Recurrent Neural Networks for Machine Translation

YAMINI RATNA THOTA
Computer and Information Sciences
The University of Texas at Dallas
Net ID: YRT190003

PHANIMADHURI NIMMALA
Computer and Information Sciences
The University of Texas at Dallas
Net ID: PXN190023

GREESHMA NAREDLA
Computer and Information Sciences
The University of Texas at Dallas
Net ID: GXN190005

AMEYA KULKARNI
Computer and Information Sciences
The University of Texas at Dallas
Net ID: ANK190006

ABSTRACT

In the initial days of the machine learning era, for people those who are new to English language found it difficult to access the net. Then came some new advancements of neural networks, which made people translate their web pages from a source language (English) to their mother tongue. So the advanced structure of neural networks called the RNNs are used for managing sequential data. Sequential data is the data in which the order of data is considered to be important. This paper mainly focuses on a special type of RNN(Recurrent Neural Networks) called the LSTM. These LSTMs form an efficient Encoder-Decoder Architecture for translating sentences from one language to another. And in this paper, we translated from English to Hindi (national language of India).

Keywords: RNN, LSTM, Encoder-Decoder Architecture, Sequential data .

I.INTRODUCTION

1.1 Initial proposal of idea of Machine Translation:

This was started in the 17th century. But the first implementation of machine translation was started in 1951 by Yehoshabar-hillel as he began his research at MIT. By the time there existed a Georgetown MT Research team, which collaborated with him to improve their research. Since then then there were some noticeable growths in the development of MT. The first conference was held in London in 1956. But in 1964 a association called ALPAC said that MT is insignificant as it is less efficient than the human translations. But the research continued to build and later MT was used on Russians tabs by translating sixty lines of some message. Slowly the use of MT was made known and then was developed. In 1970 small translations of some small texts were made available online. From 2013 the MT started to build on using the neural networks and are made more efficient. Since 2012 the growth in research of Machine translations have improved.

1.2 Idea behind the project:

Machine translation being a new advancement in NLP we tried to implement the same thing from scratch using Python without using any libraries.

II. LITERATURE SURVEY

2.1 Survey:

Article[1] talks about deep neural networks and why can't they be used for some advanced applications and the use of the LSTMs are explained. This paper also spoke about the encoder-decoder architecture and how this is used to generate some sequence from a given input. This paper they made a translation from English to French. They used five LSTMs with 380 parameters each to train and predict the output. In this paper they have also spoke about BLEU for analysing the quality of the text

Article[2] This is about the encoder decoder architecture using RNNs. This discusses in detail about how the probability of all possible words are computed. The formula about the SoftMax layer are discussed in detail in this paper; this also talks about the RNNs .Scoring the quality of the text is also done.

Article[3]] This blog in detail discusses the architecture in detail and what each unit in it does. This blog also explains in detail how the input unit is processed and is passed through different gates and outputs are being generated. This helped visualize what is happening internally in each unit.

Article[4] This talks about a new technique named BERT and also about bi-directional layers.

Article[5]] This has given a pictorial representation of all inputs and outputs and how the cross-entropy is calculated ,getting the probabilities of each word at the softmax layer.

Article[6]] This gave a basic view about how the backpropagation is taking place at each time stamp .This also has explained all the formulae related to it.So this helped in giving more ideas about backpropagation and the chain rules that are used in the formulae.

Article[7]] This paper gave a detailed view of what LSTMs are.The different ways we can use them for various applications and explains why LSTMs are novel architectures than Rnns.

III. THEORETICAL ANALYSIS

3.1 Need for this project?

Using the machine translators we can reduce the cost of translating from one language to the other. This is more effective when compared to human translators. And also when more data is to be translated it becomes tedious for a human translator to do it but a machine can deal with such huge amounts of data easily. The time taken by the machine translators to do the translation is minimal when compared to the human translators. So such projects are very much used when surveys are made in some local rural areas and if their responses are to be analysed we use such projects. So using such projects we are even able to make subtitles of movies. So this project is in huge demand in the world outside. This also made the ease of communication easy between people from different regions.

