Competition: Hindi to English Machine Translation System

Aman Tayal
180074
{amant}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

In this competition, We have to create a neural machine translation model that will help in translating Hindi sentences to English sentences. I have used an end-to-end transformer model [1] for translation task which has outperformed many other neural machine translation models such as seq2seq [2], seq2seq with attention [3]. I have achieved rank 15 out of 66 total submissions and my model achieved a BLEU score of 0.0822 and METEOR score of 0.347 while the best model in competition achieved a BLEU score of 0.1489 and METEOR score of 0.428 .

1 Competition Result

Codalab Username: A_180074
Final leaderboard rank on the test set: 15
METEOR Score wrt to the best rank: 0.347
BLEU Score wrt to the best rank: 0.0822
Link to the CoLab/Kaggle notebook: `https://colab.research.google.com/drive/18NaisVsE21cmE4u134fFXPfr`
`usp=sharing`

2 Problem Description

In this competition, we have to solve a problem to automatically translate Hindi sentences into English sentences that will help in automatically translating articles in Hindi to English without any human interference. We have given a training dataset with Hindi sentences and their corresponding English translation, which has been curated from public sources [1] . We will use this dataset to create an end-to-end Hindi-to-English neural machine translation model that will help us to solve this problem.

3 Data Analysis

3.1 Training Dataset

The training dataset provided in the competition consists of 1,02,322 Hindi-English sentence pairs. I examined the most common word occurring in Hindi sentences in the dataset and found some characters not part of Hindi Vocabulary or very less used, such as '#', '$', '%', '&', '£', '¥', '§', '©', 'Â', 'è', 'Ã', '€', '[', ']'. There are 1481 sentences pair in the dataset with either non-ASCII characters in English sentences or the characters mentioned above in Hindi sentences. I removed these sentences from the dataset as they are noisy and increase the model's vocabulary size.

The maximum length of Hindi and English sentence in the dataset is 596 and 301, respectively, and the minimum length is 1 for both. The figure 1 shows that in both English and Hindi data, most of the sentences has length less than 10. Our dataset mainly contains sentence pairs of small length, primarily in between 5-10, and there are very few pairs with a length of more than 25. We remove all

[1] `https://opus.nlpl.eu/`, `http://www.opensubtitles.org/`

the sentences from the dataset with a size of more than 25. Keeping fix maximum length of sentences helps create batches of the dataset and padding them as this will make sure batches don't have much difference between the length of sentences. It will also help in removing noise from the dataset.

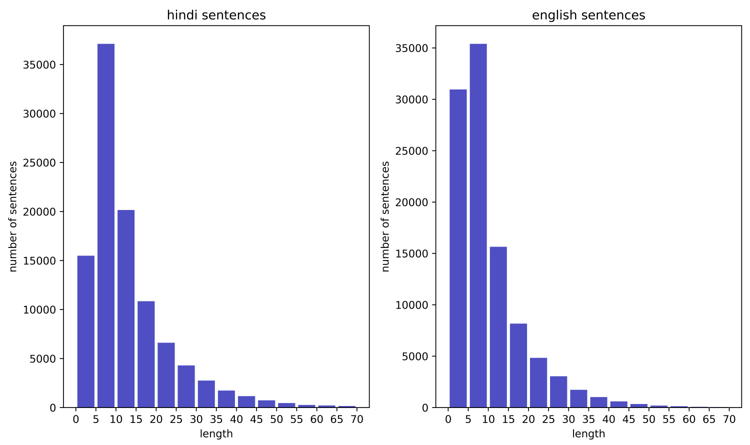

Figure 1: histogram plot of length of sentences in dataset

After removing noisy sentences and sentence with a length of more than 25, the dataset now consists of 89,076 sentence pairs. After applying some basic preprocessing techniques, I tokenized all sentences using spacy tokenizer [2] for English and indic_tokenize [3] for Hindi and created vocabulary. Hindi Vocabulary has 31,245 words and English Vocabulary has 22,933 words. The figure 2 shows the frequency distribution of word, and we can observe that the frequency distribution of words in the dataset follows Zipf's law [4].

