Journal of Engineering Sciences

Vol 11, Issue 4 , April/2020
ISSN NO:0377-9254

TRANSLITERATION OF TEXT FROM ENGLISH TO HINDI

[1]K. Naren Sai Krishna, [2]P.Krishnanjaneyulu [3]K.Mahima, [4]G. Krishna Vamsi,[5]M. Tarun Vishnu

[2]Assistant Professor, [1,3,4,5]Student

Department of Computer Science & Engineering

Anil Neerukonda Institute of Technology and Sciences, Visakhapatnam

Abstract:

Transliteration is that the method of mapping characters of text written in one language into corresponding characters of another language. Transliteration is useful when the user knows the language but does not know how to script it. Transliteration not only deals with representing the sounds of the original language but rather represent the characters, ideally accurately and unambiguously. This paper deals with the transliteration of text from English to Hindi. Given a word in English, the model represents the same word in Devanagari script (Hindi) using two Recurrent neural networks i.e. an Encoder and a Decoder. The encoder network takes the input sequence in English and maps it to an encoded representation of the sequence. The encoded representation is then given to the decoder network which generates the output sequence in Hindi. Transliteration can used effectively in the case of nouns. The application of teacher forcing has shown significant improvement during the training phase. The accuracy of the model is found to be increased further when attention mechanism used in combination with Encoder-Decoder Model.

Keywords: RNN, Encoder-Decoder Architecture, GRU, Teacher Forcing, Attention-Based Mechanism.

1. Introduction:

India is the home to numerous languages. Although every region has its own native language, most of the sign boards in metropolitan cities are written in English. This makes it difficult for a non-English reader to understand the script.

Recurrent Neural Network (RNN) are a type of Neural Networks where the output from previous step are fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next letter of a word, the previous letters are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

The Encoder-Decoder architecture is a way of organizing recurrent neural networks for sequence prediction problems that have a variable number of inputs, outputs, or both inputs and outputs.

Transliteration is a problem in natural language processing for transforming the text in one script into another script so as to preserve the phonetic structure of words as closely as possible. Our model enables a non-English reader to understand the English sentences by converting them to their native language. Since the road names, city names, organization names, shop names etc. have the same pronunciation in every language, transliteration can be used the bridge the gap between the two languages.

It is also useful when a direct method to input data in a given language is unavailable. Hence, transliteration also can be understood as the process of entering data in one language using the script of another language. The Transliteration task uses two Recurrent neural networks. The encoder network is that network of GRU cells that takes the input sequence in English and maps it to an encoded representation of the sequence. The encoded representation is then used by the decoder network to generate an output sequence in

Hindi. The power of this model lies in the fact that it can map sequences of different lengths to each other. The transliteration system developed works a reading aid for non-English readers.

Transliteration aims to only change the letters or characters of a source language into corresponding letters of the target language. It does not provide meaning unlike translation, which is converts the written or spoken meanings of words or text of a source language into a target language.

In an encoder–decoder approach, the neural network needs to compress all the necessary information that is present in the source sentence into a fixed-length vector. This might sometimes make it difficult for the neural network to handle long sentences. To overcome this, we use attention mechanism which develops a context vector that is filtered specifically for each output time step instead of encoding the input sequence into a single fixed context vector. The accuracy of the model is further increased when the attention mechanism is utilized together with Encoder-Decoder Model.

2. Literature Survey:

The following are the some of the works on machine transliteration from English to Hindi and other Indian languages.

1. Taraka Rama and Karthik Gali[6] in 2009 addressed the transliteration problem as a translation problem. They used phrase based SMT systems for this task. This approach used publicly available GIZA++ and beam search based decoder for developing the transliteration model. A well organized English-Hindi aligned corpus is used to train the model and an accuracy of 46 percent on the test set is reported by this prototype.

2. Another transliteration system was developed by Amitava Das, Asif Ekbal, Tapabrata Mandal and Sivaji Bandyopadhyay based on NEWS 2012[7]. The transliteration system that they proposed uses the modified joint source channel along with other two alternatives to transliterate script from English to Hindi. This system also uses postprocessing rules to remove any sort of errors and to improve the accuracy. They performed one standard and two standard runs in their transliteration system. The results showed that performance with the standard runs is better compared with the non-standard runs.