Neural Networks are a boon to Machine Learning. This raised the technology standards by a level. So using such techniques for predicting, translating or encoding can be also used in storing very confidential things like keeping military secrets of a country etc. Because the less the access to people the more confidential the information is. Many companies now are using such techniques to store their data. And also the techniques can be used to decode data and can be used for the country's protection.

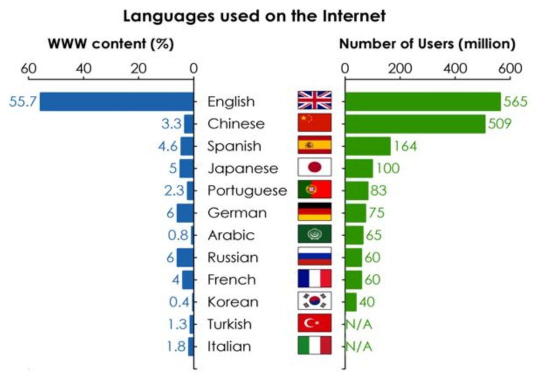


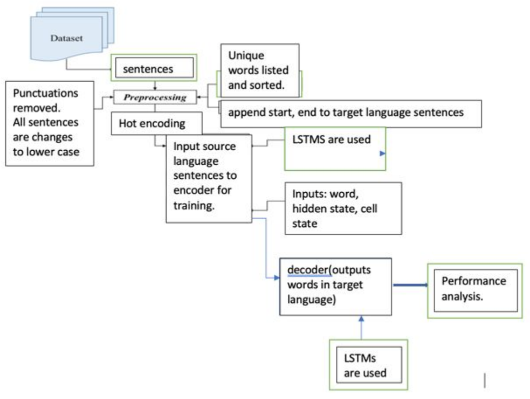
Fig 1

So the figure above would more describe the need of the project. Because the content of languages other than English is minimal but there are a quite good number of users. There is a need to improve the machine translation engines content.

IV. EXPERIMENTAL INVESTIGATIONS

4.1 OUR SYSTEM:

The architecture is an encoder-decoder architecture. The encoder contains LSTM cells, in which each cell takes previous state values and individual words of the source sentence as an input. The word in here is not a string ,it is a hot encoded vector. Basically hot encoding is a technique in which all the unique words in the dataset are given unique vector representation and are read like that. Because the machines mostly play with binary values. Representing in such a way would be easy. Each LSTM cell contains different gates like Input gate, Update gate, output gate. They internally contain tanh and sigmoid activations. Each time step takes input as Onehot encoded word and previous hidden output and cell state. So basically, the previous cell internal state will be inputted to the current time step and the last cell of Encoder gives out the hidden state value to the first LSTM cell of a Decoder. The Decoder also takes as input the word of the translated sentence with a <start> and <end> appended to starting and ending of the sentence. In the training phase irrespective of the prediction made by the decoder cells, we force the output to be as in the target sentence and compute the error between the target word and the predicted word(teacher forcing technique). In the testing phase the predicted word is given as output.



Dataset
sentences
Unique words listed and sorted.
Preprocessing
Punctuations removed. All sentences are changes to lower case
append start, end to target language sentences
Hot encoding
LSTMS are used
Input source language sentences to encoder for training.
Inputs: word, hidden state, cell state
decoder(outputs words in target language)
Performance analysis.
LSTMs are used

Fig 2

Data Preprocessing:

The input sentences are parsed to generate the vocabulary of unique English and Hindi words. This vocabulary consists of alphabetically ordered words and serves as a lookup table. Every word in every sentence in the dataset is first converted to lowercase. Any punctuation marks suchs as quotes, commas, full stops, question marks are removed. The digits in the sentence are removed. Every Hindi sentence is marked with START_ marker at the beginning and _END marker at the end of the sentence. Every word in the sentence is then represented as a one-hot vector. The entire sentence is a set of one-hot vectors for each word in the sentence. To keep the number of time steps the same, the input is padded with empty onehot vectors to match the max sentence length. The encoder-decoder model takes these set of vectors as input and outputs a set of vectors giving probabilities for the predicted words.

