Attention to Shakespeare

Soumita Das
Department of Computer Science and Engineering
University at Buffalo
soumitad@buffalo.edu

Sambo Dutta
Department of Computer Science and Engineering
University at Buffalo
sambodut@buffalo.edu

Abstract

Deep Learning Neural Networks are extremely powerful tools in classifying labeled data but an end to end mapping is required when learning complex long sequences. In this project we address the task of using Neural Machine Translation to apply artistic style transfer. While this may look like another translation task, the challenge is that the input and target maintains almost similar vocabulary and thus the model would be learning a particular writing style rather than word level translations. *Seq2Seq* model leveraging Recurrent Neural Networks like LSTM(Long Short Term Memory) has been used previously to convert modern english language to *Shakespearean* style. We aim to further improve the speed and quality of this style transfer using *Transformer* model. The *Transformer* model based solely on attention mechanisms, dispensing with recurrence and convolutions entirely reduces the training time as the system is more parallelizable yet producing better translations. Following this we present a modified version of Transformer named as the *Pizza* structure which proves to give better quality style transfer in spite of having less number of parameters. We further compare the performance of all the three models : *Seq2Seq with Attention*, *Transformer* and *Pizza Transformer*.

1 Background

Deep Neural Network is an extremely powerful tool in Machine Learning. It can be trained and achieves good results using supervised back-propagation. However, DNNs require that dimensions of the input and target variables should be shown. This is not possible for translation, or questions and answers pairs for chatbots, etc. Thus it is clear that a domain-independent method that learns to map sequences to sequences would be useful.
In this project we address the task of artistic style transfer. We aim at translating modern english language to *Shakespearean* style. While this may seem as a translation task, it has few differences, for instance the vocabulary is hugely overlapping for both input and target sequence. Here both input and target use English words. The model here should learn the essence of writing style and not word to word mapping.

1.1 Sequence to Sequence Models

Previously, Sequence to Sequence models was used for this task. In 2013 Kalchbrenner and Blunsom [3] first encoded an entire sequence to a vector. Following which Sequence to Sequence

Learning with Neural Networks [1] were introduced by Google that maps a fixed-length input with a fixed-length output where the length of the input and output may differ.
An encoder component comprising of RNNs like LSTM or GRU encodes the entire input sequence to a hidden state vector. This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions. The decoder component similarly accepts a hidden state from the previous unit and produces output as well as its own hidden state for the next word. The power of this model lies in the fact that it can map sequences of different lengths to each other.

1.2 Sequence to Sequence with Attention

According to Dzmitry Bahdanau, et al.[4] a potential issue with this encoder–decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus. LSTMs do aim to preserve this information but even it fails to remember all of it in case of very long sequences. The attention mechanism was born to resolve this problem.
The idea of **attention** is extremely intuitive. When decoding a sentence, we want the model to pay *attention* to only a part of the sentence and not the entire one.

In Xu et al.[9] attention mechanism was used to generate captions for images. CNN was used for encoder while LSTM was used for the decoder model. In this work attention was classified into *hard attention* and *soft attention*. In **hard or local attention** the model only selects one part of the input to attend to at a time saving inference time, while in **soft or global attention** the model attends to the entire input softly with different the alignment weights placed over all parts of the input which is the same model as Bahdanau et al., 2015.[4].

Different types of attention like additive [4], location-base [5], dot-product and scaled dot product [2] have been used. However these techniques were used to generate a attention-based context vector for the RNN decoder until the *Transformer* model was introduced in 2017.[2]

2 Transformer Model Architecture

The Transformer was proposed in the paper Attention is All You Need [2]. The main advantage of this model is how efficiently it uses the attention mechanism and dispenses any sequential neural networks like RNN or CNN. This allows more parallelization and reduces the training time.
The complete architecture is shown in Figure 1.

2.1 Encoder and Decoder Layers

The Transformer model can be thought of as a black box which takes an input, encodes it. The decoder part then uses the encoded sequence to generate a decoded token one at a time.

Encoder Component : The Encoder layer consists of multiple identical stacks of encoder. In this work we have used 4 layers of encoders. Each encoder consists mainly of two sub-layers : self-attention layer and feed-forward neural network (FNN).
A self-attention layer helps the encoder look at other words in the input sentence as it encodes a specific word. The details regarding self-attention will be discussed in the upcoming sections. The outputs of the self-attention layer are fed to a feed-forward neural network. The exact same feed-forward network is independently applied to each position.

Decoder Component : The Decoder layer must also contain exactly the same number of layers decoders as the encoder layer i.e. in this project number of layers =4. However each decoder has 3 sub-layers: self-attention layer, feed-forward neural network (FFNN) and another attention layer in between them. This additional attention layer helps the decoder focus on relevant parts of the input sentence, similar to what attention does in seq2seq models, while decoding. The *Masked* multi-headed attention prevents the decoder to look at the future input tokens when decoding a certain word.

