International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 06 | June 2021 www.irjet.net p-ISSN: 2395-0072

SEMANTIC SEQUENTIAL ENGINE USING RECURRENT ENCODER – DECODER MODELS

Kartik Soni[1], Dr. Boominathan P[2]

[1]*Student, School of Computer Science and Engineering (SCOPE), VIT University, Vellore, Tamil Nadu, India*
[2]*Professor, School of Computer Science and Engineering (SCOPE), VIT University, Vellore, Tamil Nadu, India*

Abstract - Machine transliteration is a new field of study that transforms words from one language to another while maintaining their phonetic properties. Machine translation and cross-language information retrieval benefit from the use of transliteration. In a machine translation system, transliteration is primarily employed to handle named entities and out-of-vocabulary words. The phonetic structure of the words is preserved. To efficiently use orthographic similarity, we present a modified neural encoder-decoder model that maximizes parameter sharing across language pairs. We further show that bilingual transliteration models generalize well to languages/language pairs not encountered during training and, as a result, perform well on the zero shot transliteration task. We used this method to transliterate English to Hindi and were able to get precise Hindi transliterations for 80-85% of English names.

Key Words*: *Transliteration, Language, Bilingual.

1. INTRODUCTION

The main purpose of our suggested application is to determine the language of a certain text and then attempt to classify and understand it. This application's usefulness is derived from its ability to combine two fields: natural language processing (NLP) and semantic web. NLP can be thought of as a means to offer a machine the ability to deal with languages in more sophisticated ways. The semantic web, on the other hand, is a technology that aims to give meaning to data on the internet.

Transliteration - The text is transliterated from one script to another. Transliteration is divided into two stages: first, the source language word is segmented into transliteration units, and then these units are aligned and mapped to target language units. For example, the word "Mera" can be divided into (m,e,r,a) units, which are then transliterated into target language units. Transliteration is mostly used to transform foreign words in a language that must be phonetically equivalent to terms in another language but are not grammatically equal.

Machine translation (MT) was the first application of natural language processing with the goal of decoding text from one language to other, and it was developed in the late 1940s.It was founded on the premise that the differences across languages are found in the vocabularies and word order that make up the language. As a result, MT condensed the entire language idea to a dictionary query to identify the translation of a given word, followed by an attempt to re-order the words to meet the rules of the target language. This reductionist approach demonstrates a limited comprehension of natural languages and demonstrates how linguistic theorists are still needed in NLP to properly grasp lexical ambiguity.

Another factor to consider is the text's context, as determining the meaning of a text is linked in some manner to the text's subject matter, whether it is social, scientific, or political. To put it another way, languages are dynamic, and there is a requirement to respond to text outside the language's normal semantics.

2. LITERATURE SURVEY

"Dzmitry Bahdanau et al. developed a novel architecture to resolve this problem" [1]. When producing each target word, they enhanced the basic encoder and decoder model by introducing a model soft search process for a set of input words. As a result, the model doesn't have to encode the entire source sentence into a fixed-length vector, and it can concentrate just on the information needed to generate the next target word. This has a significant impact on the neural machine translation system's capacity to produce better outcomes for lengthier sentences.

"Leena Jain and colleagues created a technology that converts English to Sanskrit" [2]. The process of changing the letters of typed text in one language to the letters of another language is known as transliteration. The strategy employed was to create a transliteration algorithm that employed Unicode. English and Hindi Unicode are mapped to one other. The letters are mapped to Hindi using the mapped Unicode and the input is in English. The Hindi text is the output. Conclusion and Result All of the test cases were successful. The precision is perfect. The software can be used for both machine learning and natural language translations. The user interface is straightforward.

