Competition: Hindi to English Machine Translation System

Aman Aryan
20111009
ryn20@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

This report is about the Hindi-English NMT Competition. This report contains details about the various models used during the competition. The various models I used were (GRU, LSTM, BiLSTM, Attention Mechanism). In the final phase, my rank is 17, with a BLEU score of 0.0751 and a METEOR score of 0.309.

1 Competition Result

Codalab Username: *A*_20111009
Final leaderboard rank on the test set: 17
METEOR Score wrt to the best rank: 0.309
BLEU Score wrt to the best rank: 0.0751
Link to the CoLab/Kaggle notebook: `https://colab.research.google.com/drive/10HEPkfWvNJ_AnnI2LxZ7O62rFTi578Jv?usp=sharing`

2 Problem Description

Hindi-English Neural Machine Translator: The competition describes it as the graduates' first job in the industry. The objective is to create a neural machine translation model using PyTorch, translating Hindi sentences into English sentences. The competition provided a training dataset at the beginning. Every week a dev set is release to evaluate and improve the models for four weeks. For the final evaluation, a test set is released. All these evaluations take place on the CodaLab platform. A maximum of five tried is allowed for every evaluation on the dev and test set.

3 Data Analysis

1. The training data set "train.csv" has been curated from publicly available sources like opensubtitles.com and opus. OPUS (Open Parallel Corpus) is a collection of translations collected from the web. Open Subtitles provide subtitles for movies. The data set have Hindi sentence and their English sentence translation.
2. It contains 1,02,322 pairs of Hindi and English sentences. 5697 Hindi sentences contain English alphabets. Some sentences contain symbols like Music Symbol, which is irrelevant both to Hindi and English Language. Also, many different words and phrases dictate a person's emotion or state when saying those sentences .e.g (Laughing). These noises exist because the public generally creates these subtitles for movies. There exist some large sentences with more than 200 words. About 95% sentences have the number of words less than equal to 30. The total number of unfiltered Hindi tokens and English tokens are 74591 and 71193, respectively. The total number of Hindi and English tokens with a single occurrence are 42682, 40786 respectively.
3. The 3 weekly dev set contains 5000 sentences. The final test set contains 24101 sentences.

4 Model Description

1. **Models Evolution** - This section describes the models and methods used and evolved in each phase. The various model and evolution of the models during the four phases are described below:
   (a) **Phase 1** : The first model was the Encoder-Decoder seq2seq model [1]. The encoder and decoder both were single-layer, unidirectional GRU units. In this phase, embedding feature size and hidden feature size are the same and set to 256. The model was trained with a Stochastic Gradient Descent SGD optimizer with a default learning rate. Since the training occurred one instance at a time, training took a long time. The next step was to apply mini batch gradient descent with Adam Optimizer with a learning rate of 0.01. It improved the training speed and gave a better METEOR score. The translation from this model produced sentences with the same word repeated. Next is the application of the teacher forcing method with a ratio of 0.5 on the model. The teacher forcing improved the score slightly, but the translation still contained the same word repeated in the whole sentence.
   (b) **Phase 2** : In this phase, two new models with an LSTM encoder and decoder are trained. The first model has a single layer of LSTM in the encoder and decoder. The embedding dimension and hidden size of LSTM are set as 300 and 512, respectively. The LSTM model produced relatively better translation than the GRU model. The second model has two layers of LSTM in the encoder and decoder to further improve the performance. The rest of the model is the same as the first model. The second model performed better on training data sets and produced a better result on validation data. The translation from LSTM produced multiple words, but there still exist some repetitions.
   PyTorch LSTM Seq2Seq Model Summary for Second Model :

```
seq2seq(
(encoder)
: Encoder(
    (embedding)
    : Embedding(40447, 300)(lstm)
    : LSTM(300, 512, num_layers = 2)
    )
(decoder)
: Decoder(
    (embedding)
    : Embedding(29686, 300)(lstm)
    : LSTM(300, 512, num_layers = 2)(out)
    : Linear(in_features = 512, out_features = 29686)
    )
)
```