3. Amitava Das, Asif Ekbal, Tapabrata Mandal and Sivaji Bandyopadhyay in 2009[7] addressed transliteration problem using the Letter-to-Phenome technology. In this proposed system, transliteration problem was interpreted as a variant of letter-to-phoneme subtask of text to speech processing. They applied a reimplement-ation of state-of-the-art , discriminative letter-to-phoneme to the problem without further modification. In this experiment, they demonstrated that an automatic letter-to-phoneme transducer performs well with no language specific or transliteration specific modifications.
4. Rejwanul Haque, Sandipan Dandapat, Ankit Kumar Srivastava, Sudip Kumar Naskar and Andy Way CNGL in 2009[7] proposed an English to Hindi Transliteration using Context-Informed Phrase-based statistical machine translation.The proposed transliteration system was modelled by translating characters rather than words as in character-level translating systems. They used a memory-based classification framework that enables efficient estimation of these features avoiding the data sparseness problems.

5. Gurpreet Singh Josan and Jagroop Kaur based on sta-tistical approach in 2011[8] developed a Punjabi to Hindi transliteration model. This system used a letter-to-letter mapping as a baseline and tried to find out the improvements by statistical methods. They used a Punjabi-Hindi corpus for training and openly available SMT tools for building the system.

6. Abbas Malik, Laurent Besacier Christian Boitet and Push-pak Bhattacharyya proposed an Urdu to Hindi Transliteration using hybrid approach in 2009[7]. This hybrid approach combines finite state machine based techniques with statistical word language model and this achieved better performance. The main aim of this system was the removal of diacritical words of the Urdu input text. This system improved the accuracy by 28.3

% compared to their previous finite transliteration model.

3. Models and Architecture:

3.1 Encoder-Decoder Recurrent Neural Network:

The architecture involves two components:

-An encoder and a decoder.

• Encoder: The encoder reads the entire input sequence and encodes it into an internal representation, often a fixed-length vector called the context vector.

• Decoder: The decoder reads the encoded input sequence from the encoder and generates the output sequence.

Both the encoder and the decoder sub models are trained jointly, meaning at the same time.

The entire encoded input is used as context for generating each step in the output.

Although this works, the fixed length encoding of the input limits the length of output sequences that can be generated.

An extension of the Encoder-Decoder architecture is to provide a more expressive form of the encoded input sequence and allow the decoder to learn where to pay attention to the encoded input when generating each step of the output sequence.

This extension of the architecture is called attention.

The Encoder-Decoder architecture with attention is popular in the case of natural language processing problems that generate variable length output sequences. The application of architecture to text transliteration is as follows:

• Encoder: The encoder is responsible for reading a word in source language and encoding it to an internal representation.

• Decoder: The decoder is a language model responsible for generating the word transliteration is as follows:
in the target language using the encoded representation of the source language.

3.1.1 Teacher Forcing:

Teacher forcing is a method to quickly and efficiently train RNN models that use the ground truth from a prior time step as input. *Teacher Forcing when applied in training phase makes the neural network to* converge faster. During the initial stages of training, the predictions of the model are not at all meaningful. So, we use teacher forcing and pass the ground truth as input instead of passing the predicted outputs.

Without *Teacher Forcing*, the hidden states of the model will be updated by a sequence of wrong predictions in the decoder phase which causes more errors to accumulate, and it is difficult for the model to learn. For the transliteration task, we ran the neural network with teacher forcing applied for 25% of the dataset during the training phase.

For the remaining 75% of the dataset, we trained the neural network without teacher forcing i.e.. the outputs of the previous node is passed as input to the current node instead of ground truth values.

