Name- Praneet Thakur
Roll no.- B19CSE066

NLU PROJECT REPORT

Translation from English to Indic language

Group member: Praneet Thakur (B19CSE066)
Contribution: All work done by Praneet Thakur

Link to Colab notebook:
https://colab.research.google.com/drive/1_-DDTddOHiw5yb5AkSgxVZnJjIV1P1W9?usp=sharing

Dataset chosen: en-hi (English to Hindi)

Related Research Papers and Studies :

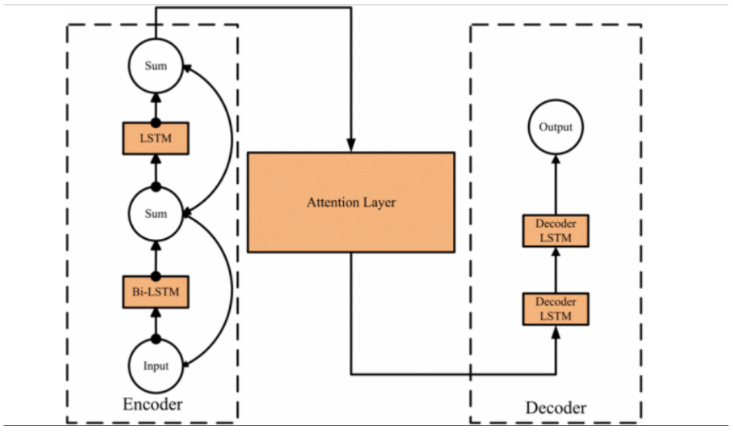
(1)Neural Machine Translation for English to Hindi:

Neural Machine Translation (NMT) is a machine translation system that uses an artificial neural network to increase fluency and accuracy the process of machine translation. NMT is based on a simple encoder-decoder based network. The type of neural networks used in NMT is Recurrent Neural Networks (RNN). The reason for selecting RNN for the task is the basic architecture of the RNN. RNN involves cyclic structure and which enables the learning of repeated sequences much easier than the other networks. RNN can be unrolled to store the sentences as a sequence in both sources as well as target languages.

In the paper, a LSTM based NMT is used in which a bi-directional encoder is used.This encoder is based on the concept that the output at any time instant may not only depend on past data but also on future data. Using this idea, the LSTM is tweaked to connect two hidden layers of opposite directions to the same output. This tweaked version of LSTM is called a Bidirectional LSTM.

Also in the paper, they have used a local Attention model. In local attention model, the model first predicts a single aligned position for the current target word. With the help of a window, which is centered around the source position, computes a context vector.

Name- Praneet Thakur
Roll no.- B19CSE066



(2) Neural Machine Translation for English to Hindi(with different config. Of NMT):
This paper has the same concepts as the one discussed above with basis being NMT , LSTM and Attention based models. But in the previous paper they had used bi-directional encoder. In this paper the different configurations are 2 layer LSTM with SGD(Stochastic gradient descent), 4 layer LSTM+SGD, 2 layer (bi-dir) LSTM+SGD and 4 layer (bi-dir) LSTM + SGD. All these configurations are compared on the basis of their training times and BLEU scores.

(3)English-Hindi Neural Machine Translation-LSTM Seq2Seq and ConvS2S:

In this paper as well once the same NMT based LSTM model is used first as in the previous papers with different configurations like uni-directional LSTM and bi-directional LSTM both with max 8 layers. Next another method is used using CNN(convolutional neural networks). ConvS2S involves encoder-decoder approach but contains convolution blocks containing layers of convolutional blocks in both encoder and decoder