   (c) **Phase 3** : In this phase, evaluation on the dev set produced terrible results from the previously saved two-layer LSTM Encoder-Decoder model. A new model with Bidirectional 2 Layer Encoder and Decoder is trained. Applied Dropout in embedding and LSTM layers to compensate for model complexity. Gradient Clipping with clip norm of one is applied while training. The model performed better than the previous LSTM model on the validation set. The saved LSTM model performed poorly in phase 3 because the word2index ordering changed on different execution due to random shuffling of the training data set. So, for the further model, the whole training data set was used for tokenization.
   PyTorch 2 layer BiLSTM Seq2Seq Model Summary :

```
seq2seq(
(encoder)
: Encoder(
    (embedding)
    : Embedding(21105, 300)(lstm)
    : LSTM(300, 512, num_layers = 2, dropout = 0.2, bidirectional = True)
    )
```

```
(decoder)
: Decoder(
    (embedding)
    : Embedding(18989, 300)(lstm)
    : LSTM(300, 512, num_layers = 2, dropout = 0.2, bidirectional = True)(dense)
    : Linear(in_features = 1024, out_features = 18989, bias = True)(softmax)
    : LogSoftmax(dim = 1)
    )
)
```

(d) **Phase 4/ Final Phase** : In this phase, an Attention Mechanism[2] is applied to both GRU and LSTM seq2seq model. The attention mechanism significantly improves the model performance. Attention LSTM performed better than the GRU Attention model. However, the model, due to its complexity, overfits the training data. Further experiments with higher dropout, lower embedding dimension, and smaller hidden size reduce overfitting on validation data. The validation loss kept decreasing for 40 epochs, but the BLEU and METEOR scores did not show any improvement.
PyTorch 2 layer BiLSTM Seq2Seq Model with Attention Mechanism Summary :

```
seq2seq(
(encoder)
: Encoder(
    (embedding)
    : Embedding(21105, 300)(dropout)
    : Dropout(p = 0.2, inplace = False)(rnn)
    : LSTM(300, 512, num_layers = 2, dropout = 0.25, bidirectional = True)(fc_hidden)
    : Linear(in_features = 1024, out_features = 512, bias = True)(fc_cell)
    : Linear(in_features = 1024, out_features = 512, bias = True)
)
(decoder)
: DecoderAttn(
    (embedding)
    : Embedding(18989, 300)(dropout)
    : Dropout(p = 0.2, inplace = False)(rnn)
    : LSTM(1324, 512, num_layers = 2, dropout = 0.2)(dense)
    : Linear(in_features = 1836, out_features = 18989, bias = True)(softmax)
    : LogSoftmax(dim = 1)(attention)
    : Attention(
        (attn)
        : Linear(in_features = 2048, out_features = 1024, bias = True)(v)
        : Linear(in_features = 1024, out_features = 1, bias = False)
        )
    )
)
```

2. **Final Model** - The model summary in phase 4 is of the model which performed best on the final test set. The encoder has a two-layer BiLSTM with a dropout of 0.25. The dimensions of hidden and cell state outputs from BiLSTM are halved using two fully connected layers. The decoder has a two-layer LSTM with a dropout of 0.25 and an attention mechanism. The output from LSTM is passed to a fully connected layer with the LogSoftmax activation layer. Since this model also suffers from overfitting. The best model weight with maximum BLEU and METEOR score is selected from the saved model weights.
3. **Loss Functions** - The CrossEntropy Loss and Negative Log-Likelihood Loss were used as loss functions. Both performed equivalent. There was no significant difference with the final result from both these losses. Using NLLLoss required the Activation Layer to be LogSoftmax. The only observed difference is that NLLLoss decreased epoch time compared to CrossEntropy Loss. NLLLoss is the loss function for most of the models.

4. **Decoding Strategies** - Greedy and Beam Search[3] are the two decoding strategies tested on the model. Using the Beam Search produced multiple sentences from which sentence with maximum score is returned. Beam Search decoding was very slow compared to greedy search. On evaluating both the strategy on a subset of validation sets (5000 pairs), the BLUE score and METEOR score were better in the greedy strategy. So for most of the model and phase, greedy decoding is used. Also, greedy decoding produced a better BLEU score in the final test set. However, Beam Search with beam length of 2 produced better METEOR score.

