Neural Machine Translation Using Transformers Project Report

By

Akhil Reddy Dooliganti

Paper Link :- https://arxiv.org/pdf/1706.03762.pdf

QR Code

Dataset :- https://huggingface.co/datasets/wmt14

Code :- https://github.com/AkhilD1/Neural-Machine-Translation-using-Tranformers

Problem Definition:-

Natural Language Processing (NLP) is an aspect of Artificial Intelligence that helps computers understand, interpret, and utilize human languages. NLP allows computers to communicate with people, using a human language. Natural Language Processing also provides computers with the ability to read text, hear speech, and interpret it.NLP breaks down language into shorter, more basic pieces, called tokens(words, periods, etc.), and attempts to understand the relationships of the tokens.

The problem we are trying to implement over the course of this project is based on the Neural Machine Translation.One of the earliest goals for computers was the automatic translation of text from one language to another. Machine translation is the process of automatically translating content from one language (English) to another (German) without any human interference. Machine Translation has been a revelation right from the beginning, but it has become more powerful.With the advancement of different architectures such as Transformers we are able to translate a sentence from one language to another language by retaining the context and meaning of the sentence more efficiently. Real Time Machine Translation applications we use on a day to day basis offer Text to Text , Text to Speech , Speech to Text , Speech to Speech , Image of words to Text translations. Some of its applications would be Google Translate, Facebook Translate. Machine translation is impacting the world by providing the users a tool to communicate, express their ideas and share with those around them with any person around the

world without any hassle and also the users with disabilities by impacting the way they communicate rely on this assistive technology to express their ideas with their emotion.Neural Machine Translation immensely helps the users of the Social Media Platforms and Media websites such as Instagram, Twitter , and The Wire. to convert tweets or posts and articles written in another language.

Project Objectives:-

The main objective of the project is to build a neural machine translation application (English - German) essentially English to German using Transformers (Deep Learning).

- For the Neural machine translation (Text- Text) system input is Text, so first, we need to preprocess the text such as removing special characters and punctuations are also included as special markers. Then generate phoneme sequences for text which are used for input for the model training.
- Preprocess the data and create tokens and build vocabulary.
- By using Huggingface tokenizer we convert the text into feature vectors.
- Design and implement the Transformer architecture to perform the neural machine translation task.
- Finally, I built a web-app using StreamLit for this Neural Machine Translation system.

Analysis:-

Dataset & Preprocessing:-

- For loading the NMT-14 English-German dataset we will use the HuggingFace Datasets library which will help us download and generally manipulate the dataset much easier.

Create Tokenizer:-

- For creating the tokenizer we use the HuggingFace Tokenizers library.
- In the paper they have used a single BPE tokenizer trained on sentences from both languages. A word level tokenizer is selected here instead, as I found that for my simpler training configurations it worked better.
- Also note that if choosing a word-level tokenizer the vocabulary size (VOCAB_SIZE param) needs to be increased compared to the 37000 word vocabulary for the BPE mentioned in the paper.
- The process of creating a tokenizer boils down to selecting a tokenization model and customizing its components. For more info on the components refer to the hugging face docs .

SPECIAL TOKENS :-

- [BOS], [EOS], [PAD] and [UNK] tokens are also added.
- [BOS] token is useful in the decoder input to signalize the beginning of a sentence, remember that the transformer decoder predicts the next word in the sequence by looking at the encoder representation and the decoder input upto the current timestep. Therefore for predicting the first word it only sees the [BOS].
- [EOS] token signals the end of the sequence and therefore the end of decoding when inferencing.
- [UNK] is to represent an unknown word while translating.

Data Preprocessing :-

- Now we tokenize the dataset. After tokenization we filter long sentences (so future batches can fit into GPU memory), perform a train/test split and sort each split by length.
- Sorting is done to aid batching of similar length sequences inside the batch sampler which speeds up training and also reduces the total count of [PAD] tokens needed.We made sure [PAD]'s id is 0 so that's why we pad with 0 here.
- We use pytorch's pad_sequence function.

Create a Batch Sampler for sampling sequences of similar lengths:-

- Batch sampler ensures that batches contain sequences of similar lengths as explained before. It iteratively returns indices of samples that should go together in a batch. We already sorted the splits by length so here we just chunk indices of sorted elements in order. We also care to shuffle the batches here.

Create DataLoaders:-

- Next, Data Loaders are constructed with the described collate and batch sampling policies.
- Design and implement the Transformer architecture according to this paper. For this project.

Transformer TTS Model Analysis:

Transformer is a sequence to sequence network, based solely on attention mechanisms and dispensing with recurrent and convolution layers entirely.

Positional Encoding:-
Positional encoding helps encoding of 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. So that the mapping between the encoder and the decoder will happen correctly.

