Machine Translation using Deep Learning

UDAY NAYAK
Assitant Professor,Information Technology Department
Don Bosco Institute Of Technology
Mumbai, India
udaysuite@gmail.com

SHUBHAM AGRAWAL
Information Technology Department
Don Bosco Institute Of Technology
Mumbai, India
sa.shubham007@gmail.com

ADITYA BHAGWAT
Information Technology Department
Don Bosco Institute Of Technology
Mumbai, India
bhagwataditya226@gmail.com

ROHIT NAIR
Information Technology Department
Don Bosco Institute Of Technology
Mumbai, India
nairrohit0612@gmail.com

DAELYN OLIVEIRA
Information Technology Department
Don Bosco Institute Of Technology
Mumbai, India
daelynoliveira3@gmail.com

***Abstract*—Neural Machine Translation using Deep Learning is a newly proposed approach to machine translation.The issue with the current translation system is that it does not translate idioms and phrases appropriately. The proposed translation system attempts to successfully translate English text to its appropriate Hindi text translation and vice versa using Recurrent Neural Networks using a novel neural network architecture which will replicate the higher accuracy levels achieved. Our project uses an encoder-decoder Long Short Term Memory (LSTM) model. The evaluation of the output text is denoted by Bilingual Evaluation Understudy (BLEU) score indicating the quality of text which has been machine-translated from one natural language to another. Our project works by training and testing the model on bilingual pre-processed and cleansed dataset. The skeleton of the project is developed on Google Colaboratory Notebooks. The Proposed system will consist of a simple Interface with Input text field and Output text field. Our project uses Keras frontend and Tensorflow backend executed on the Google Colaboratory. Our project can be used by an english or hindi amateur over the world wide web.**

***Index Terms*—Machine Translation, Neural Networks, Neural Machine Translation, Deep Learning, English-German MT, Long Short Term Memory(LSTM), Bilingual Evaluation Understudy(BLEU).**

I. Introduction

One of the most important limitations of the modern day translators is the inability to handle the translations of idioms and phrases of the source language. Due to this limitation our goal is to implement a web application based on Neural Machine Translation system which successfully translates English text to Hindi text using a special type of Recurrent Neural Network(RNN) called as Long Short Term Memory(LSTM) for encoding and decoding (with an attention mechanism) to achieve better accuracy.

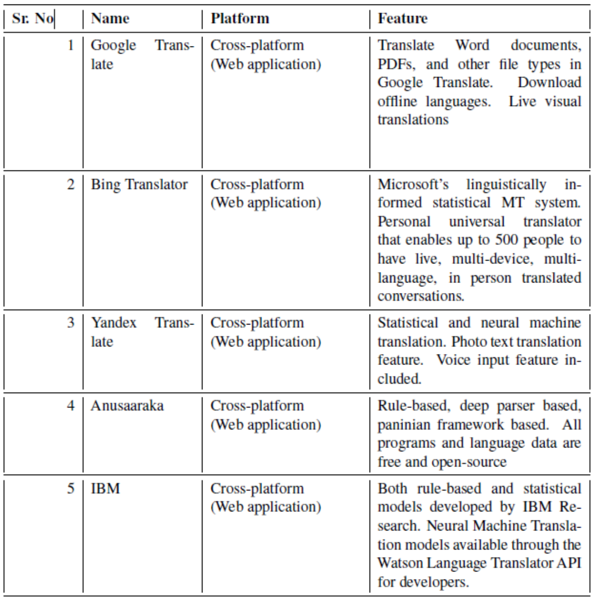
II. Scope of the Project

Implementing a fully functioning system based on Machine Translation, to successfully generate high quality and appropriate Translation from input. Scope of our project limits us to handle sentences with length more than specified(20 words), inability to handle proper nouns(name, place, animal, thing) excluding the ones already in the data set and tackling words not present in our data set. The Deliverables provided by our project would be Algorithm, Deep learning Machine Translation Model, Desktop User Interface(UI). The features included are English to Hindi Text Translation. Understanding the input English Text entered by the user and successfully translating the English text input to Hindi Text using our model.

