Project Report
Transformer Model on Translation

Group 52: Yuqi Sun, Meinan Gou, Wenxuan Wu

Abstract

Translation between different languages has many applications in education, medicine and travelling. The earliest model is Statistical Machine Translation (SMT), which requires huge memory space and efforts from linguistic experts. Later, Neural Machine Translation (NMT) made great achievement but its training is inefficient. Transformer model was proposed in 2017 and can perform better than NMT and is easy to train. Our model was trained using corss-entropy loss and Adam Optimizer on paired Portuguese and English text sentences from *ted_hrlr_translate* data in Tensorflow datasets (tfds) *pt_to_en* with 51785 training pairs and 1193 validation pairs [2]. Instead of tracking loss in each epoch, we measured the performance of our model using BLEU score for machine translation. We compared the model's performance to a baseline model with different hyperparameter settings. Our model achieves a final BLEU score of 24.7. This score can be improved by tuning other hyperparameters or training on a larger subset of the data.

1 Introduction

Transformer model is the state-of-the-art machine translation model which was proposed by Vaswani et al. in *Attention Is All You Need*[3]. Transformer model is faster to train than previous neural machine translation (NMT) models and can perform well or even better. In this project, we implemented a transformer model with multi-head attention to translate Portuguese to English with Tensorflow libraries. The BLEU score of translation result of our base model is 24.71. This score is satisfactory with only several hours training and small dataset. Based on the model, we explored how hyperparameters in a transformer model influence its performance trained on small dataset. In this project, we focused on the effect of three hyper-parameters, the number of layers, hidden units and heads. We found that small hyperparameter values can achieve better performance than larger ones using small training dataset.

2 Related Work

After the publication of Transformer model, most work focused on how to improve this model. There is also some work about hyper-parameters. Popel et al.[6] did the investigation into the influence of batch size, learning rate, warm-up steps, maximum sentence length and checkpoint averaging on training and provided some tips. However, they mainly focused on hyper-parameters related to training models instead of those hyper-parameters of transformer model. Van Biljon et al.[7] trained transformer models with different number of layers on low-resource languages and found that deep networks preform worse with small training data. Our experimental results are consistent with this finding and we explored more hyper-parameters except the number of layers.

	en	pt
0	and when you improve searchability , you actually take away the one advantage of print , which is serendipity .	e quando melhoramos a procura , tiramos a única vantagem da impressão , que é a serendipidade .
1	but what if it were active ?	mas e se estes fatores fossem ativos ?
2	but they did n't test for curiosity .	mas eles não tinham a curiosidade de me testar .

Figure 1: Examples of training data

3 Data

We used a subset of *ted_hrlr_translate* data in Tensorflow datasets (tfds) *pt_to_en* (Portuguese to English) [2] for translation tasks. The sentence-level dataset contains TED speech in both Portuguese and English languages in TensorFlow text object format. The full dataset contains 51785 training, 1193 validation and 1803 test examples. An example of the data is shown in Figure 1.

As the first step of data processing, we generated vocabulary in both languages by tokenizing each sentence on a sub-word level. One advantage of using subwords is to separate an unknown word into tokens of characters to avoid the out-of-vocabulary word problem. The tokens were written to separate vocabulary files with one token per line. After building vocabulary, each source and target sentence was encoded and padded with starting and ending tokens. To conserve system memory and train effectively, we set a maximum length. We filtered all sentences that are longer than maximum length and padded all sentences which are shorter than maximum length with 0s. After applying encoding and filtering, the training and validation datasets became trainable tensors.

4 Methods

4.1 Methods compare

Transformer model is a popular and effective model, whose purpose was to improve machine translation performance. There are several machine translation models, such as RNN , seq2seq and GNMT, while each model has some drawbacks. The connection between encoding and decoding of the seq2seq model is the only implicit vector output by the coding unit, which always lacks useful information. GNMT is hard to train. The self-attention mechanism in Transformer produces a good approach to the above problems. So our method is based on muti-head-transformer model.