An "Automatic English-Chinese name Transliteration"system has been proposed by Wan and Verspoor[3]. The method transliterated words based on how they were said. That is, a written English word was mapped to a written Chinese character using the word's spoken form. The technique functioned by converting an English word into a phonemic representation and then assigning each phoneme to a Chinese character. Semantic Abstraction, Syllabification, Sub-syllable divisions, Mapping to Pinyin, and Mapping to Han characters were the five phases of the transliteration process.Semantic abstraction was a preprocessing step that looked up words in dictionaries to see which parts should be translated and which should be transliterated. Each sub syllable's phonetic representation was converted to Pinyin, the most used standard Mandarin Romanization system.

"2008, Harshit Surana and Anil Kumar Singh developed a transliteration scheme for Hindi and Telugu, two Indian languages" [4]. A word was first categorized as Indian or foreign in their experiment using character-based n - grimes. Based on symmetric cross entropy, the likelihood of the word's origin was calculated. Transliteration was done for different classes using different strategies based on this probability value (Indian or foreign). The system employed a lookup dictionary or a direct mapping from English phoneme to IL letters to transliterate foreign words. The method split the word based on probable vowels and consonant combinations, then mapped these segments to their nearest letter combinations using specific guidelines for transliteration of Indian words. The foregoing stages generated transliteration candidates, which were then filtered and sorted using fuzzy string matching, which matched the transliteration candidates to terms in the target language corpus to produce target words. This approach does not handle terms that are not in the lexicon.

"Deep and Goyal created a Rule-based Punjabi to English transliteration method" [5], The suggested system uses a set of character sequence mapping rules to translate between the languages. The rules are written with particular constraints in mind to improve accuracy. This system was trained with 1013 people's names and tested with a variety of people's names, city names, river names, and so on. The overall accuracy of the system was assessed to be 93.22 percent.

"Melvin Johnson et al. communicate across many languages using a single Neural Machine Translation (NMT) model" [6]. There is no change to the conventional NMT system's default model architecture; instead, a fake token is added to the beginning of the input phrase to identify the desired target language. The model contains an encoder, decoder, and attention module that are all shared between languages and remain unmodified. They employed a vocabulary that they had in common. Their methodology allows Multilingual NMT to be performed using a single model with no parameter increases or additions, which is substantially easier than earlier Multilingual NMT ideas.

"Dhore et al. proposed utilizing Conditional Random Fields to transliterate Named Entities from Hindi to English" [7]. The system takes Indian location names as input in Hindi using the Devanagari script and transliterates them into English. In order to use n-gram approaches, the input is provided in the form of syllabification. In the transliterated version of English, this syllabification preserves the phonemic qualities of the source language Hindi. The goal is to use CRF as a statistical probability technique and n-gram as a feature set to generate transliteration of a named entity given in Hindi to English.A multilingual corpus of 7251 named items was built from web resources and books to test the proposed method. "Word accuracy?" was a typical performance evaluation criterion. For the bi-grams of the source language Hindi, the algorithm achieved a very high accuracy of 85.79 percent.

"Mikel Artetxe et al. fully eliminated the necessity for parallel data and presented a novel method for training an NMT system" [8] unsupervised using monolingual corpora. Their new work on unsupervised embedding mappings uses a slightly modified attentional encoder-decoder model that can be trained using a mixture of denoising and back translation on monolingual corpora alone. Small parallel corpora can also benefit the model, as it achieves 21.81 and 15.24 points when paired with 100,000 parallel sentences, respectively.

"Sanjanashree and Anand Kumar provided a deep learning-based architecture for multilingual machine transliteration in English and Tamil" [9]. The system employs a generative graphical model known as the Deep Belief Network (DBN). Preprocessing, DBN training, and testing are the three steps in the transliteration process. Tamil words are Romanized at the preprocessing phase. Both languages' data is transformed into sparse binary matrices. To keep the length of the words constant while encoding as sparse binary matrices, character padding is done at the end of each word.The Deep Belief Network is a generative graphical model made up of numerous layers of Restricted Boltzmann Machines, which are a combination of Random Markov Fields and Boltzmann Machines.