III. Need for the Proposed System

To successfully translate the English Text input to Hindi text using deep learning instead of statistical machine translation(traditional translation systems). To play around with the hyperparameters, training data size, number of training steps and compare the accuracy of the results for different combinations so as to achieve the optimum translation accuracy.

IV. Current Scenario



Sr. No	Name	Platform	Feature
1	Google Translate	Cross-platform (Web application)	Translate Word documents, PDFs, and other file types in Google Translate. Download offline languages. Live visual translations
2	Bing Translator	Cross-platform (Web application)	Microsoft's linguistically informed statistical MT system. Personal universal translator that enables up to 500 people to have live, multi-device, multi-language, in person translated conversations.
3	Yandex Translate	Cross-platform (Web application)	Statistical and neural machine translation. Photo text translation feature. Voice input feature included.
4	Anusaaraka	Cross-platform (Web application)	Rule-based, deep parser based, paninian framework based. All programs and language data are free and open-source
5	IBM	Cross-platform (Web application)	Both rule-based and statistical models developed by IBM Research. Neural Machine Translation models available through the Watson Language Translator API for developers.

Table 1.1: Current Scenario

V. Summary of the Results / Task completed

1) Completed Literature survey
2) Downloaded the data sets
3) Cleansing of data sets
4) Finding vocabulary size and maximum sentence length of english and hindi text from the parallel corpus.
5) Training
6) Validation
7) Finding the BLEU scores of the validated.

We learnt to build a neural network with hidden layer, using forward propagation and backpropagation. We understood the necessity of hyperparameters like batch size, optimizers, learning rate, number of epochs, size of the dataset, plotting to check the fitting, etc., and how the inter dependency among them work and use it to train deep neural networks.We are trying to achieve the accuracy of the already existing translation systems and also trying to overcome certain limitations faced by them. Our translation system handles idioms and phrases and generates desired outputs.

VI. Summary of the investigation in the published papers

Experiments On Different Recurrent Neural Networks For English-Hindi Machine Translation

In this paper [1], experiments using different neural network architectures employing Gated Recurrent Units, Long Short Term Memory Units and Attention Mechanism and report the results for each architecture. The Bi-directional LSTMs generally show better performance for compound sentences and larger context windows. They describe the motivation behind the choice of RNNs in detail. When they employ bidirectional LSTMs, they end up with two hidden states - one in the forward direction and one in the backward direction. This allows the network to learn from the text. Bi-directional LSTMs generally work the best, especially when complemented with the attention mechanism. Pruning was also done to remove special characters and hyperlinks from the sentences. They observe that their model is able to produce grammatically fluent translations, as opposed to traditional approaches. A bi-directional LSTM model with attention mechanism shows improvement over normal RNNs in both these aspects. They demonstrated results using this approach on a linguistically distant language pair En ! Hi and showed a substantial improvement in translation quality. They conclude that RNN perform well for the task of English-Hindi Machine Translation. The bidirectional LSTM units perform best, specially on compound sentences.

Machine Translation Using Deep Learning : A Survey

In this paper [2], the term Machine Translation is used in the sense of translation of one language to another, without any human improvement. In this approach Neural Machine Translation technique will be used to translate Japanese language into English language. Tanaka corpus will be used in this approach. Japanese will be translated into English using improved Recurrent Neural Network(RNN). For training the RNN encoder decoder, they have used Tanaka corpus. It contains 1,50,000 sentences pairs and it is publicly available database. In this approach they have tried to get accurate translation from Japanese to English. They have used small group of data to train the system. They will apply this architecture to the large amount of data with efficient processing unit. In future they will try to translate accurately Japanese to Hindi. Graphics processing unit (GPU) is very good option for parallel processing and fast computation as compare to the CPU. GPU provides a better energy efficiency and archives substantially higher performance over CPUs. The experiment perform by them in the following steps: 1) First of all collect Japanese-English vocabulary data. 2) Create RNN for Encoding texts with help of python. 3) Train the system with languge corpus. 4) Translate Japanese language into English language.