5 Experiments

1. **PreProcessing** - The noisy sentences are filtered from the training dataset.For both the language, every punctuation except the '.' and '—' are removed. The English sentences are lowered, then some abbreviated words are replaced with their complete form like " n't to not ", " 've to have." The Spacy tokenizer is used to tokenize the English sentences. The Hindi sentences are first normalized using the IndicNormalizerFactory [4]. The normalizer also removes the nuktas from the sentences. After removing the punctuations, the IndicTrivialTokenizer is used to tokenize the processed string.

   For the first 3 phases, the whole number is separated to separate digits for both languages. For example, '33103' is separated to '3', '3', '1', '0' and '3'. This step reduced the random numbers as tokens and produced only ten numbers as tokens for both languages. Also, it allowed the model to learn translation for each Hindi number to its equivalent English number. However, this step required removing spaces after generating the output from the model. This step was not allowed for the competition. So numbers were mapped as unknown tokens.

   The vocabulary for both the language is generated. This vocabulary contains a total number of tokens, count of each token, word2index, and index2word. Both the language vocabulary contain four additional tokens, START_TOKEN, END_TOKEN, UNK_TOKEN, and PAD_TOKEN.

   For all the phases, the provided data set was divided into train and validation data sets. For the first three phases, only the training set is used to generate the vocabulary. In the final phase, the whole data set is used to produce the vocabulary. Also, the tokens with a single count are eliminated and mapped to UNK_TOKEN. Also, after phase 3, all the words having single occurrence are mapped to unknown tokens.
   The maximum token length is fixed as 32 as 95% of sentences are less than 30. Two tokens for each sentence are reserved as START_TOKEN and END_TOKEN. For sentences having lower length than maximum length, PAD_TOKEN are padded till maximum length.
2. **Optimizer** - Stochastic Gradient Descent Optimizer is used only for the first model. Training Loss decreased very slowly. For all the other models, Adam Optimizer is used. The default learning rate of Adam worked well for the models.
   For phase 4, the final model on training with default learning rate started overfitting after 15 epoch. The default learning of adam is 0.001, and weight decay is 0. Applying slower learning rate 0.0005 and adding weight decay (l2 norm of weight) as 1e-6 solved overfitting. The validation kept decreasing for 40 epochs, but lower validation loss did not produce a better translation(BLEU and METEOR). So, for the final model, adam optimizer with default parameters is used.

   Only the first model GRU-based encoder-decoder with max sentence length as 12 was trained for 300 epochs. The rest of the model was trained only for 40 epochs because the max sentence length was 32. Training time for every model per epoch was around 10 minutes. However, the training time was inconsistent as different GPU were assigned on different sessions on Google Colab. Also, randomly, the GPU gets busy during the sessions, and some epochs took more than an hour. However, on average, training for 40 epochs took around 6 hours for every model.
3. **Hyper parameters** For the final model, the default learning rate of 0.01 was used. After ten epochs, the validation loss started increasing even though the training loss decreased, i.e., the model started overfitting. Using the learning rate 0.001 made the model overfit after 30 epochs,

and the learning rate of 0.0005 allowed the model to not overfit for 40 epochs. However, decreasing the learning rate did not decrease the validation loss significantly. Adding some weight decay solved overfitting, the validation loss kept decreasing significantly. Dropout layer after embedding layer is used with dropout ratio as 0.2. In 2 layer LSTM encoder-decoder, the dropout ratio is set as 0.25. For every model, the model dictionary is saved after every five epochs till 40 epochs. After testing each saved model, the best performing model in terms of BLEU score and METEOR score was the two Layer LSTM Encoder-Decoder with LSTM with default learning rate and zero weight decay trained for ten epochs. Even the weight decay improved the validation loss; it did not improve the metrics.

6 Results

- Val Set : 20% of the training dataset used as validation set.
- Test Set : The weekly dev set and final test set.
- $*$ Represent the best model for the phase and leaderboard rank corresponding to the model.

