Translation MODEL: NN and DL Project

By Ved Anand (12021002018015)

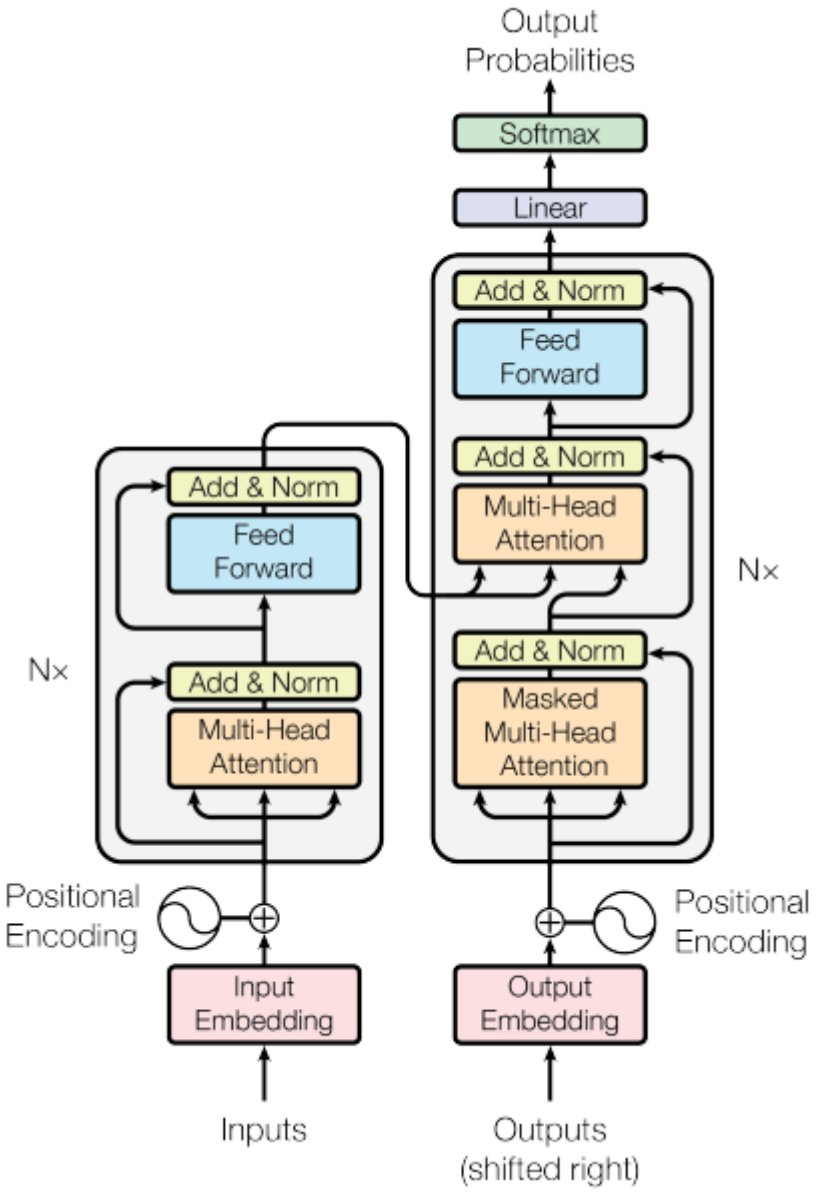
- ## Dataset Used: Multi-30K Dataset

- We are going to train our model on the Multi 30k data set from English to German.
- This data set already comes in a tokenized format for German, English, French.
- So, this is a very nice data set.
- This can be taken as a toy data set to implement and see how good our model works.
- Dataset Link: https://github.com/multi30k/dataset

- ## Model Used: Standard Transformer Model

- For the model, we'll use the Standard Transformer Model architecture.
- For a Good tutorial series on implementing seek to seek architectures from scratch in bluepython, one can refer to: Ben Trevett Implementation
- So, if one wants to understand the more minute details behind the Transformer architecture, they can refer to above link.

- ## Decoding Colab Notebook File: TranslationModel.ipynb



➢ Importing the Dataset (Multi-30K):

- So, our first step would be to get the data set.
- Get the data set from here.
- And then we install a couple of python modules.

