Transliteration for Sanskrit text

Report submitted in fulfillment of the requirements
for the Undergraduate Project of

Third Year

by

Deepanshu Gupta(15075011)

Sujal Maheswari(15075052)

Under the guidance of

Dr. Sukomal Pal

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
May 2018

Dedicated to

My parents, teachers without whom this project would have not been possible.

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date:

Deepanshu Gupta, Sujal Maheswari
B.Tech. Student
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Transliteration for Sanskrit text**" being submitted by **Deepanshu Gupta, Sujal Maheswari** (**Roll No. 15075011, 15075052**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bonafide work of our supervision.*

Supervisor :	Dr Sukomal Pal
Place :	Indian Institute of Technology (BHU) Varanasi, Department of Computer Science and Engineering, India 221005
Date :	1-May-2018

Acknowledgments

I would like to express my sincere gratitude to Dr Sukomal Pal for guiding us through the entire project and providing us with all the things that we required. We would also like to thank the anonymous reviewers for their thorough and diligent review comments, which has improved this report.

Place: IIT (BHU) Varanasi

Date: **Deepanshu Gupta, Sujal Maheswari**

Abstract

This project proposes a Devanagari to Roman script (and vice versa) transliteration system targeting old Sanskrit documents and manuscripts. Our system first transliterates a given query term containing Roman characters into Devanagari script, which will later be used to retrieve documents relevant to the transliterated and the original query. It also converts a document given in Devanagari script to Roman script.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Transliteration is changing words from one script to another, more commonly the words are proper nouns. Sometimes it also means changing sounds from one language to another. In our project, we convert a word from Roman script to Devanagari script and vice versa. Transliteration is one of the main sub tasks of Cross-lingual Information Retrieval (CLIR) which ensures a higher accuracy in the output.

Here we have used Deep Learning for transliteration instead of using conventional machine learning techniques. We have used Seq2Seq model of Keras library which uses LSTM networks(which is basically a special type Recurrent Neural Network(RNN)) to train data.

1.2 Problem Definition

Whenever we search the web with a query we often find our resulting documents to be in English language primarily because the query provided is in Roman script. What if we want the documents in other languages also? For that we will develop a

system that would return the old Sanskrit documents and manuscripts along with the English documents for a given query in Roman script. If we look at other languages like German, French etc we find that their old literature floats on the web but when it comes to Sanskrit we find very little of it on the web primarily because the work done in this area is minimal so we wanted to build a system that would return the user a set of relevant documents of Sanskrit language.

1.3 Organization of the Report

This report includes five chapters with detailed description of Long Short Term Memory(LSTMs), Seq2Seq model of Keras library and RNNs.It also provides a glimpse of previous work done in the field of transliteration and the techniques used by them. The main sections are the following:

- Background
  - Recurrent Neural Nets(RNNs)
  - Seq2Seq Model
  - Long Short Term Memory (LSTMs)
- Our Approach
- Related Works
- Results
- Conclusion

Chapter 2

Background

2.1 What is RNN?

RNN or Recurrent Neural Networks are a type of Neural network which contains loop in them, which basically allows information to persist. RNN emerged because of the

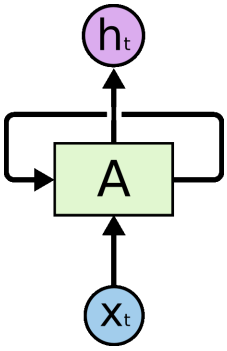


Figure 2.1 Looping in RNN.[1]

need to address the issue of long term dependencies. For example you want to predict the last word in the sentence "the clouds are in the sky." In this sentence the last word is pretty much obvious without the requirement of any further context. But if we take a look at the sentence "I grew up in France.....I speak fluent French." Traditional Neural Networks would suggest that the next word ought to be a name of language but if we want to be sure as to what language it must be, we need to remember the

context of France. A recurrent Neural Network can be considered as a repetition of the same network with each one of them passing message to its successor.