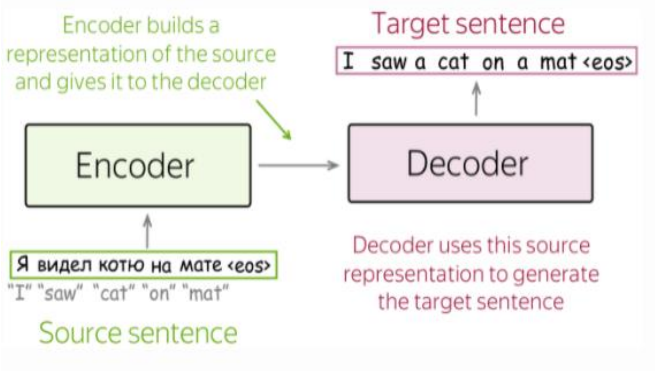
"Using a hybrid method, Mathur and Saxena created a system for English-Hindi named entity transliteration" [10]. The algorithm first uses rules to extract phonemes from English words. After that, a statistical methodology is used to transform the English phoneme to its Hindi equivalent. The authors extracted 42,371 name entities using Stanford's Name Entity Extractor (NER). These things were subjected to rules, and phonemes were extracted. These English phonemes were transliterated into Hindi, and an English-Hindi phoneme knowledgebase was constructed.

"Rico Sennrich et al. proposed a simpler and more successful method for making the NMT model" [11] capable of open-vocabulary translation by encoding rare and unknown terms as subword unit sequences. Byte Pair Encoding is a simple data compression method that replaces

the most common pair of bytes in a sequence with a single, unused byte iteratively. NMT models normally work with a defined vocabulary, while translation is an open-vocabulary problem. Their previous work focuses on using a dictionary to translate words that aren't in their vocabulary.

3. PROPOSED SYSTEM

Language modelling is the process of predicting the next word or letter. In contrast to FNN and CNN, the current output of sequence modelling is reliant on the prior input, and the duration of the input is not fixed. We want to anticipate the ith word based on the previous words or information given a set of 't-1' words. This is how we use Recurrent Neural Networks to address the language modelling problem.



The function's input is indicated in orange and represented as an x_t. The weights associated with the input are represented by a vector U, and the hidden representation (s_t) of the word is calculated as a function of the previous time step's output and the current input, as well as bias.

$$s_T = \sigma(Ux_T + Ws_{T-1} + b)$$

$$y_t = O(Vs_T + c)$$

The hidden representation's output (s_t) is generated by the following equation: we compute the hidden representation of the input, and the network's final output (y_t) is a softmax function (written as O) of the hidden representation and weights associated with it, as well as the bias.

ENCODER

• The first time step's output is provided as input to the RNN, together with the original input to the following time step.

• The hidden representation (st1) of the word is computed at each time step as a function of the previous time step's output and the current input, plus bias.

• The final hidden state vector(st) holds all of the encoded data from the previous hidden representations and inputs.

• In this case, the Recurrent Neural Network serves as an encoder.

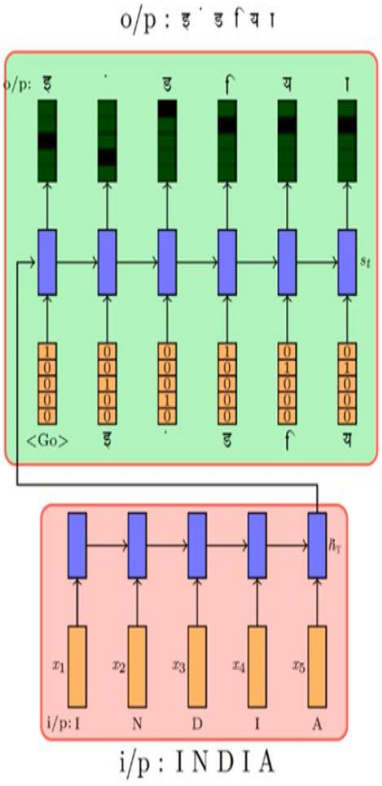
DECODER

• After passing the encoded vector to the output layer, the probability distribution of the next possible word is decoded.

