Neural Machine Translation of Indian Languages

A report submitted to the
Indian Institute of Technology, Kharagpur
in partial fulfilment of the requirements for the degree

of

Master of Technology

by

Yerra Pranay Hasan
[11CS30042]

under the supervision of

Prof. Sudeshna Sarkar

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

November 2015

Abstract

Neural machine translation is a new approach to machine translation, where we train a single, large neural network to maximize the translation performance. This is a radical departure from existing (phrase-based) machine translation approaches, where a translation system consists of many subcomponents which are optimized separately.

Various deep learning architectures such as deep neural networks, convolutional deep neural networks and recurrent neural networks have been applied in Machine Translation of English and Foreign languages which have shown to produce state-of-the-art results. We study variants in architectures of Recurrent Neural Networks in MT. But we focus on English-Hindi pair mostly due to abundance of parallel data compared to other Indian languages. Finally we analyse and compare the above Recurrent Neural Network models and provide some sample translations.

Acknowledgements

I take this opportunity to express my profound gratitude and deep regards to my guide Prof. Sudeshna Sarkar for the exemplary guidance, monitoring and constant encouragement throughout the course of this work. The blessing, help and guidance given by her time to time shall carry me a long way in the journey of life on which I am about to embark. I wish to acknowledge the encouragement received from research scholar Ayan Das for initiating my interest in this topic and also for his unparallelled help and motivation round the clock to carry out my project work. Lastly, I would like to thank all professors, lab administrators and friends for their help, moral support and wishes.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Neural Machine Translation is recently proposed approach for Machine Translation(MT). Unlike the traditional Statistical MT, Neural MT builds a single Neural Network which can be tuned to maximize the translation performance. Most of the models proposed recently for Neural MT belong to the family of encoder-decoders(Sutskever et al., 2014; Cho et al., 2014a) that encodes the source sentence into a fixed-length vector from which a decoder generates the translated sentence.

The whole encoder-decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained for maximizing the probability of the correct translation given the source sentence. Recently, Neural Machine Translation using encoder-decoder architecture has been producing comparable results with the Baseline systems for English-Foreign languages. We verify if it is true even in case of English-Indian Languages. We demonstrate basic un-regularized Recurrent Neural Network models and compare the results with that of Moses considering it as Baseline System. We also analyse the shortcomings of each of these models.

Chapter 2

Background

2.1 Word Vectors

Each languages comprises of millions of words but they all aren't completely unrelated. Thus, word tokens can be encoded each into some vector that represents a point in some sort of "word" space. This is paramount for a number of reasons but the most intuitive reason is that perhaps there actually exists some N-dimensional space (such that $N \ll$ Vocabulary size) that is sufficient to encode all semantics of our language. Each dimension (Eg. tenses, count, gender) would encode some meaning that we transfer using speech.

2.1.1 1-of-N Coding

In this representation, every word is an $\Re^{|V|\times 1}$ vector with all 0's and one 1 at the index of that word in the sorted words, where $|V|$ is the size of our vocabulary.

2.1.2 SVD Based Methods

For this class of methods to find word embeddings (otherwise known as word vectors), we first loop over a massive dataset and accumulate word co-occurrence counts in some form of a matrix X, and then perform Singular Value Decomposition on X to get a USV^T decomposition. We then use the rows of U as the word embeddings for all words in our dictionary. Some of the SVD based methods are:

1. Word-Document Matrix

2. Word-Word Co-occurrence Matrix

2.1.3 Iteration Based Methods

Model that will be able to learn one iteration at a time and eventually be able to encode the probability of a word given its context. We can set up this probabilistic model of known and unknown parameters and take one training example at a time in order to learn just a little bit of information for the unknown parameters based on the input, the output of the model, and the desired output of the model.

At every iteration we run our model, evaluate the errors, and follow an update rule that has some notion of penalizing the model parameters that caused the error. This method is called "backpropagating" the errors. Some of these *models*[1] are

1. Language Models (Unigrams, Bigrams etc)

2. Continuous Bag of Words Model[2] (CBOW)

3. Skip-Gram Model[2]

4. Negative Sampling[2]

2.2 Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a neural network that consists of a hidden state h and an optional output y which operates on a variable-length sequence $x = (x_1, \ldots, x_T)$. At each time step t, the hidden state $h_{<t>}$ of the RNN is updated by

$$h_{<t>} = f\left(h_{<t-1>}, x_t\right) \tag{2.1}$$

where f is a non-linear activation function.

