Competition: Hindi to English Machine Translation System

Shilpa Chatterjee
20111057
{shilpa20}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

For the Competition, my first model was GRU as Encoder and Attention based GRU as Decoder.It obtained a METEOR score of 0.129 and BLEU score of 0.0010.My second attempt was Bi-directional LSTM as encoder and LSTM as decoder.It obtained a METEOR score of 0.170 and Corpuslevel BLEU score of 0.0166. My next attempt was Bi-directional LSTM as both encoder and decoder,which fetched me a METEOR score of 0.165 and Corpuslevel BLEU score of 0.0145.Lastly,I experimented with Bi-directional LSTM as encoder and LSTM as decoder with Attention mechanism, that obtained METEOR score of 0.286 and Corpuslevel BLEU score of 0.0580.

1 Competition Result

Codalab Username: $S\_20111057$
Final leaderboard rank on the test set: 25
METEOR Score wrt to the best rank: 0.286
BLEU Score wrt to the best rank: 0.0580
Link to the CoLab/Kaggle notebook: https://drive.google.com/file/d/1kloyCX03JuhPEH6d2ZYWsFGuaAFa_UaA/view?usp=sharing

2 Problem Description

The competition is about creating a Hindi to English Machine Translation system. Initially we are provided with a training dataset containing 102,322 Hindi-English sentence pairs and every week we need to build a model , train it with the train dataset and evaluate it's performance on the dev set that is provided . The dev set is changed every week but the train set remains the same.The performance of the model gets evaluated on the following metrics :

1. Macro Average Smoothed BLEU score
2. Corpus level BLEU score with smoothing
3. METEOR score

The performance of the model is evaluated on the dev set for three weekly phases while in the final phase, the performance of the model is evaluated on the test set provided.

3 Data Analysis

3.1 Uni-grams in Unprocessed Training Dataset

Before, pre-processing the source(hindi) and target(english) sentences of the training set , I observed that majority of the stop-words were the most occurring words of the dataset.

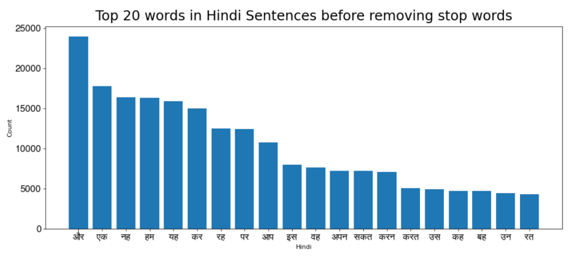


Figure 1: Most occurring Uni-grams of Hindi Sentences

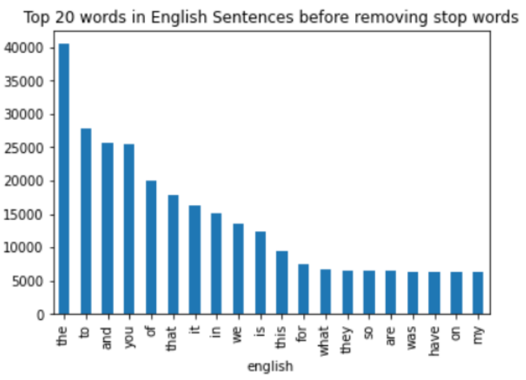


Figure 2: Most occurring Uni-grams of English Sentences

3.2 Token counts in Source and Target Sentences for Training Dataset

After data pre-processing was done(i.e. removing punctuation marks, special characters, digits,stop words), I found out that most of the hindi and its corresponding english sentences had around 150-200 tokens(i.e. words in this case).

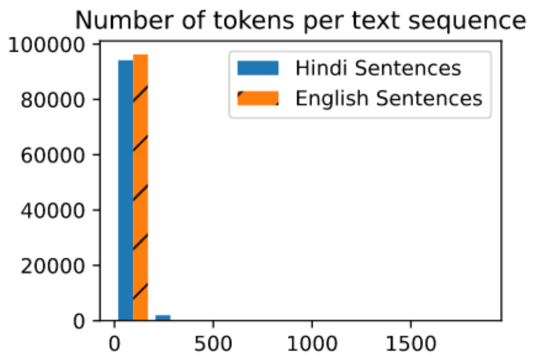


Figure 3: Analysis of token counts in Source and Target Sentences

3.3 Uni-grams in Unprocessed Test Dataset

Before, pre-processing the source(hindi) sentences of the test set , I observed that majority of the stop-words were the most occurring words of the dataset and they were similar to the top 20 hindi words of training set.

