Minor project report on

Text Summarization Using Transformer Architecture

By

Shrikant M. Patil - 202SP024
Shubham R. Tiwari - 202SP027

Under the Guidance of

Dr. Deepu Vijayasenan

Department of Electronics and Telecommunication, NITK Surathkal

Date of Submission: 9-06-2021

In partial fulfillment for the award of the degree

Master of Technology
in
Signal Processing and Machine Learning

at

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
National Institute of Technology Surathkal, Karnataka

DEPT. OF ELECTRONICS & COMMUNICATION ENGINEERING
National Institute of Technology Surathkal -Mangalore
2020 – 21

CERTIFICATE

This is to certify that the report entitled **"Text Summarization Using Transformer Architecture"** submitted by **Shrikant Madhukar Patil (202SP024) and Shubham Rajeshnath Tiwari (202SP027)**, to the NITK Surathkal in partial fulfillment of the M.Tech. degree in Signal Processing and Machine Learning is a Bonafede record of the work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Dr. Deepu Vijayasenan
Associate Professor
Dept.of ECE
NITK Surathkal
Mangalore

DECLARATION

We **Shrikant Madhukar Patil, Shubham Rajeshnath Tiwari** hereby declare that the project report **"Text Summarization Using Transformer Architecture"** for partial fulfillment of the requirements for the evaluation of Master of Technology to the National institute of technology Surathkal, Mangalore is a Bonafede work done by us under supervision of Professor **Dr. Deepu Vijayasenan.**

This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission.

Mangalore
9-06-2021

Shrikant Madhukar Patil

Shubham Rajeshnath Tiwari

ACKNOWLEDGEMENT

With immense pleasure we are presenting the **"Text Summarization Using Transformer Architecture"** Project report as a part of the curriculum of **"EC788– Minor Project**" under the department of Electronics and Communication Engineering, National Institute of Technology, Karnataka". We wish to thank all people who gave us the unending support. We express my profound thanks to our Professor, **Dr. Deepu Vijayasenan**, and all those who have indirectly guided and helped us in the preparation of this project.

ABSTRACT

Summarization is an important challenge of natural language understanding. It finds its application in many useful area like research paper summarization, news article summarization; in video transcript and many more area. The aim is to produce a condensed representation of an input text that captures the core meaning of the original. Most successful summarization systems utilize extractive approaches that crop out and stitch together portions of the text to produce a condensed version. In contrast, abstractive summarization attempts to produce a bottom-up summary, aspects of which may not appear as part of the original, also it requires language generation capabilities to create summaries containing novel words and phrases not found in the source text. In this project we have used Abstractive Summarization as this provide better results on news article. we haven't defined available novice evaluation metrics. We are using subjective evaluation. Evaluation metrics those which are available are still at research stage for summarization problems.

Keywords: Text Summarization; Natural Language Processing; Information Retrieval; Abstraction; Neural Transformer; Attention; Encoder and Decoder; RNN; LSTM; GRU; Seq2Seq

Contents

List of Figures

1. Introduction

Text summarization is the task of shortening long pieces of text into a concise summary that preserves key information content and overall meaning.
There are two different approaches that are widely used for text summarization:

a. <u>Extractive Summarization:</u>

Extractive summarisation systems form summaries by copying parts of the source text through some measure of importance and then combining those parts/sentences together to render a summary. The importance of the sentence is based on its linguistic and statistical features.

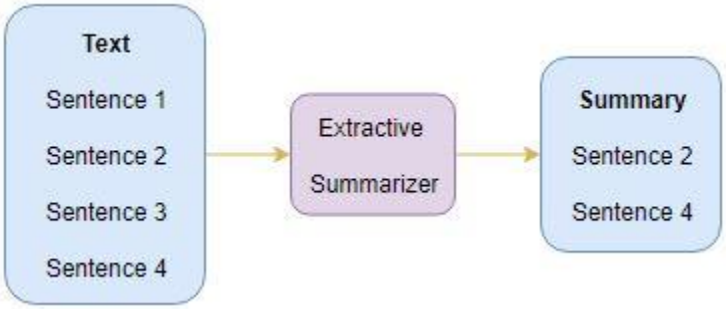

Fig 1.1 Extractive Text Summarization Block Diagram

b. <u>Abstractive Summarization:</u>

Abstractive summarisation systems generate new phrases, possibly rephrasing or using words that were not in the original text. Naturally abstractive approaches are harder. For a perfect abstractive summary, the model has to first truly understand the document and then try to express that understanding in shortform, possibly using new words and phrases. This is much harder than an extractive summary, requiring complex capabilities like generalisation, paraphrasing and incorporating real-world knowledge.