An RNN can learn a probability distribution over a sequence by being trained to predict the next symbol in a sequence. In that case, the output at each timestep t is the conditional

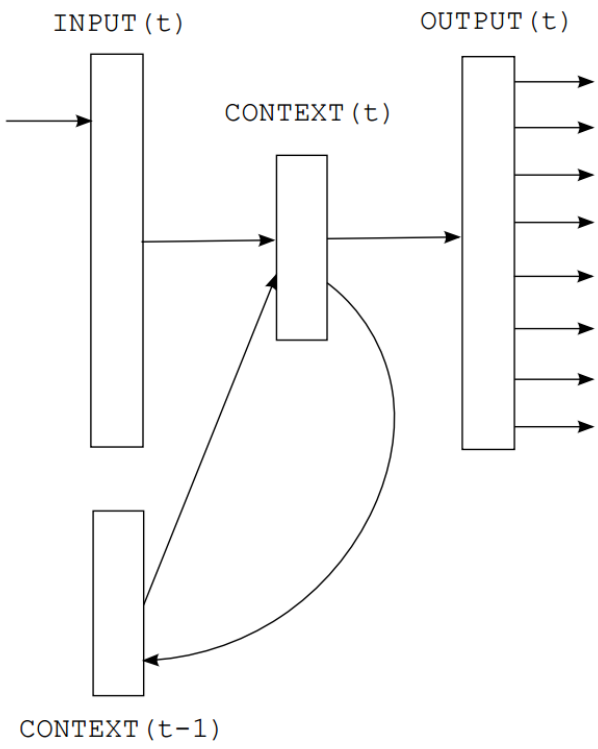


Figure 2.1: Recurrent Neural Network

probability distribution $p(x_t \mid x_{t1}, \ldots, x_1)$. For example, a multinomial distribution (1-of-K coding) can be output using a softmax activation function

$$p(x_{t,j} = 1|x_{t-1}, \ldots, x_1) = \frac{exp\,(w_j h_{<t>})}{\sum_{j'=1}^{K} exp\,(w_j h_{<t>})} \tag{2.2}$$

for all possible symbols $j = 1, \ldots, K$, where w_j are the rows of a weight matrix W. By combining these probabilities, we can compute the probability of the sequence x using

$$p(x) = \prod_{t=1}^{T} p\,(x_t|x_{t-1}, \ldots, x_1) \tag{2.3}$$

From this learned distribution, it is straightforward to sample a new sequence by iteratively sampling a symbol at each time step.

RNNs do not use limited size of context as in forward-feed neural network [4]. By using recurrent connections, information can cycle inside these networks for arbitrarily long time.

2.3 RNN Encoder-Decoder for Machine Translation

The RNN Encoder-Decoder architecture consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence. The two networks are trained jointly to maximize the conditional probability of the target sequence given a source sequence. The model is trained to learn the translation probability of a source language sentence to a corresponding target language sentence.[6]

From a probabilistic perspective, this model is a general method to learn the conditional probability distribution over a variable-length sequence conditioned on yet another variable-length sequence, e.g. $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$, where one should note that the input and output sequence lengths T and T' may differ.

2.3.1 Simple RNN

The Simple RNN or just RNN, the encoder reads the input sequence sequentially and as it reads each symbol the hidden state of RNN changes accordingly as in equation (2.1). After reading the end of the sequence (marked by an end-of-sequence symbol), the hidden state of the RNN is a summary $\mathbf{c}$ of the whole input sequence. The decoder of the proposed model is another RNN which is trained to generate the output sequence by predicting the next symbol y_t given the hidden state $h_{<t>}$.

However, unlike the RNN described in Sec. 2.2, both y_t and $h_{<t>}$ are also conditioned on y_{t-1} and on the summary $\mathbf{c}$ of the input sequence. Hence, the hidden state of the decoder at time t is computed by

$$h_{<t>} = f\left(h_{<t-1>}, y_t, \mathbf{c}\right)$$

and similarly, the conditional distribution of the next symbol is

$$P(y_t \mid y_{t1}, \ldots, y_1, \mathbf{c}) = g\left(h_{<t>}, y_t, \mathbf{c}\right)$$

for given activation functions f and g (the latter must produce valid probabilities, e.g. with a softmax). See Fig. 1 for a graphical depiction of the proposed model architecture. The