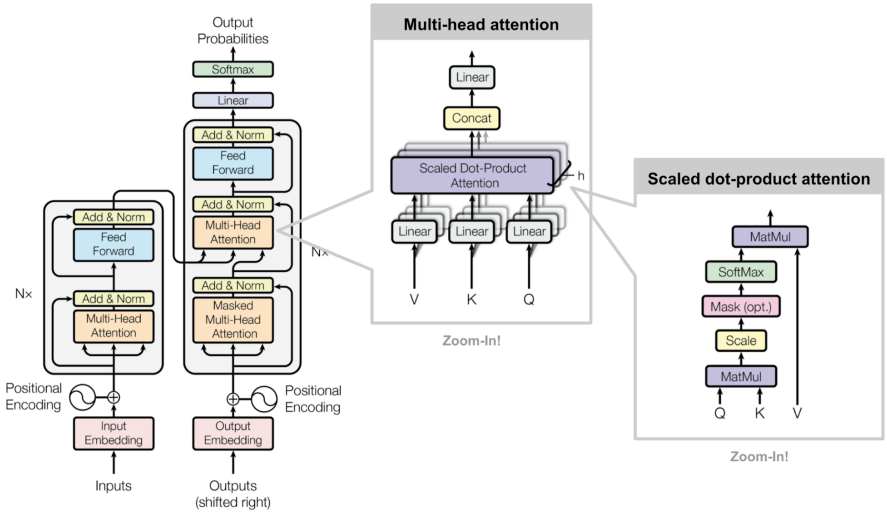

Figure 1: Architecture of Transformer (Image source: Fig 1 & 2 in Vaswani, et al., 2017.)

The Residuals : One detail in the architecture of the encoder and decoder is that each sub-layer (self-attention, FFNN) has a residual connection around it, and is followed by a layer-normalization step.

2.2 Self Attention

Attention is one of the most influential ideas in deep learning. Like humans, the attention mechanism breaks a sequence into parts and uses only parts of it to translate one part at a time rather than memorising the entire sequence.
Self-attention is the method the Transformer uses to bake the understanding of other relevant words into the one being currently processed. Self-attention. also known as **intra-attention**, is an attention operation of a single sequence in order to calculate the representation of the very same sequence.

2.2.1 Calculating Self Attention: Scaled Dot Product Attention

The first step is to create three vectors from each of the encoder's input vectors (i.e. the embedding of each word). So for each word, a **Query** vector, a **Key** vector, and a **Value** vector are created. These vectors are created by multiplying the embedding vector by three weights matrices $W_Q W_K, W_V$ that were trained during the training process. The dimension of these Key, Query and Value vectors are usually less than that of the embedding dimension (set to 128 in our model).

Using these Key, Value vectors a self-attention score is calculated for each Query vector. The self attention score is a weighted dot product divided by the $\sqrt{d_k}$ where d_k is the dimension of the Key vector. A softmax function is then applied so that all scores for a particular Query sums up to 1.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

2.2.2 Multi-Headed Attention

Transformer model leverages multi-headed attention for mainly two reason:
1) It expands the model's ability to focus on different positions.
2) It enables the model to jointly attend to multiple representation subspaces. With multi-headed attention we have not only one, but multiple sets of Query,Key,Value weight matrices (in [2] the authors have used 8 heads). Each of these sets of weights is randomly initialized. Then, after training, each set is used to project the input word embeddings into a different representation

subspace.

In our model we too have used 8 heads. Before passing to the FFNN the outputs from each of these heads are concatenated and multiplied with a weight matrix say W_0 which is also trainable.

2.3 Positional Encoding

Since the Transformer model does not have any usage of RNN or CNN, the idea of positional encoding has been used to maintain the order of the sequence. A vector is added to each input embedding. These vectors follow a specific pattern that the model learns, which helps it determine the position of each word, or the distance between different words in the sequence. The intuition is that adding these values to the embeddings provides meaningful distances between the embedding vectors once they are projected into Query, Key and Value vectors and during dot-product attention. The following equation is used for calculation:

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2}$$

2.4 The Final Linear and Softmax Layer

The decoder stacks outputs a vector of floats. The job is to convert this vector to a predicted word.
The **Linear Layer** is a fully connected neural network that converts this vector to a logits vector having dimension same as that of the length of the vocabulary.
The **Softmax Layer** then assigns probabilities summing up to one to each of the words, and the word with highest probability is predicted.