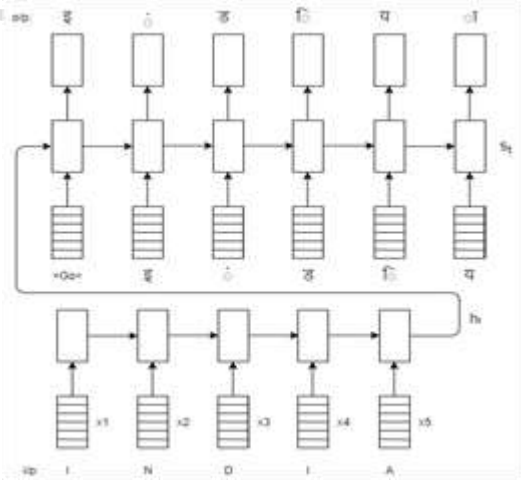

Fig. 1: English to Hindi transliteration using Encoder Decoder Architecture with teacher forcing.

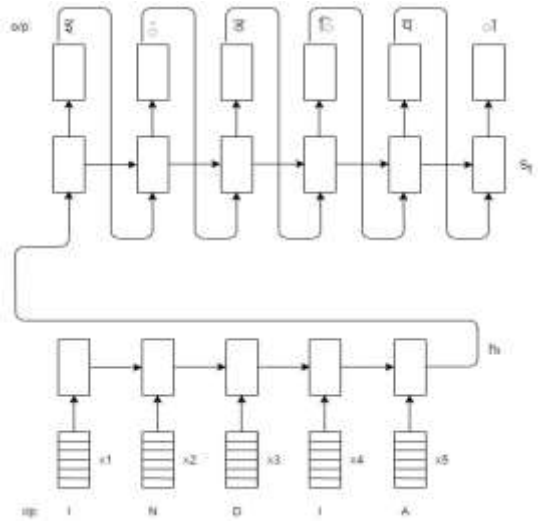

Fig. 2: English to Hindi transliteration using Encoder Decoder Architecture without any teacher forcing applied.

Encoder: In this approach, the feature vector corresponding to each word is first fed to the input of encoder. This feature vector has a fixed length for different words. At each time step, the feature vector corresponding to a source word enters the repeating encoder module. Then, the hidden state in the encoder captures the information relevant to the seen words up to that time step. This vector is computed based on the hidden state of the previous time step, and the input of the current one:

$$H_t = \Phi(H_{t-1} , x_t) = f(W^{(hh)} * H_{t-1} + W^{(hx)} * x_t) \quad (1)$$

In (1), Φ is a nonlinear activation function which is usually considered to be sigmoid function. Weight vectors $W^{(hh)}$ and $W^{(hx)}$ are learned during the training process. These vectors define the weights between hidden states at different moments, as well as the weights between hidden states and inputs. At each time step, a word from the source sentence is given to the encoder.

Decoder: Once the last word in the source sentence reaches the encoder, the input state of the encoder at that time step will represent the whole input sentence. This vector is then fed to the decoder in order to emit the first word in the destination sentence. Afterward, the decoder produces the next destination words using the previous hidden state and produced destination words. The equation of computing hidden state h_t and output Y_t at time step t is as follows:

$$H_t = \Phi(H_{t-1}) = f(W^{(hh)} * H_{t-1}) \quad (2)$$

$$Y_t = \text{softmax}(W^{(s)} * H_t)$$

3.2 Encoder-Decoder Recurrent Neural Network with Attention Mechanism:

The attention mechanism emerged as an improvement over the encoder decoder-based neural MT system in tongue processing (NLP). Later, this mechanism, or its variants, was utilized in other applications, including computer vision, speech processing, etc.

Attention is a mechanism in the RNN allows it to focus only on certain parts of the input sequence when predicting a certain part of the output sequence, which increase the quality of learning and also enables easier learning. Combination of attention mechanisms is observed many times to improve performance making it an integral part of modern RNN networks.

Consider the encoder's input vectors by x1, x2, x3, x4 and the output vectors by h1, h2, h3, h4. The attention mechanism is located in between the encoder and decoder, its input comprises of the encoder's output vectors h1, h2, h3, h4 and the decoder states s0, s1, s2, s3, the attention's output is a series of vectors called context vectors denoted by c1, c2, c3, c4. The context vectors enable the decoder to focus only on certain parts of the input while predicting the output. Every context vector is a weighted sum of the encoder's output vectors, each vector hi contains information about the whole input sequence with stronger focus on the neighbouring parts of the i-th vector of the input sequence. The vectors h1, h2, h3, h4 are scaled by weights αij which captures the degree of relevance of input xj to output at time i, yi.

