Transliteration as Machine Translation*

Devanshu Jain
University of Pennsylvania
devjain@cis.upenn.edu

Chris Callison-Burch
University of Pennsylvania
ccb@cis.upenn.edu

Abstract

Machine Transliteration is the process phonetic translation of a word across different scripts. This is an important and significant task in the field of Natural Language Processing, particularly because of the value it adds to many downstream applications such as Machine Translation, Entity Discovery, Information Retrieval in the context of multilingualism. This is also a difficult task, from the perspective of machine learning especially due to lack of high-quality training data. The objective of this paper is to report the results of many experiments conducted by the authors to improve the results of this task in a low resource environment.

1 Introduction

Machine Transliteration is the process phonetic translation of a word across different scripts. When a word is translated from its native script to foreign script, then it is called *forward transliteration*. On the other hand, when it is translated from a foreign script back to its native script, then it is called *backward transliteration*.

With growing multilingualism, machine transliteration plays an important role in many downstream applications. One such example is Machine Translation Systems. Almost always, named entities (such as names, addresses, technical terms, festivals, etc.) are transliterated while generating annotated parallel corpora. Sometimes, there are no words in the target language corresponding to a word in the source language. Here, machine transliteration proves to be an important module to improve the performance of machine translation.

Another application is Cross Lingual Information Retrieval (CLIR) Systems. Many of the search engines do not consider transliterated content while responding to a query. It can be observed that the same query (for example, song lyrics) returns significantly different results when submitted to a search engine in a transliterated form.

An interesting human behaviour is also observed on websites such as Facebook, Twitter, etc. which are sources for a lot of user generated content. A lot of these posts/tweets are written in user's native language transliterated to Roman Script. Machine Transliteration would be a useful component for text-based applications such as Question Answering, Sentiment Analysis, etc.

Machine Transliteration presents its own set of challenges. One of them is of multiple target candidates. For example hamare can be written as *humare* or *hamare* or *hamaare* and so on. This leads to the system learning various rules, all of which are correct but may introduce test error. Thus, we need an appropriate performance measure that takes into account the partial correctness of transliteration process. Moreover, there is a lack of high quality training data which hurdles the creation of accurate models.

This study is based on Irvine et.al.'s paper titled "Transliterating From All Languages" (2010). I attempt to recreate some of their experiments with focus on low resource languages to build an end to end machine transliteration system that takes a sequence, its source script and the target script and does the phonetic conversion from the source script to the target script.

The rest of the paper is organised as follows. Section 2 describes some of the related work done in this field. Section 3 describes the tools and data used for this study. Section 4 describes the Baseline System. Section 5 describes the experiments

conducted by the authors. Section 6 presents our results and provides some analysis. We have provided some possible future work in section 7.

2 Related Work

There are mainly two approaches followed for this task:

1. Phoneme-Based
2. Grapheme-Based

In Phoneme-Based approaches, the transliteration process is modelled by converting grapheme based representation in source language to source language's phonetic representation to target language's phonetic representation to grapheme based representation in target language. On the other hand, Grapheme-Based approach omits the phonetic representations altogether and maps the source language grapheme to target language grapheme.

Knight and Graehl (1997) followed a phoneme-based approach for back-transliterating *Japanese* to *English* and learnt a generative-model using Weighted Finite State Machines. They implemented Dijkstra's shortest path and k-shortest path algorithms to extract the best transliterations. Virga and Khudanpur (2003) also followed a phoneme-based approach to transliterate English named-entities to Chinese. They modelled transliteration as a machine translation task and used IBM Noisy Source Channel modelling for the same.

Kumaran and Kellner (2007) followed a grapheme-based approach to transliterate from English to Hindi, Tamil, Japanese, Arabic and vice-versa using IBM Noisy Source Channel modelling. Pingali, Ganesh, Yella, and Varma (2008) also follows a grapheme based approach but they approach it as a sequence labelling task. They also used this technique to build a Hindi-English transliteration system.

Some Oh, Choi, and Isahara (2006) have also tried to use a hybrid approach using both phoneme and grapheme based approaches for transliteration.

