Analysis of Transformers Applied to Abstractive Text Summarization

Rafe Kruse
Herbert Wertheim College of Engineering
University of Florida
rafekruse@ufl.edu

Abstract

In this paper I explore and analyze the prevailing model for solving abstractive text summarization problems, the Transformer. Using the well-known Extreme Summarization and CNN/DailyMail datasets for training and comparison with current Transformer literature. My Transformer takes the original implementation that was applied to language translation, a similar field, and reapplies it to the problem of ATS. The Transformer is unique from previous models, utilizing two attention functions and positional encoding to replace recurrent and convolutional layers. My experimentation shows that the Transformer architecture is well suited for the problem of ATS, achieving a ROUGE-1 score of 18.11 on the CNN/DailyMail dataset given a reduced dataset and smaller model complexity.

1 Introduction

Abstractive text summarization (ATS) is a field of Natural Language Processing (NLP) concerned with creating a novel text summary that accurately captures the meaning of the source text. This is unique from past extractive techniques[1,2] that assign value to certain portions of text and "effectively" summarize by removing text considered non-essential. It is the requirement to create new text, not present in the source, that makes ATS a difficult natural language processing task. Achieving this requires a model to analyze and understand a large amount of text, compress relevant information, and generate original output. ATS can be broken down into two independent steps that are common in the field of NLP. First informative feature extraction, wherein the case of ATS, creating an condensed representation of the source text containing relevant semantic information. Second converting the compressed information back into a human readable format that maintains the source meaning and is syntactically correct.

The problem of ATS has been approached using a wide variety of commonly used NLP techniques. The encoder-decoder architecture has been the dominate model for solving ATS and similar language encoding problems such as text translation[3]. In the past the application of recurrent neural network (RNN) and long-short term memory (LSTM) layers were predominately used for their well-known ability to process and remember sequential data[4]. In recent years Transformers that employ attention and fully connected layers have taken over as the go-to for ATS tasks. These Transformers based upon attention layers have shown to not only be more effective at encoding and decoding ATS problems but come with the added benefit of a much quicker training time as they remove the costly overhead of training and running RNN, LSTM, and CNN operations. Current state-of-the-art papers also rely upon unsupervised pre-training, that Transformers especially benefit from, which has been shown to improve Transformer's ability to generalize on language tasks [5].

In this work I take the well known Transformer implementation[6] and apply it too two ATS news summary datasets with the goal of comparing the performance of a reduced complexity Transformer to the original implementation and the current state-of-the-art pretrained model.

2 Problem Statement

2.1 Data

The two data sets chosen for this paper are Extreme Summarization (XSum) dataset and the CNN/DailyMail non-anonymized summarization dataset. XSum consists of ~226,000 news articles and their corresponding one sentence summaries. The CNN/DailyMail dataset contains ~311,000 news articles and summaries constructed by concatenating all highlighted article content.

In order for the above plain text datasets to be usable within the Transformer model the data must first go through a process of embedding. This is where every word is mapped to a specific word vector of dimension N within a vocabulary V. Where V is all words found within the input text and N is a model hyperparameter for the dimensionality of the word vectors. It is these embeddings that are applied to decoder outputs along with the softmax function in order to predict the summarization tokens.

Transformers are made using directly embedded data and thus have no information related to the order of the text. For the case of ATS this poses a problem as the positional information of text is very important to the semantic meaning. To resolve, this I chose to implement the current leading method described within[6] and described shortly below.

This method makes use of the sine and cosine functions below,

$$E_{p,2i} = \sin\left(\frac{p}{10000^{2i/d}}\right)$$

where p is the position, i is the dimension, and d is the dimensionality of the previously mentioned vocabulary. Applying these functions to the input embeddings allows the model to learn relative positions, for any offset k, E_{p+k} which can be represented as a linear function of the above E_p.

