NLP project report

Models ensemble for QA with BERT

Filippo Orazi - 0000928971
Andrea Rossolini - 0000954735

November 2021

Abstract

The SQuAD is a reading comprehension dataset. The input is a paragraph and a question about that paragraph. The goal is to answer the question opportunely. Open space question answering may be a field of normal language processing in which incredible strides are as of now being made. Open-domain question answering models have greatly improved in accuracy upon adopting BERT, and current state-of-the-art models fine-tune and extend this language representation model in their implementations. Our work presents different models built on the BERT encoder, intending to carry out an interesting comparison and find which model version performs better. The evaluation criteria will be *F1 score*, *Exact Match*.

Contents

1 Introduction

Question and Answering (QA) Systems is a widely researched subject in NLP and is also used extensively in the real world. We can think about systems, such as Google Assistant or Siri, as well as advanced search engines and chatbots. Many early systems frame it as an information retrieval problem, where they look for word similarities and return the most relevant passages (much like a search engine) while developing sophisticated similarity metrics such as Term Frequency Inverse Document Frequency (TF-IDF) as in Ramos et al. [1]. With the increasing ability to store and process a massive amount of data, research has shifted into building massive data storage and using NLP techniques to extract keywords and indices. An attempt at creating a simple question-answering system gave birth to Knowledge Bases (KBs) such as Freebase [2], which store information key-value tuples in a categorical and structured way. Despite the evolution of these kinds of QA KBs, the research community found its flaws. Undoubtedly, this kind of method work with fair structured information, whereas in a real-world environment, information isn't organized most of the time, and answering the question might not be as simple as discovering key-value sets in a dataset, particularly when the address requires a rational investigation.

While not exactly aiming for this level of complexity, we use the *Stanford Question Answering Dataset* (SQuAD) as a way to measure how well a machine learning model can answer general questions by extracting information based on conceptual text paragraphs. The SQuAD is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text from the corresponding reading passage [3].

This project aims to study the techniques used by the previous researcher and implement our solutions upon an established baseline model. Specifically, we developed a simple neural network that incorporates a BERT encoder and then extended the research and implementation to other models. We creatively incorporate convolutional neural networks, as introduced in [4] with Qanet. Consequently, we studied different architectures that will compose an ensemble model by which we will carry out some comparisons.

2 Related Work

The leaderboard on the SQuAD website[1] shows numerous deep learning models made for this dataset, and numerous analysts have come up with diverse structures. Moreover, the website presents and explains a more complex and new

[1]https://rajpurkar.github.io/SQuAD-explorer/

SQuAD dataset, called SQuAD 2.0. Nonetheless, this paper will focus just on the first SQuAD version. After a paper review, we wanted to focus on models and architectures based on Google's BERT. At its core, BERT could be a language model that is architected utilizing bidirectional Transformers. It can be fine-tuned to a variety of NLP tasks by including and training additional layers. Since the time of its release, its capabilities expanded far beyond question answering. The model described all through this paper takes advantage of these incredible results and fine-tunes BERT in its implementation.

3 Approaches

Our project aims to implement three main network models, that incorporate the BERT encoder, and make a comparison between these models. Then implement four ensemble models that build in different architectures. An ensemble model is a type of architecture that combines the output of different models and choose which one is the most valuable. Considering that our ensemble model (explained in the following subsections) is implemented in a way that allows us to easily customize it and its characteristics, like the hyperparameters as well as the number and type of models used. Hence, we carried out numerous experiments and approaches. Eventually, we decided that only some of these experiments are worth mentioning in this report. The first one, which we consider the baseline model, combines the encoder block with simple fully connected layers. After that, we implemented an architecture that uses convolutional layers. Usually, convolutional layers are applied in Computer Vision models, but some studies show that they might achieve good results in NLP (as explained in section 2). Therefore, we considered this kind of model worth of study. Finally, we realised a third interesting network architecture. The latter is an architecture that composes the two outputs computed by the BERT encoder (start and end token) to generate a new result. This approach aims the overall result to better capture different language features and succeed on different types of questions and evaluate if Google's BERT can be improved using different architectures.
After a comparison between the models listed above, we implemented an ensemble architecture that combines the result of different models and chose the best one. Next, we carried out numerous experiments with different models encapsulated into the ensemble and evaluated the improvements or setbacks. The experiments worth being reported in this document are listed in sections 3.7 and 4. As mentioned in section 1, each of these networks utilizes Google's BERT as the language understanding model, and details of this language model's architecture can be found in Devlin et al [5].