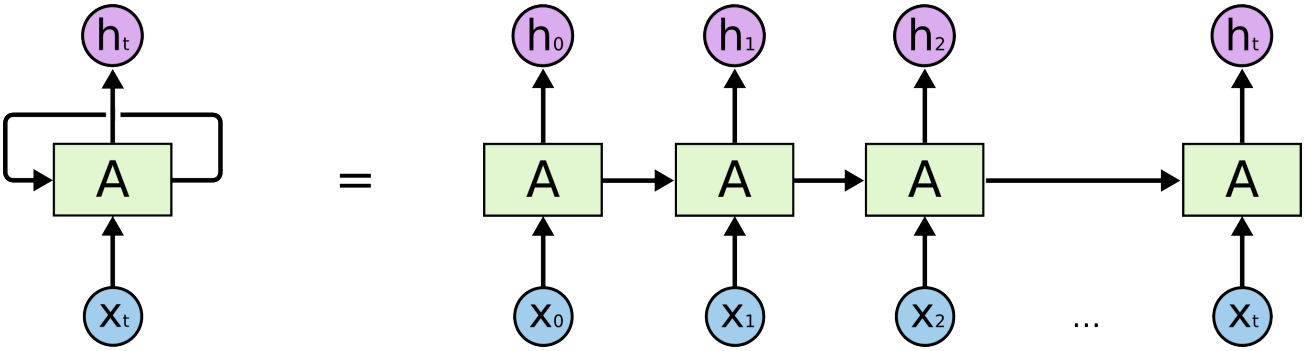


Figure 2.2 Unrolled RNN. [1]

2.2 What is Seq2Seq Model?

The main backbone of our project is the sequence to sequence model of Keras library. This model has been introduced in Cho et al., 2014[2]. It consists of two recurrent neural networks (RNNs). One of them is the encoder that processes the input given to it and the another one is the decoder that generates the output. Encoder and Decoder can share weight. This architecture is given below in the figure :-

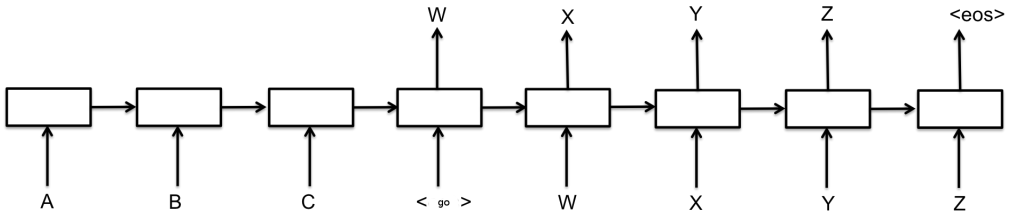


Figure 2.3 Encoder-Decoder Network.

The boxes in the picture represent a cell of the recurrent neural network. It is generally an LSTM cell. These cells are then deployed in multi-layers.

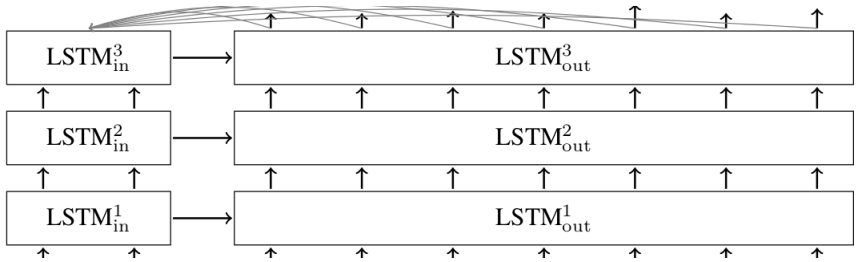


Figure 2.4 multi-layer sequence-to-sequence network with LSTM cells and attention mechanism.

In this model, the output of the decoder at time t is fed back to the algorithm and it becomes an input for the algorithm at time t+1. This is done at the testing time and when we are generating the model it is kept in mind that the input should be given correct even if the decoder has given wrong output at a previous step.

