TRANSLITERATION OF INDIAN LANGUAGES IN SOCIAL MEDIA CONTEXT

A Project Report Submitted

for the Course

MA498 Project I

by

Mitanshu Mittal - 160123020

Rahul Kumar Gupta - 160123030

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

November 2016

CERTIFICATE

This is to certify that the work contained in this project report entitled "Transliteration of social media text for low resource languages" submitted by Mitanshu Mittal (Roll No.: 160123020) and Rahul Kumar Gupta (Roll No.: 160123030) to the Department of Mathematics, Indian Institute of Technology Guwahati towards partial requirement of Bachelor of Technology in Mathematics and Computing has been carried out by him/her under my supervision.

It is also certified that this report is a survey work based on the references in the bibliography.

OR

It is also certified that, along with literature survey, a few new results are established/computational implementations have been carried out/simulation studies have been carried out/empirical analysis has been done by the student under the project.

Turnitin Similarity: 14%

Guwahati - 781039
November 2019

Prof. Gautam K. Das
(Project Co-Supervisor)
(Mathematics Department)

Prof. S. Ranbir Singh.
(Project Co-Supervisor)
(CSE Department)

ABSTRACT

Since the popularity of social media in India, the amount of unstructured text is enormous. The informal social media environment promote colloquial elements in the text. For example, in the India context, many of Indians converse in native languages using Latin (transliterated) script without following proper grammatical or transliteration rules. One way to leverage this huge data is to transliterate them back to the native script and use traditional text processing models. But most of the machine transliteration studies have so far mostly focused on formal text. We, therefore want to study state-of-the-art solutions of machine transliteration and investigate their effectiveness over noisy social media data. This will help us to better understand the strength and weaknesses of state-of-the-art transliteration methods and propose better solutions appropriate for a noisy environment. In this work, we start with introduction to the transliteration task, and then study major challenges in the field along with state of the art solutions. Finally we evaluated different variations of character level Encoder-Decoder models for word transliteration and reported the results and observations obtained.

Contents

Chapter 1

Introduction

The process of phonetically transforming the words of a language from one script to another script is called transliteration.[13] for e.g (bacche –> बच्चे), (hamara –> हमरा).Converting a word written in the native script to a foreign script(for example "खुशबू" to "Khushboo," Devanagari is the native script here) is called forward-transliteration, and the conversion of a word back to its native script is called back-transliteration for example, ("bacche" to "बच्चे"). Traditionally transliteration has been used as a subtask of translation for converting words, which are nouns, names or out of dictionary words.

1.1 Motivation:

Since the advent of Web 2.0, the amount of data on the internet has been increasing at an exponential rate. There are over 3.48 billion social media users in 2019, and the number of users are growing at around 9% each year, with about a quarter of them in India.[1] Much of the textual information contained in these social media channels is in transliterated form. So in recent

years, transliteration has become an important research topic in itself. Most of the Indians prefer to write regional language using Latin script. The reason being (1) Modern computers and mobiles provide seamless support for Latin script (2) Most of the Indian users are well-conversant with English (3) Lack of suitable software for rendering native script (4) Users are inexperienced with writing in their native script.

1.2 Application:

The inference gained from the social media data can help us enhance social security, understand public sentiment, more personalized advertisement, better information retrieval, improve machine translation, and understand customer sentiment to improve marketing and customer services. However, existing text mining models are trained to analyze traditional formal text. However, since a large amount of this data is in the transliterated form, lack of standard solutions to process transliterated data hinders the extraction of useful information from this enormous amount of data. Therefore, an effective transliteration model for social media text can aid different text processing applications in extracting useful information.We are interested in exploring models for backward transliteration of social media text.

There are two approaches we can consider to analyze social media text (1) Build new text-mining models for each task trained over noisy text (2) Perform back-transliteration to convert this text to formal text. The former approach requires us to build and train new models for every task. The latter approach leverages the existing text mining models for text analysis and eliminates the need to build one for noisy text.