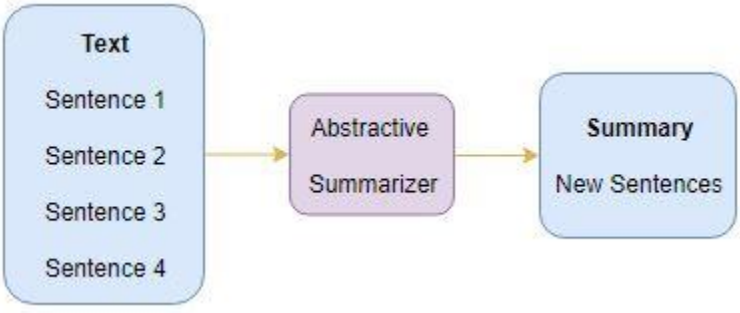

Fig 1.2 Abstractive Text Summarization Block Diagram

To achieve best in class outcomes, we are using Transformers which is having news articles data. Transformer is an architecture for transforming one sequence into another one with the help of two parts (Encoder and Decoder), but it differs from the traditional sequence-to-sequence models because it does not imply any Recurrent Networks (GRU, LSTM, etc.).
The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained.

2. Literature Survey

2.1] Sequence-to-Sequence models:

Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g., sentences in English) to sequences in another domain (e.g., the same sentences translated to French).

This can be used for machine translation or for free-from question answering (generating a natural language answer given a natural language question) in general, it is applicable any time you need to generate text.

There are multiple ways to handle this task:

2.1.1] RNN:

Recurrent Neural Network remembers the past and it's decisions are influenced by what it has learnt from the past. Note: Basic feed forward networks "remember" things too, but they remember things they learnt during training. For example, an image classifier learns what a "1" looks like during training and then uses that knowledge to classify things in production.While RNNs learn similarly while training, in addition, they remember things learnt from prior input(s) while generating output(s). It's part of the network. RNNs can take one or more input vectors and produce one or more output vectors and the output(s) are influenced not just by weights applied on inputs like a regular NN, but also by a "hidden" state vector representing the context based on prior input(s)/output(s). So, the same input could produce a different output depending on previous inputs in the series.

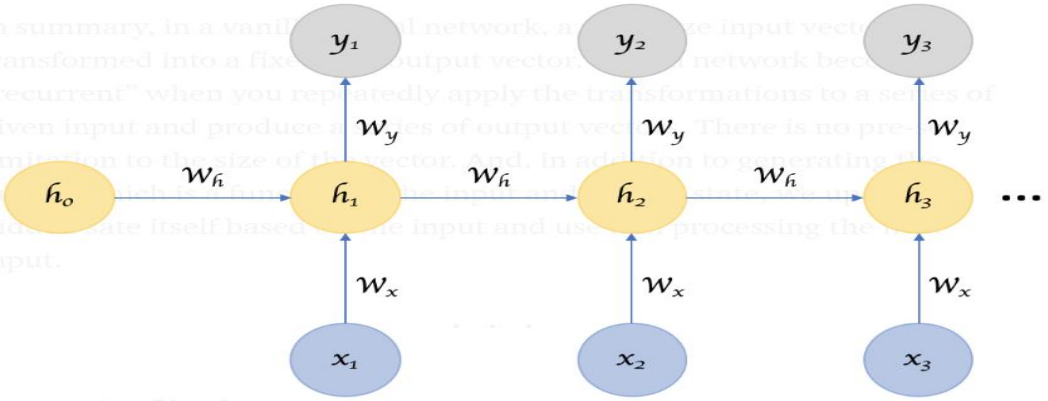

Fig. 2.1 RNN Block Diagram

2.1.2] LSTM:

Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory.

Let's say while watching a video you remember the previous scene or while reading a book you know what happened in the earlier chapter. Similarly RNNs work, they remember the previous information and use it for processing the current input. The shortcoming of RNN is, they can not remember Long term dependencies due to vanishing gradient. LSTMs are explicitly designed to avoid long-term dependency problems.