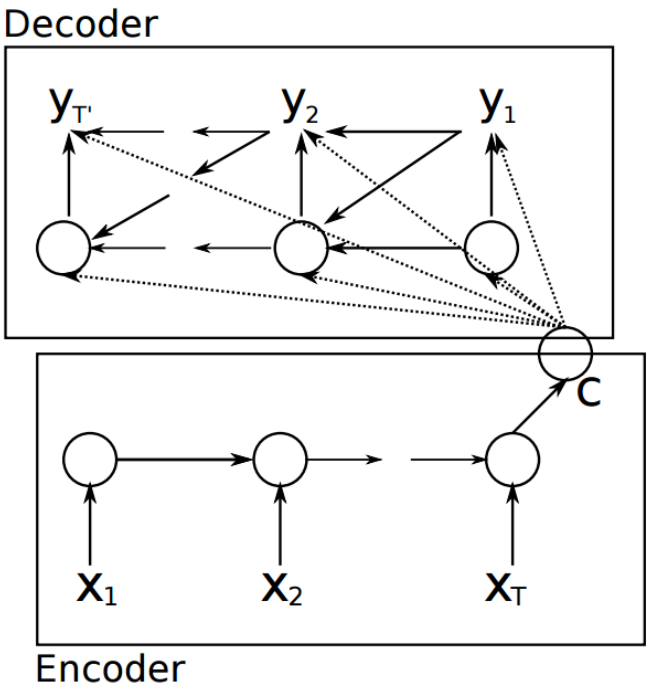


Figure 2.2: Simple RNN Encoder-Decoder

two components of the proposed RNN Encoder-Decoder are jointly trained to maximize the conditional log-likelihood

$$max_\theta \frac{1}{N} \sum_{n=1}^{N} \log p_\theta \left(y_n | x_n\right) \tag{2.4}$$

where θ is the set of the model parameters and each (x_n, y_n) is an (input sequence, output sequence) pair from the training set. In our case, as the output of the decoder, starting from the input, is differentiable, we can use a gradient-based algorithm to estimate the model parameters.

Once the RNN EncoderDecoder is trained, the model can be used in two ways. One way is to use the model to generate a target sequence given an input sequence. On the other hand, the model can be used to score a given pair of input and output sequences, where the score is simply a probability $p_\theta(y \mid x)$ from Eqs. (2.3) and (2.4).

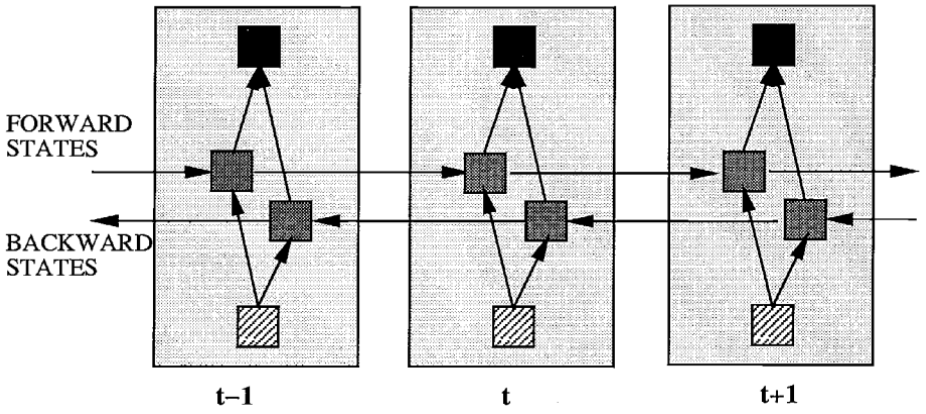


Figure 2.3: General structure of the bidirectional recurrent neural network (Bi-RNN) shown unfolded in time for three time steps

2.3.2 Bidirectional RNN (Bi-RNN)

The usual RNN, described in Eq. (1), reads an input sequence x in order starting from the first symbol x_1 to the last one x_{T_x} . However, in the proposed scheme, we would like the annotation of each word to summarize not only the preceding words, but also the following words. Hence, we use a bidirectional RNN [7].

The state neurons of a regular RNN in the encoder part are split, a part for the forward direction (forward hidden states) and a part for the backward direction (backward hidden states). Outputs from forward hidden states are not connected to inputs of backward hidden states, and vice versa. This leads to the general structure that can be seen in Fig. 2.3, where it is unfolded over three time steps. Note that without the backward hidden states, this structure simplifies to a regular unidirectional forward RNN, as shown in Fig. 2.2. If the forward hidden states are taken out, a regular RNN with a reversed input sequence. With both directions taken care of in the same network, input information in the past and the future of the currently evaluated time can directly be used to minimize the objective function without the need for delays to include future information, as for the regular unidirectional RNN discussed above.

In theory, RNNs are absolutely capable of handling such “long-term dependencies. The problem was explored in depth by Hochreiter (1991) [German] and Bengio, et al. (1994) Thankfully, LSTMs covered in next section dont have this problem!