2.2 Loss

During training after the predicted output of the decoder portion of the Transformer is obtained the loss for a sample is calculated using the well known cross-entropy loss function. Cross-entropy is a metric for quantifying the difference between two probability distributions and is well suited for training on multi-class classification problems. ATS follows a similar pattern as classification problems as the prediction of the next token is chosen from a set of probabilities corresponding to each vector within the embedded vocabulary. The equation for calculating the loss of a single embedded vector within the vocabulary is as follows,

$$loss(\Theta, E) = -\log\left(\frac{\exp(\Theta_E)}{\sum_j \exp(\Theta_j)}\right)$$

where Θ is embedded vector matrix and E is a particular embedding.

2.3 Optimization

Transformers consist of a series of attention, which is covered in section 3, and fully connected layers. This makes training them similar to other encoder-decoder networks and artificial neural networks in general. Because these models, transformers included, are made up of series of independently connected layers of nodes there is no singular optimization problem to solve for. Instead backpropogation of the above cross-entropy gradients is employed to train the Transformer network. To achieve this training I chose, and is common within the literature, to use the well known ADAM[11] optimization algorithm.

3 Algorithms

3.1 Attention

Attention is the functionality that makes Transformers unique from strictly encoder-decoder networks and is the change that allows them to out preform previous recurrent methods. Attention functions map queries to a set of key-value pairs. The output of an attention function is then computed as

the sum of the values weighted by the compatibility of the query and key. There are two common methods for calculating attention, additive and the dot product. For the purpose of this paper dot product was chosen because it is far more efficient and if scaled[6] can avoid the dot product value exploding. The attention values are calculated as follows,

$$A(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{n}}\right)V$$

Where Q is the matrix of queries, K is the key matrix, V the value matrix, and n the dimensionality of the queries.

3.1.1 Multi-Head Attention

Multi-headed attention is an extension of the above attention function that allows the model to reference different position jointly while also having access to multiple vector subspaces to learn from. This is done by having multiple sets of query, key, and value matrices linearly projected then concatenated into a single matrix for use in the feed-forward layer that is explained in section 3.2.

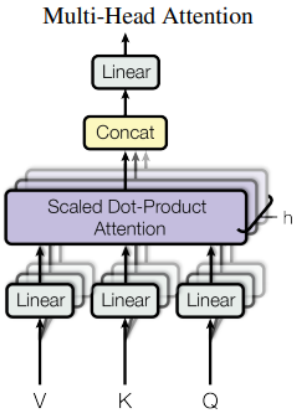

Figure 1: Multi-Head Attention component architecture

3.2 Transformer

The Transformer architecture is primarily based on the well known encoder-decoder network architecture with a few additions such as the previously covered attention functions. Within the literature it is common to use these encoder-decoders structures with six or more layers for both the encoder and decoder. However, given the resulting complexity of a model of this size, for the purpose of this paper the size was reduced which is explained in greater detail in section 4.2.

3.2.1 Encoder

The encoders within a Transformer consist of a series of identical layers made up of two components. The first being the aforementioned multi-head attention function. The second being a traditional fully connected feed-forward neural network. An additional residual is added connecting the input of a layer around to the output. This residual serves to ease the training process by giving the model an additional reference to layer inputs reducing the difficulty of learn functions without context. Finally the results of each encoder layers are normalized to prevent exploding values.

3.2.2 Decoder

The decoder structure is very similar to the encoder with the addition of a second multi-head attention function applied to the encoder stack output. When decoding data the predicted previous token is important to the prediction of all following tokens, because of this the decoder must be aware of previous tokens. Thus the first attention layer in the decoder is masked to only allow the decoder access to positions prior to the current prediction position. This limitation is also the reasoning behind