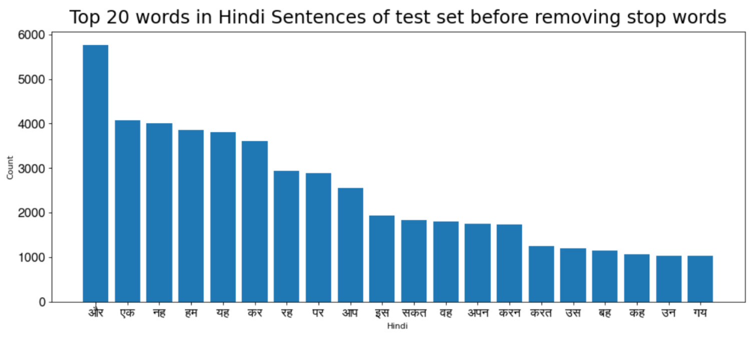


Figure 4: Most occurring Uni-grams of Hindi Sentences in test set

4 Model Description

4.1 Model Evolution

For different phases of the Neural Machine Translation Challenge I have used different Seq2Seq models. The basic architecture of a Seq2Seq model can be viewed as follows:

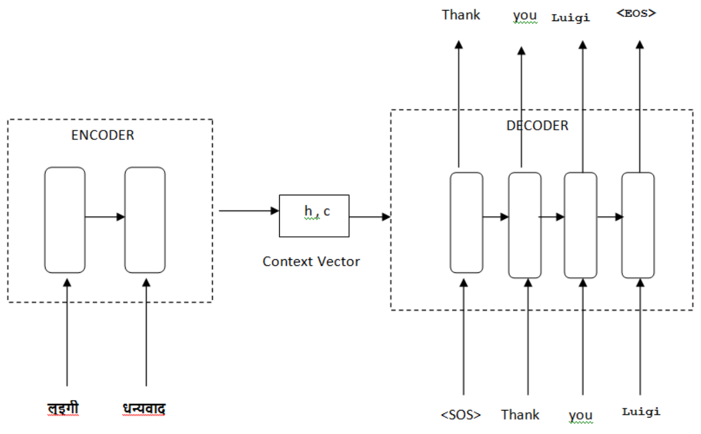


Figure 5: Basic Structure of a Seq2Seq Model

4.1.1 Model for Phase 1

For the first phase of the challenge I have used Attention based GRU Model[1]. It is a Seq2Seq model in which for my encoder I have used GRU(Gated Recurrent Unit) and for my decoder I have used GRU with Attention mechanism along with teacher forcing. For the attention scores, I have used a simple concatenation between embedding of the input passed onto the decoder and the hidden state forwarded by the encoder and applied a Softmax function to get the attention scores. For incorporating the teacher forcing mechanism I have kept my teacher forcing ratio to be 0.5 and each time I randomly selected a number between $(0, 1)$ and if the selected number was less then teacher forcing ratio i.e. 0.5 then for the decoder's next input instead of sending the previous output of the decoder , I have sent the word present in the target sentence at the that location. It needs to be noted that teacher forcing was used only during the training phase as target sentences were known and not during validation or testing phase.

[1]`https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`

4.1.2 Model for Phase 2

In the first phase of the competition , I was providing input to my encoder-decoder model one by one and it was a bit time consuming.So, for this phase I wanted to give inputs to my Seq2Seq model in small batches to reduce the training time. Also , I observed that even though GRU(Gated Recurrent Unit) was giving good results on the validation set formed by using 10 % of the training dataset (and training the model with remaining 90 % of the training dataset), it failed to provide good results on the dev set provided. So, for this phase I used a Bi-directional LSTM(Long Short Term Memory) as my encoder and a uni-directional LSTM(Long Short Term Memory) as my decoder.