Achieving Open Vocabulary Neural Machine Translation With Hybrid Word-Character Model

In this paper [4] , neural machine translation (NMT) has used quite restricted vocabularies, perhaps with a subsequent method to patch in unknown words. This paper presents a novel word character solution to achieving open vocabulary neural machine translation. Publishers build hybrid systems that translate mostly at the word level and consult the character components for rare words. Their character-level recurrent neural networks compute source word representations and recover unknown target words when needed. The advantage of such a mixed approach is that it is much faster and easier to train than character-based ones. Also it never produces unknown words as in the case of word based models. In training word-based NMT, they follow Luong et al. (2015a) to use the global attention mechanism together with similar hyperparameters. In word-based NMT, their source and target sequences to have a maximum length of 50 each, words that go beyond the boundary are ignored. Due to memory constraint in GPUs, they limit their source and target sequences to a maximum length of 150 each, i.e., they backpropagate through at most 300 timesteps from the decoder to the encoder. They have proposed a novel hybrid architecture that combines the strength of both word- and character-based models. Word-level models are faster to train and offer high-quality translation; whereas, character-level models help achieve the goal of open vocabulary NMT. Their best hybrid model has surpassed the performance of both the best word-based NMT system and the best non-neural model to establish a new state-of-the-art result for English-Czech translation with 20.7 BLEU. Moreover, they have succeeded in replacing the standard ¡unk¿ replacement technique in NMT with their character level components, yielding an improvement of +2.111.4 BLEU points.

Neural Language Models for Machine Translation

In this paper [3], they studied that deep NLMs with three or four layers outperform those with fewer layers in terms of both the inability to deal with and the translation quality. They combine various techniques to successfully train deep

NLMs that jointly condition on both the source and target contexts. They use DARPA BOLT program in the Chinese-English bitext, with 11.1M parallel sentences. They reserve 585 sentences for validation, i.e., choosing hyperparameters, and 1124 sentences for testing. They train their model for 4 epochs. For both Chinese and English words the vocabularies are limited to the top 40K frequent. All hidden layers have 512 units each and embeddings are of size 256. Their training speed on a single Tesla K40 GPU device is about 1000 target words per second and it generally takes about 10-14 days to fully train a model. In contrast, reranking with deep NLMs of three or four layers are clearly better, yielding average improvements of 1.0 TER / 1.0 BLEU points over the baseline and 0.5 TER / 0.5 BLEU points over the system reranked with the 1-layer model, all of which are statisfically significant according to the test described. In this paper, they have bridged the gap that past work did not show, that is, neural language models with more than two layers can help improve translation quality. Their results confirm the trend reported in (Luong et al., 2015) that source conditioned perplexity strongly correlates with MT performance.

VII. Comparison between the tools / methods / algorithms

1) Algorithm: **Model 1 : Statistical Machine Translation for Text-to-text**

Statistical machine translation (SMT) is a machine translation where the text are translated using parameters which are derived using bilingual text corpora. The statistical approach contrasts with the rule-based approaches to machine translation as well as with example-based machine translation. Training of millions of words training data is needed for high quality SMT results which is costly compared to modern way of approach. So SMT used to translate with a limited amount of words which created results in a less optimal way.

In Statistical machine translation (SMT) it learns to translate the human translations by analzying existing translations which is also known as bilingual text corpora. It differs from the Rules Based Machine Translation (RBMT) which uses an approach of word based translation which uses more computational power and has less efficiency, modern SMT systems are based on phrased based translation and rearranges the translations using overlap phrases. Phrase-based translation aims to reduce the restrictions of word-based translation by allowing it to translate the whole word where even the length of the word may differ.