3.1 Baseline

The first architecture is a vanilla fine-tuning of the base BERT model. It makes use of *two single fully connected layers* for each of the two output layers for predicting the start and end tokens.

Regarding the network training, the context and question are concatenated and fed into the pre-trained BERT language model to obtain a final hidden state. Consequently, the output is split and directed into two different Dense layers, one focuses on the end token prediction, and the other the start toke prediction. Finally, the result is flattened and passed through two activation layers. A softmax function computes the most probable start and end tokens. This architecture is shown in figure 1.

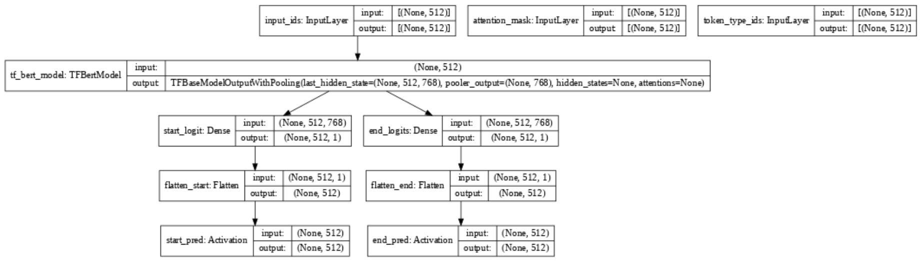

Figure 1: Baseline - BERT Vanilla architecture

3.2 Convolution Layers

The "convolutional architecture", shown in figure 2, uses *two convolutional layers* to receive the two outputs encoded by the BERT module. Consequently, these outputs go respectively into two fully connected layers, and the results get flattened. In conclusion, the final activation layer uses a softmax to compute the start and end tokens.

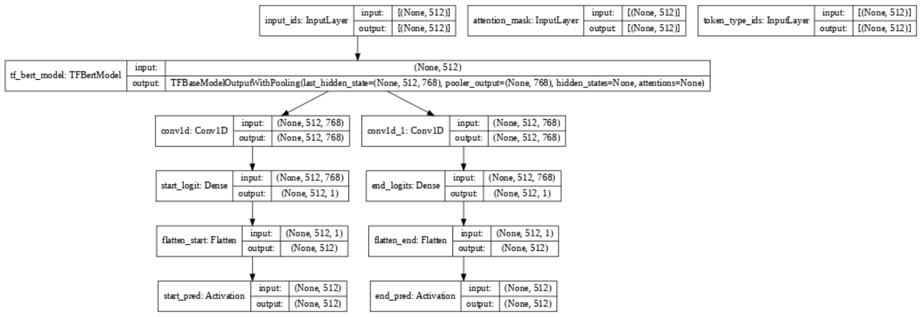

Figure 2: BERT with Convolutional Layer architecture

3.3 Multiply Layer

As for the vanilla, the context and the question are concatenated and fed into BERT to obtain the final hidden state. The output of the latter is then split between two layers.

Along the two "branches" of the neural networks, two dropout layers are implemented to slow down possible overfitting during the training. A *multiplication layer* is then applied between the start token predictions and the intermediate end token predictions to obtain the end token predictions. Finally, a softmax is computed over the final layer to find the most probable start and end tokens.