4.1.3 Model for Phase 3

For, the third phase of the challenge , I wanted to see how a bi-directional decoder will affect the output and the results on the dev set. So, in this phase I made my encoder as Bi-directional LSTM(Long Short Term Memory) and my decoder also as Bi-directional LSTM(Long Short Term Memory). The Bleu and the Meteor scores did not improve much from the last phase but it did not deteriorate either.

4.1.4 Model for Final Phase

Since, in the previous phases of the challenge I observed that even simple Bi-directional LSTM(Long Short Term Memory) gave better results than GRU with Attention mechanism[1], I wanted to incorporate attention mechanism to a uni-directional LSTM Decoder keeping my Encoder as Bi-directional LSTM(Long Short Term Memory)[2].Not to my surprise, this improved the predictions of the model thereby, providing better scores.

4.2 Loss Function and Optimizer Used

I tried using loss functions like NLLLoss(Negative Log Likelihood Loss), CrossEntropyLoss in different phases of the Neural Machine Translation Challenge. For the Optimizer I have used SGD(Stochastic Graident Descent),Adam in different phases of the Neural Machine Translation Challenge. In the first phase of the challenge I used NLLLoss(Negative Log Likelihood Loss) function with SGD(Stochastic Graident Descent) as my optimizer. But since SGD used same learning rate across all dimensions , it might often oscillate and converge to a poorer optima. So, for the remaining phases of the challenge, I used Adam as my optimizer along with CrossEntropyLoss as my loss function.

5 Experiments

5.1 Data Pre-processing

Raw data(real world data) is always noisy and it is beneficial to pre-process the data before sending it to our model for Machine Translation task.

I observed that both source and target sentences in the training dataset where noisy in the sense that it had unnecessary punctuation marks(sometimes to express some emotion like excitement), also there were numbers in the sentences which were not so useful for our task. So I thought of cleaning the source and target sentences by removing punctuation marks, special characters(like @,$) and numbers.

Once the source and target sentences were cleaned , I wanted to tokenize the sentences so that I could build the source and target vocabulary from the training dataset and use it for my Seq2Seq model.For the Hindi Sentences I used tokenizer of IndicNLP and for English Sentences I just used space delimeter as my tokenizing criteria.

Using every unique token , I constructed my vocabulary (both for source[Hindi] and target[English] languages) and added $< SOS >$(Start of Sentence),$< EOS >$(End of Sentence),$< UKN >$(Unknown), $< PAD >$(Pad token) to the existing vocabulary. Each word in source and target vocabulary was

[2]`https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b`

given a unique index , so that each sentence in the training dataset could be represented in terms of these indexes and be sent as an input to the Encoder and Decoder during training phase.

5.2 Training procedure

5.2.1 Phase 1

For the first phase of the challenge , my model was GRU(Gated Recurrent Unit) as Encoder and Attention based GRU(Gated Recurrent Unit) as Decoder. NLLLoss(Negative Log Likelihood Loss) was the loss function and SGD(Stochastic Graident Descent) was the optimizer.I tried different learning rates for my optimizer like 0.1, 0.05, 0.01 and found that 0.01 works the best on the validation set(formed by splitting the training dataset as 90% for training and 10% for validation) and therefore finalised 0.01 as the learning rate of the optimizer. The training time for Attention based GRU model was around 5hours(i.e. 5-6 epochs).

5.2.2 Phase 2

For the second phase of the machine translation challenge, I used Bi-directional LSTM(Long Short Term Memory) as Encoder and uni-directional LSTM(Long Short Term Memory) as Decoder.
CrossEntropyLoss was the loss function and Adam was the optimizer. I tried experimenting with the learning rates, but since Adam uses adaptive learning rates at different dimensions , I observed that initializing it to a very small value was working well , so I kept learning rate to be 0.0001. I trained the model for 20 epochs(i.e. for approximately 20-21hours) and stopped after that as the Bleu score of the model's prediction on validation set was decreasing thereafter.