The Basic Architecture:

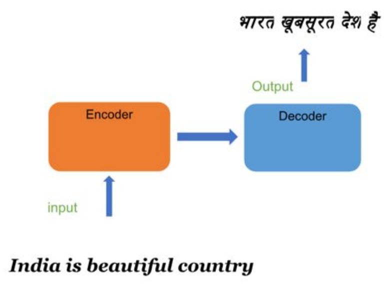


Fig 3

Embedded layer:

In this we encode all words in a sentence using hot encoding technique. This assigns unique vector values for each unique word in a dataset of a language. Then all the words in sentences are converted to Onehot Encoded strings.

Encoder:

During Training and Testing:

The encoder contains LSTM cells. Each cell takes as input the words of the sentence that we need to train(each cell takes only a single word as input).Along with this we give as input a hidden state from the previous timestep. Here the outputs are not stored but passed as inputs to the next LSTM after it(hidden state). Finally at the end the encoder forward passes an intermediate vector **<cellstatevalue,hiddenstate>** is given as input to the decoder forward pass. The encoder performs as the same in both the training and testing phase.

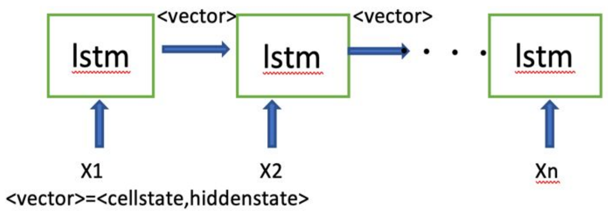


Fig 4

Decoder:

During Training and Testing:

The Decoder contains LSTM cells .In here the first cell always takes as input the<start> that denotes the start of the sentence. In the training phase the decoder takes as input the cell state of the previous LSTM and force inputs the desired word irrespective of the predicted word in the previous step. But the predicted word is used to compute the error, that is when a word is given as input to the decoder it generates probabilities of the possible words that can be got as output (SoftMax) and uses cross entropy technique to compute error.

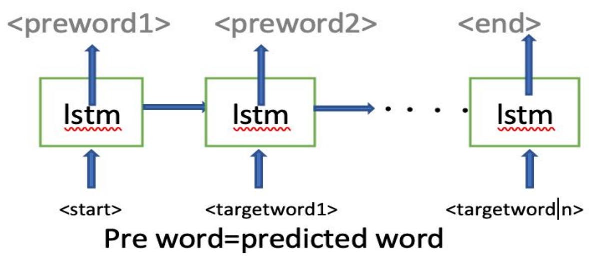


Fig 5

But in the Testing phase it words differently by inputting the predicted value of the previous LSTM as input.

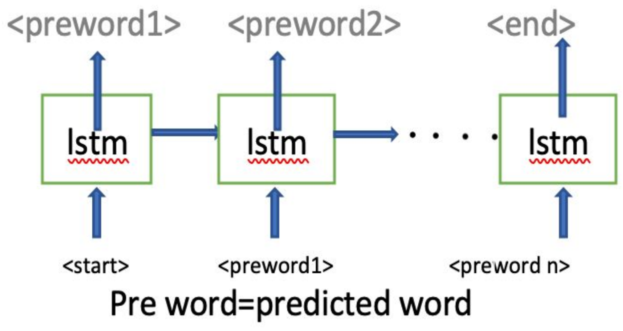


Fig 6

5. EXPERIMENTAL RESULTS

This model is implemented based on a subset of the dataset provided by Kaggle. The experimental results show a smooth decrease in error with increasing epochs in backpropagation.

5.1. Dataset :

We have used the dataset provided at Kaggle parallel corpora (https://www.kaggle.com/aiswaryaramachandran/hindienglish-corpora). A subset of the dataset was created using limited vocabulary. We picked pairs of sentences made of words present in the vocabulary. This was done to limit the vocabulary size to roughly 2500 English and 2500 Hindi words. This would ensure the size of input and output vectors and inturn reduce the number of parameters to be learned.

5.2. Dataset Analysis:

The dataset contains a total of 6861 English-Hindi sentence pairs. The maximum English sentence length is 23 words whereas the maximum Hindi sentence length is 25. The dataset consists of 2361 unique English words and 2847 unique Hindi words. Graphs in Fig 7 and 8 show the distribution of sentence lengths in English and Hindi.