2.3 What is LSTM Network?

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task. RNN can learn to use past information. **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)[3], and were refined and popularized by many people in following work.[4] 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. A useful property of the LSTM is mapping a variable length input sentence into a fixed dimensional vector representation.

2.3.1 Understanding LSTM

In normal RNNs the repeating module generally have a simple structure like a tan(h) layer but in LSTMs the repeating module has four neural network layers instead of one which interact with each other in a very trivial way.

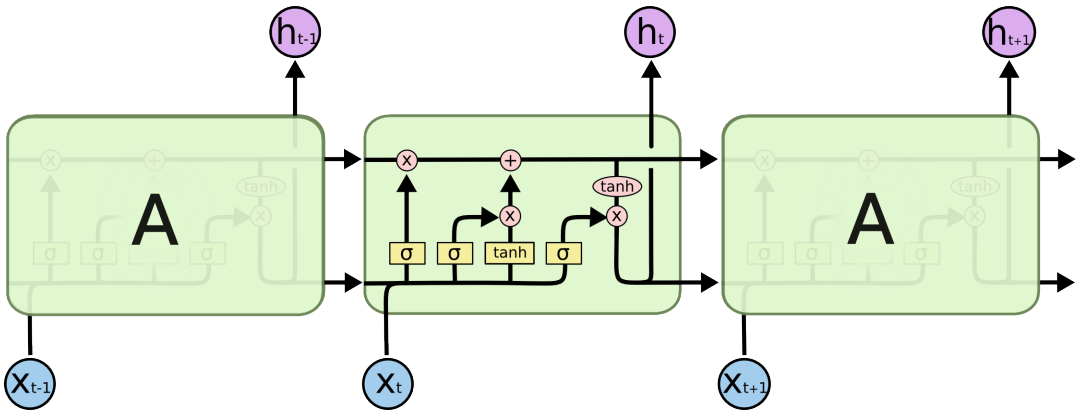


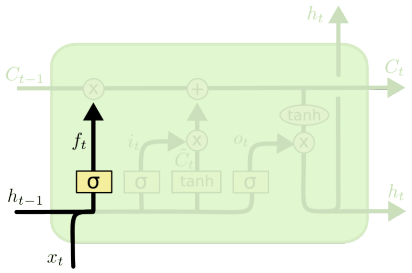
Figure 2.5 Layers of LSTM. [1]

The line on the top that runs straight through the cell is the cell state. Information can be added or removed from the cell state via structures called gates.

Gates are nothing but a sigmoid functions that gives an output between 0 1 which basically decides how much information can be added or removed from the cell state. A '0' corresponds to 'nothing' and a '1' corresponds to 'everything'. A LSTM has three of these to regulate information. *There are basically 4 steps to reach the output of a particular cell:*

Deciding what to throw away

The first step is to decide what information we want to throw away, i.e. the information which no longer will be needed. For this purpose, the "*forget gate layer*" is used. It is a sigmoid layer that looks at previous output(h_{t-1}) and the input for this layer(x_t) and generates a number between 0 and 1 that decides how much to keep and how much to throw away.

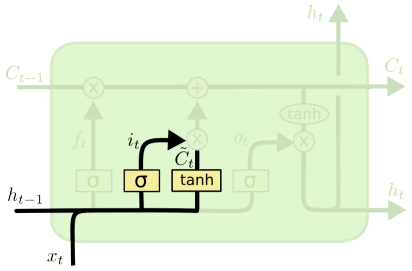


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 2.6 Forget Gate Layer. [1]

Deciding what to store

Now, after throwing away the unnecessary details we want to decide what information we need to store in our cell state, whether it is the information that was removed or some new information or both of them. To solve this purpose, our second sigmoid layer "*input gate layer*" along with a tanh layer is used. First, the sigmoid layer chooses which information is to be added to cell state then the tanh layer prepares a vector(C_t) which can be combined with the above result to update the cell state.