5.2.3 Phase 3

For the third phase of the machine translation challenge, I used Bi-directional LSTM(Long Short Term Memory) as Encoder and Bi-directional LSTM(Long Short Term Memory) as Decoder.
CrossEntropyLoss was the loss function and Adam was the optimizer. I kept the learning rate of the optimizer to be 0.0001(same as the previous phase). I trained the model for 11 epochs(i.e. for approximately 12hours) and stopped after that as the Bleu score of the model's prediction on validation set started decreasing thereafter.

5.2.4 Final Phase

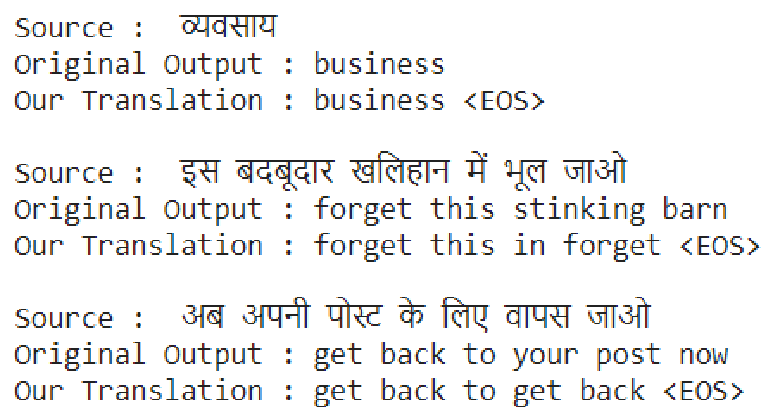
For the final phase of the machine translation challenge, I used Bi-directional LSTM(Long Short Term Memory) as Encoder and uni-directional LSTM(Long Short Term Memory)with attention mechanism as Decoder. CrossEntropyLoss was the loss function and Adam was the optimizer. I kept the learning rate of the optimizer to be 0.0001(same as the previous phase). I trained the model for 20 epochs(i.e. for approximately 20-21hours) and stopped after that as the Corpus level Bleu score of the model's prediction on validation set started decreasing thereafter.

6 Results

Hindi to English Machine Translation Challenge Results(On Dev Set)					
Phase	Model Used	Macro Average Smoothed BLEU score	METEOR score	Corpus level BLEU score with smoothing	Rank(in terms of METEOR Score)
Phase 1	GRU as Encoder and GRU as Decoder with Attention mechanism	Not Given in this phase	0.129	0.0010	33
Phase 2	Bi-directional LSTM as Encoder and Uni-directional LSTM as Decoder	0.1148	0.170	0.0166	19
Phase 3	Bi-directional LSTM as Encoder and Bi-directional LSTM as Decoder	0.10819	0.165	0.0145	21

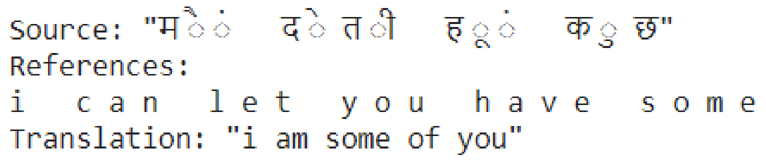
Hindi to English Machine Translation Challenge Results(On Test Set)					
Phase	Model Used	Macro Average Smoothed BLEU score	METEOR score	Corpus level BLEU score with smoothing	Rank(in terms of METEOR Score)
Final Phase	Bi-directional LSTM as Encoder and Uni-directional LSTM as Decoder with Attention mechanism	0.22234	0.286	0.0580	25

From the above table it is evident that even simple LSTM Model outperformed GRU with attention mechanism. Let us now see the outputs obtained on validation set for different models during different phases of the challenge :



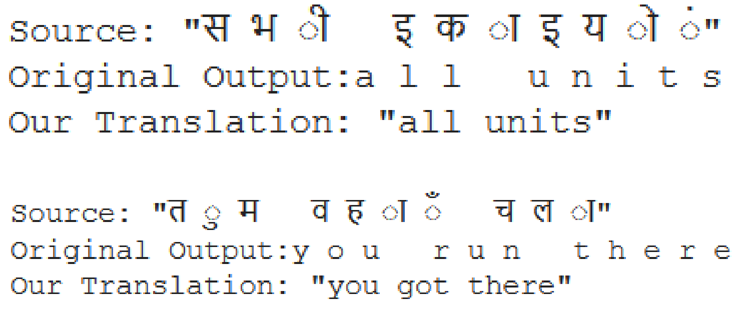
```
Source :  व्यवसाय
Original Output : business
Our Translation : business <EOS>

Source :  इस बदबूदार खलिहान में भूल जाओ
Original Output : forget this stinking barn
Our Translation : forget this in forget <EOS>

Source :  अब अपनी पोस्ट के लिए वापस जाओ
Original Output : get back to your post now
Our Translation : get back to get back <EOS>
```