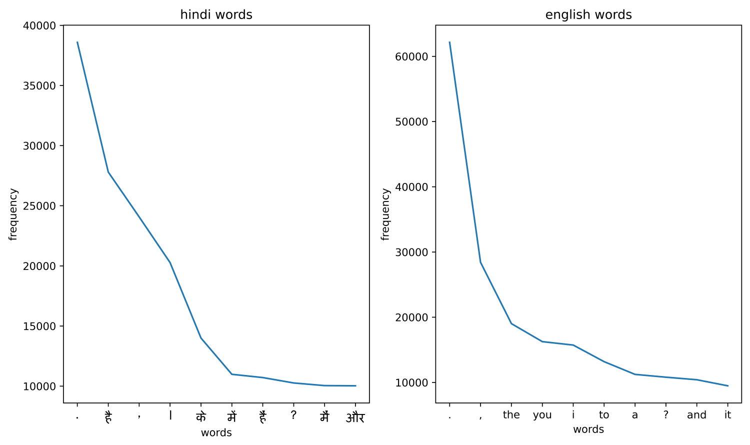

Figure 2: Frequency distribution of words

Figure 3 shows that out of 31,245 words in Hindi Vocabulary, around 20,000 words has a frequency of less than 3, and out of 22,933 words in English Vocabulary, about 13,000 words has a frequency of less than 3. We can remove some of the words from vocabulary to reduce model parameters and prevent overfitting. We choose 10,000 as a vocabulary size for both Hindi and English sentences.

[2]https://spacy.io
[3]https://pypi.org/project/indic-nlp-library/
[4]https://iq.opengenus.org/zipfs-law/

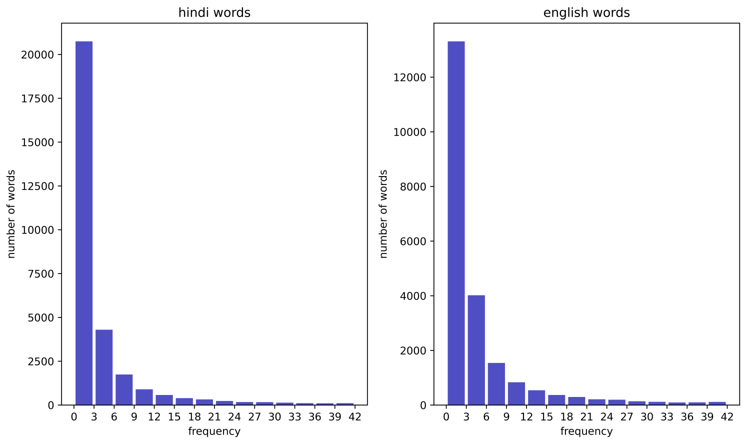

Figure 3: histogram plot of frequency of words in dataset

3.2 Test Dataset

Test dataset has about 24,102 Hindi sentences. We have restricted our vocabulary size to 10,000, so figure 4 shows the frequency of unknown words in Hindi sentences in the test dataset. About 8,463 sentences have unknown words in the test dataset, which is about 35%, but from figure 4 we can observe that most sentences with unknown words has only 1 or 2 such words.

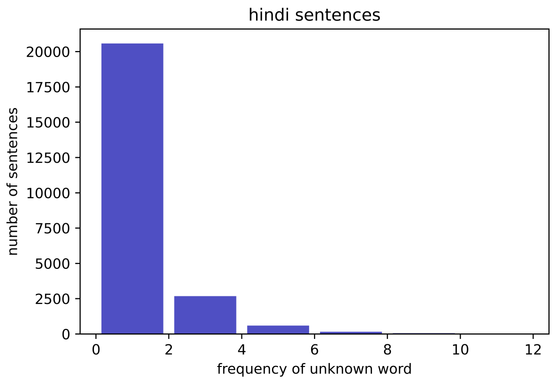

Figure 4: histogram plot of frequency of unknown words in Hindi test sentences

4 Model Description

4.1 Model Evolution

Firstly, I implemented the seq2seq model [2] in which we have two parts encoder and decoder. We first encode the input sentence into a fixed-length vector using an encoder model, and then we decode the output sentence from an encoded vector using a decoder model as shown in the figure 5.
We can use different recurrent layers in both encoder and decoder, which can be LSTM [4] or GRU [5] or just a simple recurrent layer with different hyperparameters. I trained five different seq2seq model with different recurrent layers but with the same hidden layer dimension of 512 and vocabulary size of 8196 and compared their METEOR and BLEU scores on test set as shown in table 1
From table 1, we can observe that the Seq2Seq model with Bidirectional GRU layer in encoder and GRU layer in decoder outperforms other models. The output sentence produced by our model has many repetitive words. It doesn't capture all possible sentences as simple Seq2Seq encode all information about the input sentence into a fixed-length vector, leading to the model not capturing all information