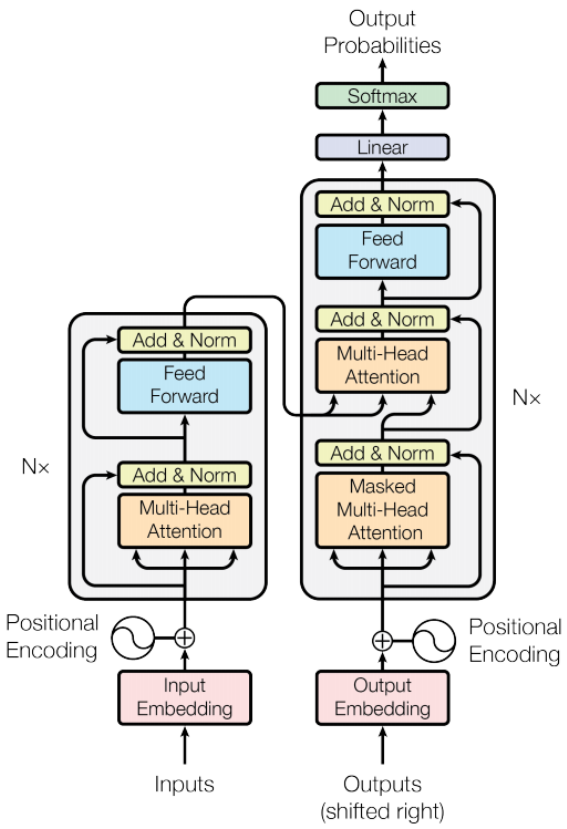

Figure 2: Transformer, model architecture[6]

the inclusion of the additional attention layer, while the first attention layer of the decoder gives information on previous tokens, the second gives greater context and helps focus on relevant tokens in the input sequence.

3.3 Regularization

Regularization typically can be applied to many deep learning models to improve results and this holds true for Transformers. The first method of regularization added in this paper is the a residual dropout applied to feed forward networks within the model in order to reduce complexity and improve performance. Well tuned dropout has been shown to be effective in reducing training times without performance detriments. The second change applied to the model is label smoothing that penalizes the model for high confidence values. While not entirely known why label smoothing improves model performance it has been empirically shown[9] to be effective and is assumed to prevent models from overfitting by approaching local minima in the confidence of certain tokens. This label smoothing function is shown below and further explained in [8],

$$S_i = (1 - \epsilon)K(p, q) + \epsilon K(u, q)$$

Where ϵ is a weight factor, K is the relative entropy loss function[7], p is the one-hot encoded distribution of a tokens word vector, u is a uniform distribution of the word vector, and q is the predicted distribution.

4 Experiments

4.1 Model Parameters

My model's hyperparameters are compared to TRANSFORMER$_{BASE}$ because it is the same model, and is the parameters/architecture the majority of state-of-the-art models use. All the hyperparameters, if possible, were kept consistent with TRANSFORMER$_{BASE}$ and otherwise were maximized for ROUGE-1 score using a grid search method with a reduced number of epochs.

Table 2: ROUGE Score comparison between

Hyperparameter	Transformer$_{ME}$	Transformer$_{BASE}$
Vocab Dimensionality	256	512
Feed Forward Output	1024	2048
Multi-Attention Heads	6	8
Residual Dropout	0.1	0.1
Encoder-Decoder Layers	2	6
Smoothing Coefficient	0.1	0.1
Dataset Size(XSum/CNN)	50k/50k	226k/311k
Batch Size	1000	<25000 (Varies)
ADAM β_1	0.9	0.9
ADAM β_2	0.98	0.98
ADAM ϵ	10^{-9}	10^{-9}

4.2 ROUGE Result Metrics

Recall-Oriented Understudy for Gisting Evaluation(ROUGE) is a metric for ATS evaluation. ROUGE has many different variants with ROUGE-N, the overlap of N-grams quantity, and ROUGE-L, the longest common subsequence, being the most popular. For the purpose of comparison with other papers in the field I chose to use the ROUGE-1, ROUGE-2 and ROUGE-L metrics for evaluation in this paper.