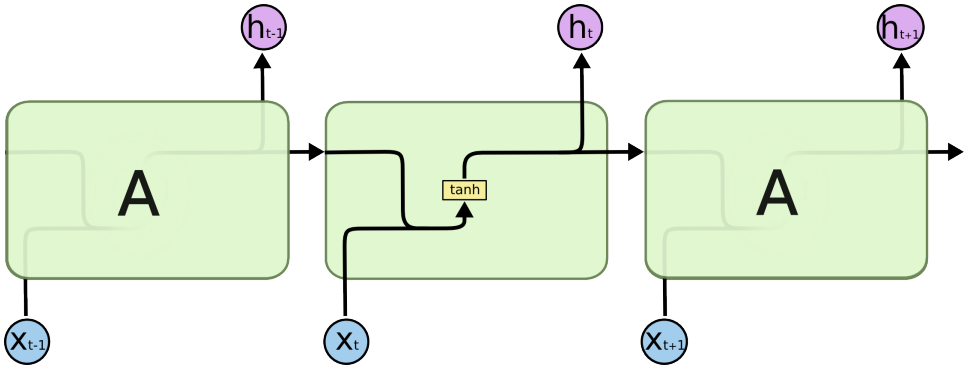


Figure 2.4: The repeating module in the standard RNN contains a single layer

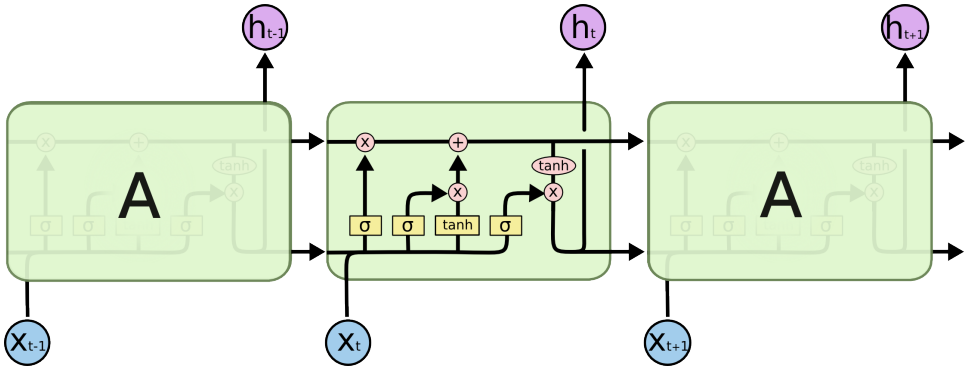


Figure 2.5: The repeating module in an LSTM contains four interacting layers

2.3.3 Long Stort-term Memory Networks (LSTMs)

Long Short Term Memory networks – usually just called LSTMs – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the recurring module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. As shown in Fig 2.6, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise

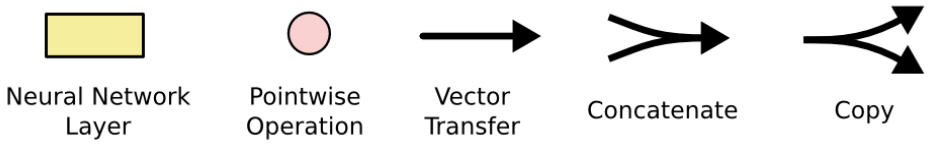
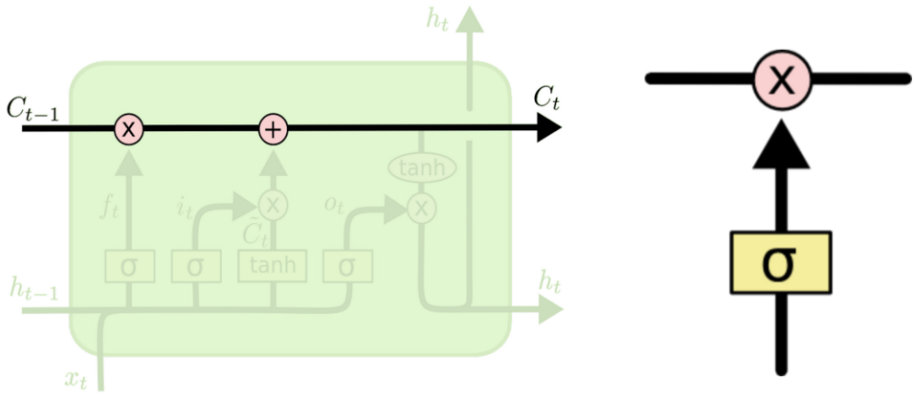


Figure 2.6:

operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.