The context vectors c1, c2, c3, c4 are given by:

$$c_i = \Sigma \alpha_{ij} * h_j$$

The attention weights are learned using the attention fully-connected network, denoted by fc, and a softmax function. The fully connected network is trained along with the encoder and decoder and backpropagation, the RNN's prediction error terms are backpropagated through the decoder, then through the fully connected attention network and finally to the encoder.

This mechanism enables the decoder to decide to which part of the input sequence it has to pay attention to. By implementing the attention mechanism at decoder, we relieve the encoder from having to encode all information in the input sequence into a single context vector. The information can be spread throughout the sequence h1, h2, h3, h4 which can be selectively retrieved by the decoder as and when required.

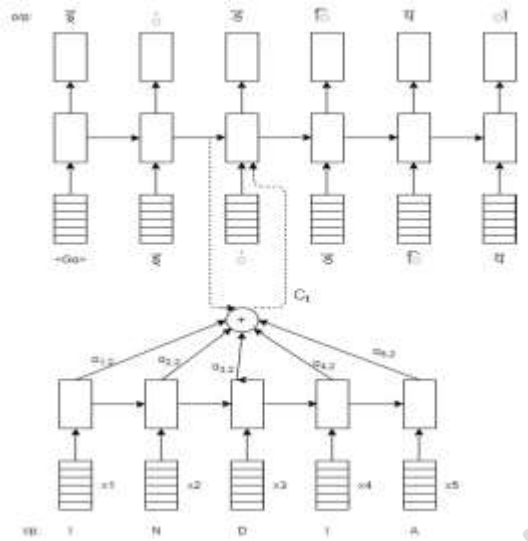

Fig. 3: English to Hindi transliteration using Encoder Decoder Architecture using attention mechanism.

3.3 Dataset loading and pre-processing

The dataset containing 13,000 examples for training and 1,000 examples for testing is procured from {Zhang, Min and Li, Haizhou and Kumaran, A and Liu, Min}[1] in XML format. The XML tree structure makes traversal, modification, and removal of data items relatively simple programmatically from the dataset. The built in Python-library, **ElementTree** has functions to read and manipulate XMLs (and other similarly structured files).

During the data pre-processing stage, we clean both the English words and Hindi words separately.

i) All the words other than the respective language words are removed.

ii) Words representations with different sizes in Hindi and English are discarded.

Finally, both the English words and Hindi words are returned.

A python dictionary containing the <key,value> pairs with keys as English alphabets and values as their respective indexes is formed. There are 26 characters in English. **Unicode range={65,90}**. Then, the English word is transformed from character representation to numerical representation in-order to perform numerical computations on the data. One hot encoding is a process by which categorical variables are converted into a form involving numerical values that could be provided to ML algorithms to do a better job in prediction. So, the categorical value represents the numerical value of the entry in the dataset. 0 indicates non-existent while 1 indicates existent.

A python dictionary containing the <key,value> pairs with keys as Hindi alphabets and values as their respective indexes is formed. There are 128 characters in Hindi. **Unicode range-{2304,2431}**

4. Results

The following table shows the results of our experimental study on **test data**.

Iterations	Accuracy without attention mechanism	Accuracy with attention mechanism
1000	**50.82 %**	**63.56 %**
5000	**73.64 %**	**77.79 %**
10000	**80.78 %**	**83.41 %**
15000	**82.17 %**	**84.04 %**

The following table shows the results of our experimental study on **training data**.