Long Short Term memory allows us to learn very long range connections in a sequence.Although GRU and LSTM can be used for same purposes but LSTM results are superior and it is complex in computation.

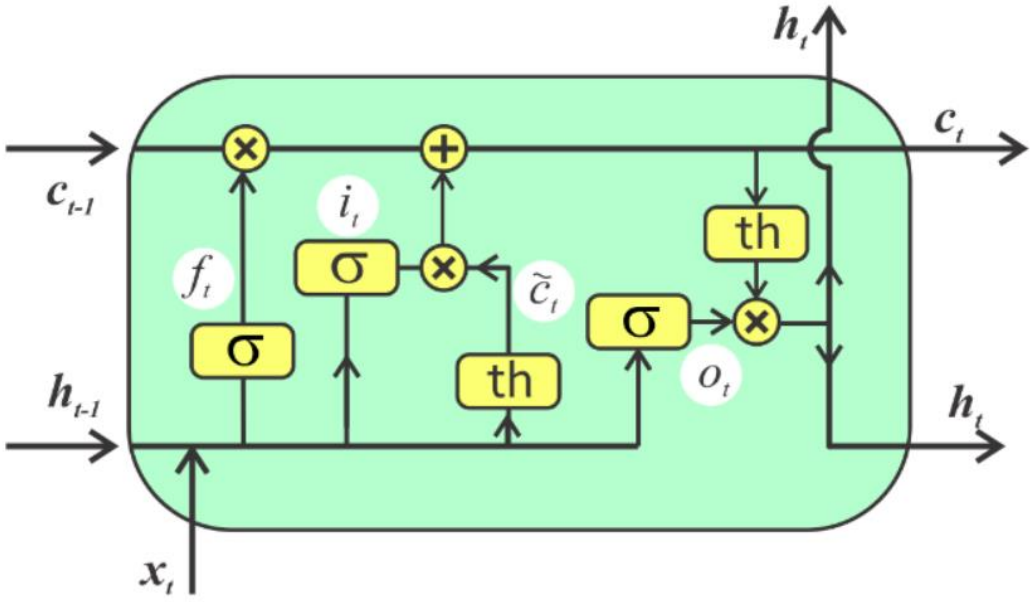

Fig. 2.2 LSTM Block Diagram

where, x=input, h=output, f=forget gate, i=input gate, c=cell update o=output gate

2.1.3] GRU:

GRUs are improved version of standard recurrent neural network. To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

GRU are less complex and requires less computation compared to LSTM also it is more scalable.