Core Idea behind LSTM:

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. Its very easy for information to just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through!". An LSTM has three of these gates, to protect and control the cell state.

Chapter 3

Dataset

In order to perform Machine Translation, mainly we need two kinds of information
(a) A Large Monolingual corpus to build the word vectors in both the source and the target languages.
(b) Parallel corpus between the source and the target languages for training and testing the corresponding Neural Network.

The Monolingual Hindi Corpus was obtained from the ILTP-DC called ILCI corpus which is about 10GB. The Monolingual English Corpus used was obtained from wikipedia dumps. The English-Hindi bidirectional Corpus comprises of a total of 337,210 parallel sentences of which 287,210 sentences were obtained from WMT2014 comprising data in multiple domains and 50,000 sentences from ILCI Corpus corresponding to Tourism and Health domains.

Chapter 4

Methodology

Simple RNN, Bidirectional RNN and LSTM models for Machine Translation from English-Hindi are created in following steps:

4.1 Corpus Cleaning

Both monolingual and parallel corpus contains characters other than word tokens. So the corpus was appropriately cleaned and tokenized by removing these special characters except for the end of sentence markers. From the parallel sentences only 96,820 were selected having the sentence length between 4 and 12 inclusive. 96,000 parallel sentences were used for training the RNN and the rest for evaluation.

4.2 Building Word Vectors

From the monolingual corpus word vectors are built using [1] word2vec with Skipgram model and negative sampling to generate the embeddings. The dimension of each embedding was chosen to be 500. So we generate word embeddings in both the languages separately. The source and the target sentences would be encoded using these word embeddings and used as i/o to the encoder-decoder RNN models.

4.3 Training and Evaluating RNN Models

Theano was used in implementing the above language models, which has support of using GPU for optimisation. The Simple RNN (or simply RNN) and LSTM models used a single encoder and two decoders one for training and other for evaluation with a single layer each. The same encoder is used both for training and evaulation as the sentence is incoded in the same way in both the cases. The Bi-RNN model used a single encoder but has two layers in encoder, one for learning forward input sequence and other for reverse input sequence. The number of hidden neurons in each hidden layer of RNNs was chosen to be 2000 both for the encoders and decoders.

Currently the simple RNN has terminated after 37 epochs giving its best result. Bi-RNN is in its 7th epoch and LSTM in 25th epoch. The results were current of these epochs respectively.

Also moses was used as a baseline system to compare with the above models.

Chapter 5

Results

We start this section by defining metric to measure the quality of MT. Next we see some sample translations by different models like RNN, LSTM etc.

5.1 Evaluation metric: BLEU

BLEU (Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one language to another. BLEUs output is always a number between 0 and 1. This value indicates how similar the candidate and reference texts are, with values closer to 1 representing more similar texts.

5.1.1 Comparison of BLEU scores for En-Hi Machine Translation

Moses is a statistical MT system that allows you to automatically train translation models for any language pair. From the trained model, an efficient search algorithm quickly finds the highest probability translation among the exponential number of choices. The BLEU score from the translation from moses can be taken as the state-of-art score or Baseline score for a language pair.

From the above parallel En-Hi data, 96,000 sentences are used for training the SRILM language model and Mert. Evaluation was done on 820 En-Hi sentences. 30,000 parallel sentences randomly generated from the above 96,000 sentences were used for tuning the

Model	Current Iteration	BLEU score
Moses	-	20.89×10^{-2}
Simple RNN	36 (final)	11.53×10^{-2}
Bidirectional RNN	8 (running)	2.31×10^{-2}
LSTM+RNN	27 (running)	11.08×10^{-2}