1.3 Challenges:

The social media data is noisy, which means that while writing a native language in Latin script, there may be different variations of a single word. For example "बच्चे" can be written as "bache" or "bachhe". The variations occur due to the following reasons:

- **Informal Transliteration:** While writing the native language in Latin script, people do not follow a common standard, e.g., ITRANS, ISO, so decision for writing vowels and other sounds relies entirely on the users
  - **Long Vowel transliteration:** Users write long vowel signs in different forms. For example, transliterating the word "पानी" to "pani" or "paani" one with a single a and another one with double a's.
  - **Borrowed words' pronunciations:**Due to influence of foreign languages like the Persian language in Hindi while transliterating a foreign word, we rely on the pronunciation. For Example, we can observe both ("आजाद" ajaad) & ("आज़ाद," azad) because some Persian sounds do not have any Hindi counterpart.
  - **Accent and Dialect:**People also write based on their accent and dialect. For example, "शाम" can be transliterated as both "shaam" and "saam".
  - **Non phonetic writing:** There may be variations because of multiple alphabets which may fit the corresponding Hindi character. For Example "मुकाम" could be transliterated as either "mukam" or "muqam", here क can be represented by both "k" and "q", since

they both have same pronunciations.

- **Informal writing pattern:** These variations are due to informal and casual writing on social media platforms.
  - **Elongation:**Users elongate some words to express different expressions or moods. For example, coooool, gooood.
  - **Escaping vowels:** Users sometimes escape vowels to write faster. For Example, "महके" may be transliterated as "mhke."

Back-transliteration models provide an efficient way to clean the noisy social media data and transform it into a normalized form. The study is more critical for the development of different linguistic tools in the low-resource languages, specifically in the Indian context. However, the present state-of-the-art of transliteration for low resource Indian languages is in its initial stages.

1.4 Scope & Plan:

In this work, we are focusing on the problem of "back transliterating the social media text where the words belong to native Indian languages but are written in Latin script."

- **Phase-I:** We focused on back transliterating Hindi text to Devanagari script from Latin script. We have done an extensive survey of different machine transliteration and translation models, starting from the basic architectures and extending to state of the art methods. We have performed experiments on the corpus, which simulates the social media data by taking motivation from the current researches. Finally, we have listed the future works for improving the current results.

- **Phase-II:** We aim (1) To improve the confusion-pair metric by incorporating phonetic knowledge of native script in our models. (2) To propose a generalized transliteration model for social media data valid for most Indian languages. (3) To extend our models to complete sentences instead of using word-to-word transliteration results. Thus the model can use the context of the previous sentence to get better results.

Chapter 2

Literature review

Earlier researches in machine transliteration were based on the phonetics of source and target languages, followed by statistical and language-specific methods.[7] The need for transliteration first arose as a subtask to translation. Transliteration and Translation are closely related, and to some extent, transliteration can be considered as a character-level translation problem[4]. Many of the works on language models are based upon machine translation, so understanding the major ideas behind machine translation and appropriately transferring them to solve the transliteration task seems to be a viable option. In this section, we will first discuss machine translation approaches that have been used for machine transliteration, and then we will look at some of the research specific to machine transliteration.

2.1 Phonetic-based Method

Many early transliteration research applied methods of speech recognition and researched transliteration within a system focused on phonetics. This family of methods is also referred to as pivot or phoneme-based in the lit-

erature.[7] The idea behind the phonetic-based method is that all languages have the same phonetic representation. Phonetic-based methods identify phonemes in the source word S and then map them to character representations in the target language to generate the target words. The words are converted to their phonetic representation using set transformation rules.[7] The main challenge of this approach is that many criteria decide the pronunciation of words in any language. For example, the location of words may decide how they are pronounced.

2.2 Sequence-to-Sequence Models

Seq2Seq models are encoder-decoder based machine translation models that maps an input of sequence to an output of sequence.
Though There are deep neural networks that have achieved a high performance on difficult tasks like voice recognition, computer vision, social network filtering, and machine translation. Despite this excellent performance, DNNs are only applicable to problems whose inputs and targets can be encoded with vectors of fixed dimensionality. Thus it poses a significant limitation for sequential data since input & output is of variable length.

Sutskever et al.[14] has shown a straightforward application of Long Short-Term Memory (LSTM)[6] architecture which can solve general sequence to sequence problems. A useful feature of the LSTM is that it learns to transform a variable length input sentence into a representation of a fixed-dimensional vector. The idea is to use one LSTM to read the sequence of inputs to get a fixed-length vector and then use another LSTM to retrieve the sequence of output from that vector. These encoder-decoder models require the source sentence to be compressed into a fixed vector, which may

give poor results for long sentences.

