Learning Multi Character Alignment Rules and Classification of training data for Transliteration

Dipankar Bose
Dept. of Computer Science and Engg.
Indian Institute of Technology
Kharagpur, West Bengal
India - 721302
dipankarcsiit@gmail.com

Sudeshna Sarkar
Dept. of Computer Science and Engg.
Indian Institute of Technology
Kharagpur, West Bengal
India - 721302
shudeshna@gmail.com

Abstract

We address the issues of transliteration between Indian languages and English, especially for named entities. We use an EM algorithm to learn the alignment between the languages. We find that there are lot of ambiguities in the rules mapping the characters in the source language to the corresponding characters in the target language. Some of these ambiguities can be handled by capturing context by learning multi-character based alignments and use of character n-gram models. We observed that a word in the source script may have actually originated from different languages. Instead of learning one model for the language pair, we propose that one may use multiple models and a classifier to decide which model to use. A contribution of this work is that the models and classifiers are learned in a completely unsupervised manner. Using our system we were able to get quite accurate transliteration models.

1 Introduction

Transliteration is the practice of transcribing a word or text written in one writing system into another writing system which may have a different script (wikipedia [1]). The rules are often quite ambiguous, and they are often related with the pronunciation of the word.

Many applications like Machine Translation (MT), Cross Language Information Retrieval (CLIR), Question Answering (QA) require transliteration of named entities, which are the major component of out-of-vocabulary (OOV) words, and they are most often transliterated and not translated, in any cross language system. For example ,'Europe' is transliterated as 'iuropa' and 'Michael' transliterates to 'maaikela' in Bengali.[2]

In this paper we develop a scheme of transliteration, which captures context by creating a dictionary of multi-character transliteration rules. We have tested our system for English and several Indian languages. For Indian Languages, we have an additional preprocessor which enhances the performance.

2 Related Work

Brown et al. (1993) have come up with their revolutionary IBM alignment models, and the Giza++ (Och and Ney, 2000) is a well appreciated implementation which work with parallel data in two languages. Though originally designed for machine translation, the package can as well be used for transliteration, where the alignment is between the characters in the languages. Moses further enhances the accuracy by using phrase based decoding, which can capture context. We have Moses[3] as our baseline system.

Li et al. (2004) have pointed out the problems of using language information. Apart from the difficulty of collecting the language information, they pointed out that, although written in the same script, the origin of the source names may vary widely. For example French and English names may vary a lot. But it is difficult to collect information for each and every language. They came up with a joint source chan-

[1] http://www.wikipedia.org

[2] above Bengali words are scripted using ITrans, instead of traditional Bengali script.

[3] http://www.statmt.org/moses/

Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, pages 61–64,
Suntec, Singapore, 7 August 2009.

nel model, to transliterate foreign names to Chinese, Korean, and Japanese, which uses, direct orthographic mapping (DOM), between two different languages, to find out how the source and target words can be generated simultaneously. Ekbal et al. (2006) also used this model for English-Bengali Transliteration. Ganesh et al. (2008) used Hidden Markov Model (HMM) alignment and Conditional Random Field (CRF), a discriminative model together. Surana et al. (2008) used fuzzy string matching algorithms to identify the origin of the source word, and then apply rules of transliteration accordingly. However the classifier makes use of labeled training data, which is often not available.

3 Issues

Transliteration is ambiguous. Firstly, the transliteration rules depend on the context. For example, 'a' in English may transliterate to 'a' or 'A' in Hindi, but 'aa' almost definitely maps to 'A'. Secondly, there can be multiple transliterations of the same source word. For example 'abhijIta' may transliterate to 'abhijit' and 'abhijeet' as well. Thirdly, the transliteration rules also vary, depending on the origin of the word. For example, when considering Hindi to English transliteration the English characters used vary depending on whether the word originated from Arabic or from Sanskrit. We elaborate more on this in the section on classification of corpus.

4 Approach

Our method is primarily based on IBM models used in machine translation based on the EM algorithm. But before we move on to the IBM models, we first preprocess the training data. Other than marking the 'Start' and 'End', for each of the parallel words, we can do further preprocessing if any of the scripts is Indian. All Indian language scripts consist of a set of consonants and vowels. Independent vowels and their corresponding diacritic markers (Matra) are considered as the same character in the standard analysis of words into their constituent characters (varna vishleshhana). Unlike ITrans, Unicode assigns different codes to them. We found in our experiment that treating them as one, improves the accuracy of the system. Our preprocessor thus transforms Unicode data to ITrans format. We have seen that preprocessor improves the accuracy by around 10-15%.

After preprocessing, we align the letters using the expectation maximization (EM) algorithm of IBM model 1, using the parallel corpus of named entities as input. We use only the IBM model 1; the subsequent models are omitted since in transliteration we need not consider the reordering of letters. Both Unicode and transliterated text are in phonetic order, and re-ordering of letters are rarely observed. As an output of the EM learner we get a table of translation probabilities TP, of source letters to target letters. If, s_i and t_j are source and target letters, $\forall s_i, t_j, TP_{s_i,t_j} \in [0,1]$, denotes the corresponding translation probability. For example after EM learning, the values of $TP_{bha,v}$ and $TP_{bha,b}$ will be much more than $TP_{bha,k}$, since 'bha' rarely transliterates to 'k'.

