Competition: Hindi to English Machine Translation System

Debanjan Chatterjee
20111016
{debanjan20}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

Machine Translation is the process of automatically translating text from one natural language to another. Neural network-based models have outperformed traditional rule-based and statistical MT models. In this competition, the task was to develop a NMT system that was capable of translating Hindi text to English. In this report, a NMT has been proposed based on sequence-to-sequence architecture using gated RNNs such as LSTM and GRU. Techniques such as teacher forcing and attention have also been incorporated. The NMT system has achieved a BLEU score of 0.0161 and a METEOR score of 0.141 in the final phase of the competition.

1 Competition Result

Codalab Username: D_20111016
Final leaderboard rank on the test set: 53
METEOR Score wrt to the best rank: 0.141
BLEU Score wrt to the best rank: 0.0161
Link to the CoLab/Kaggle notebook:
https://colab.research.google.com/drive/1vEjgzmK1VXpTyo8vguvnce7Nk8OpH_LT?usp=sharing

2 Problem Description

Machine Translation is the process of automatically translating text from one natural language to another with the help of a computer system. The language from which the text is translated acts as the source language, while the language to which the language is translated to is the target language. The use of neural networks in machine translations tasks has achieved admirable results with sequence-to-sequence models employing recurrent neural networks. The task at hand is to design a neural machine translation system to translate text in Hindi to English.

3 Data Analysis

The given dataset consists of 102322 pairs of Hindi and its translated English sentences. Therefore each training sample will be a (Hindi,English) sentence pair, and there are 102322 training samples in the dataset. In machine translation tasks, the length of the source sentences play a significant part in determining the quality of translation. Longer sentences are more difficult to translate by MT models as the models need to have the capability to retain long term dependencies.

The average length of Hindi sentences was 12.77 words in the training dataset and was 10.87, 10.92, 10.87 and 11.07 words in the development and test set. Figure 1 shows the graphical representation of the average length of Hindi sentences throughout the various phases. As the mean length of the Hindi sentences is roughly the same in both the train and test set, the quality of translation should be unaffected by length.

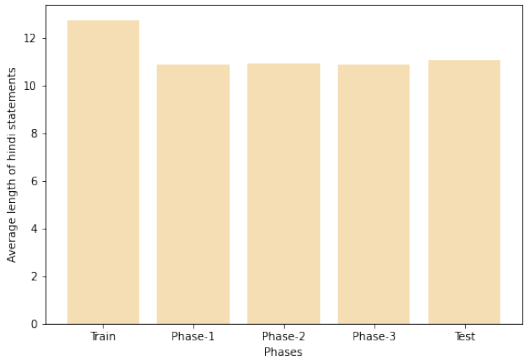

Figure 1: Average length of the Hindi statements over the various phases

The average length of English sentences is 9.68 words. Figure 2 represents the distribution of the frequency of occurrence of the words over all the English sentences. This shows that words in a language have a long tail distribution, that is, most of the probability mass is distributed over only a few words. This also confirms that word usage in language follows Zipf's law [1].

Apart from quantitative analysis, there is a need to perform a qualitative analysis on the dataset. The quality of the training data greatly influences the performance of the machine translation model developed. There are several training samples that appear to be noisy. A few examples of noisy samples is presented in the following list:

- Translation pairs like - *('Cezanne.','Cezanne.')*.
- Translation pairs where English words appear in the source Hindi sentences.
- Incorrect translations; clearly understood by a large mismatch in the lengths of the Hindi and English sentences.
- Translation pairs with inaccurate translations.

In the Section 5, some prepossessing techniques have been discussed which can filter the first three types of noise. However, sentences with inaccurate translations are hard to detect and they might lower the performance of the neural machine translation system developed.

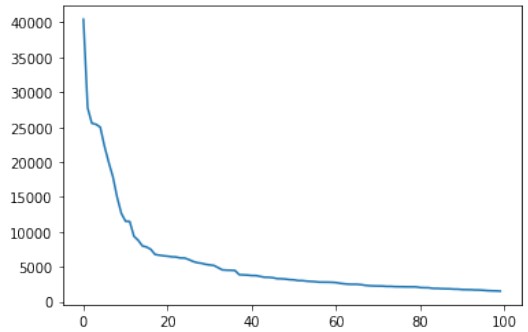

Figure 2: Word frequency distribution in the English sentences following a long tail distribution

4 Model Description

Automated translation is a very challenging problem for computers to perform because natural languages are complex and it quite difficult to retain the semantic meaning while translating text from one language to another. Most of the early work done in this field was machine translation system using rule-based models [2] and statistical models [3]. However, with the advent of deep learning,

neural machine translation has emerged as the most popular machine translation approach in recent times, showing substantial rise in performance [4].