The results of various models for various phases are tabulated below:

1. Phase 1 : Leaderboard Rank - 70
   Common : Max Sentence Length = 17, SGD Optimizer with learning rate = 0.01, Epochs = 300

Model Seq2Seq	BLUE Score	METEOR Score
Encoder - GRU Layers(1) Decoder - GRU Layers(1)	0.0000	0.0017
* Encoder - GRU Layers(1) Decoder - GRU Layers(1) Training - Teacher Forcing with tf ratio 0.5	0.0000	0.0045

2. Phase 2 : Leaderboard Rank - 28

Model Seq2Seq	BLUE Score	METEOR Score
*Encoder - LSTM Layers(2) Decoder - LSTM Layers(2) Max Sentence Length = 32 Learning Rate = 0.001 Training - Teacher Forcing with tf ratio 0.5 Epochs = 20	Val Set - 0.0221 Test Set - 0.0120	Val Set - 0.2482 Test Set - 0.1493

3. Phase 3 : Leaderboard Rank - 22
   Common : Max Sentence Length = 32, Adam Optimizer with lr = 0.001, Teacher Forcing with tf ratio = 0.5, Epochs = 40

Model Seq2Seq	BLUE Score	METEOR Score
Encoder - LSTM Layers(2) Decoder - LSTM Layers(2)	Val Set - 0.0221 Test Set - 0.0017	Val Set - 0.2482 Test Set - 0.0692
*Encoder - BiLSTM Layers(2) Dropout(0.2) Decoder - BiLSTM Layers(2) Dropout(0.2)	Val Set - 0.0341 Test Set - 0.0186	Val Set - 0.3032 Test Set - 0.1748

4. Final Phase : Leaderboard Rank - 17
   The evaluation script for bleu score and meteor in final phase is different from previous evaluation script. The updated evaluation script is case insensitive.
   Common : Max Sentence Length = 32, Adam Optimizer, Max Epochs = 40

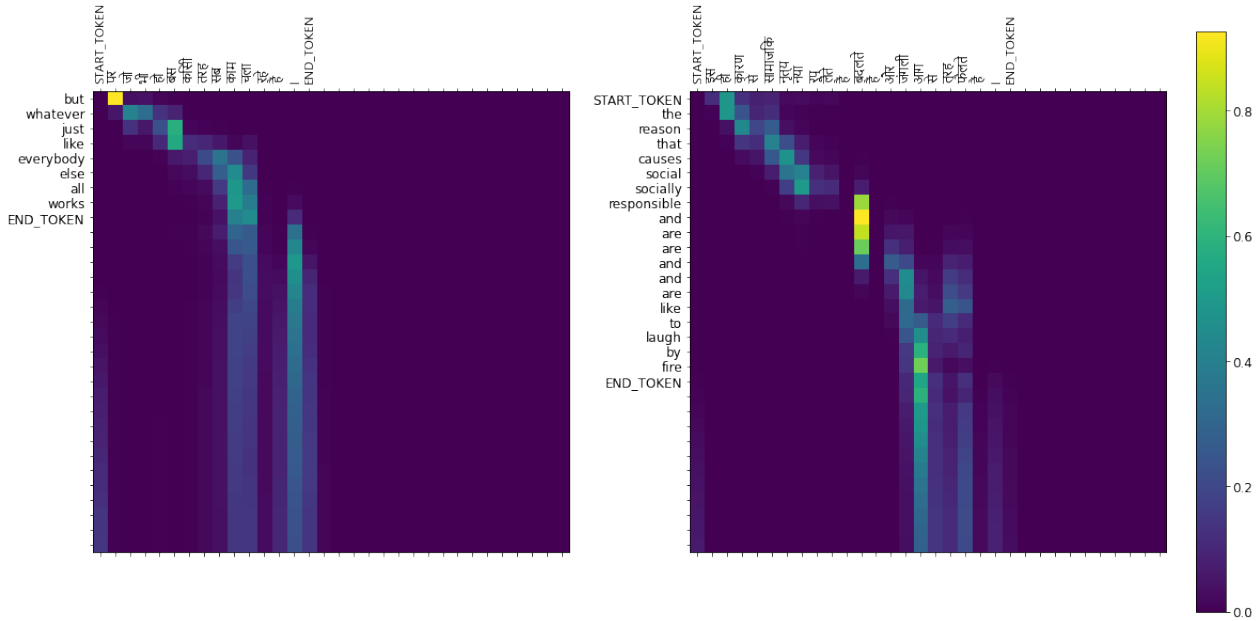
Model Seq2Seq	BLUE Score	METEOR Score
Encoder BiGRU Layers(1) Dropout(0.2) Decoder GRU Layers(1) Dropout(0.2) Attention Mechanism Training Methods: Gradient Clipping(1) Teacher Forcing Ratio = 0.5 Epochs = 40	Val Set - 0.0336 Test Set - 0.0482	Val Set - 0.3181 Test Set - 0.2650
Encoder BiLSTM Layers(1) Dropout(0.2) Decoder LSTM Layers(1) Dropout(0.2) Attention Mechanism Training Methods: Gradient Clipping(1) Teacher Forcing Ratio = 0.5 Epochs = 20	Val Set - 0.0513 Test Set - 0.0613	Val Set - 0.3726 Test Set - 0.2957
Encoder BiLSTM Layers(2) Dropout(0.25) Decoder LSTM Layers(2) Dropout(0.25) Attention Mechanism Training Methods: Gradient Clipping(1) Variable Teacher Forcing Epoch = 10	Val Set - 0.0550 Test Set Greedy Decode - 0.0751 Beam Search - 0.0733	Val Set - 0.3807 Test Set Greedy Decode - 0.3090 Beam Search - 0.3106