Bahdanau et al.[2] extended the encoder-decoder model to align and translate jointly. Each time the model generates a word in translation, it searches for a set of encoder outputs which are most relevant for generating the output on the corresponding position in decoder, this is called as attention mechanism. The model then uses the context vector based on these source positions and all previously predicted words to predict the target word. The distinguishing feature of this method from the basic encoder-decoder model is that instead of just using a fixed-length encoded vector, it chooses a subset of encoder outputs adaptively while decoding the translation. The improvement is more evident for longer sentences but can be observed for sentences of any length.

2.3 Fully Character-Level Language Models

The inability of word-level models to represent out-of-vocabulary words makes them unsuitable for low resource languages - also, the complexity of training and decoding increases as the size of the dictionary increases. Character-level models are more beneficial in following manner (1) No out-of-vocabulary issues (2) Rare morphological variants of a word are better represented (3) No need of explicit segmentation[3] (4) Easy to apply in multilingual translation settings (allows parameter sharing between languages with significant standard alphabets) (5) Not forcing model with explicit segmentation encourages it to better learn the internal structure of a sentence.

Lee et al. [11] proposed a fully character-level NMT model to generate a target sequence mapping from a given source sequence. The reduction in training time of a fully character-based encoder is achieved by substan-

tially reducing the source sentence length using a stack of convolution, max-pooling, and highway layers. Max-pooling reduces the source vector representation allowing the model to learn at the speed of sub-word level models while preserving the local features. In the many-to-one translation task, by using universal encoder for all the language pairs, their model outperforms the sub-word level models without increasing the model size. They demonstrated the feasibility of a fully character-based language model in bi-lingual and multi-lingual settings.

2.4 Multilingual Neural Transliteration

Most of the machine transliteration models are focused on bilingual training. There are many phonetically similar languages, so to leverage these similarities, multitask training approaches make sense. The aim is to (1) Jointly train multiple languages to favor superior transliteration results. Parameter sharing across phonetically similar languages can help learn richer patterns. (2) Leverage dataset of other languages for excellent training. Kunchukuttan et al. [10] proposed a multilingual transliteration models for training orthographically similar languages. Two languages are considered orthographically similar if they have: (i) immensely overlapping phoneme sets, (ii) similar grapheme to phoneme mappings, and (iii) mutually compatible orthographic systems.

They proposed an attention-based neural encoder-decoder model. An encoder is based on CNN instead of traditional bidirectional LSTM layer keeping in mind that most of the temporal dependencies are local in the transliteration task. The CNN based model is much faster to train with a slight decrease in accuracy. The decoder is a single layer LSTM network.

Encoder and decoder are shared across the source and target languages, respectively. The output layers(fully connected feedforward layer), which are specific for every target language, are used to generate target words from the decoded output. The model offers maximum parameter sharing across language pairs to leverage orthographic similarity effectively while supplying sufficient space to learn language-specific features.

Quantitative observations. (1) Better transliteration accuracy than bilingual training. Arabic and Slavic are most benefitted. (2) Notable advancement over bilingual systems with phrase-based SMT. This result is consistent with previous works. (3) The better generalization of transliteration task as verified by zero-shot transliteration. Qualitative observations (1) Major decrements in vowel errors (average decreases 20%) (2) Less confusion between consonants having similar phonetic properties (3) Canonical spellings are preferred over alternative spellings with similar phonetics.

Further for languages having one to one character-phonemes correspondence, eg. Indic-English and Indic-Indic, the results on these languages are observed to be further enhanced when phonetic embeddings are given as input to the encoder instead of traditional embeddings generated by one-hot character vector. The phonetic embedding is generated by the product of the phonetic feature vector and weight matrix.

2.5 Compositional Transliteration systems

Compositional transliteration systems refer to systems where multiple components of transliteration are composed to improve the quality of transliteration.[9] There are two forms of composition serial & parallel. By composing the transliteration engine between X&Y and Y&Z, transliteration function-

ality can be created in serial composition between two languages X&Z. Such compositions can be used when no parallel corpora exist between X&Z directly but have parallel data for X&Y and Y&Z. There are 22 recognized languages in India, for instance, but parallel information can be found between Hindi and any foreign language such as Spanish, but not for other Indian languages. In such cases, transliteration from Kannada to Spanish may be achieved by composing transliteration modules from Kannada to Hindi and Hindi to Spanish. This method leverages the existing resources and alleviate the need for developing parallel corpora for many language pairs. In parallel composition, the author explores different transliteration paths between X&Z to get better accuracy. In this approach, the availability of parallel corpus between 3 or more languages is leveraged to get more accurate transliterated results. They have used the Conditional Random Field-based transliteration model for training and further transliterating.