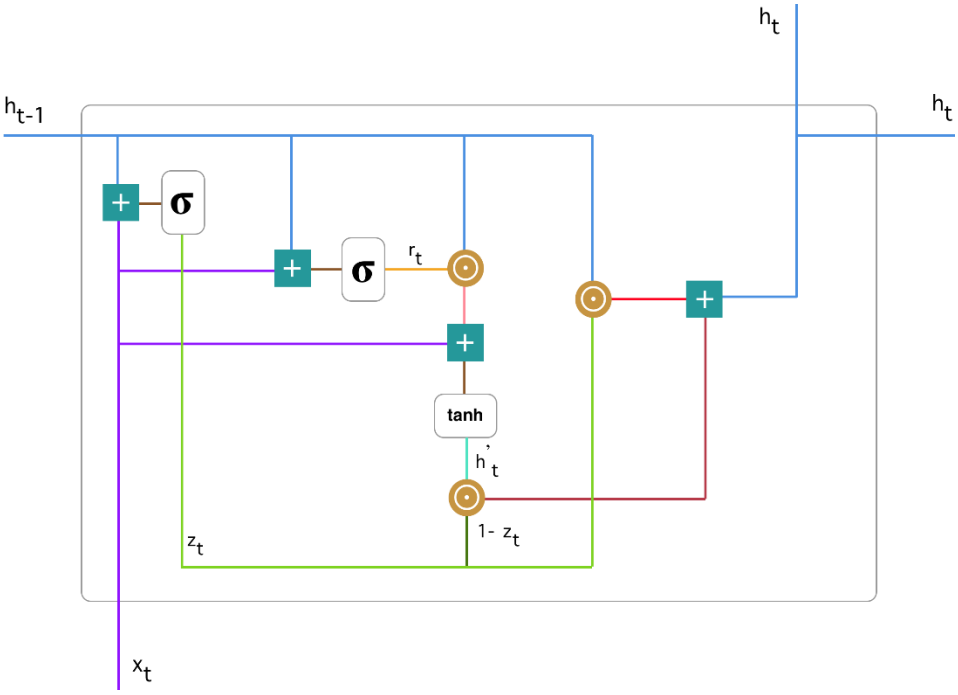

Fig. 2.3 GRU Block Diagram

2.1.4] Encoder-Decoder Models:

The encoder-decoder model is a way of using recurrent neural networks for sequence-to-sequence prediction problems.

It was initially developed for machine translation problems, although it has proven successful at related sequence-to-sequence prediction problems such as text summarization and question answering.

The approach involves two recurrent neural networks, one to encode the input sequence, called the encoder, and a second to decode the encoded input sequence into the target sequence called the decoder.

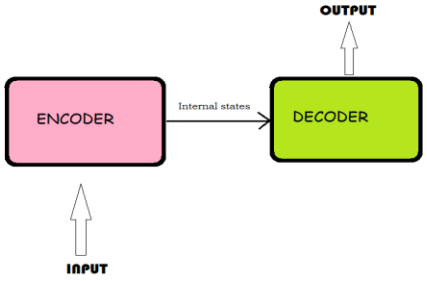

Fig. 2.4 Encoder-Decoder Architecture

It consists of 3 parts: encoder, intermediate vector and decoder.

Encoder-It accepts a single element of the input sequence at each time step, process it, collects information for that element and propagates it forward.

Intermediate vector- This is the final internal state produced from the encoder part of the model. It contains information about the entire input sequence to help the decoder make accurate predictions.

Decoder- given the entire sentence, it predicts an output at each time step.

2.2.5] Transformer:

Like LSTM, Transformer is an architecture for transforming one sequence into another one with the help of two parts (Encoder and Decoder), but it differs from the previously described/existing sequence-to-sequence models because it does not imply any Recurrent Networks (GRU, LSTM, etc.).

Recurrent Networks were, one of the best ways to capture the timely dependencies in sequences. However, the paper proved that an architecture with only attention-mechanisms without any RNN (Recurrent Neural Networks) can improve on the results in translation task and other tasks!

So, what exactly is a Transformer?

