Sequence-to-sequence Architecture Using BERT

Claudio Scheer and **José Fernando Possebon**
Pontifical Catholic University of Rio Grande do Sul - PUCRS
{claudio.scheer, jose.possebon}@edu.pucrs.br

Abstract

Abstract.

Introduction.

Related works

Related works.

Deep Learning

In this section, we will discuss the sequence-to-sequence model using recurrent neural networks and transformers. In addition, we also discuss how a BERT model works.

Sequence-to-sequence

The encoder-decoder architecture was initially proposed by (Cho et al. 2014). Although simple, the idea is powerful: use a recurrent neural network to encode the input data and a recurrent neural network to decode the encoded input into the desirable output. Two neural networks are trained.

(Graves 2013) - Generating sequences with LSTM

- Attention is all you need

BERT

BERT is a short for Bidirectional Encoder Representations from Transformers, proposed by (Devlin et al. 2018). Transformers network was proposed by (Vaswani et al. 2017) and use the attentions mechanism, proposed by (Bahdanau, Cho, and Bengio 2015), to learn representations between words that can express their contextual meaning.

The original BERT model was pre-trained in a corpus comprising Wikipedia and Book Corpus. BERT has two pre-trained models available: BERT large, with 345M parameters (24 layers) and BERT base, with 110M (12 layers).

The model was pre-trained for masked language modeling nas next sentence predictions tasks. However, with the replacement of the last layer and fine-tuning, the model can be used for other tasks, using the same parameters as the original BERT model.

discuss it here

In a nutshell, the difference is that self-attention is only applied to the input sequence, while cross-attention is applied to the input and output sentences.

Dataset

As the project focus is on automatic email reply, The Enron Email Dataset[1] was used to train the model. The dataset contains only the emails raw data. Therefore, a parser[2] was built to extract the email and the replies from the raw data of the email.

To identify whether an email has a reply or not, we look for emails that contain the string `-----Original Message-----`. After filtering only emails with non-empty replies, those emails were parsed into an input sequence (the original email) and a target sequence (the reply email). The entire extraction was done automatically, that is, we did not extract or adjust any email manually.

Two libraries were used to parse the dataset: `talon`[3], provided by Mailgun, and `email`, provided by Python. The `email` package returns the email body with the entire thread. To extract only the last reply from an email thread, the `talon` package was used.

The original dataset contains 517 401 raw emails. After parsing the raw dataset, the new dataset consisted of 110 205 input and target pairs. As the resources available to fine-tune the model were limited, only emails with less than 256 characters were used. The final dataset consisted of 40 062 emails. All of these input and target pairs were used to train the BERT model.

Implementation

A pre-trained BERT model, provided by Hugging Face[4], was used. Hugging Face also provides a PyTorch library for using the pre-trained models. Therefore, this library was used to implement the sequence-to-sequence model, with PyTorch 1.5.1.

The `BertModel` class provided by Hugging Face can behave as an encoder or decoder. The difference is that, for the encoder, only a layer of self-attention is used and, for

[1]https://www.kaggle.com/wcukierski/enron-email-dataset
[2]https://www.kaggle.com/claudioscheer/extract-reply-emails
[3]https://github.com/mailgun/talon
[4]https://huggingface.co/

decoder, a layer of cross-attention is added between the layers of self-attention. The difference between these attention mechanisms is discussed in Section BERT.

In this paper, the BERT base architecture was used. This model uses fewer resources, which allowed us to increase the batch size. To fine-tune the model, the following hyperparameters was used:

- Learning rate: $1e{-}4$;
- Warm-up steps: 5000;
- Epochs: 10.5;
- Adam epsilon: $1e{-}4$, same as in the original BERT paper (Devlin et al. 2018);
- Batch size: 10;
- Beam search hypothesis: 3;

In the fine-tune process, an EC2 spot instance on AWS was used. A checkpoint of the model was saved every $20\,000$ steps, mitigating the loss of processment if the instance was suddenly stopped. The instance was equipped with a 16 GB NVIDIA T4 GPU (CUDA 10.2).

The batch size was limited by the amount of GPU memory available. We also tested to accumulate and update the gradients after 4 steps, but we did not see any improvements.

Different values were tested for the learning rate and epochs, with and without warm-up. Figure 1 shows the final results of the learning rate update during the fine-tuning process.

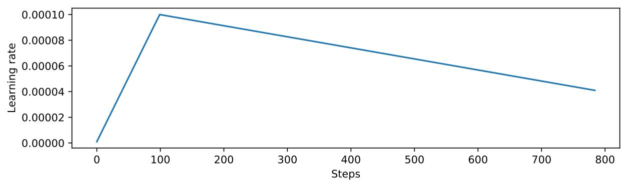

Figure 1: Learning rate schedule

Figure 2 shows the loss function value over the fine-tuning process. We do not know why, but loss function shows a greater decrease at the end of each epoch.

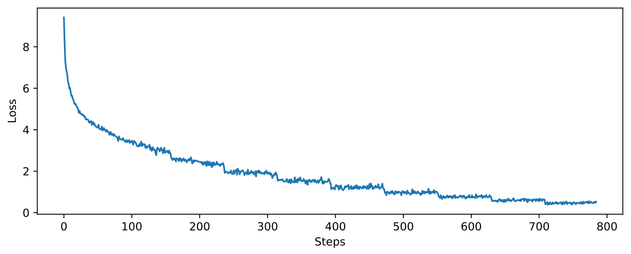

Figure 2: Loss function

Some hyperparameters configuration shows bad results in the loss function. For example:

- learning rates greater than $1e{-}4$ did not make the model converge;
- the use of a warm-up period allowed the model to converge in less epochs;
- a high number of epochs almost causes an overfitting of the model;

The last changed hyperparameter was the number of hypothesis explored in each branch of the beam search. A number greater than 3 resulted in more words being out of context in the generated reply email.

Results

The evaluation of the model was
https://huggingface.co/blog/how-to-generate

References

[Bahdanau, Cho, and Bengio 2015] Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In Bengio, Y., and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Cho et al. 2014] Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR* abs/1406.1078.

[Devlin et al. 2018] Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

[Graves 2013] Graves, A. 2013. Generating sequences with recurrent neural networks. *CoRR* abs/1308.0850.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *CoRR* abs/1706.03762.