Serial Composition Model: For this model, three languages X, Y, Z, were selected on the basis of Weighted Average Entropy between X&Y and Y&Z, which is a measure of ease of transliteration. Top k outputs were taken from the X->Y transliteration engine and passed to the Y->Z engine, which in turn takes the top 10 outputs based on the probability scores. This method is expected to give poor results since an erroneous result at an intermediate stage will result in incorrect result in the second stage, but on seeing the results and on doing the error analysis it was proved that even the erroneous result at intermediate stage might have enough information to get correct result in output. The dip in accuracy of composition models was very less as compared with the direct X->Z baseline model.

Parallel Composition Model: This model is possible if there is a parallel corpus available between the languages X, Y, Z. First, train an X→Z system,

using the direct parallel names corpora between X & Z, which is called the direct system. Then an X→Y transliteration system, and second a fuzzy transliteration system, Y→Z that is trained using a training set that pairs the top-k outputs of the X→Y system. This is called a fuzzy system. The results from direct and fuzzy systems are combined to get final output for transliteration task of X->Z. This method improves the accuracy of the task.

This paper thus builds the composition models on already built state of the art transliteration models to improve the accuracy of the given transliterated task.

2.6 Normalization of Transliteration words in code-mixed data

The amount of code-mixed data (especially in the Indian context) in social media is increasing at a fast rate. A particular concern is needed to handle code-mixed data because of grammatical inconsistencies and phonetic variations. Although there are standard transliteration tools like ITRANS, ISO, it is unrealistic to use them. People transliterate different words differently based on their phonetic judgment.

Mandal et al.[12] focused on normalizing phonetic typing variation present in code-mixed data. Identical words are transliterated differently by different people based on their phonetic judgment. They proposed a two-step modular approach consisting of two-step normalization. The first module used the Seq2Seq Encoder-Decoder model to convert text into standard transliteration. The model is trained over a (Bn-En) Roman script dataset. Then the output from the first module was looked up in (BN_TRANS) dictionary. The word having the least Levenshtein distance is returned as a final nor-

malized text. Model achieved a 90.27% accuracy on test data. It shows that a high accuracy can be achieved by using dictionary-based methods.

2.7 Conclusion

In this work,we reviewed existing works on machine transliteration. Though, phoneme based or statistical models were used earlier, most of the recent models use neural network based approaches. However, since only limited works were found in social media data, therefore you think there is a lot of scope for investigation and research in this area.

Chapter 3

Experiment Setup

We evaluate the models that were proposed in the above research papers on English-Hindi translation. So far, we have explored some of the sequence-to-sequence models for English-Hindi transliteration. The experimental set-up, datasets used, results, and observations obtained are discussed in this section.

3.1 Dataset

We have used "Datasets for FIRE 2013 Track on Transliterated Search," which has 30,000 English-Hindi Pairs for training. Hindi word transliteration pairs contain annotations collected from different users in multiple setups - chat, dictation, and other scenarios. Our dataset include English-Hindi pairs with different English spellings for the same Hindi word(for example (behke - बहके) & (baheke - बहके)) which simulates the social media data.

3.2 Data preprocessing

We prepend every word with '$' and append with '' to mark the start and end of the word, and used the words where both Hindi-English Pairs are no longer than 25 characters and divided the dataset into 80% for training and 20% for testing.

3.3 Model

We trained five different types of models with around 30,000 English-Hindi Pairs. The Encoder-decoder model has 1500 hidden units in both encoder and decoder. In all models, we have used a Unidirectional decoder. The Encoder-Decoder with attentions mechanism models have the following variations :

1. Unidirectional single layer Encoder
2. Bidirectional single layer Encoder
3. Unidirectional Multi-layer Encoder, Decoder

We have used a mini-batch Stochastic Gradient Descent (SGD) algorithm with Adam-optimizer for training. Each SGD update direction is computed using a mini-batch of 64 pairs. We trained each model for 30 EPOCHS. After the model is trained, we used Beam Search(size = 3)[8][5] for back-transliteration, which approximately maximizes conditional probability of the complete word, rather than maximizing the probability of next character based on earlier predictions.