• The hidden state representation and weights associated with it, as well as the bias, are inputs to the output layer, which is a softmax function.

• The output layer can be referred to as a simple feed-forward neural network because it incorporates the linear transformation and bias operation.

• The Decoder is a Feed-Forward Neural Network.



Model

- **Encoder:**

$$h_t = RNN(h_{t-1}, x_{it})$$

- **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$
$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$
$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

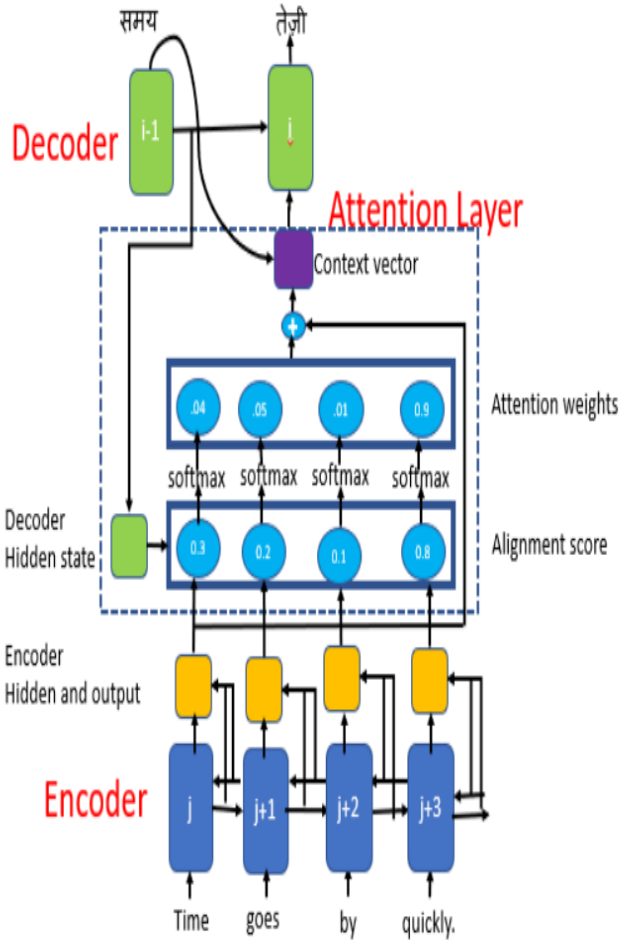
ATTENTION MODEL SOLUTION

The issue with this approach is that the encoder only reads the complete sentence once and must remember everything before converting it to an encoded vector. Longer phrases will cause the encoder to forget the beginning sections of the sequence, resulting in information loss.

Humans attempt to translate each word in the output by concentrating on only a few words in the input. We extract just important information from large sentences at each time step and then translate that specific word. *Only relevant information (encodings of relevant information) should be fed to the decoder for translation at each time step, ideally.*

What else do we require?

As a result, we assign α weight (ranging from 0 to 1) to each input word that signifies the importance of that word for the output at time step 't'. For example, α12 represents the influence of the first input word on the output word at the second time-step. To generalise, the weight associated with the j^{th} input word at the t^{th} time-step is represented by the representation αjt. For example, at time-step 2, we might simply provide the decoder a weighted average of the appropriate word representations together with the weights αjt. In this case, the weighted representation of the words is fed into the decoder rather than the entire encoded vector.



Gated Recurrent Units (GRU RNNs) are used to implement the recurrent networks. With residual connections between layers to enhance gradient flow, GRU RNNs are used. To accomplish parallelism, we connect the attention from the bottom layer of the decoder network to the top layer of the encoder network. We use low-precision arithmetic for inference, which is further accelerated by specific hardware, to reduce inference time. Experiments reveal that the created translation system's quality is comparable to that of human translators.

