Machine Translation with a Transformer

Naveen Rayapudi
40291526
rayapudinaveen777@gmail.com

December 11, 2024

Abstract.
The Transformer introduced the "Attention" mechanism, revolutionizing natural language processing by enabling efficient data processing. It uses self-attention mechanisms and positional encodings to capture long-range dependencies in text, replacing the need for models such as RNNs and LSTMs.Transformers are the foundation for many state-of-the-art models, such as BERT, GPT, and T5. In this report, we implemented a Transformer-based machine translation system using PyTorch. where we preprocessed datasets and used a RoBERTa's tokenizer to handle tokenization and padding, and fine-tuned a customized transformer model to translate from English to French. Additionally, we used the pre-trained T5 model from the Hugging Face transformers library for comparison. Metrics like BERTScore and METEOR were used to evaluate translation quality, and experimental results demonstrated the effectiveness of the pre-trained model compared to the customized model.

Contents

1 Introduction

In this report, we conduct experiments to analyze how different hyperparameters and training data impact the performance of a customized transformer model. Adjusting parameters such as attention heads, dropouts, and batch size, testing on varied datasets, and using metrics like BERTScore and METEOR to understand the model. Also, we compare the pre-trained T5 model with the performance of the Customized Transformer.

2 Experiments

2.1 Experiment 1:Customized Transformer model using PyTorch

2.1.1 Motivation

The motivation of the experiment is to understand the Transformer model architecture and the impact of different hyperparameters on the performance of customized Transformer model. And how the model's nature of processing and translation is affected by change in parameters.

2.1.2 Methodology

The methodology in the experiment starts with preprocessing the texts further by creating datasets for Training,Validating,and Testing. After the creation of the dataset we construct a customized transformer model.Then the data is preprocessed by cleaning and tokenizing the source and target text using a XLM-Roberta tokenizer.Furthermore the model is then trained on the training set and evaluated on the validation set, using metrics like BERTScore, and METEOR to assess translation quality. By adjusting the hyperparameters, the experiment aims to analyze the affect model's processing efficiency and translation accuracy.

2.1.3 Results and Analysis

In this section,We learn how these parameters affect the model by first looking at default Parameters and then looking at other parameters.

Effect of Default Parameters: By using default parameters we able preprocess the dataset size of **100000** with the Transformer model. From the

tables[1] During the process of training the Training loss shows a steady decline over the 10 epochs, starting at **5.536** and gradually dropping to **2.991** this indicates that the model is learning and Validation loss also decreases from **5.388** in epoch 1 to **3.829** in epoch 10 this is a good indication that how well the model generalizes to unseen data. After training the model we test the model on test dataset where we able to translate the English sentence: **"Hello how are you today"** to French **"comment êtesvous aujourdhui"** with metrics related to BertScore "**Precision:0.8581** ,**Recall:0.8435**, **F1 score:0.8506**,and **METEOR:0.2978**. The metrics indicate that the model performs well in terms of high precision,recall and F1 score. However, the relatively low METEOR score suggests that the model generates accurate translations, there is some part for improvement in translation.Overall,the model is effective but requires further to improve the quality of the generated translations.

Effect of tuning Embedding Size: From tables[2],[3],[4] and figure[2] increasing form **128 to 512** the embedding sizes demonstrate better learning , achieving lower losses and higher translation quality, but come with computational requirements. lowering the embedding size tend to underfit the data, producing less accurate translations.

Effect of tuning Heads: From the table[5] and figure[3] we can say that increasing number of heads form **1 to 8** results with decrease in training loss from **4.332 to 4.204** ,with slight increase in validation loss from **5.512 to 5.682** suggests that the model to capture more complex features in the data, leading to faster learning during training. where as added heads is not effectively utilized for generalization for validation set.

Effect of tuning Batch Size: From the tables[6],[7] and figure[4] we can say that as the batch size increases from **16 to 64**, both the training and validation losses increase. This indicates that smaller batch sizes lead to better model performance, as larger batches may cause the model to struggle with generalization.

Effect of tuning Dropout:
The results from thw table[8] and figure[5] suggest that a **dropout rate of 0.01** offers the best balance for model performance. As the training loss is lower than with lower dropout values, and the validation loss is also controlled, indicating better generalization. A **dropout of 0.1** is too high, leading to underfitting, while **dropout of 0.0001** is too low, causing over-

fitting and poor generalization. Overall, dropout values between **0.01 and 0.001** shows most effect on the model.

2.1.4 Limitations

- From the results the translation revealed that the model did not capture the word "Hello," indicating that the model might not be fully handling certain parts of the input.
- Although the model achieved good precision and recall, the relatively low METEOR scores highlighted limitations in generating fluent translations.
- Despite training on a large dataset 100,000, the model's performance indicated their is still need for improvement.
- Larger embedding sizes , dataset sizes and higher numbers of attention heads improved performance but significantly increased computational requirements.