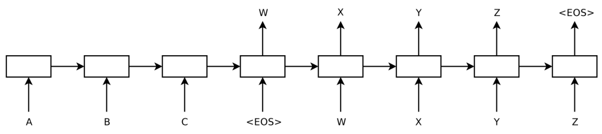

Figure 5: Seq2Seq model [2]

Encoder	**Decoder**	**BLEU Score**	**METEOR Score**
LSTM	LSTM	0.006	0.15
LSTM with 2 layers	LSTM with 2 layers	0.010	0.17
Bidirectional GRU	GRU	0.012	0.19
Bidirectional LSTM	GRU	0.007	0.18
Bidirectional LSTM	LSTM	0.011	0.17

Table 1: BLEU and METEOR scores for Seq2Seq model with different recurrent layer

about the input sentence like the position of different words in the input sentence. To resolve this problem, we updated our model with an attention layer [3]. In seq2seq with the attention model, the decoder, instead of only using a single fixed-length vector, also takes output for all words in the encoder model. The decoder now for each prediction first calculate attention for each word in an encoder and decides which word is more important for prediction. Using the attention layer in Seq2Seq model with Bidirectional GRU layer in encoder and GRU layer in decoder gives BLEU score of **0.0264** and METEOR score of **0.2861** on test set, which is more than the score of the model without attention layer.

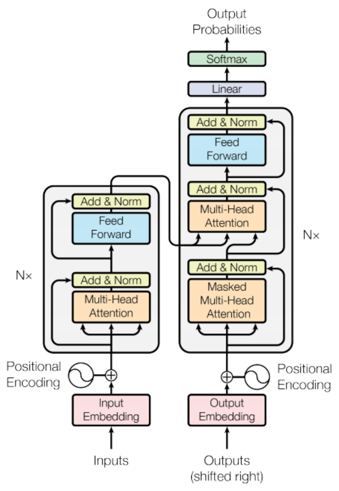

Figure 6: Transformer model [1]

Finally, I implemented a transformer model [1] which, instead of using a single encoder-decoder attention layer with RNN layers, uses multiple attentions concurrently and also uses self-attention in both encoder and decoder layers as shown in the figure 6, and it uses multiple such layers for encoding and decoding. The transformer model has outperformed many other models for neural machine translation task as it uses multiple self-attention layers in the encoder, which can capture more information than any other model and instead of only using encoder-decoder attention layer, it also uses self-attention layer in a decoder model which improves sentence prediction. The model uses positional embedding to capture information about the position of words in input as the model runs parallelly on all words instead of sequentially in RNN layers.

4.2 Final Model

4.2.1 Final Model Description

I used a transformer model with 512 hidden size, 8 attention heads and a feedforward layer with size 1024 in both the encoder and decoder layer. I used 4 layers in both the encoder and decoder model. For embedding, I used a vocabulary size of 10,004 for both Hindi and English and an embedding size of 512. I also used the look-ahead mask in the decoder self-attention layer, which prevents words in the decoder from looking at future words for prediction and used a padding mask in the encoder self-attention layer. To produce the final output, I used a single feedforward layer of size 10,004, same as vocabulary size with softmax activation on decoder outputs.

4.2.2 Optimization and Loss Function

I used the Cross-Entropy loss function as the model objective function. For optimization, I used Adam optimizer with the same hyperparameter as author used in paper [1], $\beta_1 = 0.9$, $\beta_2 = 0.98$ and initial learning rate of 10^{-4} and updated the learning rate after each step using the equation given below, where warmup steps used in the model is 4000 and d is hidden size of model.

$$lr = d^{-0.5} * \min(step^{-0.5}, step * warmup\_step^{-1.5})$$

I trained the model for 15 epochs which took around 30 mins for training on GPU. Model achieved best validation score in about 11 epochs.