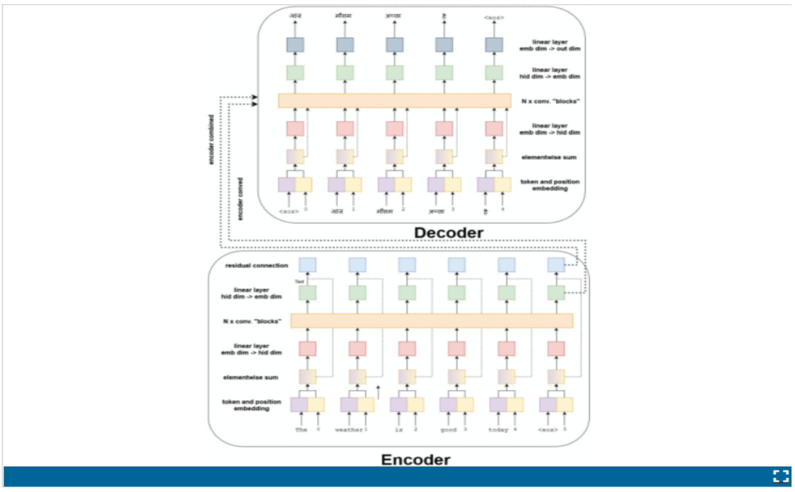

Decoder
Encoder
linear layer emb dim -> out dim
linear layer hid dim -> emb dim
N x conv. "blocks"
linear layer emb dim -> hid dim
elementwise sum
token and position embedding
residual connection
encoder combined
encoder conved

Fig. 3
ConvS2S architecture

Name- Praneet Thakur
Roll no.- B19CSE066

The encoder unit accepts the input and forms an embedding vector by summing the positional vector and token embedding. This embedding is passed through a linear layer that transforms the vector to the size acceptable by the convolutional block. The output of the convolutional block is fed to the linear layer to convert it back to the original embedding size called the conved vector. This conved vector is summed with embedding corresponding to the tokens individually generating a combined vector.The decoder has a structure similar to the encoder except that it takes token embedding, conved and combined vectors as input and convolutional blocks here contain an additional layer to compute attention scores.

Later these models are compared using bleu scores with different configurations,

All these Papers mentioned above have more or less used different datasets.

Model Information and Documentation:

(1) Model Architecture and Method Used

I have implemented a transformer-based encoder-decoder architecture and I have trained it on an English to Hindi machine translation. We know that a transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data.
My approach in this model was to :

- Read the dataset appropriately, preprocessing, tokenization etc.
- Vectorizing text of English and Hindi using the Keras TextVectorization layer.
- Implementing Encoder layer, decoder layer, and a positional embedding layer.
- Generating translations and evaluating the model using accuracy and BLEU scores.

Preparing the data:
The dataset en-hi provided had english and hindi sentences separately given. So I first brought whole corpus into a single dataframe and then formed two columns with english sentences and hindi sentences and then did preprocessing on it.

Vectorizing the data and formatting datasets:
Now I have used two instances of vectorization layers for both hindi and english where english uses default string standardization and hindi used a custom string standardization where we add the character "¿" to the set of punctuation characters to be stripped.

Name- Praneet Thakur
Roll no.- B19CSE066

Making the Transformer Layers:
Here I make the Encoder and Decoder layers which together makes up the transformer. And we make use of positional embedding so the model becomes trained for word order. A source sequence is passed through the encoder which forms a new representation of it that is passed onto the decoder with the target sequence. The decoder tries to predict the upcoming words in the target sequence. We then use accuracy and BLEU scores to evaluate our model.

And at the end we give the model vectorized english sentences as inputs and the target torken (start) and w generate token till we hit (end).

(2) Training, Validation and Testing Splits:

Here I have split the data according to the size of the dataset (the dataset has approx 8466304 sentences each of english and hindi). I have made an 80% to 20% training to test split and out of the 80% training data, I have further made an 80% to 20% training to validation split.

Training - testing : 80-20 %

(3) Hyperparameters and Others:

Total number of epochs considered were 20.

There are a total of approx. 8466304 sentences of english and hindi both. I have read these datasets to two different dataframes and then concatenated them so the resultant dataframe that has two columns (one of english sentences and one of hindi) with 8466304 sentences each.

