An Update on my Machine Translation Work

Anirudh Srinivasan

April 3, 2018

1 The Encoder-Decoder Model

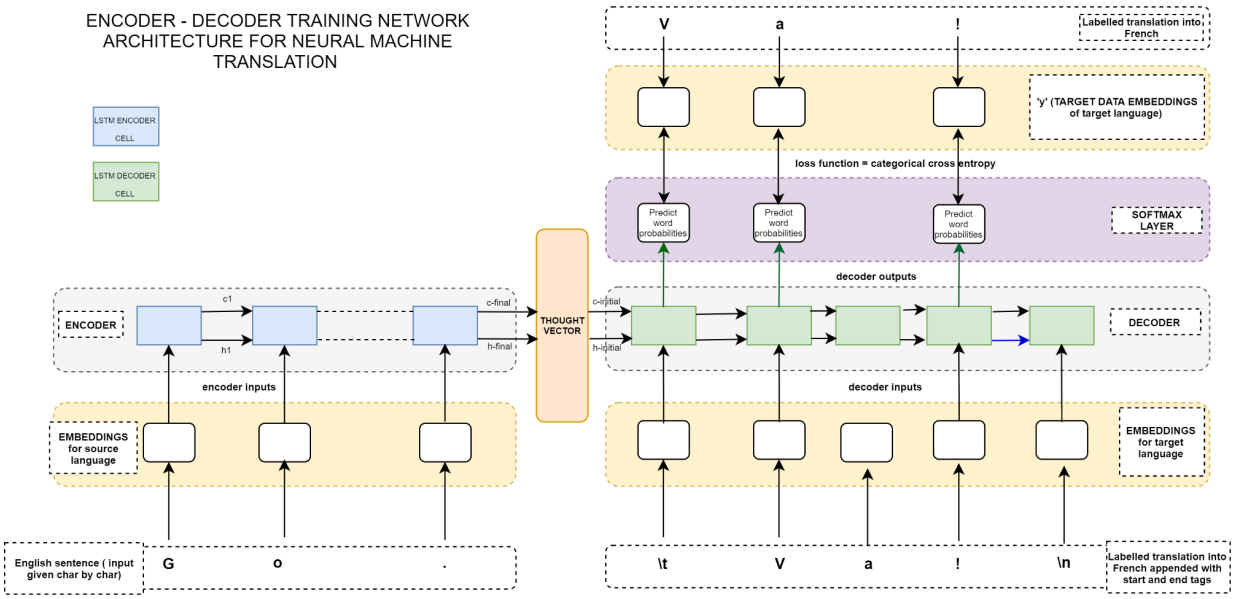

Figure 1: Overview of the Model

2 Evaluation Metrics - BLEU

- Cumilative weighted 4 gram overlap
- NLTK has an implementation of this (*nltk.translate.bleu_score*)
- Gives score in range [0,1]

- Journals report this score multiplied by 100

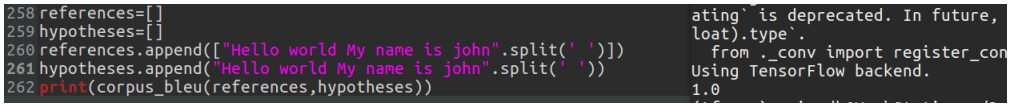

Figure 2: BLEU score of 1

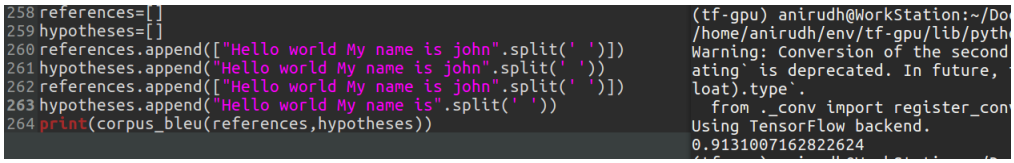

Figure 3: BLEU score smaller than 1

3 Character Level encoding

- Length of LSTM becomes very large
- Hidden state of encoder needs to propagate all the way
- Not able to capture information for such a long period
- Vanishing gradients??
- Not feasible to use character level encoding

4 Running Methodology

- Create a Model
- As of now, using word level encoding
- Run it on the fra-eng.txt corpus
- Tune the model so that it works well
- Run it for the bible sanskrit-english corpus

5 Problems discovered

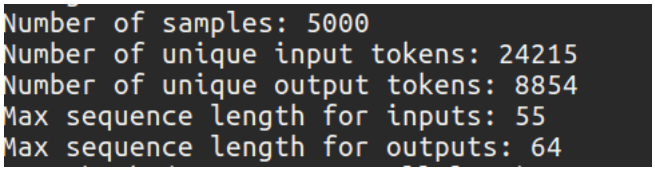

Figure 4: Sanskrit

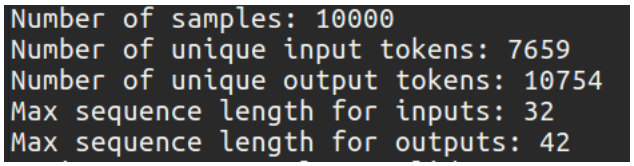

Figure 5: French

- No. of unique tokens in french vs sanskrit
- Sentence length in french vs sanskrit
- Effective number of samples that one can train on
- In sanskrit, there are a large number of words which occur only once. Hence a lot of space get's allocated for them. Some way must be found to handle this
- In sanskrit, since sentence length is much more, one pass through the LSTM Takes more time

6 Results so far

Corpus	No. Epochs	BLEU score(test set)	Sample Translation
fra-eng	500	0.0219	Click Here
Sanskrit Bible	100	0.0115	Click Here
Sanskrit Bible	500	0.0026	Click Here

Table 1: BLEU scores for fra-eng and Sanskrit Bible Corpus

The above scores suggest that the model may be overfitting

No. Epochs	BLEU score on train set	BLEU score on test set
10	0.1286	0.1243
20	0.0143	0.0058
50	0.0513	0.0290

Table 2: BLEU scores for test and train set on Sanskrit Bible Corpus

- BLEU scores, even for fra-eng were very low. Common papers report BLEU scores of around 30(multiplied by 100)
- As the number of epochs were increased, the output sentences made more sense w.r.t the expected output
- In current implementation, there are few constraints w.r.t the length of the input and output sentence
- This model was designed in such a way that the length of the sentence shouldn't be a Problems
- Will modify the code and try that

7 Future Prospects

- Vectors used for input and output are very sparse, because there are a large number of words
- Propose a new vector scheme
- Use a POS tagger($INRIA, France$) to extract the root word and POS
- Let the vocabulary contain the root words alone. To the one-hot-vectors, add the POS information as features

7.1 word2vec/GloVe

- word2vec was attempted using the original implementation from Google
- It did not parse the input correctly. The input had words separated by spaces
- It should have taken each word as a unit and split based on spaces. Instead, it took each character as a unit and ran the algorithm for it

- The implementation was done in C and probably doesn't take into account characters that are non ASCII. Will look into it later