Natural Language Processing
NLP Course Project
Question Answering with SQuAD dataset

Master's Degree in Artificial Intelligence, University of Bologna

Samral Tahirli, Vida Zahedi and Marcello Fuschi
{ samral.tahirli, vida.zahedi, marcello.fuschi }@studio.unibo.it

February 2023

Abstract

This work explores how the BERT transformer-based deep learning model performs on the Stanford Question Answering Dataset (SQuAD), a collection of more than 100,000 question-answer pairs based on a set of Wikipedia articles. The goal of SQuAD is to enable Machine Reading Comprehension (MRC) by providing questions that are designed to challenge a machine's ability to understand a text passage, reason about the passage, and answer the questions accurately. The model we trained was evaluated on metrics such as *exact match* and *BLEU*, and the results were 64% and 0.74, respectively. We used a pre-trained model that we then fine-tuned on the SQuAD dataset, encountering overfitting issues, which we managed to solve by cleverly tuning some hyperparameters like the learning rate.

1 Introduction

The field of natural language processing (NLP) has seen significant advances in recent years, particularly in the area of question answering (QA). The Stanford Question Answering Dataset (SQuAD) is a widely used benchmark for evaluating QA models and their performance. In this paper, we propose the use of a pre-trained BERT deep learning model for solving the SQuAD dataset. BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art NLP model that has achieved remarkable results in a variety of NLP tasks, including QA. The key advantage of using a pre-trained BERT model is that it has already been trained on a massive corpus of text, allowing us to leverage its learned knowledge and fine-tune it for the specific task of SQuAD. This results in a much more efficient training process compared to training a model from scratch. In comparison to other popular NLP models such as LSTMs or RNNs, BERT's bidirectional approach allows it to effectively capture context in a way that traditional NLP models cannot. This makes it well-suited for solving the SQuAD task, where context is critical for accurately answering questions.

2 System description

2.1 Tokenization

The first component that we set up was a tokenizer module that's provided by Hugging Face for using it with their pre-trained BERT models. When working with transformer models like BERT, tokenization is needed because the input data that's fed into the neural networks is not ASCII strings but

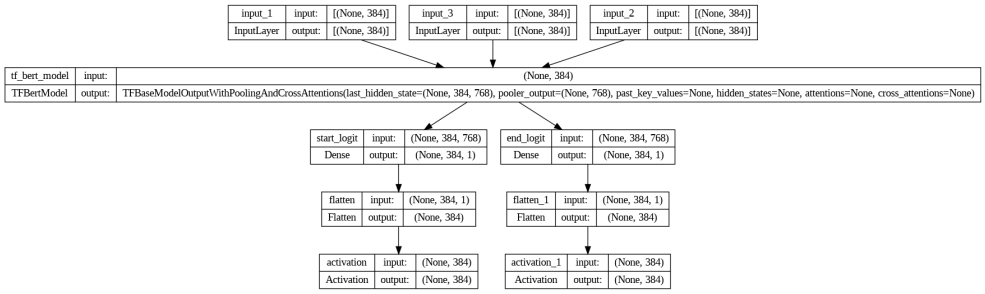
a vector of tokens representing parts of text like sub-words or full words. In order to transform a piece of text into a sequence of tokens, a model-specific operation is needed called tokenization. Wth the pre-trained tokenizer, we also handle the "out-of-vocabulary" words present in the dataset (by transforming them into tokens representing unknown words, or tokens representing a known piece of that word). The tokenizer also adds two necessary tokens to the sequence, "CLS" and a "SEP", representing the start and end of an input sequence.

2.2 Data preprocessing and representation

For the representation and manipulation of samples from the SQuAD dataset, we used a class called "SquadExample", that unpacks the data like x and y into a convenient data structure, and elaborates the necessary padding tokens and attention masks corresponding to each context+question sample in the dataset. The reason why we need padding tokens to be added to the input sequences is that BERT models such as the one we're using only accept fixed-length input sequences. The attention masks are useful for indicating to the model which tokens are original data and which are padding tokens, so that it can make more accurate predictions based on that information. The code for the SquadExample class is an adaptation of code found in this post.