3 The Modified Transformer: *Pizza Structure*

The task presented is only style transfer with the vocabulary of sentence formations staying same for both original and predicted sentence. This gives us an advantage of simplifying the Transformer model even further so as to reduce training time.
The most number of trainable parameters found in the Transformer model are the ones part of the Feed Forward Neural Networks in each of the encoder and decoder stack. Thus in the modified transformer we remove these Feed Forward Neural Networks from 1/4 of the stacks of encoder and decoder i.e. 3 out of 4 layers will follow the same configuration as that of any traditional transformer while the first layer will consist of only self attention.

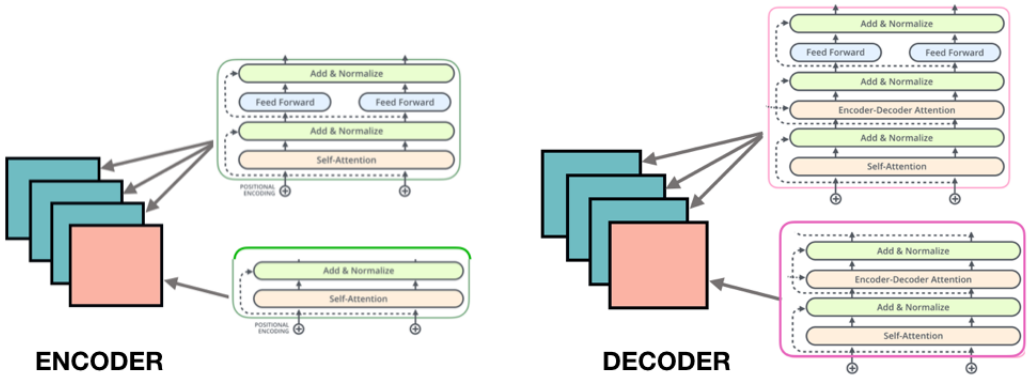

Figure 2: Architecture of Pizza Transformer

Figure 2 shows how the layers of encoder and decoder work. In encoder model 3/4 of the layers contain both Multi-headed Self Attention and Feed Forward Neural Networks while 1/4 of the layers

only has Multi-headed Self Attention. In decoder 3/4 of the layers contain all three Multi-headed Self Attention, Masked Multi-headed Attention and Feed Forward Neural Networks while 1/4 of the layers only has Multi-headed Self Attention and Masked Multi-headed Attention.
Thus we name this architecture the *Pizza structure* with the bread being 3/4 of traditional stack layers and with 1/4 layer of topping of self-attention.

4 Experiments

4.1 Data set

Sparknotes (https://www.sparknotes.com/shakespeare/) and Enotes (https://www.enotes.com) translations of portions of Shakespeare plays were used to train the model. There were 24 plays in total. The modern translations and their respective Shakespearean language were aligned. For testing the model 10% sentences were extracted from the total data set and not used for training. After preprocessing and excluding sentences with more than 40 words there were a total of 28545 sentences with a vocabulary size of 8038.

4.2 Models

We will be comparing the performance of 3 models in the upcoming sections. The first model is the *baseline* model as implemented in [8]. It is a **Sequence to Sequence** model with attention. Both the encoder and decoder module consisted of GRU (Gated recurrent unit) layers of 1024 units. The second model which outperforms the baseline model is the **Traditional Transformer** model. It has 4 stacks of both encoder and decoder module with each stack containing both 8-headed self-attention and Feed Forward Neural Network (FFNN) with 512 hidden units. Thirdly, we using our modified **Pizza structured Transformer** for comparison. It too has 4 stacks of both encoder and decoder module with 3/4 layers being the traditional layer consisting of 8-headed self-attention and FFNN with 512 hidden units. However, the first layer is the *topping* layer consisting only of multi-headed self attention (8 heads). All the 3 models used an embedding dimension of 256.

4.3 Results

4.3.1 Training speed

In this section we will compare the training performance of 3 models: *Sequence to Sequence with Attention*, *Transformer* and *Pizza Transformer*. While Seq2Seq with Attention has the highest training time owing to the fact that it has sequential layers like RNN or LSTM, the Pizza Transformer outperforms the traditional Transformer and has the lowest training time. For *Pizza Transformer* the first layer is a topping layer with only self-attention and no feed forward neural network. The table below shows number of trainable parameters for each traditional and topping layer.

Layers	Trainable Parameters
Encoder traditional layer	198,272
Encoder topping layer	66,304
Decoder traditional layer	264,576
Decoder topping layer	132,608

This tells us that for Encoder the topping layer has almost 3 times less trainable parameters than the traditional layers, while for Decoder the topping layer has half the number of trainable parameters in the traditional layer.
This reduction in total number of trainable parameters makes the training speed of *Pizza Transformer* the fastest.