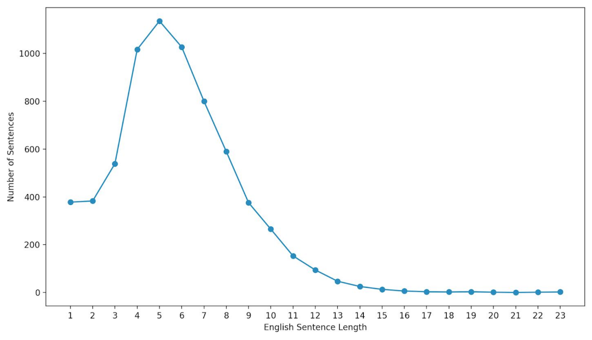


Fig 7 (Frequency of English Sentences)

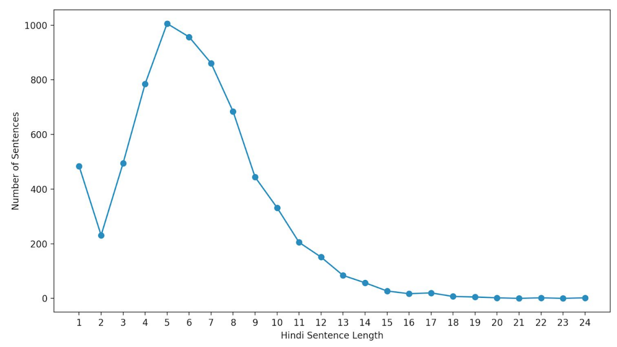


Fig 8 (Frequency of Hindi Sentences)

The frequency distribution of maximum occurring words in English are shown in Fig 9

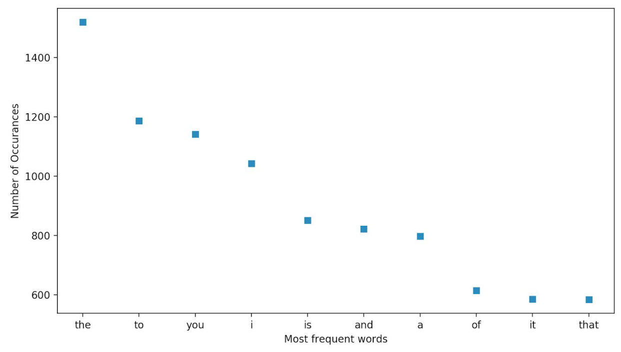


Fig 9 (Most Frequently occurring English Words)

Similarly most frequent words in Hindi with their frequencies are [('है', 2072), ('हैं', 912), ('में', 894), ('और', 841), ('नहीं', 837), ('के', 670), ('से', 635), ('मैं', 620), ('यह', 564), ('कि', 543)]. it appears that the most frequent words are indeed stopwords. In standard Natural Language Processing tasks like text summarization, these stopwords are removed in text preprocessing. However, in this task these words cannot be omitted as they are part of the sentence and represent respective time steps in the forward pass.

5.3 LSTM model:

The LSTM model consists of one input gate , one ignoring gate and one selection gate. Every gate is a small network with a number of nodes equal to vocabulary size. The outputs of ignore gate and selection gate are conditioned by sigmoid function whereas that for input gate is conditioned by tanh function. The cell internal state is a product of input and ignore gates added to the previous state cell state. The hidden state is tanh output of the cell internal state. The equations for LSTM are shown in Figure 10.

5.3.1 Formulae

Forward Pass equations:

$$I_k = \sigma(X_k . W_I + O_{k-1} . W_{recI} + b_F)$$

$$A_k = tanh(X_k . W_A + O_{k-1} . W_{recA} + b_A)$$

$$O_k = \sigma(X_k . W_O + O_{k-1} . W_{recO} + b_O)$$

$$S_k = (A_k * I_k + S_{k-1})$$

$$Out\_hidden_k = tanh(S_k) * O_k$$

$$Out_k = softmax(Out\_hidden_k)$$

I_k, A_k, O_k are input, activation and output gates of the time step **k** respectively

X_k is the input passed to the time step **k**

W_I, W_A, W_O are weights of input,activation and output gates respectively

$W_{recI}, W_{recA}, W_{recO}$ are recurrent weights of input,activation and output gates respectively

S_k is the internal cell state of time step **k**

Out_hidden_k is the hidden output of time step **k**

Out_k is the actual output of time step **k**

5.4. Training the model:

The model was trained to minimize the cross entropy error of the softmax output using backpropagation at each time step. To accommodate the uneven distribution of sentence lengths, every sentence having length less than 3 or greater than 10 was repeated 10 times. This was tried in order to give weightage to small sentences which contain less stop words and large sentences that were present in less numbers in the data set The learning rate used for best results was 0.01. The Figure 11 shows training error and Figure 12 shows testing error.

