Ensembling BERT with its variations for Question Answering

Jakub Wasylkowski, Unnati Parekh, Nimesh Tripathi, Akash Govindarajula

Abstract

BERT which stands for Bidirectional Encoder Representations from Transformers, is used to pre-train bidirectional representations, from some given text, where context is taken from both before and after the word. BERT Is widely adaptable to a lot of tasks, once the additional output layer is fine tuned. BERT proved to become a popular attention modeling, as it combined both right-to-left and left-to-right training. Several BERT offspring, like the pre-trained models AlBERT, RoBERTa etc. also gained popularity due to their versatility which catered to different uses. BERT models had achieved much superior performance, in comparison to the other models, and they function way better when ensembled with other variations, for the task of Question-Answering. We have worked on several ensembled models using the "huggingface" library which were fine-tuned for the SQuaD 2.0 database.

1 Introduction

Three models have been used here, which were ensembled with BERT to fulfil our task of Question-Answering. The three models that we have worked with over here are – 1) BERT mini (L = 4. H = 256) 2) BERT small (L = 4. H = 512) and BERT medium. (L = 8, H = 512), where L represents the transformer layers and H represents the hidden embedding sizes. These models come with their own unique tokenizer mappings. Both, the questions and the context are encoded using their respective tokenizers. They are then loaded into each model for predicting the span i.e. the start and the end tokens. We have worked with the transformers-library, which was earlier known as pytorch-transformers, which work with 32+ trained models in 100+ languages. The output scores have been explored in two ways. The first one is by taking the output scores as is, and the other one is done by applying the SoftMax function to the output tensor. We try to get the best result and compare the three models and see which ensemble method offers the maximum performance boost. We have used several benchmark methods and evaluation metrics to analyze our results.

2 Motivation

Deep learning neural networks are nonlinear methods. They offer increased flexibility and can scale in proportion to the amount of training data available. A downside of this flexibility is that they learn via a stochastic training algorithm which means that they are sensitive to the specifics of the training data and may find a different set of weights each time they are trained, which in turn produce different predictions. Generally, this is referred to as neural networks having a high variance and it can be frustrating when trying to develop a final model to use for making predictions. A successful approach to reducing the variance of neural network models is to train multiple models instead of a single model and to combine the predictions from these models. This is called ensemble learning and not only reduces the variance of predictions but also can result in predictions that are better than any single model.

3 Related Work

The Question-Answering problem has a variety of applications like information retrieval, chat-bots, dialogue frameworks etc. However, transformer-based models had exceled in this task, with far-greater efficiency than the existing methods. Similar work has been done with models like XLM, and many others. Sentence-embedding (Kiros et al., 2015; Logeswaran and Lee, 2018) was another popular method, which was based on the same idea as popular word-embeddings like word2-vec, doc2vec etc. The

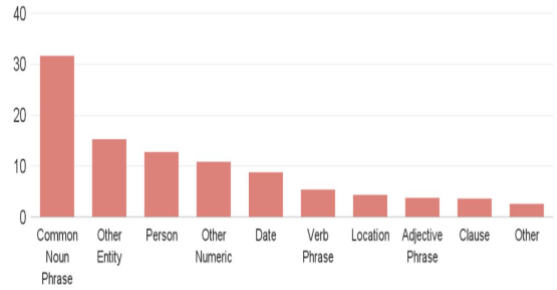

Figure 1: Distribution of answer-types

sentence embedding model which provides semantic sentence representations was called "Infersent.", which was developed by Facebook. Infersent is quite versatile with different tasks and it is trained with inference data. Here the entire sentence was provided as an input to the encoder function, which in turn, gives an output dimensional vector of 4096 dimensions, regardless of the sentence size, which was a drawback. Unsupervised models were also used, where the target variable was not defined. This approach also used the words or sentences as the output, which had the lowest distance from the given question. Several supervised learning models were also implemented, however, the size and the number of sentences required to get the contest was very large. Models with multinomial logistic regression had close to 150 million features, and the accuracy was lower than 80%. Several other models were also used for different tasks.The benchmarks were improved greatly by the ELMo (Peters et al., 2018). The task of sentiment analysis (Socher et al., 2013). Named-Entity-Recognition (Tjong Kim Sang and De Meulder, 2003). Melamud et al. (2016), gave the idea of obtaining contextual representations for the prediction of single word using both right-to-left and left-to-right contexts. This was done using LSTMs.

