News Article Summarization Using Transformers

Abstract

Due to huge amounts of data available on the internet it becomes more and more important to find a way to deliver the information to people in a small and a concise manner without any loss of information. In this project we aim to summarize text from news articles using transformers models. In this project we evaluate the performance of a generic transformer network, the T5 transformer model and the pegasus transofrmer model. We summarize our findings and decide which transformer model is more suitable for news summarization

1. Introduction

Introduction: The main goal of this project is to generate a concise summary of news articles without too much loss of information. Summarization is the ability to explain a larger piece of literature in short while conveying most of the information and meaning of the larger text. In recent times, transformer networks are used to solve this challenge. When we aim to solve text summarization as a natural language processing task , we classify summarization into two categories:

- Extractive summarization: Extractive summarization involves the task of extracting a few important sentences or text from the large corpus of text based on predefined weights assigned to important words and the sentence selection is done based on the number of important words in it.
- Abstractive summarization: Abstractive summarization involves a heuristic approach where we train the model to understand the context of the entire text and generate a summary based on the model's understanding.

Main Reasons for Text Summarization:

- Selection of the most important information from a large corpus of data.
- Reduce the amount of time for reading a specific article.

2. Methods

2.1. Transformers

Sequence to sequence models are modern deep learning models that are used for natural language processing tasks like machine translation, text summarization , image captioning among others. A sequence to sequence model is a deep learning model that takes in a sequence of words as input and outputs another sequence of words, depending on the task it is trained to perform. For example in a machine translation model , the input is a sequence of words in one language and the out is a sequence of words in an other language.

The underlying deep learning model has 2 essential building blocks to perform these tasks , namely the encoder and decoder. The encoder processes the sequence of inputs and converts it into a vector. This vector is more commonly known as the context vector. The encoder sends this created context vector to the decoder which then decodes the context vector and generates outputs. The encoder and decoder both consists of recurrent neural network units (RNN's or LSTM's) to encode and decode the context vectors.

The size of the context vector is defined by the number of RNN units present in the encoder or decoder. Basically, the RNN takes in two inputs, namely the hidden state and the word. However, the words cannot be read by the RNN's and need to be passed as vectors. The conversion of words into these said vectors is known as the process of word embeddings.

In the initial step, the RNN unit take the first word embedding vector as a vector and the hidden state(all zero vector) and produces a hidden state. In the next time step the next RNN unit will take the second word embedding and the first hidden state generated by the first RNN unit and generates a second hidden state. This goes on until the the last RNN unit generates the last hidden state. This hidden state generated is called the context vector as discussed above.The decoder follows the same procedure as the encoder.

The context vector turned out to be an issue while dealing with large sentences.This lead to the introduction of a new concept called attention by Luong et.al in 2015.Attention concentrates on the relevant parts of the sentence rather than the whole sentence and this improved the performance of sequence to sequence models.In the attention mechanism,