In 2009, Named Entity Workshop (NEWS) introduced a shared task on Machine Transliteration. The latest version (sixth) of the task was held In 2016. The best performing team this year was Finch, Liu, Wang, and Sumita (2016). They used ensembles of pure neural-network based system. They overcome the general weakness of a unidirectional sequence-to-sequence transducer: error propagation that leads to an output with high-quality prefixes but low-quality suffixes by deploying a bidirectional sequence-to-sequence model that generate from left-to-right and right-to-left.

Some have also tried to study the transliteration in the context of a low-resource language. Chinnakotla, Damani, and Satoskar (2010) presented a way to build a transliteration system for low-resource languages i.e., for which large parallel corpora isn't available. They made use of monolingual resources and non-probabilistic manually created character mappings for this purpose. Kumaran, Khapra, and Bhattacharyya (2010) explores compositional transliteration i.e. using an intermediate language Y for transliterating from X $\rightarrow$ Y $\rightarrow$ Z. This technique claims to improve the system performance parallel corpora is available for language pairs: (X,Y) and (Y,Z) but not for (X,Z).

3 Base Study

Irvine, Callison-Burch, and Klementiev (2010) paper titled "Transliterating From All Languages" was used as a basis for this study. They treated Machine Transliteration as a monotone character translation task.

3.1 Tool - Joshua

Irvine et al. (2010) used an off the shelf tool called Joshua Li et al. (2009) for their experiments. Joshua is an open source toolkit for Statistical Machine Translation. It provides a scalable, expandable and end-to-end solution for most of the translation tasks. Joshua implements efficient algorithms for Training corpus sub-sampling suffix array grammar extraction, Language Modelling, Decoding Algorithms and parameter tuning. It includes all the components that are required for phrase-based machine translation. To use them, Joshua provides a pipeline perl-script to use these components in different phases: preprocessing the data, creating word alignments, training language models, grammar extraction, tuning, testing and analysis.

Irvine et al. (2010) treated transliteration as a phoneme-based monotone character translation task, following Virga and Khudanpur (2003).

However, they used log-linear formulation instead of noisy-channel formulation. They used Berkeley aligner for aligning characters in the parallel corpora. Since transliteration is a monotone task, i.e. characters don't need reorganisation within the word, they don't extract the grammar rules that involve any hierarchial structure by restricting the number of non-terminals to 0. They used 10-gram language model. All the data was preprocessed by replacing spaces with underscores('_') and including spaces between consecutive characters. Special characters were also ended to denote the start and end of words. They used Joshua's MERT optimisation to tune the parameters for their *"translation"* model by optimising the BLEU score objective function.

3.2 Data

Irvine et al. (2010) used data comprising of named-pairs extracted from Wikipedia articles. Wikipedia maintains pairs of articles written in different languages. They made use of feature that named-entities are mostly transliterated rather than translated from their canonical language to the target language. They mined Wikipedia category pages like: "1961 births" to get a list of people. They made use of such listings and the language links associated with each person to create data consisting of people names in various languages. Data was collected for approximately 200 languages.

Data was further cleaned for use. Some titles were not consistently transliterated. Reasons for this included abbreviations (A.P.J. Abdul Kalam in Roman written as Abul Pakir Jainulabdeen Abdul Kalam in Devanagari). Another reason was that sometimes, middle name was omitted during transliteration i.e. Abbot Suger in Roman was written as Suger in Russian. They computed word-alignments for the data and chose a threshold score for removing the extreme cases of such occurrences.

They also built their own character-based language model for English by tagging and counting named-entities in the English-Gigaword Corpus.

3.3 Evaluation Metrics

Irvine et al. (2010) used Levenshtein distance to evaluate the quality of their transliteration. The distance measures the minimum number of insertions, deletions and substitutions needed to convert the source string to the target string. The distance was normalised by the length of the reference string. The score obtained by averaging the distance over the number of samples in the dataset was used as an evaluation metric for the transliteration system.

3.4 Experiments and Observations

Irvine et al. (2010) experimented with transliterating over 13 languages to English. They performed 10 fold-cross validation using 80% data for training, 10% for tuning and the remaining 10% data for testing. They observed that it was not always true that languages with the largest training corpus gives a better result over the languages with smaller corpus.