Figure 3 shows the training time in seconds for each epoch for two settings: without GPU and with GPU.

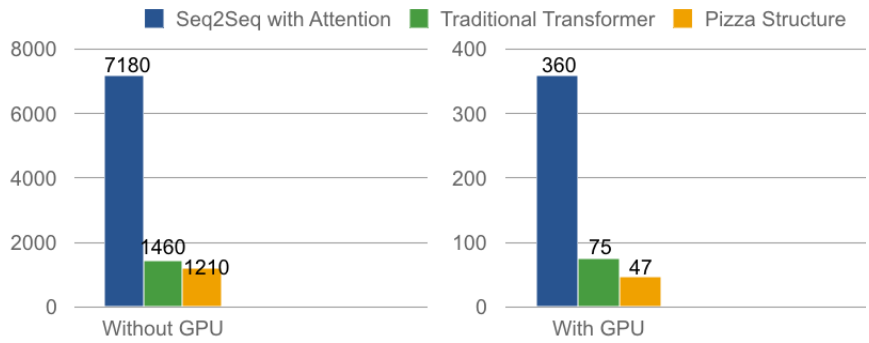

Figure 3: Comparison of training time for each epoch

4.3.2 Translation Quality

Here we will show the translations given by the same 3 models:*Sequence to Sequence with Attention*, *Transformer* and *Pizza Transformer* with the same architectures as discussed in section 4.2.
The following are few hand-picked examples of translations:

Original Text	Seq2Seq with Attention	Traditional Transformer	Pizza Transformer
have you killed tybalt ?	hast thou slain tybalt ?	hast thou slain tybalt ?	hast thou slain tybalt ?
i will hit you !	i will , beat thee !	i will strike thee !	i will strike thee !
What is wrong with you ?	what is amiss you ?	what is your passion ?	what is thy wrong with thee ?
what fight was here?	what fray was here ?	what fray was here ?	what fray was here ?
what do you want ?	what would you ?	what would you ?	what s your will ?
I am very happy .	i am very merry .	i am much merry .	i am much merry .

For generation of **BLEU (bilingual evaluation understudy)** score the re-sampling procedure **10-fold cross validation** was used. Each time we excluded 10% random sentences from the entire shuffled training data set and used the rest 90% sentences for training. The BLEU score was calculated for all the 3 models. The summary of the scores is given in the following table:

Folds	Seq2Seq with Attention	Traditional Transformer	Pizza Transformer
1st Fold	0.497	0.508	**0.517**
2nd Fold	0.503	0.510	**0.511**
3rd Fold	0.499	0.513	**0.518**
4th Fold	0.497	**0.506**	0.502
5th Fold	0.489	**0.506**	0.505
6th Fold	0.508	0.507	**0.510**
7th Fold	0.501	0.510	**0.519**
8th Fold	0.499	**0.507**	0.504
9th Fold	0.500	0.509	**0.511**
10th Fold	0.501	0.509	**0.512**

The bold cells show the best BLEU score for each of the folds. The table shows that the *Transformer* architecture beats the *Seq2Seq* model every time. It also shows that the *Pizza Transformer* has a better quality of translation than the *Traditional Transformer* in most cases even while maintaining less number of parameters.

5 Conclusion and Future Work

In this work we presented how the *Transformer* model dispensing of any sequential networks like RNN or CNN outperforms the *Seq2Seq* model. Consisting only of Feed Forward Neural Network and Self-Attention the *Transformer* model is much more parallelizable. Following this, a modified version of the Transformer was introduced: *the Pizza structure*. Taking advantage of the style transfer task where the vocabulary is overlapping for both the input and target sequence, we were able to further reduce trainable parameters by using 1/4 topping layer of only Self Attention in Encoder and Decoder modules. This set up not only reduced the training time but also improved the quality of style transfer.
In our future work we plan to extend this modified *Pizza Transformer* to other general translation or conversational tasks. We will also work on finding the optimum number of layers that can be modified to topping layer for a particular task using machine learning tools.

References

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 3104–3112.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. NIPS.

[3] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In EMNLP, 2013.

[4] Bahdanau, Dzmitry & Cho, Kyunghyun & Bengio, Y (2014). Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv. 1409.

[5] Luong, Minh-Thang & Pham, Hieu & Manning, Christoper. (2015). Effective Approaches to Attention-based Neural Machine Translation.

[6] Weng, Lilian. Attention? Attention! (2018). https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

[7] Alammar, Jay. The Illustrated Transformer. (2018) http://jalammar.github.io/illustrated-transformer/

[8] Xu, W., Ritter, A., Dolan, W.B., Grishman, R., Cherry, C. (2012). Paraphrasing for Style. COLING.

[9] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: neural image caption generation with visual attention. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15).