The shortcomings of SMT are as follows:

- Corpus creation are costly.
- Errors are difficult to detect and fix.
- Results may have less fluency of words that masks translation problems.

Model 2 : Recurrent Neural Network for Text-to-text

In feed forward neural network length of input layer is fixed to the length is the input sentence. So, to deal with these types of variable-length input and output, RNN comes into action. Whenever a single sample is fed into a feed-forward neural network, the networks internal state, or the activation of the hidden units, is computed from scratch and is not influenced by the state computed from the previous sample. The RNNs thus help in converting the input sequence to a fixed size feature vector that encodes primarily the information which is crucial for translation from the input sentence, and ignores the irrelevant information.

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and belong to the most promising algorithms out there at the moment because they are the only ones with an internal memory. Because of their internal memory, RNNs are able to remember important things about the input they received, which enables them to be very precise in predicting whats coming next. In a RNN, the information cycles through a loop. When it has to make a decision, it considers the current input and also what it has learned from the inputs it received previously. Long Short Term Memory (LSTM) units are a type of RNNs which are very good at preserving information through time-steps over a period of time. In an LSTM we have three gates:-

1) Input gate: This gate determine whether or not to let new input in
2) Forget gate: This gate deletes the information because it isnt important
3) Output gate: This gate let the output at te current time step.

Model Architecture

- Text Vectorization (One-hot encoding)
- InputLayer : Input :(None,12) , Output :(None,12)
- Embedding : Input :(None,12) , Output :(None,12,20)
- LSTM : Input :(None,12,20) , Output :[(None,256), (None,256), (None,256)]
- InputLayer : Input : (None,13) , Output : (None,13)
- Embedding : Input :(None,13) , Output :(None,13,20)
- LSTM : Input : [(None,13,20), (None,256), (None,256)] , Output :(None, 13, 256)
- TimeDistributed : Input : (None, 13, 256) , Output : (None, 13, 43213)

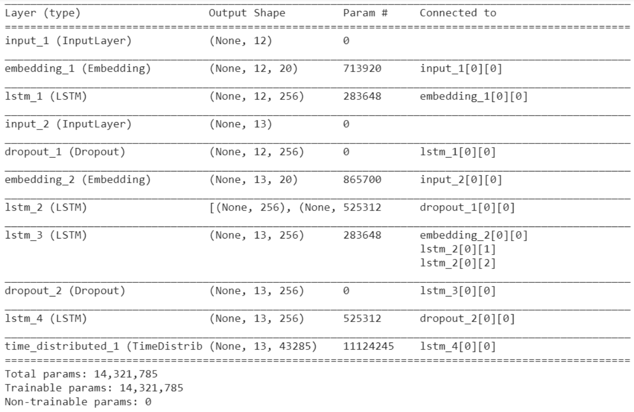
VIII. Analysis and Design

A. Methodology Adapted

A typical life cycle was divided into two phases:

Phase 1:

The fundamentals of Deep learning in context to the model was the main task in the first phase that we intended to learn and understand . In this step we intended to get to know all of the available resources and technologies that can be used in our project for the model to be implemented. Literature survey



```
____________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
====================================================================================================
input_1 (InputLayer)            (None, 12)           0
____________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 12, 20)       713920      input_1[0][0]
____________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 12, 256)      283648      embedding_1[0][0]
____________________________________________________________________________________________________
input_2 (InputLayer)            (None, 13)           0
____________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 12, 256)      0           lstm_1[0][0]
____________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 13, 20)       865700      input_2[0][0]
____________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 256), (None, 525312      dropout_1[0][0]
____________________________________________________________________________________________________
lstm_3 (LSTM)                   (None, 13, 256)      283648      embedding_2[0][0]
                                                                 lstm_2[0][1]
                                                                 lstm_2[0][2]
____________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 13, 256)      0           lstm_3[0][0]
____________________________________________________________________________________________________
lstm_4 (LSTM)                   (None, 13, 256)      525312      dropout_2[0][0]
____________________________________________________________________________________________________
time_distributed_1 (TimeDistrib (None, 13, 43285)    11124245    lstm_4[0][0]
====================================================================================================
Total params: 14,321,785
Trainable params: 14,321,785
Non-trainable params: 0
```

Fig. 1. Model Summary

and downloading of the datasets needed to train the model was part of this phase.