Encoder

• When comparing the encoder operation to the vanilla form of encoder-decoder architecture, there isn't much difference.

• The representation of each word is computed at each time step as a function of the previous time step's output and the current input, plus bias.

• The final hidden state vector(s_t) holds all of the encoded data from the previous hidden representations and inputs.

• A RNN encoder is utilized.

Encoder:

$$h_t = RNN(h_{t-1}, x_t)$$

$$s_0 = h_T$$

Decoder

• In a traditional encoder-decoder paradigm, we'd send the complete encoded vector to the output layer, which would decode it into the probability distribution of the next probable word.

• Rather of transmitting the complete encoded vector, we must calculate the attention weights using the fancy equation we discussed in the last part to determine **e**jt. Then, using the softmax function, normalize the **e**jt weights to get αjt.

• We'll compute the weighted combination of all the inputs and weights to produce the resultant vector Ct once we have all the inputs and weights to feed into the decoder (thanks to the fancy equation!).

• We'll feed the Decoder RNN the weighted combination vector Ct, which will decode the probability distribution of the next probable word. This decoding operation applies to all of the time-steps in the input.

• The bias, as well as the hidden state representation and related weights, are all inputs to the output layer, which is a softmax function.

Decoder:

$$e_{jt} = V_{attn}^T tanh(U_{attn} h_j + W_{attn} s_t)$$

$$\alpha_{jt} = softmax(e_{jt})$$

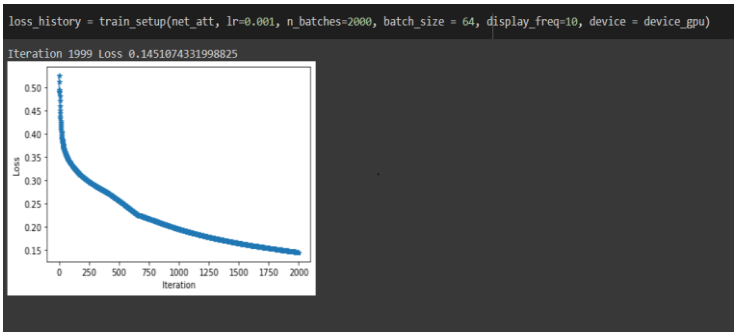
$$c_t = \sum_{j=1}^{T} \alpha_{jt} h_j$$

$$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])$$

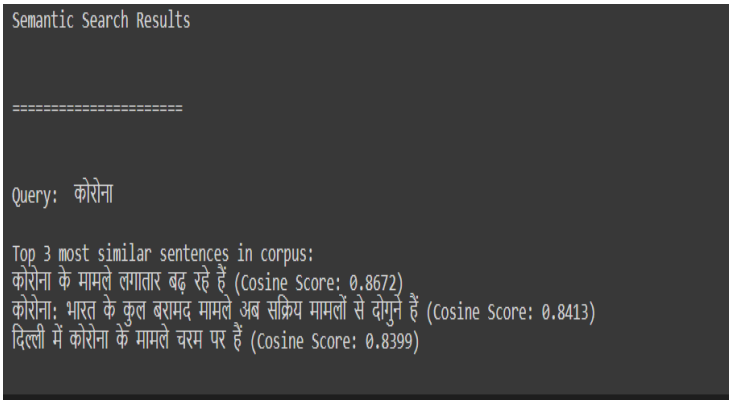
$$\ell_t = softmax(Vs_t + b)$$

4. IMPLEMENTATION AND RESULTS

Attention Loss:



Query Results:



5. CONCLUSIONS

Training an NMT system on a large-scale translation dataset takes a significant amount of time and computational resources, limiting the rate of experimental turnaround time and innovation. They are often significantly slower than phrase-based systems when it comes to inference because of the usage of large parameters.