4.3 Results

The model within this paper was trained using a single NVIDIA 1080Ti GPU. Given the above hyperparameters each training step took approximately 0.45 seconds and the ideal model was trained for 10,000 epochs. I also applied the 80/10/10 training/validation/test split found within other papers, however, given computational constraints I only utilized 50k samples from each dataset.

Table 2: ROUGE Score comparison between

Dataset	Dataset Size	Transformer$_{ME}$	Transformer$_{BASE}$	PEGASUS$_{LARGE}$
XSum	50k/226k	15.67/6.12/12.45	30.83/10.83/24.41	47.21/24.56/39.25
CNN/DailyMail	50k/311k	18.11/9.89/15.91	38.27/15.03/35.48	44.17/21.47/41.11

Scores are given in ROUGE-1/ROUGE-2/ROGUE-L format. And dataset size is split based on the size my model was trained with and the other papers.

The hyperparameters applied to my model were far smaller is size reducing training time to something doable given time constraints, however, with this comes a far smaller model complexity and a decrease in score.

5 Conclusion

In this paper I explained the well known Transformer model as well as going into detail on a few of the models parameters. Additionally, I implemented the Transformer model and applied it to the XSum and CNN/DailyMail datasets.

It is clear from my results presented that my model didn't preform comparably to any other ATS Transformer implementations. However, this was to be expected as the complexity of my model was greatly reduced along with the dataset size used for training. The results of my model are a testament to Transformers strength for the ATS task. Taking into account the short amount of time training, relative to other methods, my model was able to achieve a ROUGE score where the gist is hard to understand but present.

The Transformer implementation in this paper is also still used within the current state-of-the-art ATS models with improved training methods that utilize pretraining text corpuses. Given that Transformers have far more efficient training times and ability to be parallelized makes the models ideal for being scaled up to use a large amount of computation. I'm interested to see and would like to in the future explore the application of Transformers to other NLP problems as they have already been shown to be ideal for ATS and language translation tasks.

References

[1] Duy Duc An Bui, Guilherme Del Fiol, John F. Hurdle, Siddhartha Jonnalagadda, "Extractive text summarization system to aid data extraction from full text in systematic review development", Journal of Biomedical Informatics, Volume 64, 2016, Pages 265-272, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2016.10.014.

[2] J. N. Madhuri and R. Ganesh Kumar, "Extractive Text Summarization Using Sentence Ranking", 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, 2019, pp. 1-3, doi: 10.1109/IconDSC.2019.8817040.

[3] P. Li et al. 2017. "Deep Recurrent Generative Decoder for Abstractive Text Summarization." In Proceedings of EMNLP 2017

[4] Khatri, C., Singh, G., Parikh, N. (2018). "Abstractive and Extractive Text Summarization using Document Context Vector and Recurrent Neural Networks." ArXiv, abs/1807.08000.

[5] Zhang, Jingqing, et al. "PEGASUS: Pre-Training with Extracted Gap-Sentences for Abstractive Summarization." ArXiv.org, 10 July 2020, arxiv.org/abs/1912.08777.

[6] Vaswani, Ashish, et al. "Attention Is All You Need." ArXiv.org, 6 Dec. 2017, arxiv.org/abs/1706.03762.

[7] Joyce J.M. (2011) "Kullback-Leibler Divergence". In: Lovric M. (eds) International Encyclopedia of Statistical Science. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04898-2$_3$27

[8] Pereyra, Gabriel, et al. "Regularizing Neural Networks by Penalizing Confident Output Distributions." ArXiv.org, 23 Jan. 2017, arxiv.org/abs/1701.06548.

[9] Müller, Rafael, et al. "When Does Label Smoothing Help?" ArXiv.org, 10 June 2020, arxiv.org/abs/1906.02629.

[10] The Annotated Transformer, 3 Apr. 2018, nlp.seas.harvard.edu/2018/04/03/attention.html.

[11] Kingma, Diederik P. and Jimmy Ba. "Adam: A Method for Stochastic Optimization." CoRR abs/1412.6980 (2015): n. pag.