4.2.3 Inference

I used the auto-regressive technique for inferencing model with beam search of width 3 and length normalization. I used length normalization so that beam search doesn't give preference to lower length sentences.

5 Experiments

5.1 Pre-Processing

For pre-processing, as explained in section Data Analysis, I removed some noisy sentences with a character not typical in Hindi and English language and removed all sentences with a length of more than 25. Then I convert all characters to lower case and replaced short-form words in English sentences, such as replacing "I'm" with "I am", "'ll" with "will", etc. These basic pre-processing helps in reducing vocabulary size and removed noise from the training set.

5.2 Hyper-parameters

I tested the Seq2Seq model with Bidirectional GRU layer in encoder and GRU layer in decoder on different vocabulary size and parameter initialization as shown in the table 2 and 3.

Vocabulary Size	**BLEU score**	**METEOR score**
8,196	0.0074	0.1789
12,292	0.0057	0.1686
16,388	0.0053	0.1725

Table 2: BLEU and METEOR scores on different vocabulary size

Initialization	BLEU score	METEOR score
normal with mean 0 and std 0.01	0.0074	0.1789
xavier_normal	0.0076	0.1891
xavier_uniform	0.0076	0.1834

Table 3: BLEU and METEOR scores on different initialization

Table 2 shows that the model with 8196 as vocabulary size performs best. Increasing vocabulary size instead of improving performance hurts the model as the model tends to overfit. We can also observe this from figure 3, which shows that apart from about the top 10,000 words, the rest words come only three times in the whole dataset. Table 3 shows that using xavier normal initialization [6] improves the performance model. Xavier normal initialization sample parameters from $\mathcal{N}(0, std^2)$, where std is given by equation,

$$std = \sqrt{\frac{2}{d_{in} + d_{out}}}$$

Inference method	BLEU score	METEOR score
Greedy search	0.0352	0.3052
Beam search (with 3 beam width)	0.0396	0.3158

Table 4: BLEU and METEOR scores on transformer model using different inference methods

I tested the transformer model on inference using greedy search and beam search with beam width 3 and length normalization, and the result shown in table 4. Table 4 shows that using beam search increases performance over using greedy search. But using beam search significantly increases inference time that's why I couldn't test on higher beam width.

6 Results

Phase	Best Model	BLEU Score	METEOR Score	Rank
1	Seq2Seq model with BiGRU encoder and GRU deocoder	0.0066	0.169	12
2	Seq2Seq model with BiGRU encoder and GRU deocoder initialized using xavier_normal	0.0305	0.184	14
3	Seq2Seq model with BiGRU encoder, GRU deocoder and attention layer	0.0577	0.268	3
Final	Transformer model with beam search in inference	0.0822	0.347	15

Table 5: Best Model, their scores and rank on different phases

Table 5 shows that the transformer model with beam search outperforms other models. The transformer model uses multiple attention layers to capture more information about the input sentence. Apart from encoder-decoder attention, it also uses a self-attention layer while decoding that significantly improves the prediction of the model. It can capture long term dependencies, and also size of the input sentences does not affect the model much because the model runs parallelly on all words instead of sequentially as in recurrent layers used in the Seq2Seq model. Using beam search improves score as instead of greedily selecting the best word, beam search considers multiple options and select one with higher conditional probability.

7 Error Analysis

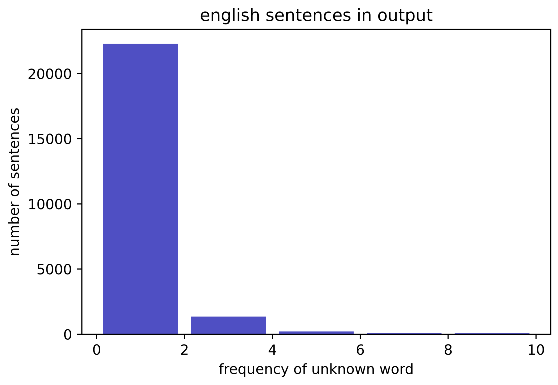

Figure 7: histogram plot of frequency of unknown words in sentences created by model

Out of 24,102 sentences generated by the model, 5597 sentences has unknown words in them. Figure 7 shows that most of the sentences have less than two unknown words. Out of 15,639 Hindi sentences without unknown words, the model generated an output English sentence with unknown words for 660 sentences which shows that vocabulary used for Hindi might have some words for which equivalent word is not present in English vocabulary. Figure 4 and 7 also shows that our model has a much better vocabulary for English sentence that's why out of 8,463 Hindi sentences with unknown words, 3526 sentences has no unknown word in English translation by model. Our model either guesses these words using other words in input or overfitted to some of the words in English vocabulary.