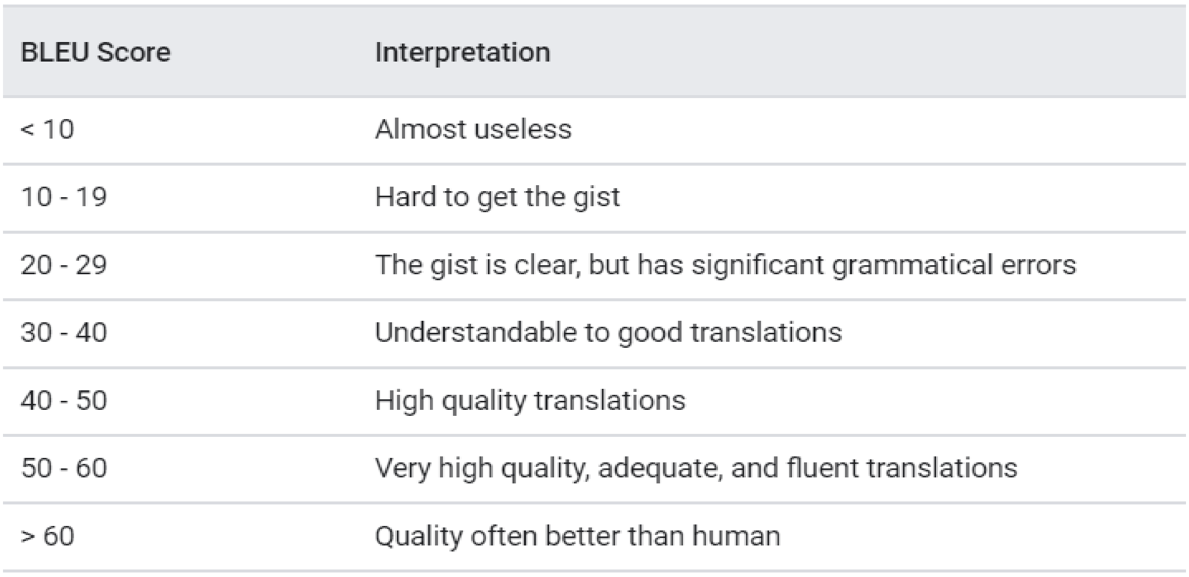
Evaluation Metric:

There are quite a few ways to evaluate a machine translation model like BLEU scores, TER (Translation error rates), Perpexlity matrix ,etc.
Here the Evaluation Metric we are using to judge the performance of our model and compare it with others are accuracy scores and BLEU scores.

BLEU scores:

BLEU (Bilingual Evaluation Understudy) is a method to evaluate the quality and accuracy of a machine-translated text (sentence) from one language to another.

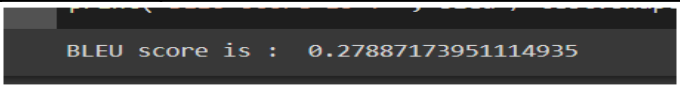
Scores are calculated for individual translated sentences by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality.
This BLEU algorithm is said to correlate well with human evaluation and is quite dependable.
The following is a simple way to interpret the BLEU score (a very rough approach) however BLEU follows a complex algorithm itself. (Here the scores are in percentages).



BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

Results and Observations: BLEU Score for our transformer model: 27.887%

Paper no.	**Model**	**BLEU score %**
	Transformer based (our project)	**27.887**
1	NMT based LSTM	12.34
2	2-layer LSTM+SGD	12.512
2	4-layer LSTM+SGD	13.523
2	2-layer(bi-dir)LSTM+SGD	12.854
2	4-layer(bi-dir)LSTM+SGD	13.863
3	8-layer LSTM Seq2seq	14.992
3	8-layer (bi-dir) LSTM Seq2seq	15.203
3	8-layer Convs2s	16.282

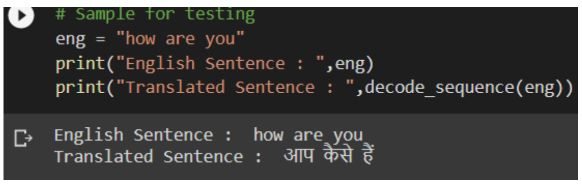


```
BLEU score is :  0.27887173951114935
```

Name- Praneet Thakur
Roll no.- B19CSE066

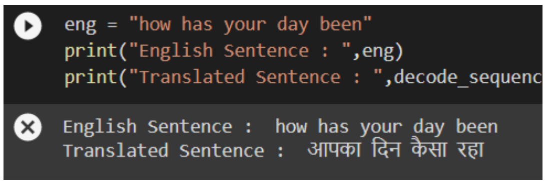
Conclusion: The BLEU score obtained between our model and rest of the models mentioned above are different due to different datasets and size of datasets but all in all it can be concluded that the transformer model performs better than other model involving LSTM methods. At the same time, the model is doing alright for small sentences with some errors and sometimes there are unknown words present but it was doing quite poorly for larger sentences with words more than 15 or 20 because of lesser amount of training.

