

Generalizing this beyond just BERT, a promising approach could be to train multiple cutting-edge models that leverage slightly different approaches such as ALBERT + XLNet + ELECTRA. Each of these could be trained for question answering, while a separate ensemble could be done for classification on its own, and then combined.

Further, the models could each be calibrated based on their overall accuracy, as opposed to based only on model confidence. A diagram of such an architecture is included in the Appendix as figure 7. In the future, I plan on applying this approach on the SQuAD v2.0 dataset, as its unique question composition makes it an ideal candidate to explore these approaches.

References

Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova. 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* arXiv:1810.04805

Jesse Dodge and Gabriel Ilharco and Roy Schwartz and Ali Farhadi and Hannaneh Hajishirzi and Noah Smith. 2020. *Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping.* arXiv: 2002.06305

Yige Xu and Xipeng Qiu and Ligao Zhou and Xuanjing Huang. 2020. *Improving BERT Fine-Tuning via Self-Ensemble and Self-Distillation.* arXiv: 2002.10345

Chenglei Si and Shuohang Wang and Min-Yen Kan and Jing Jiang. 2019. *What does BERT Learn from Multiple-Choice Reading Comprehension Datasets?* arXiv: 1910.12391

Yichan Liang and Jianheng Li, and Jian Yin. 2019. *A New Multi-choice Reading Comprehension Dataset for Curriculum Learning.* Proceedings of Machine Learning Research 101:742-757

Zhu, H and Wei, F and Qin, B and Liu T. 2018. *Hierarchic attention flow for multiple-choice reading comprehension.* In Proceedings of AAAI-18

Sun, C and Qiu, X and Huang, X. 2019. *How to Fine-Tune BERT for Text Classification?* In Chinese Computational Linguistics. CCL 2019

Shivaji Alaparthi and Manit Mishra. 2020. *Bidirectional Encoder Representations from Transformers (BERT): A sentiment analysis odyssey.* arXiv: 2007.01127

Appendix

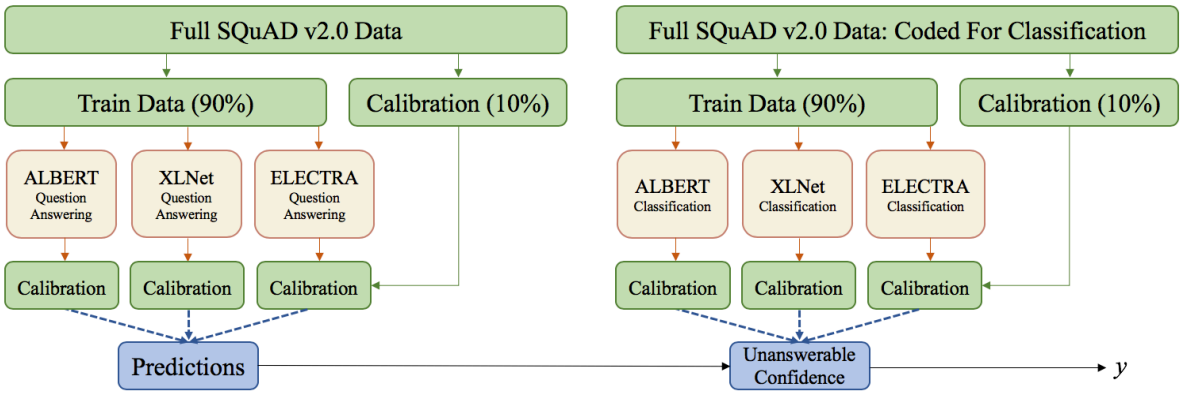

Figure 7: Proposed architecture to apply findings to achieve optimal results on the SQuAD v2.0 dataset. An ALBERT, XLNet, and ELECTRA model would each be trained for Question Answering, and Classification (answerable/unanswerable). The predictions would be calibrated by overall model accuracy, then combined with prediction confidence to determine the best prediction. On the classification side, the three models would again be calibrated and then combined with prediction confidence to determine an "unanswerable confidence" for each. The predictions would then be passed through, and if the unanswerable confidence outweighs the prediction confidence, that particular example would be turned to the null answer.