Project Proposal
Text Summarization with Transformer

Sungkyun Yoo

Overview

Transformer architecture, introduced in Attention is All You Need [1], has helped increase the performance on the Large Language Models we are using in the modern time. The aim of this project is by using a base Large Language Model, tailor the model to perform well on summarizing the dialogue in English.

1. Project Stages

	Task	Delivery/ Report
Plan	- Choose appropriate dataset for text summarization - Choose a pretrained model	Description of Dataset* Model description*
Analysis	- Process the dataset - After construct phase, choose	Describe in the report
Construct	- Finalize the model strategies - In-context learning - Fine-tune the model	Describe in the report
Execute	- Finalize and evaluate the results - Share the limits	Describe in the report

2. Methodology

2.1. Transformer architecture

There are different types of Recurrent Neural Network introduced, LSTM, GRU, and Bidirectional RNN. For example, however, since LSTM is known to have vanishing and exploding gradient problems dealing with Sequence to Sequence architecture for long sentences, Vaswani et al. [1] introduced the transformer architecture, encoder-decoder structure (Figure 1).

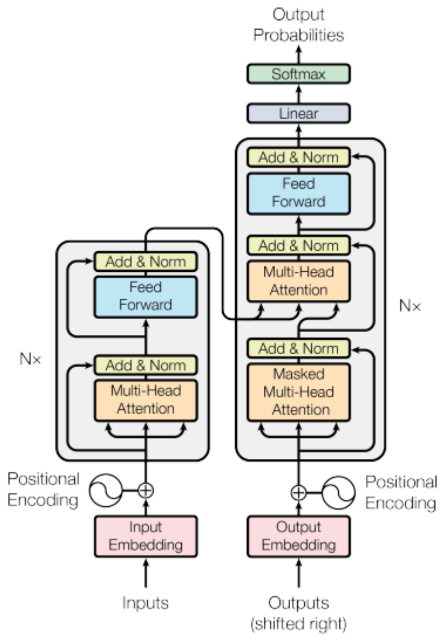

Figure 1: Anatomy of Transformer

The transformer model uses sinusoidal positional encoding to determine the relative position of each word in a sequence.

$$PE(pos, 2i) = sin(pos/10000^{2i/d_{model}})$$
$$PE(pos, 2i+1) = cos(pos/10000^{2i/d_{model}})$$

, where i is dimension. This method has a continuous nature of waves, which creates smooth transitions between the adjacent positions in a sequence.

Encoder

The encoder contains several layers which are a multi-head attention mechanism, Add&Normal, and a feed forward network.
Self attention is a scaled dot-product mechanism

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$

, where Q is query vectors, which help the model focus on position in the sequence, K is key vectors, which are the entire tokens in the sequence, V is value vectors, containing the information to be weighted by the attention scores, and d is embedding size. This method scales down with a square root of embedding size so that the gradients become more stable as the dimension size value is large. Which means it is used to avoid any exploding gradient problem. Additionally, the softmax function is applied to avoid any vanishing gradient problem, by converting into a probability distribution. At the end, attention calculates the weight sum of values to aggregate information from the entire sequence and to maintain its dimensionality.
Multi-Head is to perform the self attention function in parallel to avoid any domination of the actual word itself. The transformer uses 8 attention heads, which means there would be 8 sets.

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$

,where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

The feed forward neural network was implemented to provide non-linearity by rectified linear unit function. Since the attention mechanism retains the relationship between the tokens, this neural network was used to process those positions independently.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

In transformer architecture, The residual connection [3] was employed around each multi-head self attention and feed forward network. Since the architecture uses 6 layers of each encoder and decoder, the gradients of loss function with respect to the weights might be small. In order to avoid this vanishing gradient problem, the residual connection was employed. Layer Normalization [2] was implemented to improve training stability for vanishing and gradient problems and to help converge faster.

Decoder
Masked multi-head attention was used to pad the outputs containing different dimension sizes. The techniques used were look ahead mask and padding mask. Causal mask was used to ensure each position can attend to itself and previous positions. Padding mask was used to handle different sizes of sequences in a batch. Transformer model combined these techniques and set it to -∞ and then the softmax function was used to calculate its probability distribution and pass the attention vector Q to the next layer.
At the next layer after Add & Norm, the output, the set of attention vectors V and K are going to be concatenated and generate a linear projection.
The final layer projects the vectors generated by the decoder into logits and applies the softmax function for multi-class classification with log of probabilities, in order to choose the word in vocabulary with the highest probability.
Due to the additional layers with activation functions, the transformer architecture has an advantage of processing sequence to sequence architecture.

2.2. Model

There are different types of models to consider
1. Google Flan-T5-base: a variant of T5 model refined with Fine-tuned Language Net. This free-version model’s capacity is 250 million parameters.
2. Ollama llama3.2:1b: A free version model provided by Meta, this model’s capacity is 1 billion parameters.