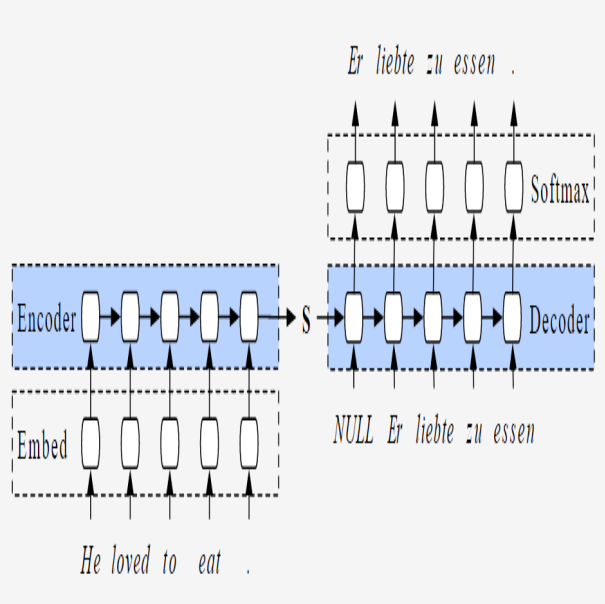

Figure 1. Sequence-To-Sequence Models

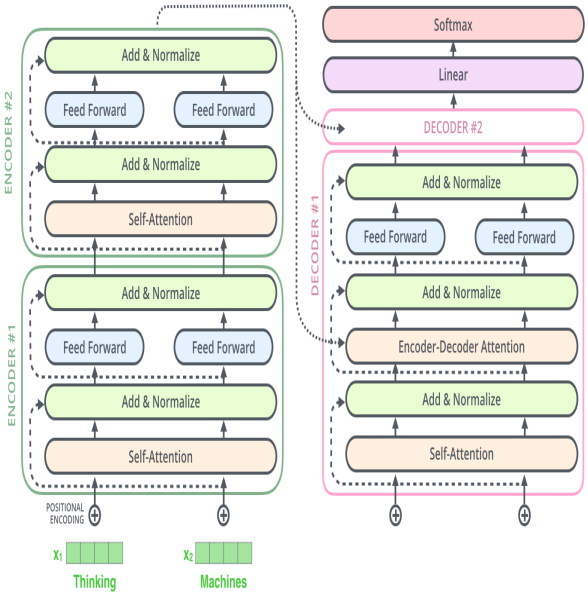

Figure 2. Transformer Model

the encoder passes all the hidden states it generates to the decoder instead of the last hidden state. The decoder then performs an extra step before sending all these hidden states to the decoder RNN units. It looks at each hidden state and the word it is associated with and then scores each hidden state.

Transformer networks use this attention mechanism to perform NLP tasks. However transformer networks differ from the generic sequence to sequence models. The sequence to sequence models consist of one encoder and one decoder to perform specific tasks. However a transformer network has a stack of encoder and decoder block stacked on top of on another.

Each encoder has 2 sub blocks namely the self attention block and the feed forward block. The input is passed onto the self attention block and then to the feed forward block. The decoder also consists of these layers but in the middle of the self attention block and the feedforward block , it consists of a encoder- decoder attention block which performs the same operation as the pre decoder step in the sequence to sequence models.

The bottom most encoder converts the words into word embedding vector but all the encoder layer above that will receive the output vectors from the encoder below them. Self attention layer understands the relevance of words with one another.There is a normalization layer after every self attention layer and the feed forward layers.The procedure is similar even for the decoder. However at the end of the decoder , we would get a set of vectors as an output. The process of conversion of these vectors to words is done by the linear layer and the softmax layer.[1]

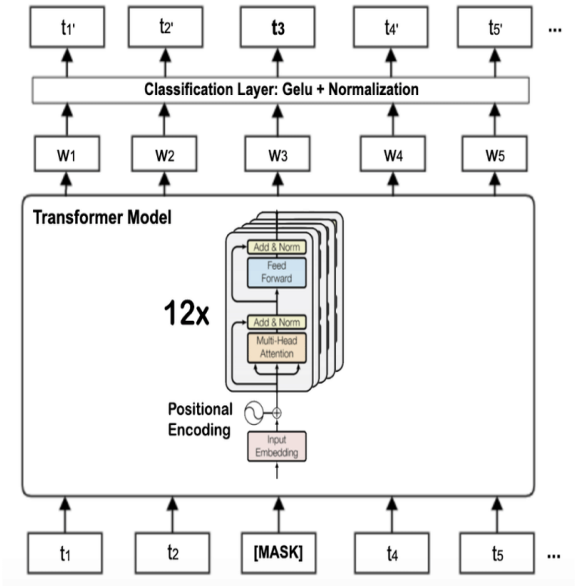

Figure 3. T5 Model

2.2. T5 Model

The T5 model or transfer learning with a unified Text-To-Text Transformer was created by google to handle multiple natural language processing tasks. It consists of 12 encoder-decoder blocks with a similar configuration to BERT base model architecture.[2]

2.3. Pegasus

Pegaus or Pre-training with Extracted Gap-sentences for Abstractive Summarization is intentionally similar to sum-