4 Dataset and Statistics

Here in the table below we see the benchmarks to which we shall compare our model. The Stanford Question Answering Dataset or SQuaD 2.0 is a dataset of reading comprehension. It consists of reading passages from articles in Wikipedia and they cover a variety of topics and domains like, celebrities to abstract scientific concepts. Passages from different articles are of different sizes and there are reading comprehension questions with each paragraph. The entire context of the passage is taken into consideration, before answering the question. The answer may be present in a span, which is a segment of text, or the question might be unanswerable as well. This is a very challenging problem, where the questions than can be asked to a passage of text can be very diverse.
To obtain the correct answers to the questions, the systems must select the answers from all the possible text in the passage, which in reality, turns out to be a lot of candidates to skim through. The dataset consists of publicly available train and dev sets. The test set is obviously hidden, from which we can get the results and change the leaderboard of performance accordingly.

5 Model Implementation

BERT's model architecture and its multi-layer Transformer Encoder was described in Vaswani et al. (2017), by using the tensor-2-tensor library. The GPT transformer which is also discussed in Vaswani et al., uses constrained self-attention.

Dataset	**Total Examples**
BERT-mini	130, 319
BERT-small	11873
BERT-medium	8862

However, BERT uses a bidirectional self-attention method.For the question-answering problem, as we discussed earlier, we have ensembled different models consisting of variations of BERT. We work with BERT mini, BERT small and BERT medium which have their respective tokenizers. The two scores are mentioned in the table below, namely score suffix and probability suffix. The start and the end token output scores/ probabilities are combined using a product. The way that the scores are combined are in the following ways- 1) Best Model Confidence (best_model_confidence) 2) Weighted-average of model outputs (weighted_average) and 3) Guided random search on weights. For the best model confidence, those predictions of the model are chosen which output the highest scores/ probabilities for the start and the end tokens.

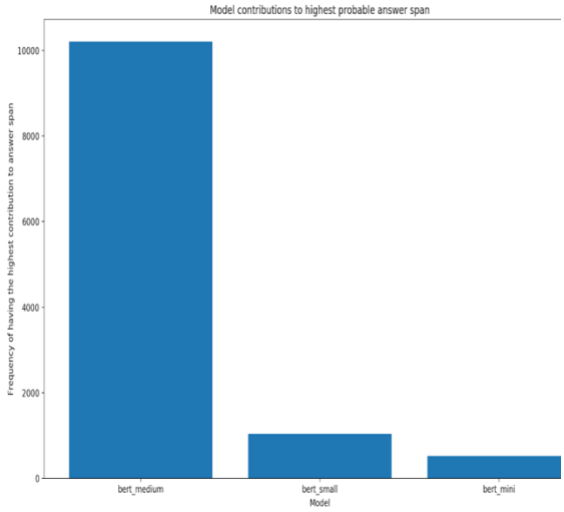

Figure 2: Model contributions to highest probable answer span

For the weighted average of model outputs, the models are ranked based on their individual F1 scores on dev set. Meanwhile, for the guided random search on weights, we add to the second method, by calling a routine to learn the best possible weights for the models that we use for the final prediction. The weights are initialized for the model randomly, per iteration, and this weight is used to predict answers for the dev set, simultaneously calculating the F1 scores. This is repeated for a limited number of iterations. The weights that obtained the highest F1-scores, over all iterations are then used to make the final predictions on the test set.

The primary evaluation metrics that we use are the Exact Match (EM) and the F1 score. EM is a binary measure that is used to verify if the ground truth matches the output exactly. The F1 score is a harmonic mean of precision and recall.

6 Method

Ensembling for Question Answering task can be challenging as we are dealing with two different probability distributions per model, for the start token and the end token over the complete vocabulary. We use a basic product Ensembling strategy for combining the probabilities for a model k as: $\kappa p_{ij} = p_i * p_j$, where p_i is the probability that token i is start token and p_j is the probability that token j is the end token Finally, we use three different weight selection strategies to combine these probabilities for the ensemble and select the span (i.e. tokens i and j) such that the sum for that span is the highest: $e_{nsemble} = \text{argmax}_{(i,j)} \Sigma w_k * \kappa p_{ij}$.