4.1 Learning Phrase Mappings

We now move on to capture context. For each word in the parallel data, we compute an alignment array, A_e, where $e \in [0, E]$, and I and E are the corresponding lengths of the words in Indian and English script respectively. So, we have, $\forall e \in [0, E], A_e \in [0, I]$. Following is an example: Let, source word be: Start s_1 s_2 s_3 End, target word be: Start t_1 t_2 t_3 t_4 End, and Alignment array be: 0 1 1 2 3 4. This means that s_1 maps to t_1 and t_2; s_2 maps to t_3 and so on. We further enforce $A_{e_1} \leq A_{e_2}$ *iff* $e_1 \leq e_2$, since we neglect re-ordering of letters. The aim is to figure out null mappings, filter out noises in the TP-table, and finally create a phrase to phrase mapped dictionary. Using the TP-table values, we propose an iterative algorithm to find the alignment array A. $W_L[i]$ denotes the i^{th} letter of a word in language 'L'. Initially $A_i = 0$ if $i = 0$, $A_i = I - 1$ if $i < E$, otherwise $A_i = I$. The first and last characters are always the 'Start' and 'End' tags, in all the words.

Initially letters are allowed a larger window to fit to. After each iteration, the window size decreases and thus the margins are made more stringent. Using iterations we are being less greedy in deciding the alignment, so that noises in the TP-table are filtered out. Finally after 5 iterations, we freeze the alignment array. It may happen that $\exists i \in [0, I]$, such that $\forall j \in [0, E], A_j \neq i$. It means that the letter, $W_{Ind}[i]$ maps to 'null' in this case, and thus it is a 'Schwa' character.

4.2 Scoring the alignment

In spite of all our attempts, it may happen that the words are not well aligned; the reason may be a

Algorithm 1 Method to compute Alignment

for $window$ = 5 to 1 **do**
  for e = 1 to $E-1$ **do**
    $left = Max(1, A_{e-1} - window + 1)$
    $right = Min(I, A_{e+1} + window)$
    $A_e = s : s \in [left, right]$ such that $TP_{W_{Ind}[s],W_{Eng}[e]} \times (1 - |s/I - e/E|)$ is maximum
  end for
  for e = 1 to $E-1$ **do**
    if $\neg(A_{e-1} \leq A_e \leq A_{e+1})$ **then**
      {try to smooth out anomalies}
      $A_e = (A_{e-1} + A_{e+1})/2$
    end if
  end for
end for

deficiency in the Algorithm 1, or a badly transliterated parallel word as input. For example the training data may contain 'mississippi river' transliterated to Bengali as 'misisipi nadI'. In this case we see that the second word is translated and not transliterated. Retaining this in the training set will introduce noise in the model. There may also be typographical errors also. We have developed a filtering mechanism, so that we can eliminate these words, otherwise we will end up learning spurious mappings. We find the score of an alignment,
$SA = \sum_{e=1}^{N-1}(TP_{W_{Ind}[A_e],W_{Eng}[e]} \times (1 - |A_e/I - e/E|)$.
We were trying to maximize SA under certain constraints in algorithm 1. The value of SA is an estimate of how good our alignment is. Next we set thresholds to distinguish between different "Classes" of alignments.

4.3 Classifying the training corpus

The training corpus may consist of words from varied origins. Though they are written in the same script, pronunciation varies widely. For example Urdu origin names like Farooque (pharUka), Razzaq (rajjAka) tend to replace 'q' in place of 'ka', but Hindi names like Latika (latika), Nakul (nakula), tend to replace 'k' for 'ka'. Unlike Surana et al. (2008) who extracted 5-gram models from labeled data in different languages, we propose Algorithm 2, to classify the parallel corpus into groups, which does not need any labeled data. We define, Classes C_1, C_2, ..., C_N, where C_i consists of a set of parallel words $< I_j, E_j >$, (I_j, E_j being the j^{th} word in Indian and English language, in the training corpus), such that the alignment score of the word pairs, lie between the predefined thresholds, th_{i+1} and th_i. Let us assume that C_1 is initialized with the parallel training corpus from input.

Algorithm 2 Classify the Corpus

for i = 1 to N **do**
  Set threshold, th_i for Class C_i: $th_i \leq th_{i-1}$
  while size of Class C_i does not decrease **do**
    Compute TP-table using IBM model 1. on C_i
    for each parallel word pair $< I_j, E_j >$ in C_i **do**
      Compute Alignment using Algorithm 1.
      Compute Score of Alignment, SA.
      if $Score < th_i$ **then**
        {Move the word pair to the next class}
        $C_{i+1} = C_{i+1} \cup < I_j, E_j >$
        $C_i = C_i \setminus < I_j, E_j >$
      end if
    end for
  end while{move on to next $Class$}
end for

We continuously discard word pairs from a class until there is no word pair to be discarded. We use IBM Model 1 to re-learn the TP-table, on the latest content of the class. Since the poor word pairs have been removed, learning the TP-table afresh, helps in improving the TP_{s_i,t_j} values. It helps in removing the bad word pairs yet left, in the subsequent iterations. It is to be noted that C_N consists of word pairs, which are of no use, and we discard them completely. We had 5 useful classes, and the thresholds of C_1 to C_5 were 0.4, 0.35, 0.3, 0.25, 0.2 respectively. In each class, for each word pair, we extract all possible ngrams on Indian language side and collect their corresponding English characters, using the alignment array. We keep frequency counts of these ngram mappings, and use this score in decoding. We use a language model, which uses Good Turing smoothing technique. We have used greedy beam search based decoder.