Sequential data such as natural languages have innate long-term dependencies, which ordinary feed-forward neural networks cannot capture. Therefore to capture the long-term dependencies, a special class of neural networks that uses recurrence to mimic internal memory. This type of neural network is called the recurrent neural network and a typical RNN cell is illustrated in Figure 3. Formally, we express the computations occurring inside RNN cell as follows.

$$h_t = f_W(x_t, h_{t-1}) \tag{1}$$

where, h_t is the hidden state at time step t, which is expressed as a function, of x_t, input of time step t, and h_{t-1} the hidden state of previous time step, with weight parameters W. For $t = 0$, we have $h_t = 0$ as well.

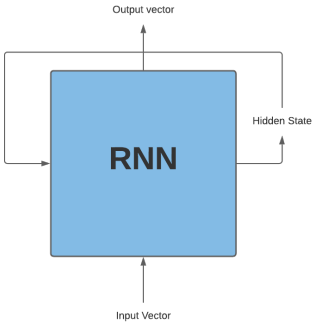

Figure 3: A recurrent neural network unit

Recurrent neural networks are capable of tacking variable length sequences, tracking long term dependencies, preserving ordering information in natural language sentences, thus they are a great fit for machine translation. However, recurrent neural networks suffer from two drawbacks, namely the vanishing gradient problem and the exploding gradient problem.

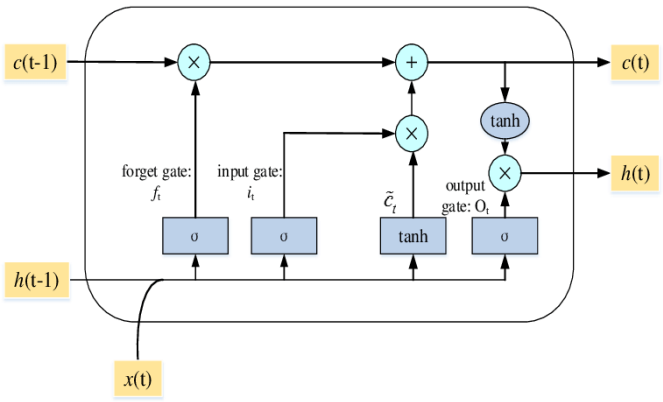

Figure 4: Long Short Term Memory (LSTM) unit [5]

For training recurrent neural networks, a technique called back-propagation through time, vanishing gradient, and exploding gradients are two problems associated with computing gradients during back propagation through time. Since gradient computation use chain rule, it involves a long series of multiplication of gradients. If a large number of gradients are less than 1, then the computed gradient will become almost zero, or vanish. The vanishing gradient problem can be mitigated by using special recurrent units using gates to regulate the flow of information through them. Two of the most

popular recurrent neural networks using gated cells are Long Short Term Memory(LSTM) and Gated Recurrent Unit (GRU). Both of them are excellent for tracking long term dependencies in sequences and handling the vanishing gradient problem [6]. The exploding gradients problem takes place when most of the gradients computed via chain rule are greater than 1, and this can lead to an extremely large gradient. This problem can be solved by a technique called gradient clipping. Gradient clipping is the process of clipping the gradient if its norm crosses a threshold value.

LSTMs work on the basic principle of forgetting irrelevant parts of the previous states through the *forget* gate, storing relevant new information into the cell state through the *input* gate, updating the cell state values selectively via the *update*, and the information passed to the next time step via the *output* gate. Figure 4 represents the graphical structure of an LSTM cell.