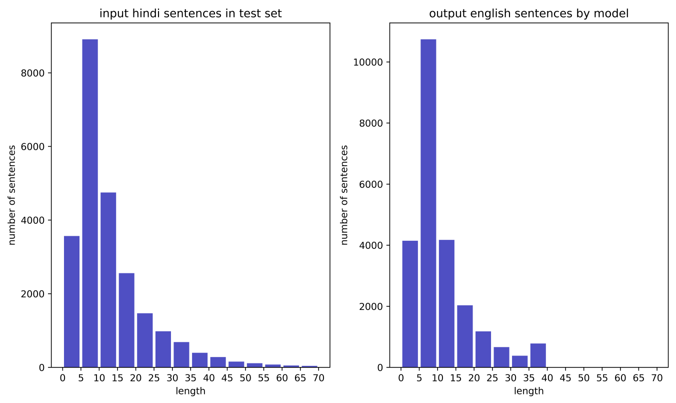

Figure 8: histogram plot of length of sentences in dev set and output from the model

Figure 8 shows that model is more likely to produce smaller sentences. The test dataset has 13,700 sentences with a length less than 10, while output from the model has 16,065 sentences with a length less than 10. Also, for 4,559 input sentences with a length of more than 15, the model translated 365 sentences to English sentence of less than 10. Analysis of the length of input and output sentences shows that our model is more prone to create a sentence with a smaller length because either beam search prefers smaller sentences or most of the sentences in our training dataset have a length of less than 10. In inference, I set the maximum length to 40, which might result in some sentences not translate completely.

Our model gives a BLEU score of 0.0822 and a METEOR score of 0.347 on the test set. When tested only on sentences of size less than 10, our model gives a BLEU score of 0.1496 and a METEOR score of 0.389, which shows a significant increase in the BLEU score. Then I tested only for translated sentences that don't have any unknown word in the sentence, and our model gives a BLEU score of 0.1044 and a METEOR score of 0.3893. It shows that if we use better techniques to capture the vocabulary of the dataset, such as subword tokenization, we can increase the performance of our model.

	True Translation	**Predicted Translation**
1	“but anyway we’re only just holding it together”	“all the way that are gone”
2	“goodbye max”	“come on”
3	“you’re smart”	“you are so clever”
4	“nothing ever turns out as planned ever”	“it does not have anything”
5	“right”	“yeah”
6	“gainfully employed”	“it ’s successful”
7	“another 200”	“and two hundred”
8	“don’t worry guys”	“do not bother”
9	“let’s do another experiment”	“let us just take one more”
10	“that a district of no more than 10 miles square”	“for the new government ’s new york”

Table 6: True and Predicted Translation of sentences with least METEOR score

Table 6 shows some sentences translated incorrectly on the basis of the METEOR score. We can see some sentences are completely mistranslated, as in sentence 1, 2, 4 and 10 in table 6. For some sentences, the word is translated to their synonym in true prediction as in the sentence 3,5 and 8 in table 1. METEOR score is low for these sentences because METEOR metric doesn’t check for synonyms and other similar slangs like “yeah” for “right”. Also, for sentence 7 in the table 6, “200” in true sentences, but it is ”two hundred” in sentence predicted by the model, both have the same meaning, but the METEOR score doesn’t take that into consideration. Our model has mistranslated some sentences, but for some other sentences, METEOR metric do not give a correct score.

8 Conclusion

First, we analyze various Seq2Seq model for translation and found that using the attention layer in the Seq2Seq model significantly improves the performance. We show that using different hyperparameters and initialization changes the overall performance model. Then we finally implemented the Transformer model, which outperformed the Seq2Seq model and delivers the best result when used with beam search for inference. We can further improve performance of our model by using subword tokenization [7], contextualized embeddings [8] and better hyperparameter tuning.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” 2014.

[3] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 2016.

[4] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, 1997.

[5] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” 2014.

[6] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and*

Statistics (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[7] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," 2018.

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.