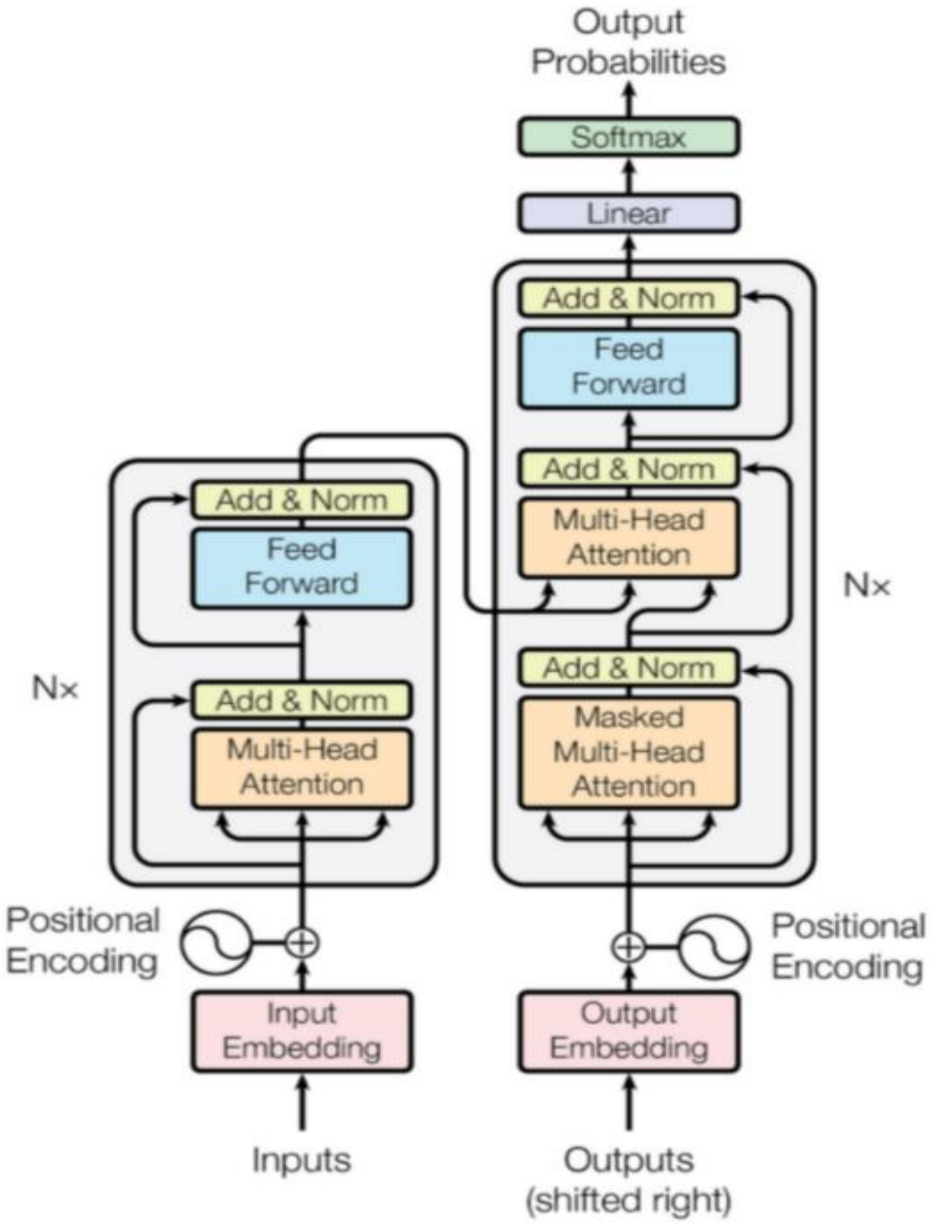

Fig. 2.5 Transformer Architecture[From Original Paper]

The Encoder is on the left and the Decoder is on the right. Both Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times, which is described by Nx in the figure. We see that the modules consist mainly of Multi-Head Attention and Feed Forward layers. The inputs and outputs (target sentences) are first embedded into an n-dimensional space since we cannot use strings directly.

One slight but important part of the model is the positional encoding of the different words. Since we have no recurrent networks that can remember how sequences are fed into a model, we need to somehow give every word/part in our sequence a relative

position since a sequence depends on the order of its elements. These positions are added to the embedded representation (n-dimensional vector) of each word.

2.2] Self Attention Mechanism :

Attention-based mechanism is published at 2015, originally work as Encoder-Decoder structure. Attention is simply a matrix showing relativity of words. Self-Attention is compression of attentions toward itself. The main advantages of Self-Attention Layer compares to previous architectures are:

1. Ability of parallel computing (compares to RNN)
2. No need of deep network to look for long sentence (compares to CNN)

Self-Attention Layer check attention with all words in same sentence at once, which makes it a simple matrix calculation and able to parallel computes on computing units. Also, Self-Attention Layer can use Multi-Head architecture to broaden the vision (associated word's distance in sentence). Self-Attentions shows dominance the fields these years, and brought us to new era.

Multi-Head is features that can create multiple Attentions Matrix in one layer. By simply double the Query, Key and Value combinations in Self-Attention Layer, and independently calculates Attention Matrix. With Multiple-Head, the Self-Attention Layer would create multiple outputs.

Basic structure of self attention :

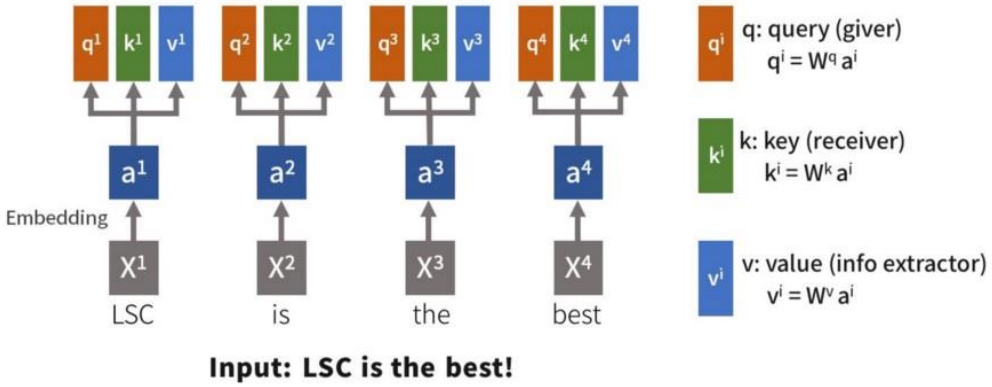

Fig. 2.6 Self-Attention model

2.3] Applications of sequence-to-sequence models:

- Chatbots
- Machine Translation
- Text summary
- Image captioning
- Video understanding

3. Implementation Details

3.1] Dataset:

We have used the Inshorts news data set for our application. Inshorts is a news service that provides short summaries of news from around the web. This dataset contains headlines and summary of news items. It contains 55104 training examples. Sample data is as shown below.

	Headline	Short
0	4 ex-bank officials booked for cheating bank o...	The CBI on Saturday booked four former officia...
1	Supreme Court to go paperless in 6 months: CJI	Chief Justice JS Khehar has said the Supreme C...
2	At least 3 killed, 30 injured in blast in Sylh...	At least three people were killed, including a...
3	Why has Reliance been barred from trading in f...	Mukesh Ambani-led Reliance Industries (RIL) wa...
4	Was stopped from entering my own studio at Tim...	TV news anchor Arnab Goswami has said he was t...

Fig 3.1 Data Set Example

3.2] Positional Encoder:

Since this model doesn't contain any recurrence or convolution, positional encoding is added to give the model some information about the relative position of the words in the sentence.

The positional encoding vector is added to the embedding vector. Embeddings represent a token in a d-dimensional space where tokens with similar meaning will be closer to each other. But the embeddings do not encode the relative position of words in a sentence. So after adding the positional encoding, words will be closer to each other based on the *similarity of their meaning and their position in the sentence,* in the d-dimensional space.

The formula for calculating the positional encoding is as follows:

PE(pos,2i)=sin(pos/10000^[2i/dmodel])
PE(pos,2i+1)=cos(pos/10000^[2i/dmodel])

3.3] Scalar Dot Product Attention:

The attention function used by the transformer takes three inputs: Q (query), K (key), V (value). The equation used to calculate the attention weights is:

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

The dot-product attention is scaled by a factor of square root of the depth. This is done because for large values of depth, the dot product grows large in magnitude pushing the softmax function where it has small gradients resulting in a very hard softmax.

The mask is multiplied with -1e9 (close to negative infinity). This is done because the mask is summed with the scaled matrix multiplication of Q and K and is applied immediately before a softmax. The goal is to zero out these cells, and large negative inputs to softmax are near zero in the output.

3.4] Multi-head Attention:

Each multi-head attention block gets three inputs; Q (query), K (key), V (value). These are put through linear (Dense) layers and split up into multiple heads.

The scaled_dot_product_attention defined above is applied to each head (broadcasted for efficiency). An appropriate mask must be used in the attention step. The attention output for each head is then concatenated (using tf.transpose, and tf.reshape) and put through a final Dense layer.

Instead of one single attention head, Q, K, and V are split into multiple heads because it allows the model to jointly attend to information at different positions from different representational spaces. After the split each head has a reduced dimensionality, so the total computation cost is the same as a single head attention with full dimensionality.