This last architecture described has been taken as a reference point for the development of similar architectures using layers other than multiply. These architectures compose the ensemble models (described in section 3.4) used for the study. Within the others, we deiced to describe the *multiplication layer* model specifically, since it is the one that alone achieves the best performances.

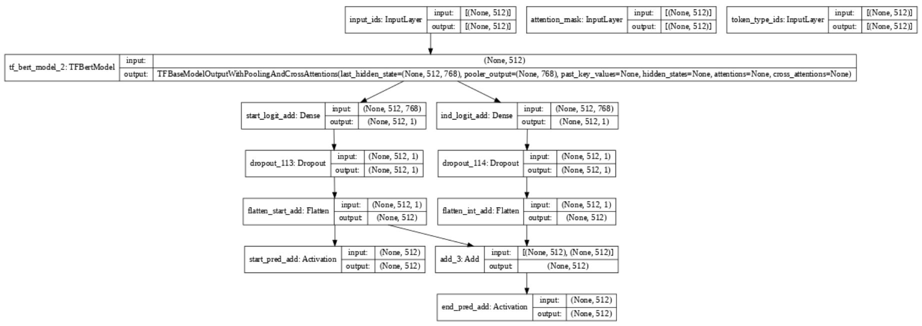

Figure 3: BERT Attention Network architecture

3.4 Ensemble

An Ensemble model consists of the union of multiple diverse models that are created changing some initial setting as the architectures or the training data sets. The ensemble obtains a prediction from each of the different trained models, and then it aggregates the obtained solutions with specific given techniques. The use of an ensemble model aims to reduce the generalization error of the prediction by using base models that are diverse and independent. In the ensemble model, the prediction error decreases with respect to the single models. This is because of the collective knowledge. Even though the ensemble model contains different architecture inside, it behaves and performs as a normal model. The general downside of an ensemble of models is the augmented prediction and training time as each part of the ensemble has to process the inputs.

We considered different versions of the ensemble based on which models are inside:

- Ensemble VC: in this ensemble contains the two models that performed best as stand-alone, namely Bert Vanilla and Bert CNN
- Ensemble VCM: Here we considered the three best models (Bert Vanilla, Bert CNN and Bert Multiply
- Ensemble 6Mod: it contains six different models that use different merge layers in an architecture analogue to the Bert Multiply. We use average, min, max, multiply, sum and subtract layers
- Ensemble 8Mod: contains all considered models (Vanilla, CNN, Multiply and the five variations).

The predetermined aggregation technique is an average of the results of each model.

3.5 Data

The data used is from the SQuAD V1 dataset as described in the introduction. The dataset is divided into training, validation and test sets:

- 64812 elements in the training set
- 21719 elements in the validation set
- 1027 elements in the test set

3.5.1 Exploratory analysis

To better understand the dataset, we conducted an exploratory analysis and understood that the set has a tree-like structure.
The structure is represented in the following scheme where the numbers at the start of each row and the indentation represent the nesting level of each element:

```
0. dataset
   1. data:
      2. title
      2. paragraphs
         3. context
         3. qas:
            4. answers:
               5. answer_start
               5. text
            4. question
            4. id
   1. version
```

During the analysis, we found some wrong or invalid *qas* that have been classified as dataset errors and removed from the training process. Examples of these are questions that have a length of 5 characters or less and a particular

question that contains the string:
"I couldn't could up with another question. But I need to fill this space because I can't submit the hit.".
We also verified the presence of repeated question-answer pairs that have been removed.

3.5.2 Data preprocessing

The data preprocessing is performed in conformity with the classical BERT preprocessing. Each question-answer pair is cleaned and then tokenized using the BERT tokenizer that corresponds to the BERT model used during training. The data is then processed into the three different inputs needed for BERT.
The only problem found during this process is the fact that BERT base uncased can only take up to 512 tokens in input, so we had to modify the context when this limit was exceeded.
During training, the cut of the context was done maximising the space around the answer as it was the best approach for the model to learn. During the testing time, the context is simply clipped at the maximum space available.