➢ Installing Required Python Modules:

- One is WandB for login into wandb.ai,
- And Sacremoses for tokenizing text,
- And fair-seq for training and evaluating our model.

➢ To use W and B, Creating and Logging in Account

- If you want to use wnb, you have to create your account and then give the command specified in Notebook.
- Authorize your login Credentials by putting your API key from WandB Profile.
- API key for this project: b910992fda2c68afc731d31d0623442d3bbe5df0
- And it directly shows up to your particular project, all the parameters, all the values that it keeps track of during the training process.

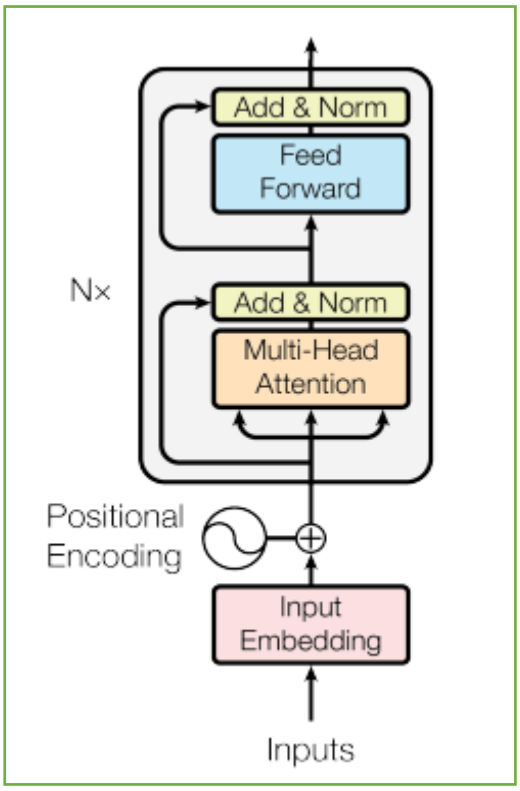
➢ Pre-process and Binarize to build Vocabularies:

- Aftr that we have logged into our wnb account, the first step is the preprocessing step.
- This step takes all the tokenized data in our data set, which is a multi 30k data set, which is there in the task1 under tokenized folder.
- It takes all the text and then try to learn vocabulary from it and then binarize the data so that it becomes faster during the training to batch them and use them.
- The parameters that we are passing for the fair-seq preprocess is the source language that we want to translate and the target language that we want to translate it to.
- The train prefix, valid prefix and test prefix are the entire path of the train data set, validation data set and the test data set except the last extensions for the language.
- For the test preference, we are using the “2018 flickr Data set”.
- The destination directory for all the vocabulary learnt by the fair-seq preprocess and the binary data set are stored in data bin.
- Threshold SRC and TGT what they specify is: The minimum number of times (minimum threshold) a token repeats so that it is added into the vocabulary.

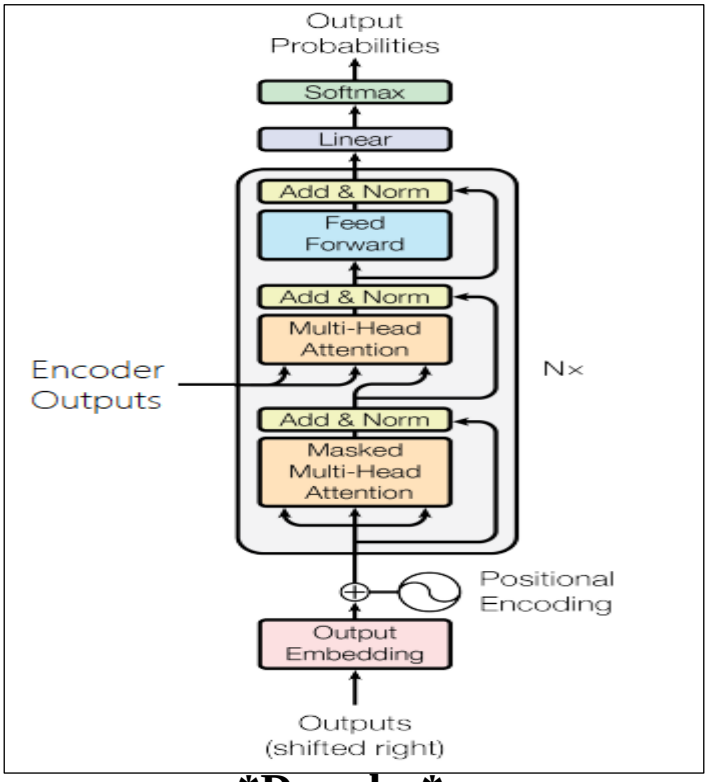
➢ Training the model:

- In the train step, we have specified a lot of parameters. Let's discuss them one by one.
- So, most of the hyperparameters are chosen or directly taken from Ben Trevett tutorial.
- We will try explain what they do with the help of the fairseq documentation.
- Dropout, attention-dropout and activation-dropout are three types of dropouts in the entire transformer architecture.
- Attention dropout is done after the data passes through the attention layer and the other one in the above diagram in the feed forward network.
- There are two feed forwards, and there are three attentions that you can see in above diagram and then the activation dropout comes.
- In the below picture architecture, the feed forward part block is made of two layers.
- The size of the hidden layer in the middle the second layer is the hyperparameter that we are setting here to 512.

- Number of encoder layers is the repetition of each encoder layer, i.e., all these encoder layers are stacked on top of each other and all the decoder layers are stacked on top of each other.
- That is what this Nx in the picture signifies.
- So, we have used three encoder layers, and similarly, three decoder layers and it is important that "The number of encoder layers = The Number of decoder layers".



Encoder



Decoder

- And the rest parameters/commands are referred from [fairseq documentation]():

➢ Evaluating the Model:

Additional command-line arguments

`--dropout`	dropout probability
`--encoder-embed-dim`	encoder embedding dimension
`--encoder-embed-path`	path to pre-trained encoder embedding
`--encoder-layers`	encoder layers [(dim, kernel_size), ...]
`--decoder-embed-dim`	decoder embedding dimension
`--decoder-embed-path`	path to pre-trained decoder embedding
`--decoder-layers`	decoder layers [(dim, kernel_size), ...]
`--decoder-out-embed-dim`	decoder output embedding dimension
`--decoder-attention`	decoder attention [True, ...]
`--share-input-output-embed`	share input and output embeddings (requires –decoder-out-embed-dim and –decoder-embed-dim to be equal) Default: False

- ‘fairseq-generate’ is a command within the fairseq Library, which can be used to generate translations.
- After running the translations of our model over the test-data set, at the end we would get a Bleu Score.
- We are running the fairseq-generate on the checkpoint_best.
- Checkpoint_best is the checkpoint where the validation loss is the lowest.
- On the bleu score we are getting 25.99 which is actually a very decent Bleu Score if compared to the other state of the art models.
- Being a NOOB, our MODEL is not fully optimized for all our hyperparameters to fit the data perfectly.
- Given that the highest bleu score is 38.4 in 2020, we have still managed to reach 25.99 with very less amount of code. So, this even is a great plus point for US.
- Meanwhile we can improve the bleu score further using a few more techniques like:
  - Beam size is a technique which is used during the decoding of the transformer model.
  - Increasing the Beam size from 5 to 10, **BLEU SCORE increased from 25.99 to 26.17.**
  - Another great technique that we have that is **Averaging Checkpoints**.
  - What it does is, it takes all the Last five checkpoints and take an average of it and save it to a different file and in most of the time the average checkpoint acts like an ensemble of all the last five models, and hence provides better results usually.
  - After performing it, we got a **BLEU SCORE RISE from 26.17 to 26.74** for beam of 5.
  - **Our Final BLEU SCORE is = 26.74 and Highest in World is 49.3.**

➢ Exporting the Model:

- In order to export the model, we need to have a dictionary of both the languages and then just our model checkpoint.
- And while exporting we'll exporting the Averaged Checkpoint Model of beam 5 as for that we got the highest Bleu Score.
- Then, there is another command which is called as fairseq-interactive in which you can type sentences as inputs as you go and it will translate and keep running.