Table 5.1: Comparison of BLEU scores for English-Hindi MT

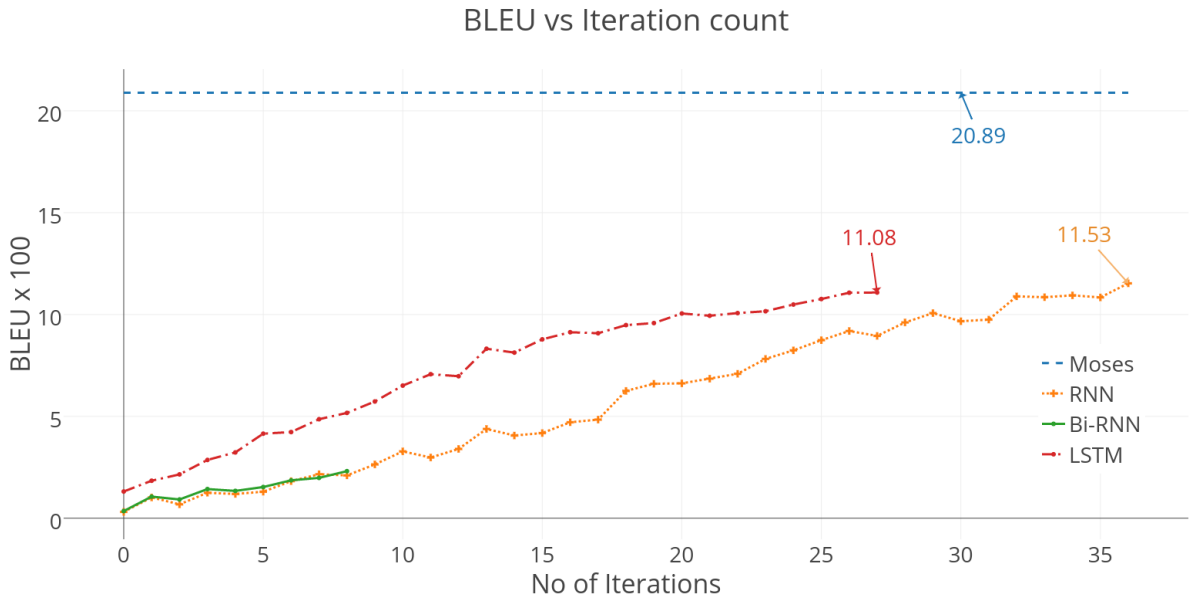


Figure 5.1: Comparison of BLEU scores in En-Hi MT

weights. We found a BLEU score of 0.2089 for English-Hindi translation using Moses on the above dataset.

After 37 iterations, the simple RNN gives a BLEU = 0.1153. Bi-RNN after 8 iterations gives BLEU = 0.0231 and LSTM after 27 iterations gives BLEU = 0.1108 .

As it can be seen in the Fig. 5.1 among RNNs LSTM appears to outperform Bi-RNN which is slightly better compared to simple RNN. The BLEU score of Moses undoubtedly shows that it is long way to go for RNN models to reach this standard. One of the reasons behind this is due to the fact that regularization hasn't been used while building the RNN models. Also a single layer has been used, which could be made deep enough by stacking multiple layers in both encoder as well as decoder.

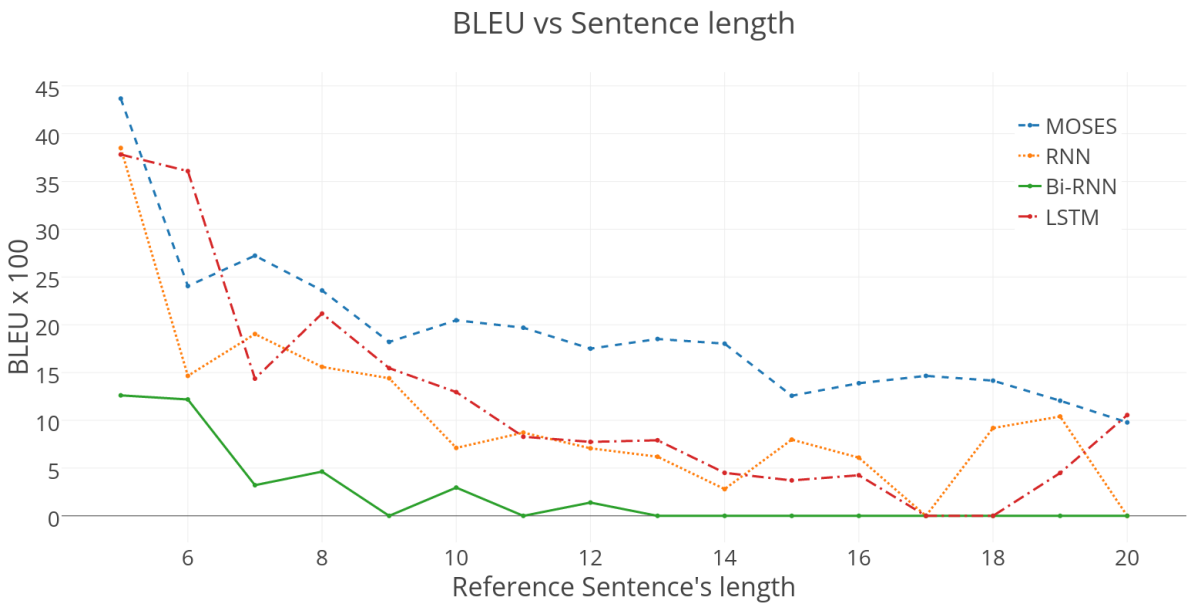


Figure 5.2: BLEU scores vs sentence lengths in En-Hi MT

5.1.2 Variation of BLEU scores with sentence lengths in English-Hindi MT

It is expected that as length of the sentence increases, it is difficult to make a good translation as it is difficult for the words in beginning of the sentence to propagate through the Encoder. It can be seen in the Fig. 5.2 LSTM's perform better as expected as they information for long periods of time. The Bi-RNN has low values as it hasn't completed enough iterations to be comparable with other models.