Phase 2:

We shift from the details and theory of deep learning algorithms to implementations, this involves an Agile like model. We build and improvise until we get an optimally performing model. In an iterative fashion we take one module at a time. At each step of project development requirements of customers are checked upon. A rapid delivery of the products is done. Importance is given to the stake holder's and customer's requirements.

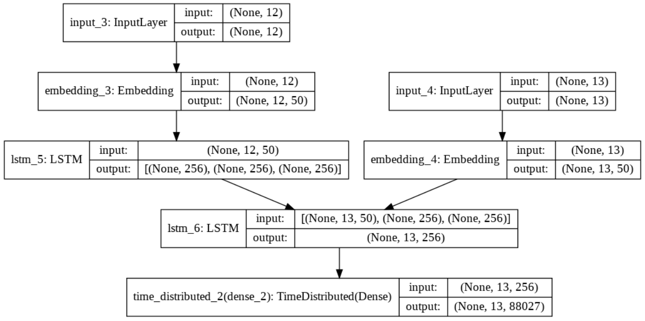
When shifting from one module to other these phases can overlap. Literature survey is carried out throughout the project as per the requirements. We have weekly assignments where we try to read about some technology or resources relevant to our project. Also a weekly meeting is conducted either in person or on a shared conference call. We use GitHub, Google Docs for collaboration. By setting milestones and checking them at each stage the progress of the project is measured

B. *Analysis*

Requirement gathering of our system was done in the initial stages. We came to a conclusion that we would use Google CoLab to implement our model as it has its own GPU which can be used to train the model using the datasets. Some of the requirements were modified and looked into as suggested by our guide and the panel.

C. *Proposed System*

Architecture and Design Model



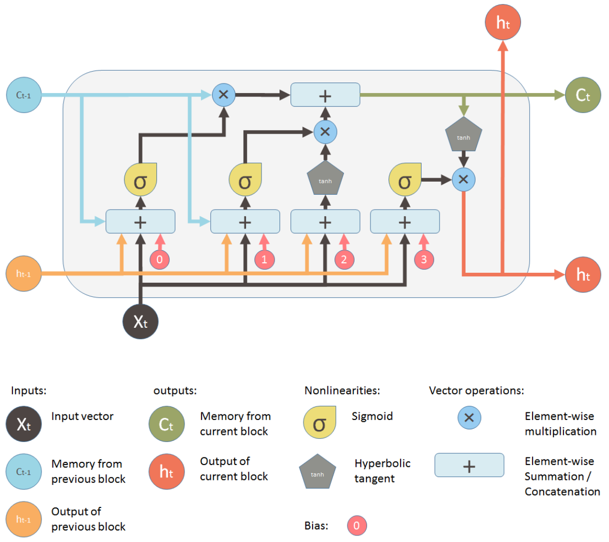
We have created a novel architecture for the proposed system to translate from English to Hindi text. In the first Input layer an English text is taken as an input and is passed to the embedding layer. The embedding layer creates an array of the word vectors for each of the English words which is being mapped from the English tokenizer . The maximum length of the English sentence is 12 which is being embedded into a 50 length vector representation. The output from the embedding layer is being passed to LSTM layer where the sentence is being encoded. There are 256 units used in the encoding layer is which is being passed to the decoder.