All that remains is to guess the class of an unknown word. Given a test word, in source script we calculate probability P_i of it being in class, C_i, based on ngram similarities. The decoders of each of the classes returns a list of feasible transliteration candidates along with their 'local scores'

Language	Accuracy in Top1	Mean F-Score	MRR	MAP_{ref}	MAP_{10}	MAP_{sys}
En2Ta	0.404	0.883	0.539	0.398	0.182	0.182
En2Hi	0.366	0.854	0.493	0.360	0.164	0.164
En2Ka	0.335	0.856	0.457	0.328	0.154	0.154

Table 1: Transliteration Accuracies. En2Ta: English to Tamil, En2Hi: English to Hindi, En2Ka: English to Kannada

(score according to that class), We denote the local score of a candidate from Class C_i as $LS[C_i]$. We calculate the global score, GS for each candidate, using GS= $\sum_{i=1}^{N-1}(LS[C_i] \times P_i)$. The candidates are sorted in decreasing order of their global scores and top 'K' of them produced as output.

5 Results

We have evaluated our system, against datasets with Hindi, Tamil, Kannada and English parallel named entities (Kumaran and Kellner, 2007). The results are in Table 1. The data consists of named entities from varied origins: almost all Indian languages and English. We combined the training and development sets to create the new training set. There are about 9000 parallel words in the training sets and 1000 words for testing.

Algorithm 2 classifies the training corpus, into 5 sets of corpus. Following are some details after classifying the Tamil-English dataset. Corpus 1, consists of Sanskrit derived words mostly; they get perfectly aligned and Schwa deletions rarely occur; Ex: Keena, Asiya, Nehra, Hemaraaj, Vijendra. This corpus contains 2167 words. Corpus 2 also is mostly comprised of Sanskrit derived words and also English words which easily align; like Wilton, Natesh, Raghu, Gerry, Achintya, Amaanat. Schwa deletions does occur, and hence the alignment scores are a little low. Size of this corpus is 2168.

Corpus 3 consists more of Urdu origin and English words, which are not fit for the normal transliteration rules. The corpus consists of words like Tarzan, Anoife, Sevier, Zahid Fazal, Floriane, where letters like 'q', 'zz', 'y' are more likely than 'k', 'j', 'i' respectively. The size of Corpus 3 is 1835. Corpus 4 & 5 consists largely of English origin words, like Lucky number, Ian Healy, Cleavant, Fort Vancouver, Virginia Reel, Bundesverdienstkreuz. These words need completely different set of rules, and moreover if these words were in any other class, it would corrupt their learning rules. Size of these corpora are 1234 and 1455 respectively.

6 Conclusion

Our system is robust in the sense that it can filter out noise in the training corpus, can handle words of different origins by classifying them into different classes. Our classifying algorithm improves the accuracy, but we believe that there is scope of further improvement and we are working on it.

References

Asif Ekbal, Sudip Kumar Naskar, Sivaji Bandyopadhyay. 2006. *A modified joint source-channel model for transliteration.* Proceedings of the COLING/ACL on Main conference poster sessions.Sydney, Australia.

Harshit Surana and A. K. Singh 2008. *A More Discerning and Adaptable Multilingual Transliteration Mechanism for Indian Languages.* The Third International Joint Conference on Natural Language Processing (IJCNLP). Hyderabad, India.

Kumaran A. and Kellner Tobias. 2007. A generic framework for machine transliteration *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 721–722.

Li Haizhou, Zhang Min, Su Jian. 2004. *A joint source-channel model for machine transliteration*. Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. Barcelona, Spain.

Och Franz Josef and Hermann Ney. 2000. *Improved Statistical Alignment Models.* Proc. of the 38th Annual Meeting of the Association for Computational Linguistics, pp. 440-447, Hong Kong, China.

Peter F. Brown, Vincent J. Delta Pietra, Stephen A. Delta Pietra and Robert L. Mercer. 1993. *The mathematics of statistical machine translation: parameter estimation.* MIT Press Cambridge, MA, USA.

Surya Ganesh, Sree Harsha, Prasad Pingali, Vasudeva Verma. 2008. *Statistical Transliteration for Cross Language Information Retrieval using HMM alignment model and CRF.* CLIA-2008, 2nd International workshop on Cross Language Information Access, 3rd International Joint Conference on Natural Language Processing (IJCNLP 2008), January 7-12, 2008, Hyderabad, India.