Machine Translation using Transformers

1 Project Overview

In this project, we are seeing how one language can be converted to another language using network models. Specifically, we need to know how a sentence is interpreted by a machine, how data is fed to train the computer and what are the requirements needed for the process of translation. Choosing an appropriate network model is a challenging task. Machine translation can be achieved through different network models. We are exploring about Encoder-Decoder Network model along with Attention mechanism on the whole called Transformer model. This project work includes Seq-to-Seq Attention model, Gated Recurrent Unit (GRU). Basic implementation of Machine translation involves Encoder-Decoder model. It was found that, for long sentences, the model was not able to keep the meaning of each word till the end. As a result, a refined model was built called Network with Long-Short Term (LSTM) model in which gates are introduced to make the computer when to have a word's meaning in memory, when to update its memory, when to forget and so on. Again, researchers found a drawback in this. All the refinement doesn't say the computer where to focus or find the important terms, given a sentence for translation as how we humans do. Hence came the Attention Mechanism addition with Encoder-Decoder model which will be discussed part by part in this work.

2 Tools

Let's see what are the tools used in achieving this research work.

2.1 Jupyter Notebook

The Jupyter Notebook is an open source web application that is used to create and share documents that contain live code, equations, visualizations, and text. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R.

3 Problem Formulation

In this section, we will be discussing about the terms involved in this work.

3.1 Seq-to-Seq Attention

In this application, our inputs and outputs are in the form of words. Computers are comfortable with numbers compared to words. So, researchers started thinking about how to convert or map words to numbers. Each source language should be mapped with its corresponding target language so that computer knows the correct answer for the question. This is achieved by Seq-to-Seq model. Sequence to Sequence model maps a source sequence to target sequence. Source sequence is input language to the machine translation system and target sequence is the output language. Seq-to-Seq model are built using Encoders and Decoders.

Encoder : reads the input sequence of words from source language and encodes that information into a real valued vectors also known as the hidden state or thought vector or context vector. Thought vector encodes the ‘‘meaning’’ of the input sequence into a single vector. The

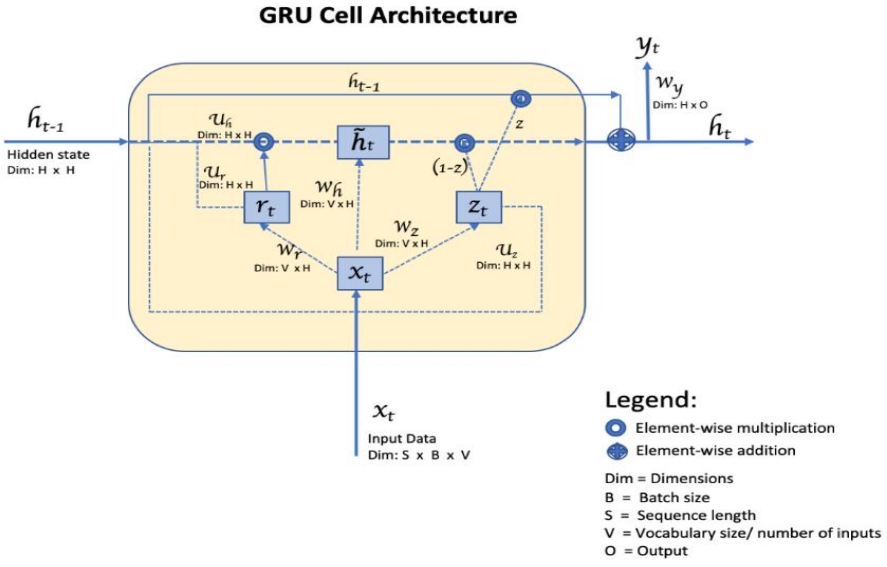
encoder outputs are discarded and only the hidden or internal states are passed as initial inputs to the decoder.

Decoder: takes the thought vecto r from the encoder as an input along with the start-of-string <START> token as the initial input to produce an output sequence.