They performed a 2^{nd} experiment in which they measured the system performance with progressively increasing training data for each of the 13 systems. Here, it was observed that the performance always increases although the rate of increase slows down as more and more data is added.

Another experiment was performed to measure a value of n to maximise the accuracy (exact transliterations) of n-best lists. They observed that not much performance is achieved beyond 10-best list.

4 Experiments

4.1 Tool

Following Irvine et al. (2010), we have used Joshua toolkit for phoneme-based transliteration treating the task as a monotone character based translation. We used 8-gram character based language model for our study. We pre-process the data just like them, however, we omit the inclusion of special characters at the word boundaries. The underscore itself ('_') should be able to handle the individual words.

4.2 Data

We used the same data extracted by Irvine et al. (2010) using the link given in their paper. In addition, we mined some extra data from Wikipedia to augment it. Wikipedia API provides the service to get backlinks i.e. all the pages that link to a given page. Using this API, we obtain the title of all the pages that link to every a given entity in the original dataset. Using this, we can capture spelling variations for a given named entity. For example, for a page titled "Barack Obama", we ob-

tained pages with titles such as: "Barak Obama", "Barach Obama", "Borrack Obama", "Barack H Obama", "44th President of the United States", etc. Augmenting the original dataset with 1^{st}, 2^{nd} and 3^{rd} options seem correct, 4^{th} seems okay but 5^{th} should not be included. To clean the data, we computed a decent threshold for the edit distances between the original page title and backlinked page title. If the edit distance is less than threshold, then the dataset is augmented with the backlinked page title, otherwise not. The threshold was computed manually by the authors.

Huang (2005) have shown that having the data corresponding to the word's source of origin may boost transliteration performance. They do it by using a statistical clustering technique to group the data into different classes and then using the transliteration system on each of those classes. To get the named-entity source of origin for our data, we first tried to get the entity's nationality by parsing the original Engish wikipedia article and using regular expression to parse B from "A is a B" (Amitabh Bachhan is an Indian actor) type of structure in the first sentence. We also tried to replicate Huang (2005) experiment with our dataset and compare the performance for both the techniques.

For our experiments, we only considered the transliteration from Hindi (Devanagri) to English (Roman). There were around 10,000 samples for this pair in the original dataset. Using backlinks augmentation, the sampe size increased to 100,000.

4.3 Experiment 1

The objective of this experiment was to confirm the hypothesis that with increase in training data, the system performance increases. The Devanagri-Roman data was divided into 3 parts: training, tuning and testing for each such repetition in the following way: For each of the repetition, we held back 60%, 70%, 80% and 90% data for training and used half of the remaining data for tuning and the remaining half for testing.

4.4 Experiment 2

The objective of this experiment was to observe how the system performance changes when we accumulate training data with languages belonging to same script family For example, Hindi, Sanskrit, Marathi, etc. belong to same family. The hypothesis here was that grouping languages with similar script will significantly increase the training data and if the above hypothesis (Experiment 1) is true, then this should increase the system performance. This experiment was created in the same way as above.

4.5 Experiment 3

The objective of this experiment was to observe the changes in system performance when we try compositional transliteration. Given the source language script X and target script Z, we wanted to observe the system performance when an intermediate language is chosen in the process. That is, X $\rightarrow$ Y $\rightarrow$ Z. The hypothesis here was that the error will accumulate over the two conversions and the system performance should decrease drastically. The experiment for each repetition was created as follows: Some percentage of data for X $\rightarrow$ Z was held for testing and the rest of the data along with script Y was used to train 2 different models. We used Arabic as our intermediate language (Y) for this experiment.

4.6 Experiment 4

The objective of this experiment was to observe whether the presence of additional information such as the source of origin for a language can affect the system performance. The hypothesis here was that since geographically-distant languages have a very different structure, the variety of structure in the training data may make it hard for the learning algorithm to learn the correct rules. We used simple regular expression to parse wikipedia articles corresponding to the entities in the training data and obtained their nationalities. Thereafter, we divided the training data into 2 parts: those which were Indian and those which were English and then, trained separate models for them.