The plot of BLEU score for each prefix of the sentence is plotted from prefix of lengths 4 until 20 as shown in Fig. 5.3. For RNN models, the BLEU score is found to be monotonically decreasing unlike Moses. The reason is due to the fact that the decoder output function is based its previous outputted words. This can't be avoided as we don't know the actual length of the target sentence. This also shows the biasness of RNN models to give a good translation at the start of the sentence and as the sentence's length increases, the probability of correctly predicting the next word decreases. This can also be observed from below Fig. 5.3.

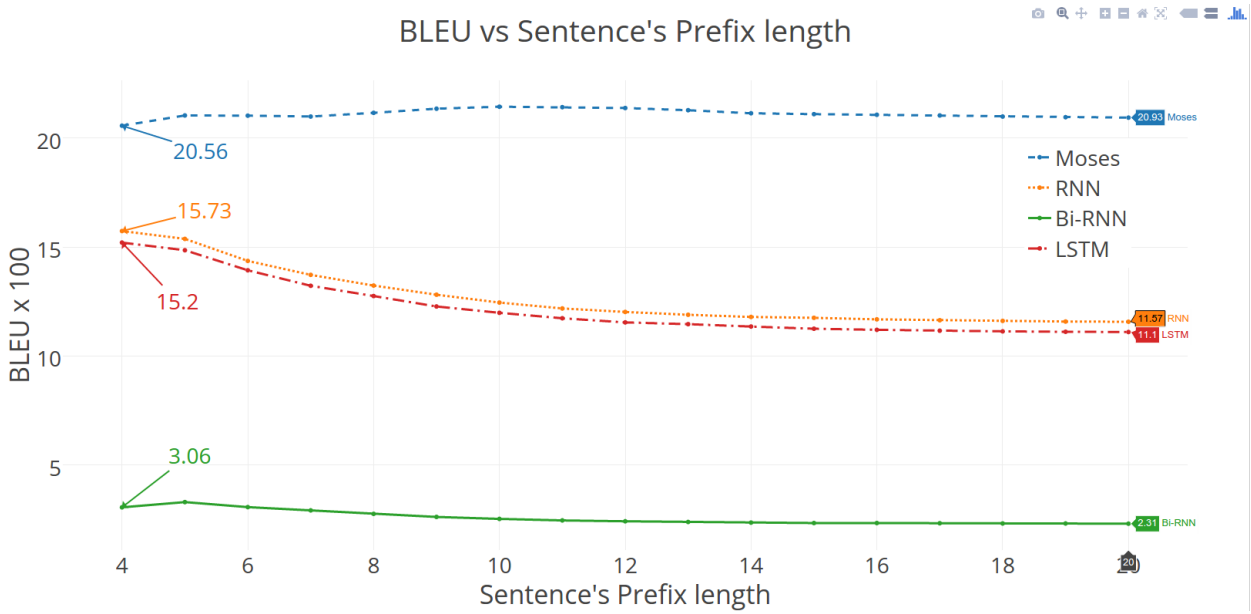


Figure 5.3: Comparison of BLEU scores for sentence prefixes in En-Hi MT

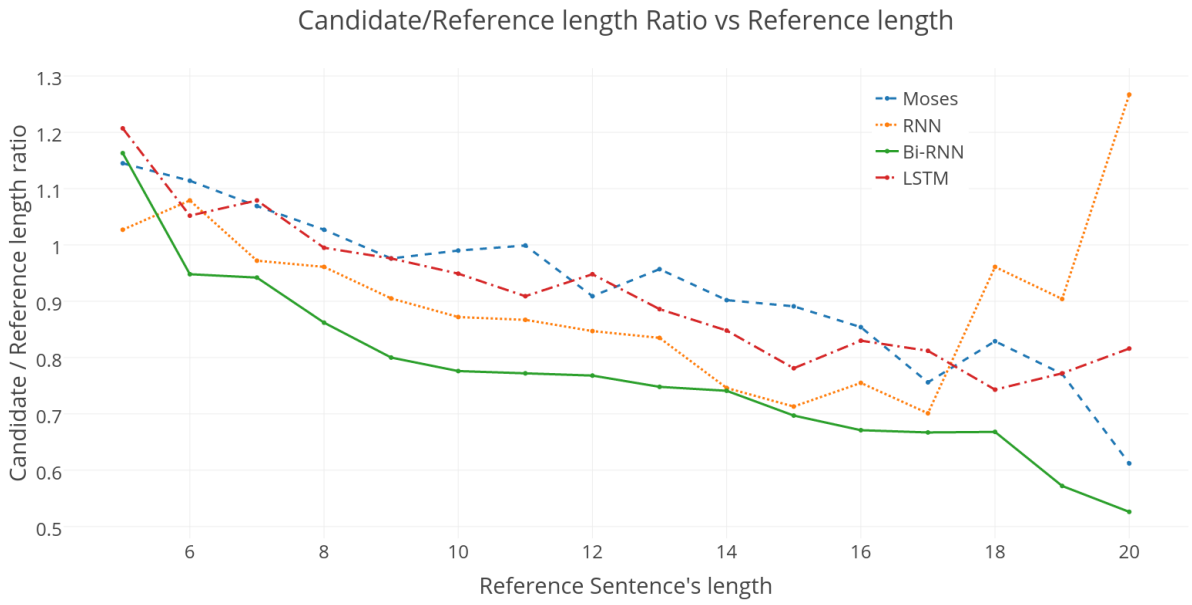


Figure 5.4: Candidate to Reference sentences Ratio comparison for En-Hi MT

5.1.3 Candidate-Reference sentences ratio for En-Hi MT

In the above Fig. 5.4, the ratio of Candidate to reference sentence's length close to 1 indicates an ideal length of translation according to the reference's sentence length. LSTM and RNN are closer to 1 and Bi-RNN is expected to perform better in future iterations.

5.2 Example Translations En-Hi

Source : in the case of the indus script .
Reference : सिंधु लिपि के मामले में ।
Moses: के मामले में सिंधु लिपि ।
RNN: सिंधु सिंधु के कारण पर ।
Bi-RNN: इस में में में में ।
LSTM: सिंधु लिपि उत्पत्ति विकास ।
Google: सिंधु लिपि के मामले में।

Source : remember password until you logout .
Reference : लॉगआउट होने तक कूटशब्द याद रखें ENG_NNP ।
Moses: कूटशब्द याद रखें जब तक आप लॉगआउट ।
RNN: लॉगआउट होने तक कूटशब्द याद रखें ENG_NNP ।
Bi-RNN: आपका UNK UNK UNK UNK ।