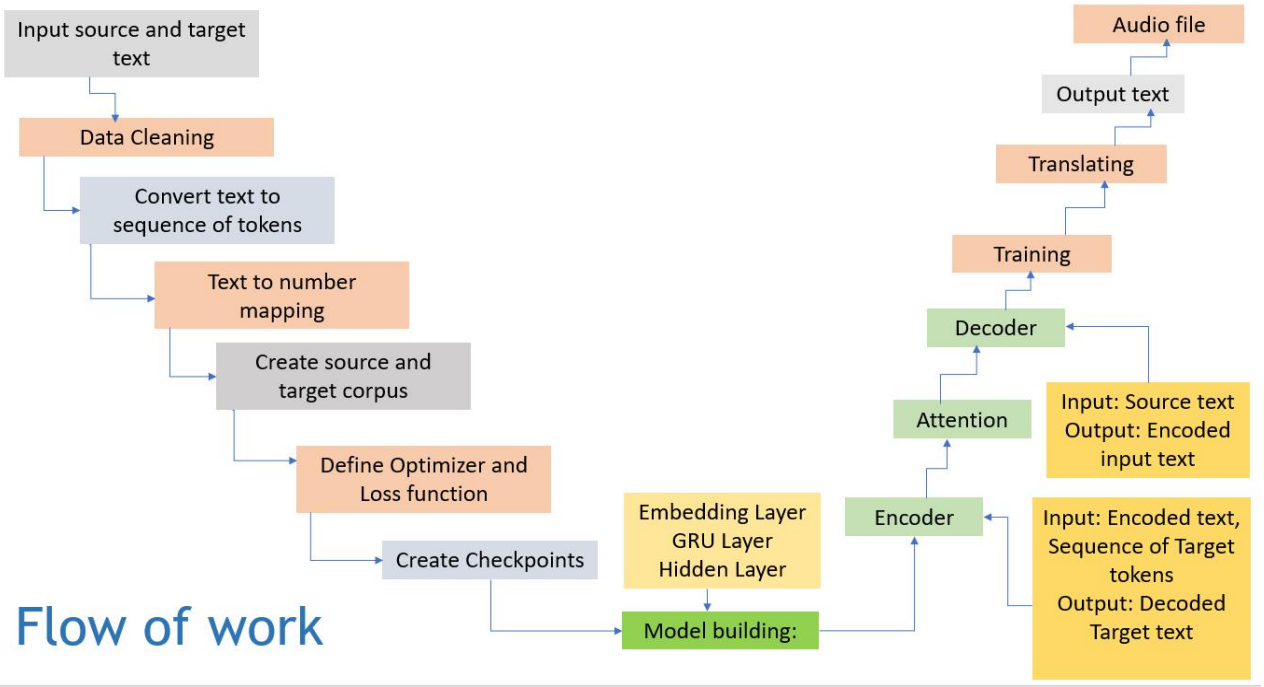
Attention: The basic idea of Attention mechanism is to avoid attempting to learn a single vector representation for each sentence, instead, it pays attention to specific input vectors of the input sequence based on the attention weights.

3.2 Gated Recurrent Unit

It is a type of Recurrent Neural Network that addresses the issue of long term dependencies which can lead to vanishing gradients larger vanilla RNN networks experience. GRUs address this issue by storing ‘‘memory’’ from the previous time point to help inform the network for future predictions.



The flow of the project in simple terms is, given a input sentence – feed to a machine translation model – get translated output sentence. Clear flow of this project work is clearly depicted in the figure below.



Each term in the above diagram will be explained in detail in the next section which is the implementation section.

4 Implementation

The work has been divided into many parts. Each part will be clearly explained in this section.

4.1 Data Cleaning

The input sentence we get will not be a perfect one with no mistakes like grammar, spelling, punctuations etc., all the time. When we work with real life data, definitely we will come across erroneous data. Hence, it is always advised to clean or It includes removal of

punctuations, and extra spaces, replacing numbers to words and so on. Once done with cleaning, <start> and <end> are added in the starting and ending of the sentence to make the computer know where a sentence starts and end.

4.2 Text to number mapping

Each text in converted to a sequence of tokens with the help of tensorflow tokenizer. The mappings are done based on the corpus created with unique words in the whole source and target text. Sequence of tokens are done for both source and target text. Padding is done for all the sentences with the sentence with long length as base as while training all the datapoints should be of equal length. Now all the words are converted to computers comfortable form which is numbers. Training and testing data are formulated.

4.3 Model building

Step-by-step building of the model will be explained in this section.

Encoder: Input text are sent to an Embedding layer. An embedding is a mapping of a discrete — categorical — variable to a vector of continuous numbers. In the context of neural networks, embeddings are low-dimensional, learned continuous vector representations of discrete variables. Neural network embeddings are useful because they can reduce the dimensionality of categorical variables and meaningfully represent categories in the transformed space. In simple words, the Encoder takes the input as the source tokens, passes them to an embedding layer for the dense representation of the vector, which is then passed to GRU.