2.2 Experiment 2: Pre-trained Transformer model

2.2.1 Motivation

The motivation in this approach is to use pre-trained T5 model, which has been fine-tuned on a large dataset, to perform high-quality translation tasks. By using the T5 model, we can perform NLP task translation from English to French. The goal is to evaluate the model's ability using metrics like BERTScore and METEOR which measure different aspects of translation quality, such as Semantic similarity and Semantic matching. This approach is used to handling large datasets efficiently in real world Nlp tasks.

2.2.2 Methodology

In the methodology, the pre-trained T5 model is loaded along with its tokenizer. The model uses the source text with a task-specific prefix: **"translate English to French:"** for encoding it ,intiating to perform translation to french and decoding the output back to text. Beam search is used to enhance the translation's quality by keeping track of multiple possible sequences at each step. The test function processes the test dataset in batches, generating translations for each source sentence and comparing the results with reference translations using the evaluation metrics.

2.2.3 Results and Analysis

From the table[1] results of T5 model with **Precision:0.8960, Recall:0.8987, and F1-scores:0.8972, METEOR score:0.5160** suggesting that it is generating translations that are both accurate and complete. But they were multiple warnings about empty candidate sentences durting the process of testing. which implies that the model sometimes generates no output for certain inputs that leads to show effect on metrics like BERTScore, which assigns raw scores of 0 for these cases. Moreover BERTScore metrics are high, the METEOR score is relatively lower. This suggests that while the generated translations are good, they lack in variation.

2.2.4 Limitations

- From the meteor results we can say there might still issues with the semantic matching.
- The testing process is slow, taking approximately 1 hour and 43 minutes for 269 samples.
- The model sometimes generate empty translations,which affects both BERTScore and METEOR metrics.

3 Conclusion and Future Work

3.1 Summary of your work and your findings

The report focuses on two approaches customized transformer model and pre-trained T5 model from the figure[1] we can say that the pre-trained transformer outperforms the customized Transformer in every aspect of metrics which says that the pretrained is preferred over the customized one where it struggels to handle over translation part as well as the customized model focused on hyperparameter tuning, revealing that larger embedding sizes and more attention heads improved translation quality but increased computational costs. Smaller batch sizes showed better generalization. Moreover **0.01** dropout rate showcases the best balance for the model performance.

3.2 Limitations

The customized transformer model in not accurate in translation. And the pre-trained T5 model sometimes generated empty outputs, affecting

evaluation metrics. Additionally, testing was slow, shows effect if we used for large-scale applications.

3.3 Future Work

It includes enhancing the customized transformer model by using beam search instead of greedy for sequence generation tasks and fixing the issues with the model to translate overall sentence, using other tokenizers for better performance like T5 tokenizer. Also invloves exploring more efficient way to handle large datasets to improve testing speed for the pre-trained model. Moreover developing better methods to reduce the occurrence of empty translations during the process of testing.

References

[1] GeeksforGeeks, *Transformer model from scratch using TensorFlow*, Available at: `https://www.geeksforgeeks.org/transformer-model-from-scratch-using-tensorflow/`.

[2] H. Ozbolat, *Text Summarization using BertScore*, Medium, Available at: `https://haticeozbolat17.medium.com/text-summarization-how-to-calculate-bertscore-771a51022964`.

[3] *Hugging Face*, Available at: `https://huggingface.co/docs/transformers/en/tasks/translation`.

A Appendix

Model	Precision	Recall	F1-Score	METEOR
Customized Transformer Model	0.8581	0.8435	0.8506	0.2978
T5 Pre-trained Model	0.8960	0.8987	0.8972	0.5160

Table 1: Performance Comparison of Customized Transfomer Model vs Pre-trained T5 Model (Over 10 epochs and 100000 Dataset Size)

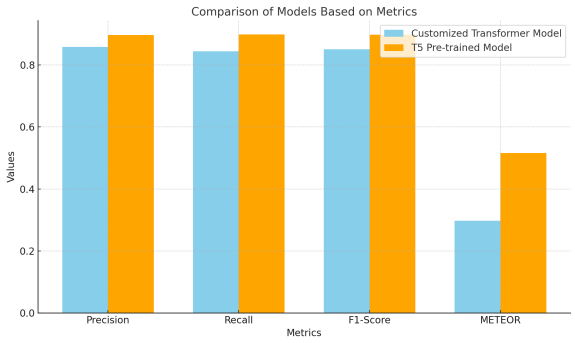

Figure 1: Customized Transfomer vs Pretrained T5 model

Embedding Size	Precision	Recall	F1 Score	METEOR
512	0.8048	0.8070	0.8057	0.1636
256	0.8079	0.7998	0.8036	0.1327
128	0.7997	0.7907	0.7950	0.0913