Outputs and test runs:



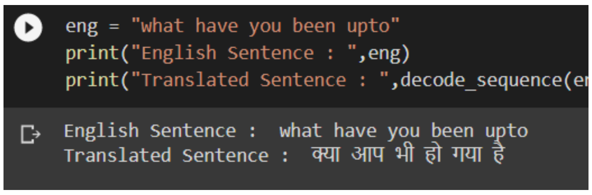
```
# Sample for testing
eng = "how are you"
print("English Sentence : ",eng)
print("Translated Sentence : ",decode_sequence(eng))

English Sentence :  how are you
Translated Sentence :  आप कैसे हैं
```



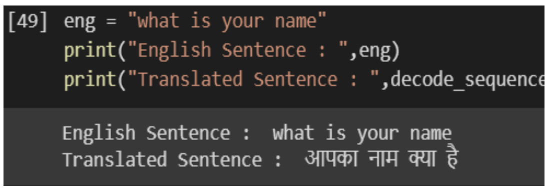
```
eng = "how has your day been"
print("English Sentence : ",eng)
print("Translated Sentence : ",decode_sequenc

English Sentence :  how has your day been
Translated Sentence :  आपका दिन कैसा रहा
```



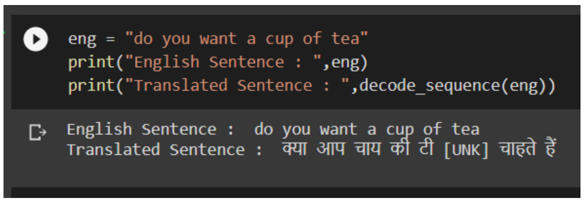
```
eng = "what have you been upto"
print("English Sentence : ",eng)
print("Translated Sentence : ",decode_sequence(e

English Sentence :  what have you been upto
Translated Sentence :  क्या आप भी हो गया है
```



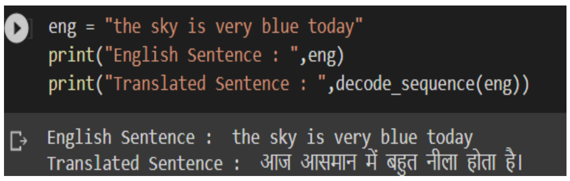
```
[49] eng = "what is your name"
     print("English Sentence : ",eng)
     print("Translated Sentence : ",decode_sequence

     English Sentence :  what is your name
     Translated Sentence :  आपका नाम क्या है
```



```
eng = "do you want a cup of tea"
print("English Sentence : ",eng)
print("Translated Sentence : ",decode_sequence(eng))

English Sentence :  do you want a cup of tea
Translated Sentence :  क्या आप चाय की टी [UNK] चाहते हैं
```



```
eng = "the sky is very blue today"
print("English Sentence : ",eng)
print("Translated Sentence : ",decode_sequence(eng))

English Sentence :  the sky is very blue today
Translated Sentence :  आज आसमान में बहुत नीला होता है।
```

We can see here the model has translated pretty accurately but it has made error in 3rd output for “what have you been upto” and last one “the sky is very blue today”. It also gives “Unk” in the translations for the words it doesnt know. All this could be avoided if I had trained the model for larger data and epochs.

Challenges faced and what can be done to improve

Here the accuracy scores and the BLEU scores are okay and not very high (27.887%). They are higher compared to other papers results. The main challenges I faced while doing this project was the lack of computational power mainly. I think I could have gotten a higher accuracy score and a better bleu score if I had trained the model for more epochs (like 50).
The dataset has english and hindi sentences separately and hence I have read these datasets to two different dataframes and then concatenated them so the resultant dataframe that has two columns (one of english sentences and one of hindi) with 8466304 sentences each. But due to lack of proper resources and

computational power I had to cull the dataset and consider lesser amount of sentences for both english and hindi datasets and also had to settle for 20 epochs for the training because my computer was taking very long to preprocess and then run the model for all these sentences which was making it overheat and as a result of which my computer was crashing. And I was working alone so I didnt have many options. But there is a lot of room for improvement in this model as conclusion.

References:

https://ieeexplore.ieee.org/document/9182117

https://ieeexplore.ieee.org/abstract/document/8464781?casa_token=cjt1wcj_RIAAAAAA:qdQh7XYLYJZcy046WpQ_RsnsFrCDgFXcTUoY7HWATJE53p0Vk6SWoaTDD1CwQJ3HpuFfUc_ciJw

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8882969

https://keras.io/examples/nlp/neural_machine_translation_with_transformer/

https://keras.io/api/callbacks/