Attention layer: All hidden states of the Encoder (forward and backward) and the Decoder are used to generate the context vector. The attention mechanism aligns the input and output sequences, with

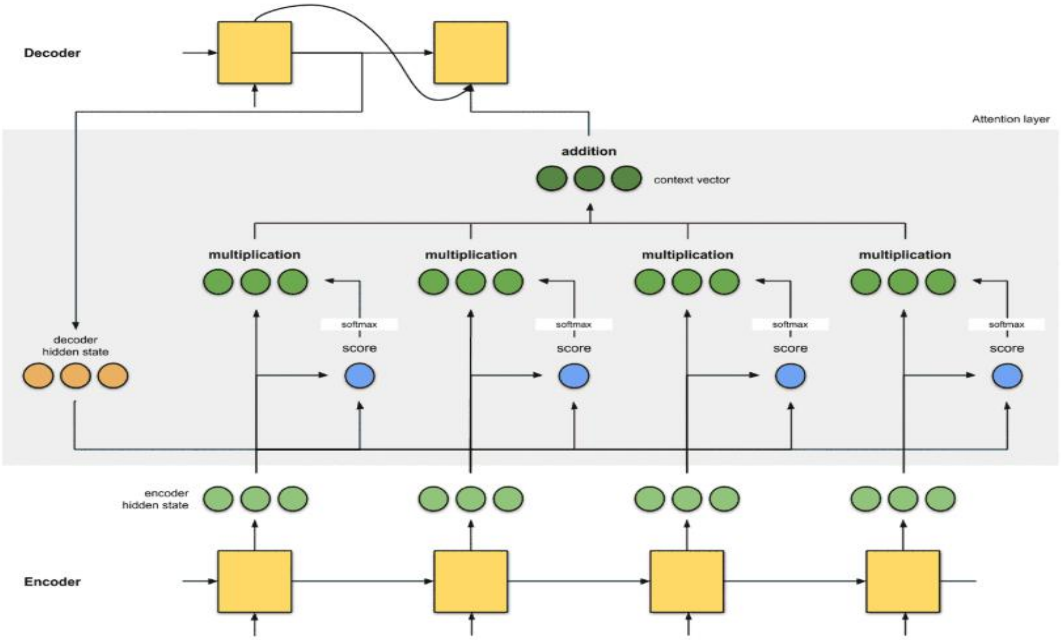
an alignment score parameterized by a feed-forward network. It helps to pay attention to the most relevant information in the source sequence. Seq2Seq Attention model predicts a target word based on the context vectors associated with the source position and the previously generated target words. This layer mainly consists of:

- **Alignment score:** Alignment means matching segments of original text with their corresponding segments of the translation. That is, when focusing on a single word to translate, what are all the other words that are aligned to it should also be seen. That is recreated in the model through this layer.
- **Attention Weights:** This tells the importance of each word given a sentence. It is calculated based on the Embedding layer and hidden layer outputs.
- **Context vector:** It is the sum of Attention weights and Encoder output.

This layer is mainly made up of Dense layers. For calculating scores tanh and softmax functions are used. This layer returns the context-vector and attention-weights.

Decoder: It consists of an embedding layer, a GRU layer, and a fully connected layer. To predict the target word decoder uses, context vector, decoder's output from the previous time step and previous decoder's hidden state.

The figure below shows overall view of the whole model with attention.



4.4 Optimizers and Loss

- Adams optimizer is used in this project work. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.
- Sparse Categorical cross-entropy loss is used. It is refined version of categorical cross-entropy. Entropy tells how pure the model. Cross-entropy is like cross-checking of our results. Categorical cross-entropy is used for categorical variable. The only difference between sparse categorical cross entropy and categorical cross entropy is the format of true labels.

With all these, our model is ready to train and test the results with these optimizers and loss functions.

5 Testing and Evaluation

In this section we will see the overall flow of what happens when a testing input comes.

- It will be sent to a pre-processing function which gives out a cleaned or perfect sentence.
- Perfect sentence is fed to tokenizer function where the text is converted to sequence of tokens.
- Padding is done for the tokenized sentence.
- Implementation of models are done with the help of tensorflow as it yields better results. Hence, the padded sentence is converted to tensors.
- The iteration starts with <start> tag.
- The tensor input is fed to encoder which encodes the text with help of already trained embedding and hidden layers.
- The encoded sentence then enters the decoder part. It returns the prediction id which is compared with trained results.
- With the prediction id, its corresponding sequence of target token is returned which is given by the target tokenizer.
- The resultant word is fed to the model to predict the next word till <end> tag is reached.
- The final translated sentence is converted into audio format for ease of use.
- In this work, English-to-French translation is done.

References

1. Daniel Jurafsky and James H. Martin, ‘‘Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition’’, Prentice Hall, 2009.
2. Blogs related to Neural Machine Translation using Encoder-Decoder architecture and Illustrated Attention.