Table 2: Impact of Different Embedding Sizes on Precision, Recall, F1 Score, and METEOR (Over epochs =10 and Dataset Size: 7000)

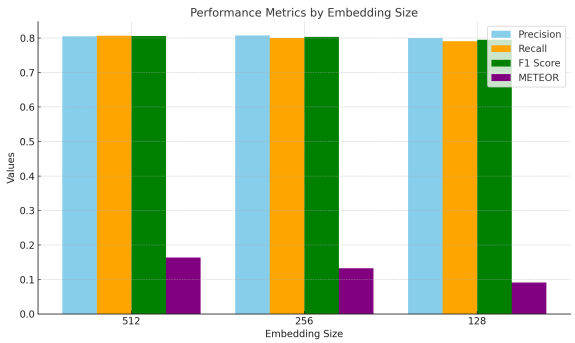

Figure 2: Embedding size vs Performance

Embedding Sizes	Training Loss	Validation Loss	Training Time
512	4.204	5.682	1:26:29
256	5.042	6.181	41:08
128	5.646	6.399	32:10

Table 3: Impact of Different Embedding Sizes on Training Loss, Validation Loss, and Training Time (Over epochs =10 and Dataset Size: 7000)

Embedding Size	Translated Output
512	”comment vous voyez comment vous pouvez voir comment comment comment comment”
256	”vous pouvez voir vous avezez vous avezez le monde”
128	”vous savez vous pouvez vous pouvez vous avez”

Table 4: Impact of Embedding size on Translated Output (Over epochs =10 and Dataset Size: 7000)

Number of Heads (n)	Training Loss	Validation Loss
8	4.204	5.682
4	4.208	5.661
2	4.264	5.695
1	4.332	5.512

Table 5: Training and Validation Loss for Different Attention Heads (Epochs=10 and DataSet size=7000)

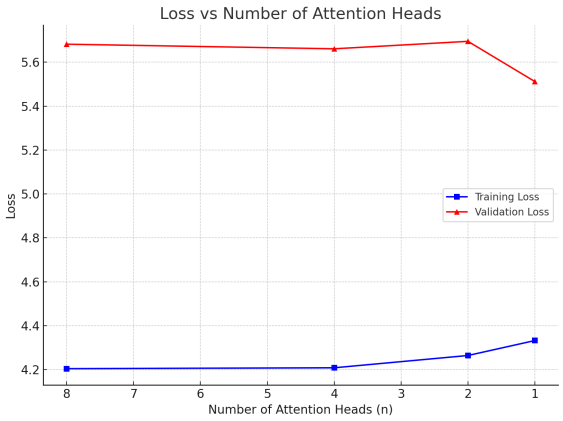

Figure 3: Loss vs Heads

Batch Size	**Training Loss**	**Validation Loss**
16	3.808	5.509
32	4.204	5.682
64	4.687	5.830

Table 6: Impact of Batch Size on Training Loss and Validation Loss (Over Epochs=10 and Dataset Size: 7000)

Batch Size	**Translated Output**
16	”comment vous pouvez voir comment aujourdhui comment aujourdhui”
32	”vous pouvez voir vous avezez vous avezez le monde”
64	”vous avez un peu de la vie sont des gens de la vie”

Table 7: Impact of Batch Size on Translation (Over Epochs=10 and Dataset Size: 7000)

Dropout Rate	**Training Loss**	**Validation Loss**
0.1	4.204	5.682
0.01	3.721	5.620
0.001	3.647	5.637
0.0001	3.670	5.650

Table 8: Impact of Different Dropout Rates on Train and Validation Loss(Over Epochs=10 and Dataset Size: 7000)

Figure 4: Loss vs Batch Size

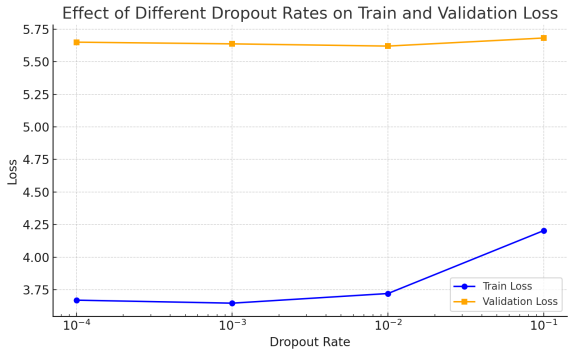

Figure 5: Loss vs DropoutRate