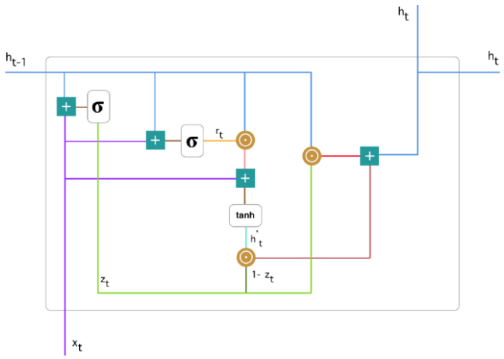

Figure 5: Gated Recurrent Unit (GRU) cell [7]

Gated Recurrent Units or GRUs were first proposed in [8].GRUs combines the *forget* and *input* gates into a single gate. It also merges the cell state and hidden state [9]. The resulting model is simpler than LSTMs, and has also shown excellent results. Figure 5 represents the graphical structure of a GRU unit or cell.

Deep neural networks have excelled on difficult learning problems if they are trained with huge labeled datasets. However, they fail to learn tasks which involve mapping one sequence to another sequence. [10] proposed an end-to-end approach for sequential learning tasks using a recurrent neural network (*encoder*) to map the input sequence to fixed dimensionality-vector (*context vector*), and then another recurrent neural network (*decoder*) to decode the target sequence from the vector. Figure 6 illustrates the diagrammatic representation of a standard sequence-to-sequence architecture.

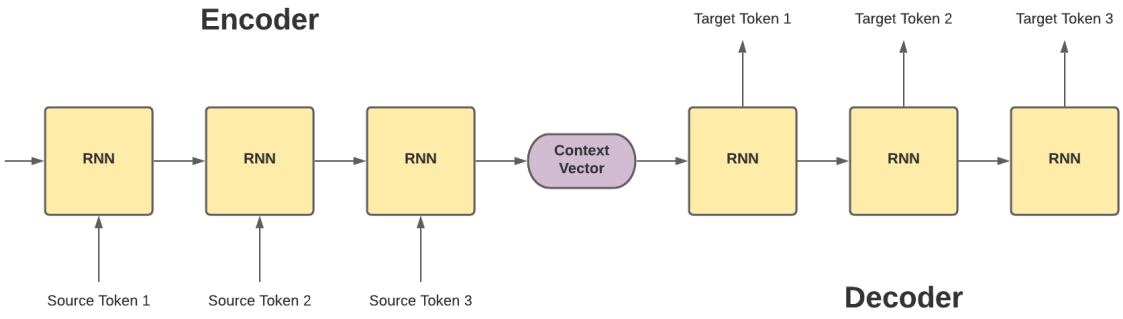

Figure 6: Sequence-to-sequence architecture for machine translation.

One drawback of the encoder-decoder based sequence-to-sequence approach is that the encoder

needs to compress the relevant information of the Hindi sentence into a vector of fixed dimensionality. This will hinder the ability of the model to process long input sequences. A solution to this problem was proposed in [11] which is the use of an attention mechanism. Attention helps to deal with both the task of alignment which identifies which portions of the source sentence are important to each decoded word in the target sentence. Instead of relying on the single fixed-dimensionality context vector which represents the entire encoding of the source sentence, the attention model develops a context vector that is filtered specifically for each decoded word by storing the context vectors at each time step of the encoder in simpler words, all the encoded outputs.

Teacher forcing is another special technique used which can be used during training to make the sequence-to-sequence architecture more efficient. Instead of passing the predicted or decoded word at a time step as input for the next time step, one can pass the actual word which was expected. This is referred to as teacher forcing and it can make the training more effective, and in turn, increase the accuracy of the model. However, always teacher forcing the actual word might lead to some kind of over-fitting. Hence, in practice, it is better to sample the use of teacher forcing through some probability distribution.

The optimization of recurrent neural networks using back-propagation through time is an example of a non-convex optimization problem hence the choice of learning rate can greatly impact the quality of training. Optimizers with adaptive learning rates such as Adam are recommended to tackle this problem. Cross Entropy Loss function was chosen as the objective function, as it is a standard loss function across machine translation literature. For the challenge, a sequence-to-sequence architecture has been presented, which explores various combinations of LSTMs and GRUs in different phases of the competition. All the experiments performed have been explained in detail in the next section.