3.6 Evaluation methods

Model and baselines were evaluated using exact match (EM) and F1 scores, standard metrics for the question answering problem.
The Exact match is the ratio between the prediction that fully corresponds with the correct answer and the number of predictions. The F1 score is computed with the formula:

$$F1 = 2\frac{precision * recall}{precision + recall} \tag{1}$$

3.7 Results

Here we present the results of the fine-tuned models. As explained in section 3, we consider three basic models (Bert Vanilla, Bert CNN and Bert Multiply) and four ensemble models:

- Ensemble VC: contains Bert Vanilla and Bert CNN.
- Ensemble VCM: contains Bert Vanilla, Bert CNN and BERT Multiply.
- Ensemble 6Mod: contains five different variations of Bert Multiply and Bert Multiply itself.
- Ensemble 8Mod: contains all considered models (Vanilla, CNN, Multiply and the five variations).

Model	EM	F1
Vanilla	71.96	83.01
CNN	72.73	83.33
Multiply	73.03	82.96
Ensemble VC	72.93	84.32
Ensemble VCM	**74.39**	**84.82**
Ensemble 6Mod	72.54	83.99
Ensemble 8Mod	74.00	84.76

Table 1: This table shows the results obtained from each of the considered models ordered by complexity. The best results are highlighted

4 Analysis

We randomly sampled some predictions and compared them with the true answer keeping context and question as reference. Often the non-exact matches are still valid responses from a human point of view, as we can see in the following example: The question is:

When did Beyoncé receive the Legend Award?

Model	**Answer**
Ground Truth	the 2008 World Music Awards
Vanilla	2008 World Music Awards
CNN	2008
Multiply	2008 World Music Awards
Ensemble VC	2008
Ensemble VCM	2008 World Music Awards
Ensemble 6Mod	2008 World Music Awards
Ensemble 8Mod	2008 World Music Awards

We can see that the ground truth (GT) is equivalent to many of the answers. Despite this, the EM metric does not consider these as correct. The following example shows that, in some rare cases, the models' answers are more precise than the GTs. The question in example is:

In what year did Beyonce embark on her Dangerously in Love tour of Europe?

Model	**Answer**
Ground Truth	November 2003
Vanilla	2003
CNN	2003
Multiply	2003
Ensemble VC	2003
Ensemble VCM	2003
Ensemble 6Mod	2003
Ensemble 8Mod	2003

Here, the question asks "in which year..." and while all the models return the precise response, the ground truth provides a more extensive but unnecessary answer.

5 Conclusions

The basic BERT architecture produced good results for this particular problem, but after our experiment, we can conclude that small modifications to the baseline model can lead to better results, as shown by the CNN and the Multiply model.
Moreover, the results show that we can create an ensemble of models that achieves higher results. On the other hand, these improvements come at a price: significant training times, and long waits if we need to predict numerous responses (excluding the complications that occur when implementing a complex model). The experiments also suggest that the benefit of an ensemble tend to vanish if we increase excessively the complexity bringing even a negative influence to the overall metrics score. Indeed, in the table 1, the negative effect of complexity is clear when comparing *Ensemble VCM* with *Ensemble 8Mod*. Even if both of them are made up of the Vanilla, the CNN, and the Multiply models, the presence of many other architectures brings a negative effect to the metrics and a huge decrease in performances. Moreover, we can observe that the model *Ensemble 6Mod* achieve worse results than the *Ensemble VC* in both EM and F1, despite the latter being way more simple than the 6Mod, proving that quantity is not synonymous with quality.

6 Technical Info

The training was done on machine provided by CoLab.
The encoder used in the models' training and the prediction tests is the *bert-base-uncased*[2].

[2]https://huggingface.co/bert-base-uncased

References

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. 01 2008.

[2] Juan Enrique Ramos. Using tf-idf to determine word relevance in document queries. 2003.

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

[4] 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, 2018.

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