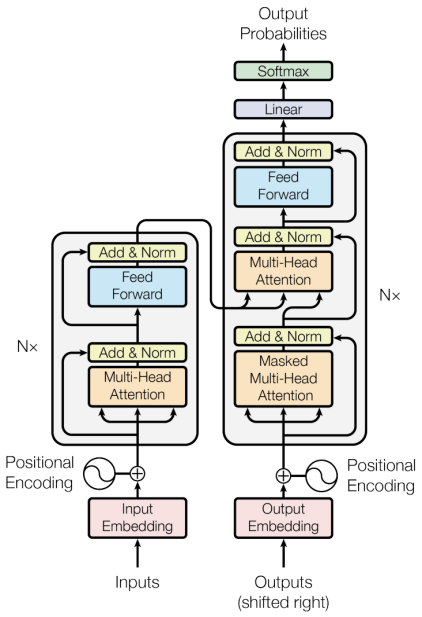
Encoder and Decoder:-
The encoder's inputs first flow through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word.

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.

The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence.

By using transformers, it parallelizes training and Inference tasks and also it helps in learning and handling long term dependencies very well.

- **Encoder** :- 3-layer CNN
- **Embedding Layer :-** 2-layer Fully connected network
- **Positional Encoding :-** A fully-connected layer.
- **The Multi-Head Attention Layer-Self-Attention :-** a fully-connected layer, predicts the stop token for each frame.Self-attention is the method the Transformer uses to make the "understanding" of other relevant words into the one we're currently processing.
- **Multi-Head Attention** :- This improves the performance of the attention layer as it expands the model's ability to focus on different positions.
- **Decoder :-** In the decoder, the self-attention layer is only allowed to attend the earlier positions in the output sequence and also from the encoder. This is done by masking future positions.



Discussion:-

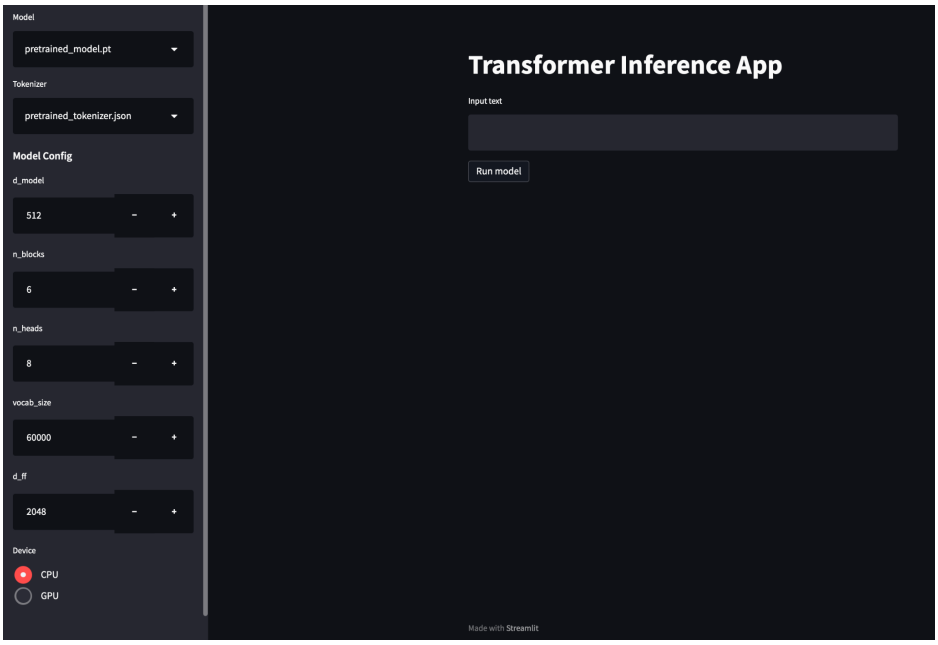
- After using transformers in the model architecture, the model is able to learn attention between encoder and decoder better than the RNN's structures.
- Model is performing well on most of the words we use colloquially in daily life and in some cases our model is not able to recognize difficult words due to limited vocabulary size taken to train the model.
- The training speed is better compared to the previous models because of the transformers architecture, which enables parallel training, and the capability of learning long-distance relationships in the text.

Results:-

- For this project, I have used the pretrained model, because training the model from scratch required a lot of computation power and training time taken by this model is approximately 25 hr(according to the paper).
- **Generated Machine Translations :-**
  - **Text 1:-** I have a dream and its german translation
  - **Text 2:-** I am a good boy and its german translation

- I have compared my model predictions with the Google translate predictions, the results generated by my predictions have not been very accurate on every occasion but it has been decent.

- The below web-app screenshot for the NMT model where the english input text is prompted , where we can choose the Model Config where we can select the vocabulary size. It will generate the German translation.



<u>Evaluation and Reflection</u>:-

- The german translations generated by the model have not always been accurate but it's decent, when the generated translations are compared with the Google translate , Google translate is providing more contextual and accurate translation because they use the current State of the art models and the training data used by the google translate is far superior to that of our data.
- And due to the limited computation power we are not able to train the model so we have to use a pre-trained model.
- Although the transformer has enabled parallel training, the model suffers from slow inference.
- Machine translation applications will eliminate the language barriers and help the people to communicate with people from any parts of the world and will make them feel inclusive.

References:-

[1] Dzmitry Bahdanau, KyungHyun Cho , Yoshua Bengio. Neural Machine Translation
By jointly learning to align and translate (2015)
[2]Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading (2016).
[3] The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time.” Jay Alammar.
[4] Towards Data Science article on Neural Machine Translation, Quinn Lanners.
[5] https://github.com/bkoch4142/attention-is-all-you-need-paper#weights-and-biases-logs