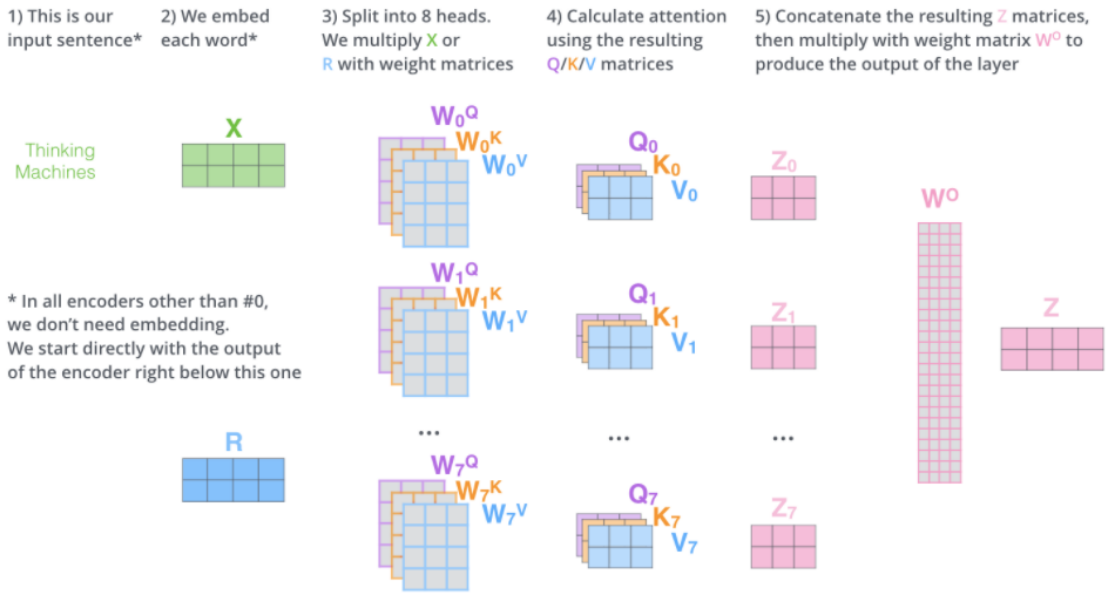

Fig. 3.2 Multi-head Attention [Image Credit: https://jalammar.github.io/illustrated-transformer/]

3.5] Encoder Layer(Individual Encoder):

Each encoder layer consists of sublayers:

1. Multi-head attention (with padding mask)
2. Point wise feed forward networks.

Each of these sublayers has a residual connection around it followed by a layer normalization. Residual connections help in avoiding the vanishing gradient problem in deep networks.

The output of each sublayer is LayerNorm(x + Sublayer(x)). The normalization is done on the d_model (last) axis. There are N encoder layers in the transformer.

3.6] Decoder Layer(Individual Decoder):

Each decoder layer consists of sublayers:

1. Masked multi-head attention (with look ahead mask and padding mask)
2. Multi-head attention (with padding mask). V (value) and K (key) receive the *encoder output* as inputs. Q (query) receives the *output from the masked multi-head attention sublayer*.
3. Point wise feed forward networks

Each of these sublayers has a residual connection around it followed by a layer normalization. The output of each sublayer is LayerNorm(x + Sublayer(x)). The normalization is done on the d_model (last) axis.

There are N decoder layers in the transformer.

As Q receives the output from decoder's first attention block, and K receives the encoder output, the attention weights represent the importance given to the decoder's input based on the encoder's output. In other words, the decoder predicts the next word by looking at the encoder output and self-attending to its own output. See the demonstration above in the scaled dot product attention section.

3.7] Encoder(stack of encoder layers):

The Encoder consists of:

1. Input Embedding
2. Positional Encoding
3. N encoder layers

The input is put through an embedding which is summed with the positional encoding. The output of this summation is the input to the encoder layers. The output of the encoder is the input to the decoder.

3.8] Decoder(stack of Decoder layers):

The Decoder consists of:

1. Output Embedding
2. Positional Encoding
3. N decoder layers

The target is put through an embedding which is summed with the positional encoding. The output of this summation is the input to the decoder layers. The output of the decoder is the input to the final linear layer.

3.9] Model Hyperparameters:

num_layers = 6
d_model = 128
dff = 512
num_heads = 8
EPOCHS = 40

3.10] Loss Function:

Sparse Categorical cross entropy is used as loss function from TensorFlow. We already know the categorical cross entropy function. The only difference between sparse categorical cross entropy and categorical cross entropy is the format of true labels. When we have a single-label, multi-class classification problem, the labels are mutually exclusive for each data, meaning each data entry can only belong to one class. Then we can represent y_true using one-hot embeddings.

3.11] Optimizer:

Adam optimizer is used with a custom learning rate scheduler according to the formula in the paper.

Learning Rate=[d^(−0.5model)]∗[min(step_num^(−0.5),step_num*warmup_steps^(−1.5))]

3.12] Masking:

Padding Mask: Mask all the pad tokens in the batch of sequence. It ensures that the model does not treat padding as the input. The mask indicates where pad value 0 is present: it outputs a 1 at those locations, and a 0 otherwise.