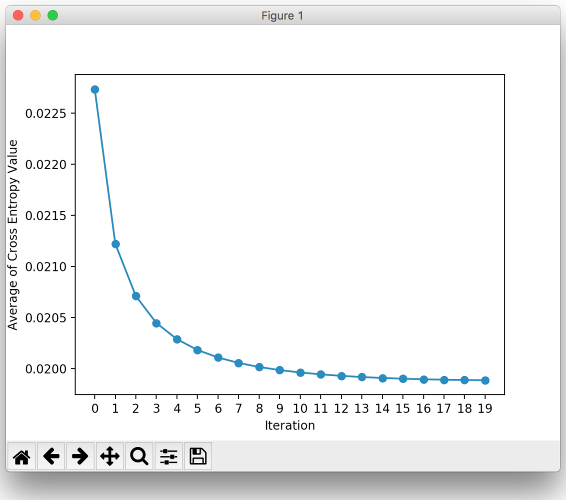


Fig 10 (Train error for every iteration)
Every iteration consists of pass through entire dataset

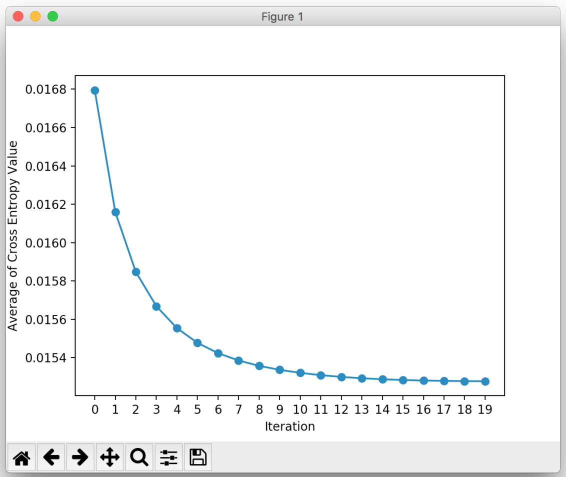


Fig 11 (Test error for every iteration)
Every iteration consists of pass through entire dataset

5.5 Error Measure :

The model outputs a vector of probabilities for the predicted word. The output vector is compared with the expected one-hot encoded vector. The error measure used is cross entropy.

5.6 Sample Result:

The initially trained model was outputting a sequence of _END for every word. This was because the most frequently occurring tag is _END. Hence we considered the second most probable word for the output. For training the network we used a 80:20 test-train split. Following table summarizes the least error values we got different runs

Epochs	Learning Rate	Average Error Over Entire Training Data	Average Error Over Entire Test Data
5	0.01	0.022	0.01554
10	0.01	0.021	0.01542
20	0.01	0.019	0.0152

We observed that after 10 iterations itself the error values change very minimally.
Following is a sample output of the trained model :

Input Sentence : the trouble is that they have no time
Expected Output Sentence : _START_ मुश्किल यह है कि उनके पास समय नहीं है _END
Predicted Sentence : है
Expected Word : मुश्किल
Predicted Probability: 0.000391374445322081
Expected Word : यह
Predicted Probability: 0.009953354596984147
Expected Word : है
Predicted Probability: 0.057574897933559695
Expected Word : कि
Predicted Probability: 0.011679099748519252
Expected Word : उनके
Predicted Probability: 0.000511428990090936
Expected Word : पास
Predicted Probability: 0.003186173020716165
Expected Word : समय
Predicted Probability: 0.0034025816546996344
Expected Word : नहीं
Predicted Probability: 0.025966927515275545
Expected Word : है
Predicted Probability: 0.05756899617319514
Expected Word : _END
Predicted Probability: 0.00010009932396344048

We also tried to implement similar code using Keras. Following is Sample Output from Reference Implementation using Keras [10].