Algorithm 1: *Make_predictions*

Input:
1. Set of k models: M
2. Weights w
3. Prediction set $\{X_i\}_{i=1,2,..,n}$

Output:

Predictions on prediction set: $\{Y_i\}_{i=1,2,..,n}$

for *each prediction set example* Xl **do**

$^{\kappa}p_{ij}$:= P (start_pos = i, end_pos = j) predicted by the k-th model in set M

$Yl = (start\_pos, end\_pos)'l = \text{argmax}_{(i,j)} \Sigma w_k * {}^{\kappa}p_{ij}$

End

Return prediction set $\{Y_i\}_{i=1,2,..,n}$

Algorithm 2: *Guided_random_search*

Input:
1. Set of k models: M
2. max_iter
3. dev set: (Xl, Yl = (start_pos, end_pos) ,) $_{i=1,2,..,n}$

Output:

best_weight

Initialize best_F1, best_weight, = 0.0, None

while num_iter < max_iter **do**

Initialize w as an array of non-increasing random floats

for each *dev set example (*Xl , Yl) **do**

$\{Y'_i\}l \in 1,2...n$:= *Make_predictions()* for weight w

end

F1 = compute_F1($\{Yl\}l \in 1,2...n$, $\{Y'_i\}l \in 1,2...n$)

if F1 > best_F1 **then**

best_F1 = F1

best_weight = w

end

Num_iter += 1

end

Return best_weight

For weight selection, we are using the below strategies to combine the probability distributions for the start and end token of different models:

1. Equal weights assigned to all models: No model is given preference and all models are assigned equal weights. This becomes a simple average of the output predictions.
2. Higher weights assigned to models with higher F1 scores: The model predictions are weighted by normalizing ranks of models based on their F1 score. Higher weights are given to models with a higher F1 score on dev set.
3. Guided random search on weights (Algorithm 2): This is a slight variation of the above method. Instead of assigning static weights to the models,

we use randomly generated weights for the ensemble and compute the F1 score for each set of random weights. After a fixed number of iterations, we choose the weights which resulted in the highest F1 score.

7 Results

Given in Table 2 are the benchmarks that we compare our models with. And on the table3, we have our ensemble results.
Starting at two model ensembles, we see that (as expected) the performance of the larger models i.e. BERT small / medium is better than the other BERT ensembles. We use this high performing double ensemble and combine it with Distilbert and BERT-mini models. But Ensembling them with other smaller models does not improve performance much. The models do almost as good as the small / medium double ensemble but do not outperform this model. Thus, it is clear that Ensembling powerful models increases performance than the single models in that ensemble but the model choice for Ensembling is crucial too. As we can see, adding a third model does not help the performance.

Table 2

A comparison of the different weight selection methods showed that there was no difference in

Model	F1 Score	EM Score
BERT-mini	55.82	52.03
BERT-small	60.65	56.49
BERT-medium	66.83	61.95
DistilBERT	49.49	49.45

results based upon how the weights were being chosen. To explore this more, we tried to analyse the joint probability distribution obtained from each model in the BERT (mini / medium / small) triple model ensemble. We see that most of the time, the most powerful model contributed highest to the final ensemble summation step performed during prediction. This can be a reason that slight changes in weights did not change the output much.

	Equal		Ranks	
	EM	F1	EM	F1
mini / small / medium	63.2	67.31	**63.2**	**67.31**
small / medium / distilbert base	**63.15**	**67.62**	**63.15**	**67.62**
mini / medium	**62.83**	**67.32**	52.03	55.82
mini / small	57.46	61.26	52.03	55.82
small / medium	63.23	67.68	56.49	60.65

Table 3

8 Conclusion

We use ensemble approach for combining different variations of the powerful BERT model for the Question Answering task. We see that Ensembling is a powerful method to improve performance of the models. A model can help improve performance when the other models predict incorrectly. But it is also clear from the results that model selection for Ensembling is as important as the weight selection.

9 References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. *Bert: Pre-training of deep bidirectional transformers for language understanding.*
[2] Huggingface. Huggingface/pytorch-pretrained-bert. Github
[3] Christopher D Manning. Cs224n: Assignment 4.
[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. *Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473* .
[5] https://towardsdatascience.com/building-a-question-answering-system-part-1-9388aadff507
6] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. *Character-level language modeling with deeper self-attention. arXiv preprint arXiv:1808.04444.*
[7] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. *A large annotated corpus for learning natural language inference. In EMNLP. Association for Computational Linguistics.*
[8] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. *Class-based n-gram models of natural language. Computational linguistics, 18(4):467–479.*