3.4 Result

Observations			
Model	word accuracy	character accuracy	avg. mismatch
Unidirectional encoder	51.3544201	92.9746489	0.108467
Bidirectional encoder	52.87915	94.1402	0.11
Unidirectional Multi-layer	55.8637	94.7373	0.17462

We got similar observations for all the above listed models.

1. Word accuracy is very less as compared to character level accuracy because even if a single character is wrong, then that word is considered as incorrect.

2. The average mismatch for all the models which is around 0.1. Most of the words have lengths in range of 10-20, so, on average, each word has 2 mismatch characters.

3. Most of the mismatch pairs were the ones which have similar pronunciations for e.g. '(क ख), (ग घ), (द, ड) .'

4. Other error are between the Hindi "maatras" are also known as vowel signs for example '(आ अ), (ी ि), (ू ु) '.

Chapter 4

Conclusion & Future Work

In this work, we have summarized the performance and ideas behind different machine transliteration models along with the challenges in the field. We also presented observations and results recorded by us while implementing various traditional Seq2Seq language models.

We observed that high-frequency mismatch pairs occur for phonetically similar Hindi characters, e.g. (त, ट).In the future, we aim to reduce the number of mismatch pairs. One of the approaches is to pass the predicted word through a filter. The filter will first mark the characters of the output of our transliteration model, which have the high probability of getting mismatched and construct all possible variations of that word by replacing the marked character with their mismatch counterparts. This approach will give us a set of probable output words. We can apply a dictionary lookup for these words to get the most appropriate word.

Another improvement can be made over dictionary-based lookup methods. Dictionary-based lookup is one approach to decide the most suitable word, as suggested in [12]. The proposed word lookup based on the least Levenshtein distance (LD). In the measurement of LD all the character sub-

stitution pairs are given an equal weight of 1. In case of Indian languages like Hindi some characters are phonetically similar for example(if "hamara" is transliterated as "हमरा", will compute Levenshtein Distance with both "हमारा" & "हमला" as 1 but we can clearly see the latter one is not correct). Granting weights based on their phonetic similarity can result in better word matching. The pairs with similar phonetic representation should be given less weights to assist the words with similar phonetic representation.

Currently, we have performed experiments over word pairs corpus. We aim to perform transliteration over sentences using character-level models. The major challenges here are (1) preserving the context or dependency over long-distance words which is initially addressed using attention mechanism [2] (2) Increase in model complexity due to long source sentences can lead to longer training time, one way to reduce the source representation is using convolution layers over source sentence representation [11]. We aim to improve upon the ideas already presented to make them more relevant to Indian languages.

Bibliography

[1] Digital trends 2019: Every single stat you need to know about the internet. `https://thenextweb.com/contributors/2019/01/30/digital-trends-2019-every-single-stat-you-need-to-know-about-the-internet/`. Accessed: 2019-10-30.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*, 2016.

[4] Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney. A deep learning approach to machine transliteration. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 233–241. Association for Computational Linguistics, 2009.

[5] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. Machine transliteration survey. *ACM Comput. Surv.*, 43:17, 04 2011.

[8] Philipp Koehn. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Conference of the Association for Machine Translation in the Americas*, pages 115–124. Springer, 2004.

[9] A Kumaran, Mitesh M Khapra, and Pushpak Bhattacharyya. Compositional machine transliteration. *ACM Transactions on Asian Language Information Processing (TALIP)*, 9(4):13, 2010.

[10] Anoop Kunchukuttan, Mitesh Khapra, Gurneet Singh, and Pushpak Bhattacharyya. Leveraging orthographic similarity for multilingual neural transliteration. *Transactions of the Association for Computational Linguistics*, 6:303–316, 2018.

[11] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017.

[12] Soumil Mandal and Karthick Nanmaran. Normalization of transliterated words in code-mixed data using seq2seq model & levenshtein distance. *arXiv preprint arXiv:1805.08701*, 2018.

[13] Dinesh Kumar Prabhakar and Sukomal Pal. Machine transliteration and transliterated text retrieval: a survey. *Sādhanā*, 43(6):93, 2018.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.