At the same time the English encoded sentence is passed to the to the decoder there is an another layer being passed to the decoder. We take a second input for the system which is an Hindi text. The second input text is being passed to an embedding layer where the all unique words are mapped to the Hindi vocabulary and we get an array of all the words mapping it to Hindi tokenizer. The embedding layer is being is passed to the decoder along with the English encoded sentence. On the decoder layer when English encoded sentence is being decoded each of the English sentence is being mapped to its corresponding Hindi sentence through which the training of the model is efficient. The system learns better in this way. The output from the decoder layer is being passed on the Time Distributed Dense layer which is more of a wrapper layer in which the Hindi sentence as an output in the form of a matrix where the rows of the matrix represents the maximum length of the sentence and the columns represent the vocabulary size of Hindi

The drawback of LSTM can be overcome by attention mechanism, attention is simply a vector, normally the outputs of dense layer using softmax function. Before Attention mechanism, translation relies on reading a complete sentence and compress all information into a fixed-length vector, as you can imagine, a sentence with hundreds of words represented by several words will surely lead to information loss or inadequate translation, etc.

Long Short Term Memory(LSTM)

The network takes three inputs, X(t) is the input of the current time step, h(t-1) is the output from the previous LSTM unit and C(t-1) is the memory of the previous unit, which is the most important input. h(t) is the output of the current network. C(t) is the memory of the current unit. This single unit makes decision by considering the current input, previous output and previous memory. And it generates a new output and alters its memory. On the LSTM diagram, the top pipe is the memory pipe. The input is the old memory (a vector). The first valve is called the forget valve. Now the second valve is called the new memory valve. Both are controlled by a one layer simple neural network that takes the same inputs as the forget valve. And finally, we need to generate the output for this LSTM unit. This step has an output valve that is controlled by the new memory, the previous output h(t-1), the input X(t) and a bias vector. This valve controls how much new memory should output to the next LSTM unit.



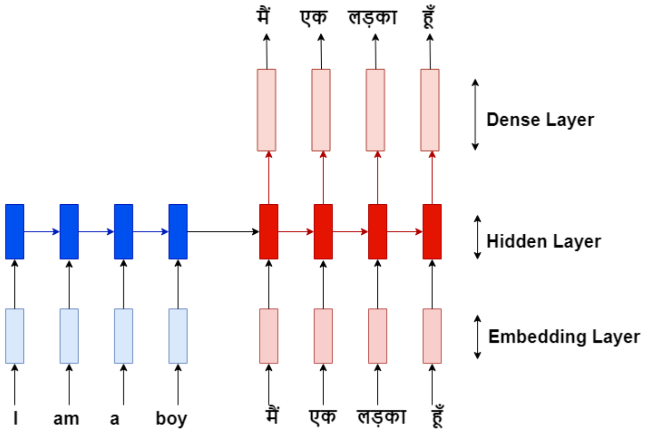
1) *Design Details:* **Encoder Decoder**

There are two models in this architecture : one is for reading and encoding the input sentence into fixed length vector, other is for outputting the predicted sequence by decoding the fixed length vector.This gives the architecture its name of Encoder-Decoder LSTM.

The Encoder-Decoder LSTM was developed for natural language processing problems. It illustrated the state-of-art performance in the text translation area. The use of fixed sized internal representation in the heart of the model which reads the input and output sequences to and from is the innovation of this architecture. Thus the method may be referred to as sequence embedding. Each word of the English sentence is passed on to an embedding layer where the each sentence is represented in a form of vectors. The embedding layer is then passed to the hidden layer.

The hidden layer is like a black box where both encoding and decoding of the sentence is done. In this layer both English and Hindi sentences are passed to the encoding layer where English the sentences are encoded. In the second part of the hidden layer i.e decoding layer each of the encoded English sentences are decoded and the corresponding Hindi sentences are mapped to the English sentences so that the system learns and understands better. The LSTM take cares of all the weights while encoding and decoding the sentences.

According to the above figure, blue represents encoder and red represents decoder, the last part of the encoder-decoder model is the dense layer which finally gives the output for the English sentence. It represents the output in form of a matrix where rows are length of Hindi sentence and columns are Hindi vocab size



IX. Results And Discussion

We have implemented the model that can successfully from English to Hindi. Dividing the data into batch size of 64 and then training helped us to train the model without the use of external hardware. We increased our knowledge of working in python language and also his knowledge of different deep learning architectures.