Iterations	Accuracy without attention mechanism	Accuracy with attention mechanism
1000	**56.14 %**	**69.60 %**
5000	**86.41 %**	**89.22 %**
10000	**96.51 %**	**96.39 %**
15000	**97.97 %**	**98.10 %**

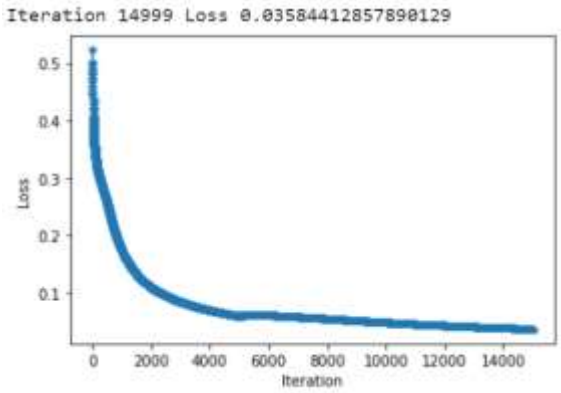

Fig 4: Loss vs Iterations graph for English to Hindi transliteration (Training phase)

5. Conclusion:

Machine transliteration can play an important role in application involving natural language processing such as information retrieval and machine translation. This paper discusses the transliteration process for text from English to Hindi in two methodologies, one using Encoder Decoder architecture and the other by adding attention mechanism. The attention mechanism has achieved much greater accuracy. However, the limitation of the model that has been trained by us is that it cannot represent the silent words in English language. The limitation can be solved by including more words with silent letters during the training phase.

6. References:

[1]**DatasetSource:**

@InProceedings{zhang2012report,author={Zhang,MinandLi,HaizhouandKumaran,AandLiu,Min},title={ReportofNEWS201MachineTransliterationSharedTak},booktitle={theACL2012NamedEntitieWorkShop(NEWS-2012),JejuIsland,SouthKorea}, url = {https://www.microsoft.com/enus/research/publication/report-of-news-2012machine -transliteration-shared-task/},edition={the ACL2012NamedEntitiesWorkShop(NEWS-2012),JejuIsland,SouthKorea}, }

[2]RanbeerMakin.,NikitaPandey.,PrasadPingali.,and VasudevaVarma.Experimentsincrosslingualiramongindianlanguages.InInternationalWorkshoponCrossLanguageInformation Processing(CLIP),2007.

[3]PeterM.Klausler.Evolutionaryalgorithmtodiscover betterkeyboard layouts.Inhttp://klausler.com/evolved.html.

[4]AnimeshNayan.,RaviKiranRaoB.,PawandeepSingh.,SudipSanyal.,andSanyal.Ratna.Namedentityrecognitionforindianlanguages.InWorkshoponNERforSouthandSouthEastAsianLanguages(NERSSEA),InternationalJointConferenceonNaturalLanguageProcessing (IJCNLP),2008.

[5]PrasadPingali.andVarma.Vasudeva.Wordnormalizationinindianlanguages.In4thInternational ConferenceonNaturalLanguageProcessing(ICON),2005.

[6]Taraka Rama, Karthik Gali (2009), 'Modeling Machine Transliteration as a Phrase Based Statistical Machine Translation Problem', Language Technologies Research Centre, IIIT, Hyderabad, India.

[7]Amitava Das, Asif Ekbal, Tapabrata Mandal and Sivaji Bandyopadhyay (2009), 'English to Hindi Machine Transliteration System at NEWS', Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, page 80-83, Suntec, Singapore.

[8] Gurpreet Singh Josan & Jagroop Kaur (2011)' Punjabi to Hindi Statistical Machine Translaiteration', International Journal of Information Technology and Knowledge Management July-December 2011, Volume 4, No. 2, pp. 459-463.

[9] Machine Transliteration for Indian Languages: A Literature Survey, Antony P J and Dr. Soman K P, International Journal of Scientific & Engineering Research, Volume 2, Issue 12, December-2011 ISSN 2229-5518.

[10] M. Mahdi Mahsuli and R. Safabakhsh, "English to Persian transliteration using attention-based approach in deep learning," *2017 Iranian Conference on Electrical Engineering (ICEE)*,Tehran,2017,pp.174-178.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7985375&isnumber=7985087

[11] Rosca, M., & Breuel, T. (2016). Sequence-to-sequence neural network models for transliteration. ar Xiv preprint ar Xiv: 1610.09565.

[12] Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint ar Xiv: 1406.1 078.

[13] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv: 1409.0473.

[14]https://en.wikipedia.org/wiki/Machine_learning

[15]https://en.wikipedia.org/wiki/Deep_learning