5 Experiments

As for all machine learning related tasks, the quality of the training data can play a big role in determining the performance of the trained model. Hence, instead of directly working with the raw dataset, there is a need to clean the dataset, to remove the dataset. Our dataset consists of 102,322 pairs of Hindi and its translated English sentences. The proposed technique of pre-processing and cleaning the dataset makes the following assumptions. Firstly, the length of the Hindi and English sentences must not be excessively large, as long sentences might be noisy and will confuse our neural translation model. Secondly, if the Hindi and English sentences consist of only one word, we filter such pairs as they are too short to be considered as sentences. Thirdly, the length of the Hindi sentence and the English sentence should be roughly the same, and that makes intuitive sense. To ensure these assumptions the entire dataset needs to be tested against the following conditions.

1. Length of sentence in terms of words of both the Hindi and its English counterpart must be less than thirty.
2. Length of both the Hindi and English sentences must be greater than one.
3. Length of both sentences must be less than two times its counterpart.

After filtering the data based on the above conditions, we are left with 69,589 clean Hindi-English pairs. Figure 7 is a graphical representation of the dataset distribution. For the task of machine translation, removing punctuation from the source or target languages might improve the neural machine translation system as it helps to reduce the vocabulary size.

After cleaning the dataset, the Hindi and English vocabularies are created for the respective sentences. Also note, the *Spacy* and the *IndicNLP* libraries were used to tokenize the respective sentences. The Hindi vocabulary consists of 37,397 and the English vocabulary consists of 27,714 words or tokens, including the start of sentence and end of sentence tokens. Now that the pre-processing pipeline and the vocabularies have been set, experiments on the models were performed phase-wise.

Phase 1: A sequence-to-Sequence architecture was used, with both Encoder and Decoder being LSTMs with two recurrent layers and a hidden state of 1240. Sampled teacher forcing was also adopted during training. Dropout and early stopping were used for regularization to prevent the NMT model

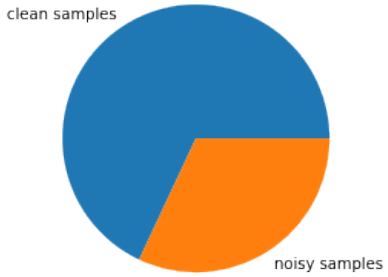

Figure 7: Dataset distribution with respect to clean and noisy samples

from over-fitting. The Adam optimizer was used to solve the non-convex optimizer. The use of LSTM is necessary to deal with the vanishing gradient problem, and gradient clipping was used to deal with exploding gradients. The cross entropy loss function was used, which is a standard loss function used in machine translation. Greedy search was used as the decoding strategy.

Phase 2: A sequence-to-Sequence architecture was used, with both Encoder and Decoder being LSTMs with three recurrent layers. Sampled teacher forcing with a probability of 0.5 was also adopted during training. Dropout and early stopping were used for regularization to prevent the NMT model from over-fitting. The Adam optimizer was used to solve the non-convex optimizer. The use of LSTM is necessary to deal with the vanishing gradient problem, and gradient clipping was used to deal with exploding gradients. The cross-entropy loss function was used, which is a standard loss function used in machine translation. A batch size of 20 samples was used. The decoding strategy used was greedy search.

Phase 3 and Final Phase: Sequence-to-Sequence Architecture was used, where the encoder was a GRU with the following design choices (Number of recurrent layers- 1 , number of features in hidden state- 512). The decoder was also a single layered-GRU implemented with an inner product attention mechanism, along with the following design choices. I have used dropout during training for regularization. Teacher forcing was also employed while training. Adam was used as the optimizer (with an initial learning rate 0.0001). Also, the loss function used was cross-entropy loss and a batch size used was 1. The decoding strategy used was greedy search.

6 Results

Initially, a standard sequence-to-sequence was developed. The encoder was a LSTM with 2 recurrent layers having 512 hidden state features, and the decoder was also an LSTM with the same configuration. This yielded a METEOR score of 0.033. The reason for poor performance might be due to over-fitting, as no techniques to regularize the model were used. The same sequence to sequence architecture was trained again, but this time along with dropout with probability 0.5. Sampled teacher forcing with probability 0.5 was also used. This model improved the METEOR score to 0.082 and attained a BLEU score of 0.0003. This was the best-performing model of phase 1.