Look Ahead Mask: The transformer is an auto-regressive model: it makes predictions one part at a time, and uses its output so far to decide what to do next. As the transformer predicts each word, *self-attention* allows it to look at the previous words in the input sequence to better predict the next word. To prevent the model from peeking at the expected output the model uses a look-ahead mask.

The look-ahead mask is used to mask the future tokens in a sequence. In other words, the mask indicates which entries should not be used. This means that to predict the third word, only the first and second word will be used. Similarly, to predict the fourth word, only the first, second and the third word will be used and so on

3.13] Gradient Tape Training:

Gradient Tape is a brand-new function in TensorFlow 2.0. It is based on the idea of Automatic differentiation **(**also called computational differentiation**)** refers to a set of techniques that can automatically compute the derivative of a function by repeatedly applying the chain rule.

3.14] Evaluation:

The following steps are used for evaluation:

- Encode the input sentence using the Portuguese tokenizer (tokenizers.pt). This is the encoder input.
- The decoder input is initialized to the [START] token.
- Calculate the padding masks and the look ahead masks.
- The decoder then outputs the predictions by looking at the encoder output and its own output (self-attention).
- The model makes predictions of the next word for each word in the output. Most of these are redundant. Use the predictions from the last word.
- Concatenate the predicted word to the decoder input and pass it to the decoder.
- In this approach, the decoder predicts the next word based on the previous words it predicted.

4. Results

Following are some of summaries we got.

1. **Original text:** The Supreme Court recently reminded the Centre that Aadhaar cannot be made mandatory for any services. The apex court also ordered the Centre to remove its condition of making Aadhaar mandatory in scholarship schemes for students. "The Aadhaar card Scheme is purely voluntary and cannot be made mandatory till the matter is finally decided by this Court," the SC added.
   predicted summary: aadhaar card to tackle a minor cuts
   True summary: Aadhaar cannot be mandatory, SC reminds govt

2. **Original text:** Researchers at the University of Stuttgart have built wall-climbing mini robots that work together to create architecture from carbon fibre. The robots carry carbon fibre thread spools that they pass back and forth after affixing to points on a wall. The researchers are planning to increase the number of robots, allowing them to attach fibre to ceilings and curved walls.
   predicted summary: california could install tree near lights
   True summary: Robots create architecture with carbon fibre

3. **Original text:** Indira Gandhi has been the only woman till date to have presented the Union Budget of India in 1970-71. This came after Indira Gandhi, the then Prime Minister, took over the Finance portfolio after Morarji Desai resigned as the Minister of Finance. So far, she has been the only woman Finance Minister of India.
   predicted summary: it dept detects demonetisation in india 39 s dream
   True summary: Indira Gandhi only woman to have presented the budget

4. **Original text:** Gangster-turned-politician Mukhtar Ansari has won from the Mau constituency in Uttar Pradesh after polling 96,793 votes, defeating the nearest candidate by over 8,000 votes. Ansari, who was the sitting MLA from the constituency, had allied with the Mayawati-led Bahujan Samaj Party before the elections. Ansari has been accused of murdering a BJP MLA.
   predicted summary: former miss universe passes away at 72 000 yr old
   True summary: Gangster-turned-politician Mukhtar Ansari wins by 8000 votes

5. Compilation of Work

The python code for the model has been compiled as a python notebook and can be successfully compiled on Google Collaboratory. The notebook can be accessed HERE. The files required for running this notebook has been compiled into a publicly accessible Google Drive directory, which can be found HERE.

6. Conclusion

In this work, we presented the Summarizer using Transformer Architecture which is the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

Summaries obtained are really meaningful but somewhat out of context to the input message. Data set used has the more generalized and very short summaries which may be the cause for this.
It is seen that for Summarization tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers.

We are excited about the future of attention-based models in summarization tasks and plan to apply them to other data sets.
.

References

[1] *Attention Is All You Need,* Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Lion Jones, Aidan N. Geomez, Lukasz Kaiser, Illia Polosukhin, (Google Brain and University of Toronto team), 6 Dec 2017.

[2] *The Illustrated Transformer*, by Jay Alammar, Visualizing Machine Learning Concept One at a Time Blog.

[3] *Transformer Model for Language Understanding*, by TensorFlow.org Tutorials.