X. Conclusion

Translation has become essential tool in the modern and ever more globalized world that we live in. Machine translation is a tool that can help businesses and individuals in a variety of ways. Machine translation developed using deep learning based LSTM encoder-decoder techniques is one of the path breaking achievement in the field of translation. The use of LSTMs help in improving the accuracy of the translation. The model is effectively developed using Python language and run on a normal desktop with GPU. This model aims at translating the English text to Hindi Text and vice versa with context using the RNNs by receiving the text input from the user and displaying the output text. We have used more than two layers and it has improved the translation quality. The Bilingual Evaluation Understudy Score (BLEU), is a metric for evaluating a generated sentence to a reference sentence. After analyzing all the above research papers we have learnt that deep learning is an effective method for giving accurate results for translating text. The time to process the data is too high and requires a lot of time to train the data.

Acknowledgment

We would like to express our special thanks of gratitude to our guide Prof. Uday Nayak, our Head of Department Prof. Prasad Padalkar, as well as our principal Dr. Prasanna Nambiar who gave us the golden opportunity to do this useful project on the topic Machine Translator using Deep Learning, which also helped us in doing a lot of Research and we came to know about so many new things, we are really thankful to them. A lot of effort has been put into the entire project with wholehearted dedication of the project team. Our project guide has helped us immensely in the entire project management aspect, which is a complete new domain to us.

We would also like to thank the staff on information technology department for their guidance and support in every possible way. We are also thankful to our project coordinator Prof Sunantha Krishnan, for her constant support and encouragement.

REFERENCES

[1] Ruchit Agrawal and Dipti Misra Sharma *Experiments On Different Recurrent Neural Networks For English-Hindi Machine Translation* 2017 DOI : 10.5121/csit.2017.71006

[2] Janhavi R. Chaudhary, Ankit C. Patel *Machine Translation Using Deep Learning : A Survey* 2018 International Journal of Scientific Research in Science, Engineering and Technology [IJSRSET]

[3] Francisco Guzman Shafiq Joty Llus Marquez Preslav Nakov *Machine translation evaluation with neural networks* 2017 Qatar Computing Research Institute

[4] Minh-Thang Luong and Christopher D. Manning *Achieving Open Vocabulary Neural Machine Translation With Hybrid Word - Character Models* 2016 Computer Science Department, Stanford University, Stanford, CA 94305 Translation

[5] Minh-Thang Luong Michael Kayser Christopher D. Manning *Deep Neural Language Models for Machine Translation* 2014 Computer Science Department, Stanford University, Stanford, CA, 94305

[6] Aditi Kalyani and Priti S. Sajja. *A Review of Machine Translation Systems in India and different Translation Evaluation Methodologies* 2015 International Journal of Computer Applications

[7] Francisco Guzman Shafiq Joty Llus Marquez Preslav Nakov. *Machine translation evaluation with neural networks* 2016 Qatar Computing Researching Institute, Qatar Foundation

[8] MT Luong Sutskever O. Vinyals W. Zaremba *Addressing the Rare Word Problem in Neural Machine Translation* 2017 Computer Science Department, Stanford University, Stanford, CA, 94305

[9] Nitish Srivastava Geoffrey Hinton Alex Krizhevsky Ruslan Salakhutdinov *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* 2014 Journal of Machine Learning Research

[10] Kishore Papineni Salim Roukos Todd Ward Wei-Jing Zhu *BLEU: a Method for Automatic Evaluation of Machine Translation* 2002 Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia

[11] Ondej Bojar Vojtch Diatka Pavel Rychl Pavel Strak Vt Suchomel Ale Tamchyna Daniel Zeman *HindEnCorp Hindi-English and Hindi-only Corpus for Machine Translation* Charles University in Prague, Faculty of Mathematics and Physics,Institute of Formal and Applied Linguistics