In an effort to improve the model developed in phase one, one additional recurrent layer was added to the encoder and decoder LSTMs, and the hidden size was reduced to 512. This resulted in a significant boost to the BLEU score jumping up to 0.00802. However, increasing the number of recurrent layers to 4, reduced the BLEU score to 0.0071. Another improvement was observed by incorporating the gradient clipping technique to prevent exploding gradients, this resulted in the best performing model of phase 2 with a BLEU score of 0.0089 and METEOR score of 0.13

After exploring different models using LSTMs, it was time to experiment GRUs. A sequence-to-sequence model with GRUs was developed. The encoder was a GRU with single recurrent layers having

512 hidden state features, and the decoder was also a GRU with the same configuration. It achieved a score BLEU score of 0.0026. Adding sampled teacher forcing with a probability 0.5 improved the BLEU score of 0.0035. Implementing the decoder along with attention, achieved a BLEU score of 0.0051 and METEOR score of 0.118, which was my best performing model for phase 3. The same NMT system was trained for 10K fewer epochs and submitted in the final phase. In the final phase of the competition a BLEU score of 0.0161 and METEOR score of 0.141. Table 1 summarizes the results based on the best model in each phase of the competition.

Phase	BLEU	METEOR	Rank
1	0.0003	0.082	46
2	0.0089	0.13	33
3	0.0051	0.118	44
Final	0.0161	0.141	53

Table 1: Results during various phases of the competition

7 Error Analysis

As observed from the various experiments and their corresponding results, the best model seems to be a sequence-to-sequence model using LSTMs/GRUs as the encoder and decoder. Special techniques like teacher forcing and attention also seemed to improve the model significantly. During training, incorporating gradient clipping and dropout is also beneficial. Despite using these techniques, the neural machine translation systems seem to be frequently making some frequent errors and some of these errors have been discussed below.

A Hindi statement which should be translated to "they need you at the ship', instead has been translated to "they're just need to you" which is interesting because the neural machine translation model was able to successfully translate the first part of the sentence relatively well. This indicates that the current NMT model is still unable to capture the longer dependencies towards the end of the sentence. A remedy to this would be using bidirectional gated recurrent neural networks as the encoders and decoders or using a transformer-based model. Both of them handle long-term dependencies better than standard LSTMs or GRUs.

There is another observation, one of the sentences by the NMT system is "you should be to be a a a a a", now in this translation, there is a repetition of 'a'. This might be resolved by using a different decoding strategy like beam search [12]. Since beam search selects the beam with the largest joint probability as the decoded words, it is unlikely a long pattern of repetitive words will be decoded by beam search.

8 Conclusion

To conclude, a sequence to sequence architecture, with a gated recurrent neural network like LSTM and GRU works well. Advanced techniques like teacher forcing and attention mechanism seem to improve performance. During training, over-fitting needs to be avoided, hence deep learning regularization techniques like dropout and early stopping should be adopted. Since the optimization problem while training RNNs is non-convex, optimizers with adaptive learning rates such as Adam seem to work well in practice. For future work, the first change that could have been made to possibly improve performance is to change the decoding strategy from greedy search to beam search. Another interesting change would be to try out transformer-based architectures, as they have shown extraordinary performances in most natural language processing-based tasks.

References

[1] S. T. Piantadosi, "Zipf's word frequency law in natural language: A critical review and future directions," *Psychonomic bulletin & review*, vol. 21, no. 5, pp. 1112–1130, 2014.

[2] M. L. Forcada, M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J. A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F. M. Tyers, "Apertium: a free/open-source platform for rule-based machine translation," *Machine translation*, vol. 25, no. 2, pp. 127–144, 2011.

[3] P. Koehn, *Statistical machine translation*. Cambridge University Press, 2009.

[4] P. Koehn and R. Knowles, "Six challenges for neural machine translation," *arXiv preprint arXiv:1706.03872*, 2017.

[5] X. Yuan, L. Li, and Y. Wang, "Nonlinear dynamic soft sensor modeling with supervised long short-term memory network," *IEEE transactions on industrial informatics*, vol. 16, no. 5, pp. 3168–3176, 2019.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[7] S. Kostadinov, "Understanding gru networks." `https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be`, Dec 2017. Accessed on 2021-04-26.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[9] C. Olah, "Understanding lstm networks." `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, Aug 2015. Accessed on 2021-04-26.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*, 2014.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[12] M. Freitag and Y. Al-Onaizan, "Beam search strategies for neural machine translation," *arXiv preprint arXiv:1702.01806*, 2017.