2.3 Model architecture

As mentioned, the algorithm that we used for solving the question-answering task is a pre-trained deep learning model called BERT, provided by Hugging Face and that we fine-tuned on the SQuAD dataset. Besides the pre-trained BERT model, we added two branches of logit dense layers and softmax activation layers, whose output is respectively the probability given to each token of the context sequence to be the start and end token of the predicted answer. The total number of parameters in the network is 109,483,776 (all trainable).



3 Data description

The Stanford Question Answering Dataset (SQuAD) is a large, publicly available corpus of questions and answers for machine learning research. It was developed by Stanford University in 2016 and contains more than 100,000 questions and answers based on over 500 Wikipedia articles. The dataset is formatted as pairs of questions and answers, with each answer corresponding to a span of text within the related Wikipedia article.

SQuAD can be used to train NLP models for question answering, a task in which a model must answer a question based on a provided context. In the case of SQuAD, the context is the related Wikipedia article and the answer is a span of text within the article. For each paragraph, there are five questions and five answers. The dataset is given in JSON format.

4 Preprocessing

Preprocessing plays a major role in all NLP projects. It eliminates irrelevant elements from our data, making it ready for the next step. Before running the model, the dataset goes through the preprocessing stage to fix inconsistencies, missing values, and other noisy data. In this project, after loading the dataset, we divided it into train and validation sets. The train set was 90% of the entire dataset, with the remaining 10% used for validation.

The first step of preprocessing was data quality assessment such as cleaning all the unnecessary blanks from the paragraphs, answers, and questions and making all the examples lowercase. We then did a normalization step in which we removed all punctuation and articles. When creating the input, we decided to use padding and an attention mask. The attention mask is a binary tensor that shows the position of the padded indexes so that the

model does not pay attention to them. For the BertTokenizer, a value of 1 indicates details that should be attended to, while 0 indicates a padded value.

For creating each example, we created a function "create_squad_examples()". This function made a list consisting of a question, context, start_char_idx, answer_text, and all_answers for each example called squad_examples. This list was used by another function "create_inputs_and_targets" which we call X and Y respectively. X is a list of lists containing (input_ids, token_type_ids, attention mask) of each example while Y consists of start and end token_ids.

In the image below, we can see an example of the data samples.

```
# Squad Example data
for prop, value in vars(train_squad_examples[0]).items():
    print(prop, ":", value)
```

```
question : To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?
context : Architecturally, the school has a Catholic character. Atop the Main Building's gold
start_char_idx : 515
answer_text : Saint Bernadette Soubirous
all_answers : ['Saint Bernadette Soubirous']
skip : False
input_ids : [101, 6549, 2135, 1010, 1996, 2082, 2038, 1037, 3234, 2839, 1012, 10234, 1996, 23
token_type_ids : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
attention_mask : [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
start_token_idx : 114
end_token_idx : 121
context_token_to_char : [(0, 0), (0, 13), (13, 15), (15, 16), (17, 20), (21, 27), (28, 31), (
```

5 Experimental setup and results

While training our model, we encountered some errors that we had to address, like running out of RAM (which we solved by reducing the batch size from 64 to 8). To make the training faster, we used TPUs instead of regular GPUs.

During training, the metrics that we used for evaluating the performance of the model on the validation set are exact match (defined as the percentage of the answers found by the model corresponding word-by-word with the ground-truth answers) and BLEU (a sentence similarity score that was devised as an evaluation method for machine translation of natural language).

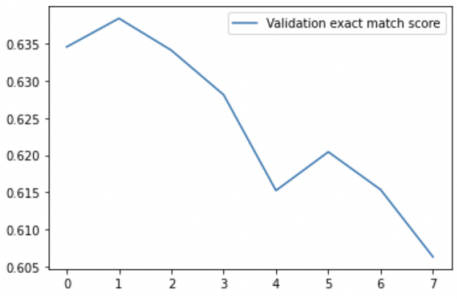
We used the softmax activation function and the Sparse Categorical

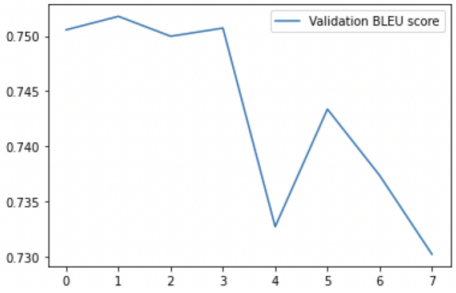
Cross Entropy as loss function since our targets were integers. Our optimization algorithm is Adam, using 8 epochs and a batch size of 16.