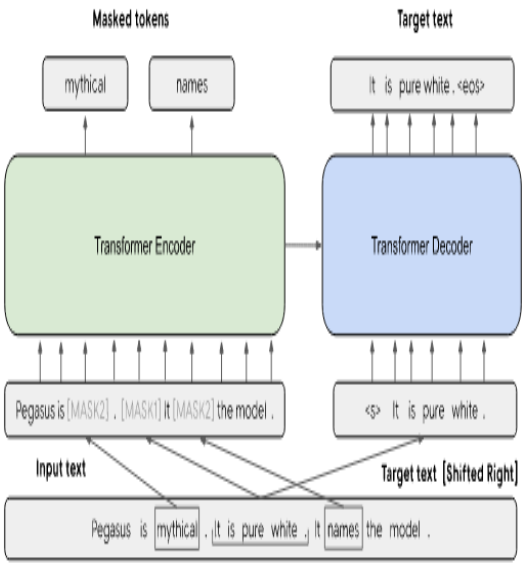

Figure 4. Pegasus Transformer

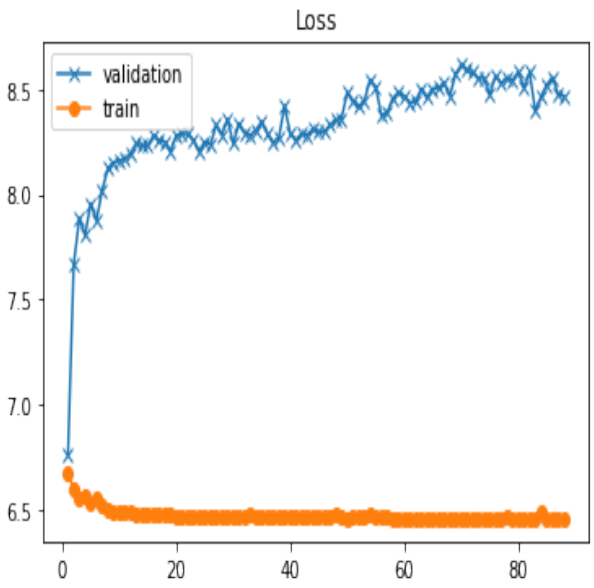

Figure 5. Training and Validation loss of Transformer network

marization: important sentences are removed/masked from an input document and are generated together as one output sequence from the remaining sentences, similar to an extractive summary.

2.4. Dataset

We used the XSUM dataset from the huggingface library to conduct these experiments.It consists of 3 features, namely the document where the entire article is stored, the summary feature where the one sentence summary is present and the id which gives the BBC ID of the article.

3. Experimentation

I conducted 3 experiments in this project.

3.1. transformer Model

In this experiment, i implemented the general transformer model for text summarization. it consists of 1 encoder block and 1 decoder block. We loaded the dataset, and used the glove embeddings of 300B 100D to convert the words into word embedding vectors.We then generated the vocabulary of words that repeat a minimum of 3 times to generate the weights of words that are important and repeat frequently.We then tokenized the words in the document split the data into train and test splits.I then implemented the attention layer and the feed forward layers at the encoder and decoder and a linear and softmax layer at the end of the decoder. The results were not very promising. The training loss was saturating near the 6.5 mark and the validation loss was saturating near the 8.5 mark. This was not an ideal result. A graph of the training and validation loss is shown below

3.1.1 T5 Model

Once the results for the transformer network were obtained i used the T5 pretrained model and fine tuned it for the XSUM dataset. I used the AutoTokenizer method from the huggingface library to tokenize the training and test data. I then used the sequence to sequence model of the huggingface library to design the T5 model to perform the summarization task. The training loss was 2.7 and the validation loss was 2.4. Then i generated the ROGUe scores to evaluate the performance of the model. The rogue scores were rouge1: 28.4834 - rouge2: 8.3339 - rougeL: 22.6874 - rougeLsum: 22.6527.

3.2. Pegasus

Then i fine tuned the pegasus transformer model for text summarization task. I used the pegasus tokenizer to perform the tokenization of the data. The training loss was 0.9709793740868569 and the validation loss was 1.591928243637085.

4. Discussion

During the course of this project , i learned about sequence to sequence models, the underlying architecture what makes sequence to sequence NLP tasks possible, how a transformer works, the different concepts behind the transformer network , the drawbacks of transformers and how transfer learning can be applied to design and implement

our own transformer network from standard pre trained transformer networks. From the above experimentation, it is safe to conclude that pegasus transformer works better than the T5 or the generic transformer models for news summarization purposes.

5. References

[1]: https://jalammar.github.io/illustrated-transformer/
[2]: https://doi.org/10.48550/arXiv.1910.10683
[3]: https://huggingface.co/docs/transformers
[4]:https://huggingface.co/docs/transformers/modeldoc/t5
[5]:https://doi.org/10.48550/arXiv.1706.03762