Second, NMT is unreliable when it comes to translating uncommon terms. Though this can be addressed in theory by training a "copy model" to mimic a traditional alignment model or by using the attention mechanism to copy rare words, both of these approaches are unreliable at scale, as the quality of the alignments varies across languages and the latent alignments produced by the attention mechanism are unstable when the network is deep. Furthermore, when dealing with unusual words, such as when a transliteration is more appropriate, simple copying may not always be the ideal method. Finally, NMT systems occasionally generate output sentences that do not fully translate the original sentence.

We have offered a survey on problems, diverse techniques, and assessment criteria for several machine transliteration systems in this academic work. We've also included a list of some of the current transliteration systems.

According to the results of the survey, practically all existing language machine transliteration systems use a statistical or hybrid methodology. We used a bilingual transliteration model for languages (English and Hindi); the basic encoder-decoder model had an accuracy of around 72 percent, while the Attention encoder-decoder model had an accuracy of around 84 percent.

Given that transliteration is frequently employed as part of machine translation systems, and since these systems are increasingly character-based end-to-end systems, the question of whether distinct transliteration models are even necessary emerges. Internal graphemic and phonetic representations inside transliteration modules are likely to be rather different from internal semantic representations required for translation, hence transliteration is likely to remain a separate submodule of such systems. Separate and unique processing of proper nouns and other nouns is also supported by human experimental research.

REFERENCES

[1] "NEURAL MACHINETRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE," by D. Bahdanau, K. Cho, and Y.

[2] "English to Sanskrit Transliteration: an Effective Approach to Design Natural Language Translation Tool," by L. Jain, In. Journal of Advanced Research in Computer Science, Volume 8, No. 1, Jan-Feb 2017.

[3] "Automatic English-Chinese name transliteration for the construction of multilingual resources," S. Wan and C. M. Verspoor, Proceedings of the 17th international conference on Computational linguistics-Volume 2, 1998, pp. 1352-1356.

[4] "An Adaptable Multilingual Transliteration Mechanism for Indian Languages," by H. Surana and A. K. Singh, in IJCNLP, 2008, pp. 64-71.

[5] K. Deep and V. Goyal, "Punjabi to English transliteration system," International Journal of Computer Science and Communication, vol. 2, pp. 521-526, 2011

[6] "Multilingual Neural Machine Translation System: Zero-Shot Translation," by M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, and N. Thorat (2017)

[7] " Using conditional random fields Hindi to English machine transliteration of names," International Journal of Computer Applications, vol. 48, pp. 31-37, 2012.

[8] K. Cho, M. Artetxe, G. Labaka, and E. Agirre, "UNSUPERVISED NEURAL MACHINE TRANSLATION," ICLR 2018 conference paper.

[9] "Joint layer based deep learning framework for bilingual machine transliteration," , Advances in Computing, Communications and Informatics, P. Sanjanaashree (ICACCI, 2014 International Conference on, pp. 1737-1743, 2014).

[10] S. Mathur and V. P. Saxena, "Hybrid approach to English-Hindi name entity transliteration,", 2014, pp. 1-5.

[11] "Neural Machine Translation of Rare Words with Subword Units," R. Sennrich, B. Haddow, and A. Birch, submitted on August 31, 2015

[12]"Complete Urdu-Hindi Transliteration System Development", 2012, pp. 643-652. G. S. Lehal and T. S. Saini.

[13] "Algorithms for Arabic name transliteration," IBM Journal of Research and Development, vol. 38, pp. 183-194, 1994. M. Arbabi, S. M. Fischthal, V. C. Cheng, and E. Bart, " Transliteration Algorithms for Arabic name,", vol. 38, pp. 183-194, 1994.

[14] B.-J. Kang and K.-S. Choi, "Decision Tree Learning for Automatic Transliteration and Back-Transliteration," in LREC, 2000.

[15] "Neural Machine Translation challenges" by P. Koehn and R. Knowles, Proceedings of the First Workshop on Neural Machine Translation, pages 28–39. (2017).

[16] 2010 International Conference , "Kernel method transliteration," in Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on, pp. 336-338.