Figure 6: Predictions made by Attention GRU model on validation set



```
Source: "मैं  देती  हूं  कुछ"
References:
i  c a n  l e t  y o u  h a v e  s o m e
Translation: "i am some of you"
```

Figure 7: Predictions made by Bi-directional LSTM model on validation set



```
Source: "सभी  इकाइयों"
Original Output:a l l  u n i t s
Our Translation: "all units"

Source: "तुम  वहाँ  चला"
Original Output:y o u  r u n  t h e r e
Our Translation: "you got there"
```

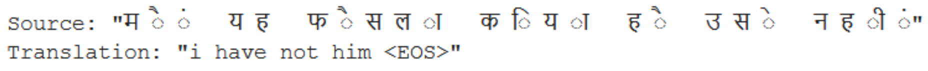
Figure 8: Predictions made by Attention LSTM model on validation set

Even though the predictive performance of Attention GRU was good on the Validation set(separately taken from the training set) it failed to perform well on the Dev set provided , while simple Bi-directional LSTM outperformed it . The reason for this being the memorising power of LSTM is much more than that of GRU , thus when the source sentences became long enough GRU failed to remember the sequence that appeared way back.

7 Error Analysis

7.1 Error Analysis on Dev Set

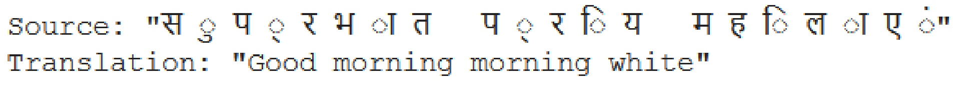
Let us consider an example for error analysis on development set provided.



```
Source: "मैं  यह  फैसला  किया  है  उसे  नहीं"
Translation: "i have not him <EOS>"
```

Figure 9: Predictions made by Attention GRU model on development set

The source sentence wanted to express "I have decided it and not him" but the attention based GRU model could correctly identify terms like "I", "have","not" and "him" but failed to interpret the hindi word "Faisla"(meaning decision in english) and hence missed the essence of the sentence.

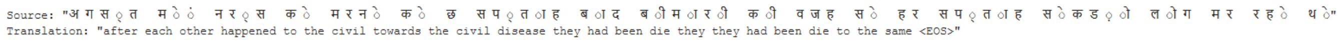


```
Source: "सुप्रभात  प्रिय  महिलाएं"
Translation: "Good morning morning white"
```

Figure 10: Predictions made by Attention LSTM model on development set

The source sentence wanted to express "Good Morning lovely ladies" but the attention based LSTM model could correctly identify "Suprabhat"(Meaning Good Morning in English) but failed to identify words like "Mahilayein"(in hindi) and "priyo"(in hindi).

However, if the sentences are short , LSTM and GRU Models could correctly identify the meaning of the sentence and could do proper translations. But, when sentences are too long the models fail to capture the meaning properly.



```
Source: "अगस्त में नर्स के मरने के छ सप्ताह बाद बीमारी की वजह से हर सप्ताह सेकड़ो लोग मर रहे थे"
Translation: "after each other happened to the civil towards the civil disease they had been die they they had been die to the same <EOS>"
```

Figure 11: Predictions made by Attention GRU model on development set(for long sentences)

7.2 Error Analysis on Test Set

8 Conclusion

In this Hindi to English Machine Translation Challenge , I have tried different Seq2Seq Models and LSTM with Attention Mechanism gave me the best results so far. In future , I would also like to implement Transformer Based Models as the memorizing capabilities of the transformers are usually considered to be much more than LSTMs and GRUs.

References

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