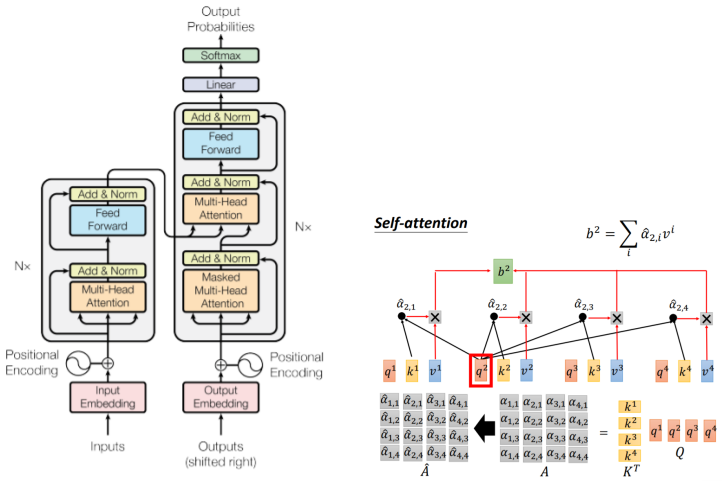
4.2 Transformer architecture

As the figure(a) shows, transformer model in origin paper contains two parts: Encoder and Decoder. An encoder consists of six same layers and each layer has two submodules: Multi-Head Self-Attention and Position-Wise Feed-Forward Network. A decoder also consists of 6 same layers but each layer has 3 submodules: Multi-Head Self-Attention, Multi-Head Context-Attention and Position-Wise Feed-Forward Network.

4.3 Self-Attention

q: query(to match other words), $q^i = W^q a^i$

k: key(to be matched words), $k^i = W^k a^i$
v: information to be extracted, $v^i = W^v a^i$



(a) Transformer-Architecture[3] (b) Self-Attention[1]

Figure 2: Transformer Architecture and Self-attention

As the Figure 2 (b) shows, each word has $< q, k, v >$ structure. The self-attention refers to the internal attention existing in the source sentence or target sentence. Each word queries other words in the same sentence and calculates the value. Here a^i is the previous self-attention layer input b^i is the next self-attention layer input, b^i could be computed parallel. A big improvement of self-attention is that it could capture the relation between the words' features and also the sentence grammar. As it could connect any two words in the sentence and could be computed parallel so it is independent on time sequence. And there is no position information in self-attention model. Positional Encoding is an innovation method on input encoding, as the figure(a)'s inputs and outputs' part shown, each input word vector x_i appends a one-hot vector p_i as position information.

4.4 Multi-Head-Self-Attention

Multi-head-self-attention divides each $< q, k, v >$ into multiple heads, allowing the model to pay attention to different aspects of sentence's information. Some research results shows that different heads focus on different token in the sentence. Its effect is similar to the convolution kernel in CNN because it could capture the different local information of a picture. However, there is difference that multi-head-attention does not share the same weights like convolution kernel, so it could divide sentences into different subspaces and obtain different features.

4.5 BLEU (Bilingual Evaluation Understudy)

We use BLEU instead of translation accuracy to evaluate the model. The general idea of BLEU is to compare the degree of overlap between the n-grams in the candidate translation and the reference translation (from unigram to 4-gram in practice). Here we use NLTK package in Python. The higher the degree of overlap, the higher the translation quality. The accuracy of unigram can be used to measure the accuracy of word translation, and the accuracy of n-gram can be used to measure the fluency of sentences.

5 Experiments

To train our model, we used teaching-forcing method so models only predicted one word during training. We computed the accuracy and cross entropy loss of predicted words during training, which can show the ability to correctly translate next word with correct input. However, the accuracy and loss is not directly related to the performance of translation without teaching-forcing. Thus, we use the BLEU score on validation dataset as stopping criteria instead of validation loss or accuracy in the following experiments. The method is setting and saving checkpoints and compute validation BLEU score at each epoch during training. After training, We used the model which achieved the highest BLEU score on validation dataset for testing by retrieving the corresponding checkpoint.