The best performing model was trained for 40 epochs, but the weights after 10 epochs performed best on validation set. The final model with Greedy Decode produced best BLEU score 0.0751, but using Beam Search produced best METEOR score 0.3106 on final test set.

7 Error Analysis

1. In the first phase, the GRU models were not appropriately trained. Due to some error in the training process, the model failed to perform. The translation from these model was just the first word repeated throughout the whole sentences. In the second phase, the training process was corrected, and the LSTM models helped to improve the translation. In the third phase, the saved LSTM failed to perform due to randomness in generating Language token indexes. In the BiLSTM model with some dropout, the whole training dataset with no randomness is used to generate the Language (word2index, index2word). In the final phase, the Attention mechanism was applied to GRU based and LSTM based seq2seq models. The scores after the attention mechanism were significantly improved. The result shows a lower score on the validation set because the evaluation script was case sensitive.

2. The final model severely overfits after ten epochs. Several other models with higher dropout (0.5), lower embedding dimension, hidden size, and optimizer with lower learning rates and weight decay to overcome the overfitting. After training for 40 epoch, this model did not overfit the training data. However, the validation score was significantly lower than the final model. The model was learning very slowly, and it needed more training to achieve the optima.
3. Aftering removing the infrequent words (words having single occurence), the translations from training and validations set started having UNK_TOKENS very much. The model were becoming biased toward the UNK_TOKENS . For longer sentences, the model were not able to predict proper translation. The model produced proper translation for sentence with no unknown tokens. As the number of unknown tokens like proper nouns, numbers etc. increased, the quality of the translation started deteriorating. Also, the models faced difficulty when translating paragraph type texts (single instance containing multiple sentences).
4. Attention Heatmaps : Below is the attention heatmap for two of the sentences from Final Test Dataset. The heatmap shows on what words are being focussed while decoding the english words.



8 Conclusion

In this competition, the seq2seq model with various rnn cells in different configuration were explored. Some of the key findings were:

- LSTM seq2seq models perform better than GRU based models.
- Attention mechanism improved the translation significantly.
- Increasing the number of RNN layers improves performance.
- Decreasing the model complexity and adding regularizers like Dropout and Weight Decay improved overfitting but require more epochs to converge.

Possible future direction includes:

- Applying the Transformer models.
- Using the Pretrained embeddings.
- Training the final model with regularizers for more epochs to converge.

References

[1] S. Robertson, "Pytorch Tutorial." https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html.

[2] elvis, "Attention Mechanism Tutorial." https://medium.com/dair-ai/neural-machine-translation-with-attention-using-pytorch-a66523f1669f, 2019.

[3] J. Brownlee, "Beam Search Tutorial." https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/.

[4] A. Kunchukuttan, "The IndicNLP Library." https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf, 2020.