3. Potential Models to consider: Bart can be considered, OpenAI base model provides good accuracy, however, this model performs well enough that I might not need to fine-tune.

The decision of model selection would need to be considered during constructing.

2.3. Tuning

1. In-context learning [6]: this helps a model to learn and perform tasks by giving n number of examples within the input prompt. However, the performance might not be fine-tuned.
2. Full Fine-Tuning: this is to update the entire parameters of a model based on a new dataset. This can perform well on the specific task, however, this method will be expensive since it is updating the entire parameters.
3. Low-Rank Adaptation [7]: unlike full fine-tuning, this reduces the number of trainable parameters and maintains most of the pre-trained weights frozen. This would be more cost-efficient than full fine-tuning, however, the performance would not be the best enough compared to the full fine-tuning.

4. Potential method: Reinforcement Learning
 Proximal Policy Optimization (PPO) [4]: a policy gradient method for reinforcement learning that clips objective function to limit the policy that makes a model

$$L^{VF} = \frac{1}{2}||V_\theta(s) - (\sum_{t=0}^{T} \gamma^t r_t | s_0 = s)||_2^2$$

 The value function, V, estimates the expected total reward for state, s. The value loss function, L, calculates the difference with the actual future total reward. This function makes the estimates for the future rewards more accurate.

$$L^{POLICY} = \min(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \cdot \hat{A}_t, \text{clip}(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon) \cdot \hat{A}_t)$$

,where the denominator of policy loss is the probabilities of the next token in the previous LLM, this is frozen. The numerator is the probabilities of the next token in the updated LLM. \hat{A_t} is an estimated advantage timer for the given action. The trust region, clip function, means where those two policies are near and also there are bounds to be stable. The PPO ensures the model to update within this trust region.

$$L^{ENT} = \text{entropy}(\pi_\theta(\cdot|s_t))$$

 This entropy function generates entropy to the model. Which means this helps the model generate more diverse completions in LLM.

$$L^{PPO} = L^{POLICY} + c_1 L^{VF} + c_2 L^{ENT}$$

,where c_1 and c_2 are hyperparameters. If I would like the model to generate more diverse completions, the coefficient of entropy function should increase. The final PPO loss function is a weighted sum, which the PPO objective updates the weights through back propagation over iterations.
 This reinforcement learning technique can be used to filter out any possibility of profanity that the model would generate.
 There is another reinforcement technique, Q-Learning, however, applying this tuning might be expensive and also my understanding of reinforcement learning is not deep enough to interpret, in this report, it might not be added.

2.4. Evaluation Metrics

 Recall-Oriented Understudy Gisting Evaluation [5]: Since the outputs from the Large Language Models are not deterministic and they are language based, this metric calculates recall, precision, and f1 by n-gram.

$$\text{ROUGE-1 Recall} = \frac{\text{unigram matches}}{\text{unigrams in reference}}$$

$$\text{ROUGE-1 Precision} = \frac{\text{unigram matches}}{\text{unigrams in output}}$$

$$\text{ROUGE-1 F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{ROUGE-L Recall} = \frac{\text{LCS(output, reference)}}{\text{unigrams in reference}}$$

$$\text{ROUGE-L Precision} = \frac{\text{LCS(output, reference)}}{\text{unigrams in output}}$$

$$\text{ROUGE-L F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

This metric will provide the accuracy of how much the generated output by a model would be matching with the reference.

Few examples for human evaluation: there are going to be a few examples to assess by a human with few-shot inferences.

3. Description of Dataset

This DatasetDict contains only one key, which is train. This should be split into train, validation, test sets. It has 52,480 rows with ‘dialogue’ and ‘summary’, with 29.2 mb.

4. References

[1] Ashish Vaswani et al. (2017). Attention is all you need. Retrieved from: https://arxiv.org/pdf/1706.03762
[2] Jimmy Lei Ba et al. (2016). Layer Normalization. Retrieved from: https://arxiv.org/pdf/1607.06450
[3] Kaiming He et al. (2016). Deep Residual Learning for Image Recognition. Retrieved from: https://arxiv.org/pdf/1512.03385
[4] John Schulman et al. (2017). Proximal Policy Optimization Algorithm. Retrieved from: https://arxiv.org/pdf/1707.06347
[5] Chin-Yew Lin, Eduard Hovy. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. Retrieved from: https://aclanthology.org/W04-1013.pdf
[6] Tom B. Brown et al. (2020). Language Models are Few-Shot Learners. Retrieved from: https://arxiv.org/pdf/2005.14165
[7] Edward Hu et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. Retrieved from: https://arxiv.org/pdf/2106.09685

5. Appendix

Dialogue and summary dataset, retrieved from: https://huggingface.co/datasets/Isotonic/DialogSumm