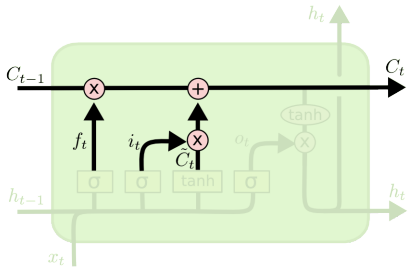


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 2.7 Input Gate Layer. [1]

Now to update the cell state from $\mathbf{C}_{t-1}$ to $\mathbf{C}_t$ we multiply the old state by output from forget layer $\mathbf{f}_{t-1}$,to forget the old information and add it to the multiplication of $\mathbf{i}_t$(what information to add) and $\mathbf{C}_t$(the vector for new candidates). So the new state

$\mathbf{C}_t$ becomes:

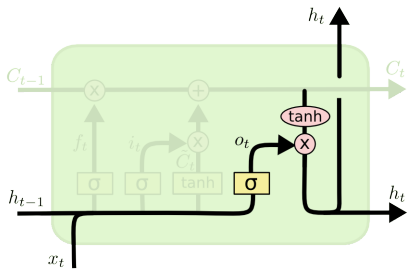


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 2.8 Updating C_{t-1} to C_t. [1]

Giving the output

After updating the cell state we need to give output that can be used by the next layer. The output we give is not the cell state but a filtered version of it because to predict the next thing maybe only some part of the information updated is required. This is done by another sigmoid layer known as "*output layer*" which controls which information should be passed ahead. This output o_t is multiplied to tanh of the cell state(to reduce it to [-1,1]) to give the output.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.9 Output Layer. [1]

Chapter 3

Related Works

- In 2009, Taraka Rama and Karthik Gali they have proposed the transliteration problem as translation problem.They have used phrase based statistical machine translation system and deployed it for transliteration. They have used GIZA++ and beam search based decoder . They have got an accuracy of 46.3% on their test set.[5]

- Another transliteration system was developed by Amita-va Das, Asif Ekbal, Tapabrata Mandal and Sivaji Bandyo-padhyay based on NEWS 2009 Machine Transliteration Shared Task training data sets. This paper reports about their works as part of the NEWS 2009 Machine Transliteration Shared Task. They have used the modified joint source channel model along with two other alternatives to generate the Hindi transliteration from an English word (to generate more spelling variations of Hindi names). They also devised some post processing rules to remove errors. During standard run, they obtained the word accuracy of 0.471 and mean F-score of 0.831. In nonstandard run, they used a bilingual database obtained from the web. The non-standard runs yielded the word accuracy and mean F-score values of 0.389 and 0.831 respectively in the first run and 0.384 and 0.823 respectively in the second run. [6]

Chapter 4

Our Approach

- Main problem faced by us in this task was to gather enough data so that our model could be trained efficiently.
  - To handle this problem we first converted Sanskrit text to Itrans notation.
  - The "Indian languages TRANSliteration" (ITRANS) is an ASCII transliteration scheme for Indic scripts, particularly for Devanagari script.
  - This Itrans notation was then converted to Roman script by creating all possible mapping from Itrans to Roman characters which was created manually based on the phoneme that the Itrans was capturing.
  - The main advantage of this technique was that our data set was enriched with multiple ways of writing a given word in roman script. Finally we trained our model on 1,52,000 words and cross-validated it on 38,000 words to fine tune the parameters and tested it on 10,000 words.
- To build our model we have used Seq2Seq model and then while decoding an unknown input sequence we go through a slightly different process.
  - Encode the input sequence into state vectors.

  - Start with a target sequence of size 1 (just the start-of-sequence character).
  - Feed the state vectors and 1-char target sequence to the decoder to produce predictions for the next character.
  - Sample the next character using these predictions (we simply use argmax).
  - Append the sampled character to the target sequence
  - Repeat until we generate the end-of-sequence character or we hit the character limit.

- For Devanagari script to Roman script we have used Harvard-Kyoto mapping scheme which is basically based on the phoneme of given Devanagari characters.

Chapter 5

Results

We have experimented our model by taking different test-sets. BLEU score, Word-error rate and accuracy were different metrices that we used to capture the quality of the system developed.