In this project, we focused on the hyper-parameters of model architecture instead of training. Thus, all models have the same training hyper-parameters, learning rate is computed by a custom learning rate scheduler[3], batch size is 64, max length of sentences is 40, dropout rate=0.1. The model code can be found at the following Github link: `https://github.com/KatieGou/DL-Project` .

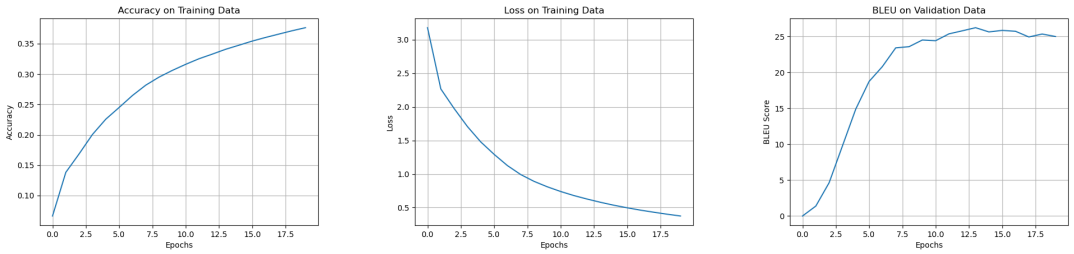
5.1 Base model

In order to find how different hyper-parameters affect the performance of the transformer model, we firstly made a coarse grid search to find a base model which has good performance and modify the hyper-parameters based on its setting. The hyper-parameters we modified during experiments are the number of layers in encoder and decoder (L_{layers}), the number of hidden units in each layer (N_{hidden}) and the number of heads in each layer (N_{head}).

The hyper-parameters of the base model we found were

$$L_{layers} = 4,\ N_{hidden} = 256,\ N_{head} = 8.$$

We plotted accuracy and loss on training dataset and the BLEU score on validation dataset as is shown in Figure 3. The final test BLEU score of base model is 24.4037. Table 1 shows some examples of the translation results with base model.



(a) Training Accuracy (b) Training Loss (c) Validation BLEU score

Figure 3: Plots of Base Model

According to the base model, we modified one hyper-parameter in one experiment to analyze its effect. Because the base model has good performance, the comparison can provide more meaningful information.

5.2 Layer Number Experiments

The number of hidden layers in base model is 4, so we experimented with 2, 3, 5 and 6 hidden layers to check its effect (base model in bold).

Reference Translation	**Predicted Translation**
then , predictions can be made and tested .	then , you can become tested and tested them up .
it had forced multiple labs that were offering brca testing to stop .	she forced to stop multiple labs that offered multiple tests that supports van to teach them .
where are economic networks ?	where are the economic networks ?
ants are a classic example ; workers work for queens and queens work for workers .	the parse are a classic example of classic ; the opec work for the quet and vice versa.
one of every hundred children born worldwide has some kind of heart disease .	one in every hundred children are born with a heart disease .

Table 1: Reference translation and Translation result of model on test dataset

L_{layer}	1	2	3	**4**	5
Test BLEU	21.2858	22.4674	23.7441	**24.4037**	4.9455

Table 2: Test BLEU score of models with different L_{layer}

As is shown in Table 2, the model achieves highest score with $L_{layer} = 4$, which means lager number of layers do not necessarily produce better performance than small number of layers. This indicates that a transformer model is different from traditional NNs, in which applying more layers would never worsen the results.

5.3 Multi-Head Attention Number Experiments

The number of Multi-Head Attention in base model is 8, so we experimented with 2, 4, 16 and 32 multi heads to check its effect (base model in bold).

N_{head}	2	4	**8**	16	32
Test BLEU	24.5919	24.2948	**24.4037**	21.9549	20.0654

Table 3: Test BLEU score of models with different N_{head}

As is shown in Table 3, too many heads would worsen the performance of the model. For this specific translation task, small number of heads outperforms large number of heads.

5.4 Hidden Units Number Experiments

The number of hidden units, N_{hidden}, in base model is 256, so we experimented with 64, 128, 512 and 1024 multi heads to check its effect (base model in bold).