LSTM: लॉगआउट होने रखें कूटशब्द कूटशब्द रखें ENG_NNP ।
Google: यदि आप लॉग आउट तक पासवर्ड याद है।

Source : how to find out more .
Reference : अधिक जानकारी केसे प्राप्त करें | ।
Moses: अधिक जानकारी कैसे मिलेगी ।
RNN: अधिक जानकारी कैसे लें करें ।
Bi-RNN: कि कैसे UNK ।
LSTM: अधिक जानकारी कैसे मिलेगी ।
Google: कैसे और अधिक जानकारी प्राप्त करने के लिए।

Source : the teachers harsh words UNK her countenance which was kind and encouraging .
Reference : शिक्षिका के कठोर शब्द उनके दयालु और UNK UNK को UNK रहे थे ।
Moses: टीचर कठोर शब्दों countenance जो UNK UNK था और प्रोत्साहित ।
RNN: कई ने ही ही ही ही ही ही ही ही ही ही ही ही ही ।
Bi-RNN: UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK UNK ।
LSTM: उनके के UNK के के से से कई कई कई कई कई कई ।
Google: शिक्षकों को कठोर शब्दों तरह की और उत्साहजनक था जो उसकी मुखाकृति UNK।

Bibliography

[1] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[2] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[3] Mikolov, Tomas, et al. "Recurrent neural network based language model." INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010. 2010.

[4] Yadav, Neha, Anupam Yadav, and Manoj Kumar. "Preliminaries of Neural Networks." An Introduction to Neural Network Methods for Differential Equations. Springer Netherlands, 2015. 17-42.

[5] Jeffrey L. Elman. Finding Structure in Time. Cognitive Science, 14, 179-211

[6] Cho, Kyunghyun, et al. "Learning phrase representations using rnn encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[7] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." Signal Processing, IEEE Transactions on 45.11 (1997): 2673-2681.

[8] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).