BLEU (bilingual evaluation understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" this is the central idea behind BLEU. BLEUs output is always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts. We have used it to capture the closeness of Transliteration as in this also we are having a sequence as input and a sequence as output.

Word error rate (WER) is a common metric of the performance of a speech recognition or machine translation system. The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one).

Results			
Test Set	**BLEU Score**	**Accuracy**	**Word-Error Rate**
Test Set 1	0.8167	0.4882	0.3808
Test Set 2	0.7959	0.4548	0.3729
Test Set 3	0.7980	0.4699	0.3712
Test Set 4	0.8065	0.4730	0.3755
Test Set 5	0.8001	0.4699	0.3728

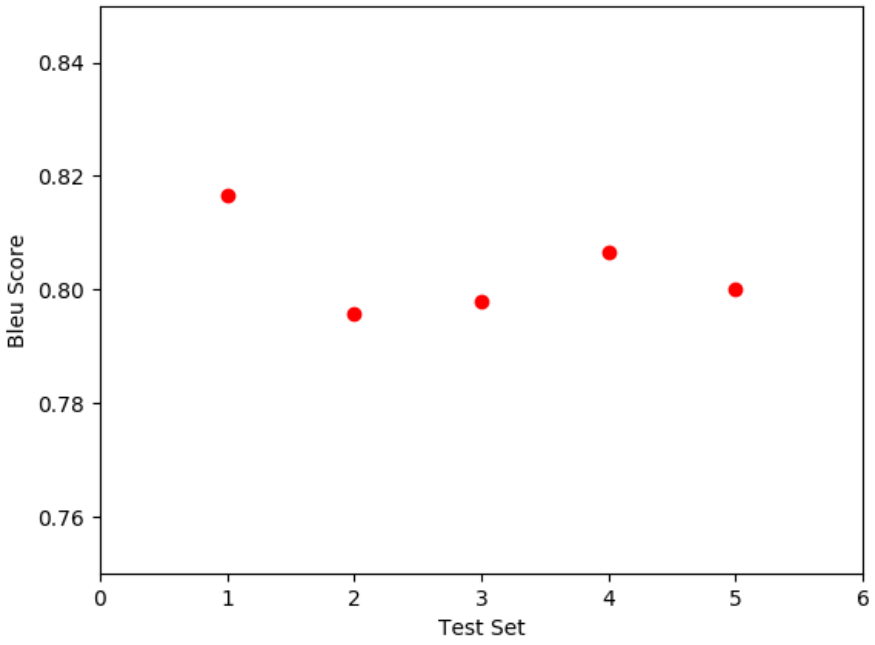


Figure 5.1 BLEU score vs Test-set

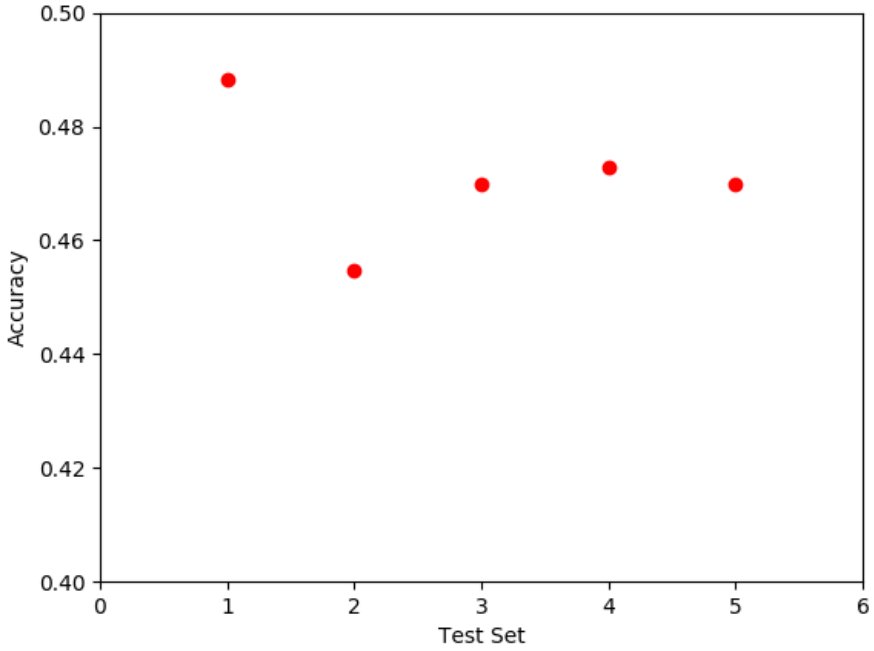


Figure 5.2 Accuracy vs Test-set

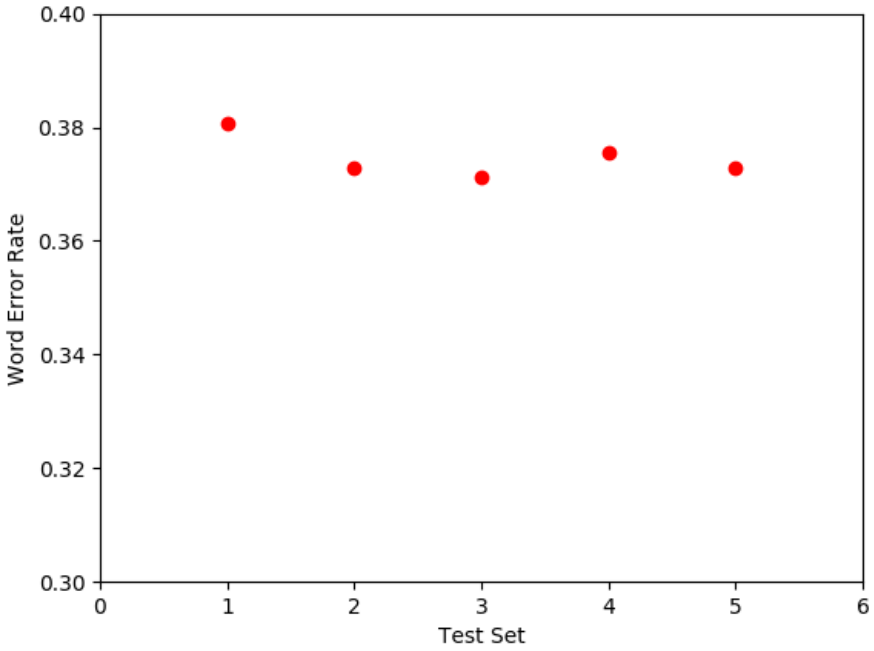


Figure 5.3 Word Error-Rate vs Test-set

Chapter 6

Discussion

The data set we have used is automatically generated. It is not able to capture the human cognition. But since every user has different ways of writing a particular word hence we need to feed the system with some manual data which would capture the different possibilities.

Future Directions

We plan to extend our model for cross-lingual information retrieval and apply information retrieval techniques on both, the transliterated and the original query in order to retrieve the set of relevant documents of the both the source and target language.

Conclusions

We have achieved an accuracy of 47.11% and a BLEU score of 80.34 The accuracy is lower than the already discovered tools but we believe that if we train the model with better data-set, there is a good hope of getting our accuracy improved. The accuracy is low because it is calculated by matching the whole word with one another(which is not a good method to measure the quality of the output). So for a better measure

we calculated BLEU score and word-error rate. The word-error rate was found to be 37.43%.

Bibliography

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[2] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: http://www.aclweb.org/anthology/D14-1179

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[4] F. C. Felix Gers, "In addition to the original authors, a lot of people contributed to the modern lstm. a non-comprehensive list is," in *Proc. of SECON*, 2008, pp. 170–178.

[5] K. G. Taraka Rama, "Modeling machine transliteration as a phrase based statistical machine translation problem," in *Language Technol-ogies Research Centre, IIIT, Hyderabad*, 2009.

[6] T. M. Amitava Das, Asif Ekbal and S. Bandyopadhyay, "English to hindi machine transliteration system at news," in *Pro-ceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009,Suntec, Singapore*, 2009.