5 Results

5.1 Performance Measures

We used the following evaluation metrics to calculate the system performance following Duan et al. (2016). Instead of having a list of reference transliteration in the dataset (which is assumed by Duan et al. (2016)), we only have 1 reference transliteration. The evaluation formulae are simplified according to this. Here, N is the number of datapoints. The transliteration system outputs a list of transliterated candidates of length n_i for each word w_i. c_{ij} is the j^{th}-best transliterated

word for word w_i. r_i is the correct transliteration (reference) for w_i

1. **Top-1 Word Accuracy:** It measures the proportion of data which produces the correct transliteration (exact matching) as the top transliterated candidate with respect to the reference transliteration.

$$accuracy_i = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 & \text{if } r_i = c_{i1}\\ 0 & \text{else}\end{cases}$$

2. **Top-1 F-Score:** The F-score is the harmonic mean of Precision and Recall. Precision and Recall is calculated using Edit Distance between the top transliterated candidate and the reference. Following formulae are calculated:

$$LCS(c,r) = \frac{1}{2}(|c| + |r| - ED(c,r))$$

Here, LCS is Least Common Subsequence and ED is the edit distance. Then,

$$precision_i = \frac{LCS(c_{i,1}, r_i)}{|c_{i,1}|}$$
$$recall_i = \frac{LCS(c_{i,1}, r_i)}{|r_i|}$$

3. **Mean Reciprocal Rank:** It calculates the average of the minimum reciprocal rank of a correct transliteration. This score does not really makes sense for our case since we only have one reference string to compare against. This is because it just turns out to be top-k accuracy measure (if there exists a reference string in the output list).

$$RR_i = \begin{cases}1 & \text{if } \exists k : r_i = c_{i,k}\\ 0 & \text{else}\end{cases}$$

4. **Mean Average Precision:** It calculates the mean of average precision over all the sources words.

$$MAP = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}(\sum_{k=1}^{n_i} num(i,k))$$

Here num(i,k) is the number of correct transliterations in k-best output list with respect to the correct transliteration.

5.2 Results and Discussion

6 Future Work

References

Manoj K. Chinnakotla, Om P. Damani, and Avijit Satoskar. 2010. Transliteration for resource-scarce languages 9(4):14:1–14:30. https://doi.org/10.1145/1838751.1838753.

Xiangyu Duan, Rafael Banchs, Min Zhang, Haizhou Li, and A. Kumaran. 2016. Report of news 2016 machine transliteration shared task. In *Proceedings of the Sixth Named Entity Workshop*. Association for Computational Linguistics, Berlin, Germany, pages 58–72. http://www.aclweb.org/anthology/W16-2709.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2016. Target-bidirectional neural models for machine transliteration. In *Proceedings of the Sixth Named Entity Workshop*. Association for Computational Linguistics, Berlin, Germany, pages 78–82. http://www.aclweb.org/anthology/W16-2711.

Fei Huang. 2005. Cluster-specific named entity transliteration. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 435–442. https://doi.org/10.3115/1220575.1220630.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. 2010. Transliterating from all languages. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, 7-12 July 1997, Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain.*. pages 128–135. http://aclweb.org/anthology/P/P97/P97-1017.pdf.

A. Kumaran and Tobias Kellner. 2007. A generic framework for machine transliteration. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '07, pages 721–722. https://doi.org/10.1145/1277741.1277876.

A. Kumaran, Mitesh M. Khapra, and Pushpak Bhattacharyya. 2010. Compositional machine transliteration 9(4):13:1–13:29. https://doi.org/10.1145/1838751.1838752.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An open

source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Stroudsburg, PA, USA, StatMT '09, pages 135–139. http://dl.acm.org/citation.cfm?id=1626431.1626459.

Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. 2006. A machine transliteration model based on correspondence between graphemes and phonemes. *ACM Trans. Asian Lang. Inf. Process.* 5(3):185–208. https://doi.org/10.1145/1194936.1194938.

Prasad Pingali, Surya Ganesh, Sree Harsha Yella, and Vasudeva Varma. 2008. Statistical transliteration for cross language information retrieval using HMM alignment model and CRF. In *Third International Joint Conference on Natural Language Processing, IJCNLP 2008, Hyderabad, India, January 7-12, 2008*. pages 42–47. http://aclweb.org/anthology/I/I08/I08-6006.pdf.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-language applications. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*. pages 365–366. https://doi.org/10.1145/860435.860503.