Input English sentence: that if other people know it or see it
Actual Hindi Translation: कि अगर दूसरे लोग इसे जान जाएंगे या देख लेंगे
Predicted Hindi Translation: वह उसने मुझे में से नहीं उसने _END

The above output shows the input English sentence , the expected Hindi Sentence and predicted Hindi sentence. The predicted Hindi sentence is calculated by taking the word with maximum probability, other than the tags START_ and _END. The output also shows the softmax output (predicted probability) for the expected words.We observed that, though the error minimizes and saturates after 20 iterations. The model is biased towards stopwords..

5.7 Analyzing the Result:

The trained model outputs the same word for the given input sentence. We tried to find out why the model behaves in such a manner. The reduction in error with iterations shows that the model is getting trained and the predictions are coming closer to expected values. However, the probabilities of highest occurring words (stopwords) are always the highest. As the model gets trained the probabilities for other words also get increased, but not as much as the most frequently occurring words.

We interpreted that the cause of this behaviour lies in the data. If we look at this problem from a classification perspective, the Hindi vocabulary consists of 2847 unique words, however the number of occurrences of stopwords is much higher than the number of occurrences of other words. Hence, the model learned to misclassify most of the words other than stopwords.

5.8 How can this be improved:

We tried to improve the model by considering very short and very large sentences, sentences that have less frequency multiple times. This indirectly increased the dataset size from 6861 to 11000 sentences. We also studied about various other techniques to improve the model. “Attention based Machine Translation”[8] is a technique which assigns weights to every word in the input sentence. This mechanism can help improve emphasis on particular words in the input sentences. “Beam Search”[9] is another technique which tries to find all the probable sentence structures from the output probabilities and computes the syntactically best sentence output.

VI. SUMMARY, CONCLUSION AND RECOMMENDATIONS

6.1. CONCLUSION:

The implemented system is able to reduce the cross entropy when the number of iterations increases. We have also computed the results using KERAS in which the drop in the cross entropy is noticed. But when the word is getting predicted it is mostly predicting the stop words.

6.2 SUMMARY:

Using machines to translate is efficient because human translations would need a lot more effort and include more cost in contrast with Machine translations.Huge amount of data can be translated in less time using machines.Confidentiality will exist when machines are used to translate.

VII. REFERENCES/BIBLIOGRAPHY

[1] Ilya Sutskever,Oriol Vinyals and Quoc V. Le "Sequence to Sequence Learning with Neural Networks": http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[2] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares,Holger Schwenk,Yoshua Bengio
"Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation"arXiv:1406.1078v3 [cs.CL] 3 Sep 2014
: https://arxiv.org/pdf/1406.1078v3.pdf

[3]Shreyasrivastava, "MachineTranslation(Encoder-Decoder Model)!"
Available:https://medium.com/analytics-vidhya/machine-translation-encoder-decoder-model-7e4867377161

[4]Thomastracey,"Language Translation with RNNs"
Available:https://towardsdatascience.com/language-translation-with-rnns-d84d43b40571

[5]John Hewitt,RenoKrizz,"Sequence- to- sequence models"slides.
Available:https://nlp.stanford.edu/~johnhew/public/14-seq2seq.pdf

[6]Jae Duk Seo,"Deriving backpropagation on simple Rnns"
Available:https://towardsdatascience.com/back-to-basics-deriving-back-propagation-on-simple-rnn-lstm-feat-aidan-gomez-c7f286ba973d

[7]Sep Hochreiter, Jurgen Schmidhuber,"Long short term memory",Neural Computation 9(8):1735{1780, 1997}
https://www.bioinf.jku.at/publications/older/2604.pdf

[8]Jason Brownlee PhD,"How does attention work in encoder-decoder Recurrent neural network"
Available:https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/

[9]Jason Brownlee PhD,"How to implement a beam search decoder for Natural Language processing"
Available:https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/

[10] We also tried reference implementation in Keras.The code is available at
Available:https://www.kaggle.com/aiswaryaramachandran/english-to-hindi-neural-machine-translation