N_{hidden}	64	128	**256**	512	1024
Test BLEU	21.8904	23.7861	**24.4037**	23.2561	14.7473

Table 4: Test BLEU score of models with different N_{hidden}

The model achieves the best performance with $N_{hidden} = 256$. Large and small number of hidden units do not produce better results. There is a score peak corresponding to the number of hidden units, which means it has an optimal value.

5.5 Hidden Units vs. Multi-head experiments

We wanted to know whether the number of hidden units of each head can influence the performance of the model, so we modified both N_{hidden} and N_{head} in this part. The Test BLEU score shown below is from best models trained for about 20 epochs.

N_{hidden}	32	64	128	**256**	512	1024
$N_{head} = 2$	19.9181	23.7607	24.7118	24.5919	23.6217	7.8668
$N_{head} = 4$	16.9513	22.6534	24.1287	24.2948	22.7206	11.3328
$N_{head} = 8$	20.0465	21.8904	23.7861	**24.4037**	23.2561	14.7473
$N_{head} = 16$	13.4255	20.4369	22.2887	21.9549	21.5714	15.1701
$N_{head} = 32$	8.1178	18.0121	19.6671	20.0654	18.7337	5.3977

Table 5: Test BLEU score of models with different N_{hidden} and N_{head}

As is shown in Table 5, the number of heads has little influence on performance, and too many heads may worsen performance. This may be caused by the small size of our dataset. Compared to the number of hidden units of each head, the number of total hidden units is more important. When N_{hidden} is too large, the validation score quickly reached the peak and began to fluctuate, while when N_{hidden} is too small, the validation score still had increasing trend after 20 epochs. Thus, suitable N_{hidden} can help to train efficiently. We think the learning rate should be more negatively related to N_{hidden} to solve this problem.

5.6 Summary

Our results indicate that for small training datasets, small hyperparameter settings would achieve better performance than larger ones. When doing a coarse grid search for the best hyperparameters in a transformer model, we suggest to start with small setting values. This is for good performance and saving time. First, small hyperparameter setting values can usually achieve better performance than larger ones, as is shown in Table 2 and Table 4. Also, training with small hyperparameters values takes much less time than training with larger ones.

In summary, for a transformer model, good hyperparamters settings rely heavily on the size of training dataset. For very large dataset (millions and more), larger hyperparameter values would behave well. However, for small datasets (thousands), smaller hyperparameter values would perform better.

6 Conclusions

Transformer model is not the same as a neural network. Based on different dataset, transformer model has a range of good architectures. Larger models do not perform better or even worse in our project. In addition, although we have gotten good BLEU score, the translation may contain some unrelated words which make sentences hard to understand. This means that the model cannot be used as an application without language models, even though it has higher score than other machine translation models.

There is some work left for future experiments. We are also interested in whether different languages share similar good hyper-parameters when the size of their dataset are similar. Moreover, the problem of unrelated words should be solved. For instance, a transformer model with improved architecture may produce more natural translation. Finally, to get more accurate BLEU score, a corpus with more references corresponding to one hypothesis would be better.

References

[1] Hongyi Li, NTU, Machine Learning course 2020 spring.

[2] Ted_hrlr_translate—TensorFlow Datasets. (n.d.). TensorFlow. URL: https://www.tensorflow.org/datasets/catalog/ted_hrlr_translate#ted_hrlr_translatept_to_en.

[3] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. arXiv preprint arXiv:1706.03762, 2017.

[4] Shuhao GU and Yang FENG, Improving Multi-Head Attention with Capsule Networks, 2019

[5] Tensorflow tutorial: https://www.tensorflow.org/tutorials/text/transformer

[6] Popel M, Bojar O. Training tips for the transformer model[J]. arXiv preprint arXiv:1804.00247, 2018.

[7] Van Biljon E, Pretorius A, Kreutzer J. On optimal transformer depth for low-resource language translation[J]. arXiv preprint arXiv:2004.04418, 2020.