6 Discussion

To train the model, at the beginning we chose 8 as the number of epochs. As we can see below, as the training progressed, the training loss correctly decreased. However, the validation metrics results showed that generalization performance was getting worse after every epoch.

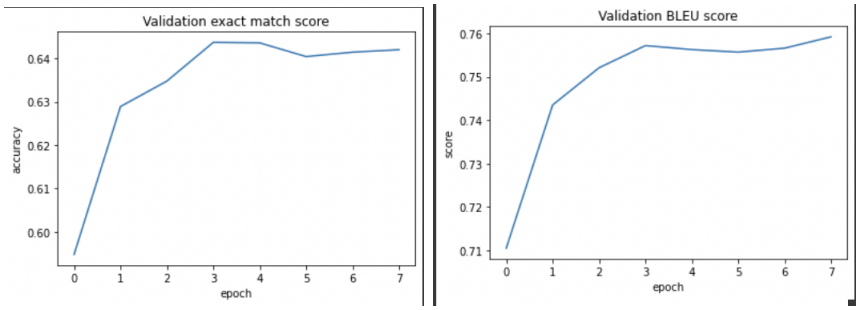
Epoch	Training loss
Epoch 1	2.5071
Epoch 2	1.5770
Epoch 3	1.1436
Epoch 4	0.8605
Epoch 5	0.6726
Epoch 6	0.5510
Epoch 7	0.4646
Epoch 8	0.4170





To prevent overfitting, we decreased the learning rate of one order of magnitude, down to 5e-6. As we can see from the graphs below, the validation metrics started to go down as the training progressed.

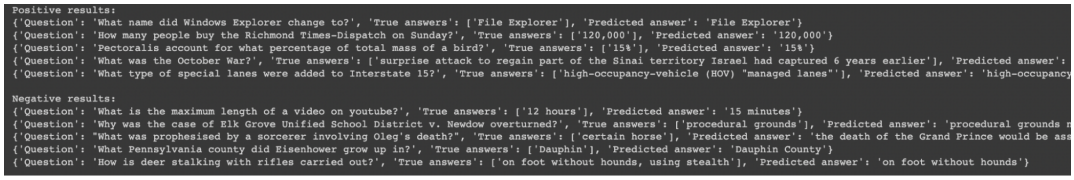
Epoch	Training loss
Epoch 1	3.2678
Epoch 2	2.1319
Epoch 3	1.7816
Epoch 4	1.5233
Epoch 5	1.3041
Epoch 6	1.1220
Epoch 7	0.9678
Epoch 8	0.8283



Our theory on why the model started improving when we decreased the learning rate is that, with a high learning rate, the model's parameters change significantly during the early stages of training (due to a high gradient of the loss function). As a result, the model was training too fast on the initial data samples it encountered, over-correcting on them. Then, it was unable to improve because it had already gotten into an area of the parameter-space with too low gradients.

6.1 Error analysis

After training the model, we did some error analysis of the validation results. Specifically, we verified that the results with the best scores were actually the ones that the model predicted perfectly, while most of the negative results were either not-too-far off, like 'Dauphin County' instead of 'Dauphin', or were plainly wrong but semantically close, like '15 minutes' instead of '12 hours'.



7 Conclusion

We successfully used the BERT transformer-based deep learning model for solving the Stanford Question Answering Dataset (SQuAD), a collection of more than 100,000 question-answer pairs based on a set of Wikipedia articles. The main problem we had was with overfitting, as can be inferred by the fact that the training loss was decreasing but the validation metrics were increasing. To fix this issue we decreased the learning rate by one order of magnitude, and this was enough to reverse that tendency, making the validation metrics go up instead of down without affecting the training loss.