ON THE UTILITY OF A SYLLABLE-LIKE SEGMENTATION FOR LEARNING A TRANSLITERATION FROM ENGLISH TO AN INDIC LANGUAGE

Sowmya V. Balakrishna[1], Shankar M. Venkatesan[2]

[1] University of Tuebingen, Tuebingen, Germany [2]Philips Research India, ManyataTech Park, Nagavara, Bangalore 560045, India
svajjala@sfs.uni-tuebingen.de, Shankar.Venkatesan@Philips.com

ABSTRACT

Source and target word segmentation and alignment is a primary step in the statistical learning of a Transliteration. Here, we analyze the benefit of a syllable-like segmentation approach for learning a transliteration from English to an Indic language, which aligns the training set word pairs in terms of sub-syllable-like units instead of individual character units. While this has been found useful in the case of dealing with Out-of-vocabulary words in English-Chinese in the presence of multiple target dialects, we asked if this would be true for Indic languages which are simpler in their phonetic representation and pronunciation. We expected this syllable-like method to perform marginally better, but we found instead that even though our proposed approach improved the Top-1 accuracy, the individual-character-unit alignment model somewhat outperformed our approach when the Top-10 results of the system were re-ranked using language modeling approaches. Our experiments were conducted for English to Telugu transliteration (our method will apply equally well to most written Indic languages); our training consisted of a syllable-like segmentation and alignment of a large training set, on which we built a statistical model by modifying a previous character-level maximum entropy based Transliteration learning system due to Kumaran and Kellner; our testing consisted of using the same segmentation of a test English word, followed by applying the model, and re-ranking the resulting top 10 Telugu words. We also report the dataset creation and selection since standard datasets are not available.

1. INTRODUCTION

Transliteration finds application in various Natural Language Processing applications like Machine Translation, Mono lingual as well as Cross lingual information retrieval. It has been used for various scenarios like named entity recognition [5], [19], [13], cross lingual spelling variation detection [15], [4], text input [14], [23], etc.

There are different approaches to transliteration. The more obvious one (rule-based) for source and target languages that have a compact alphabet and fairly simple pronunciation is a manually created set of Transliteration rules between a finite set of 'segments' of the source and target

David C. Wyld (Eds) : ICCSEA, SPPR, CSIA, WimoA, SCAI - 2013
pp. 421–429, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3544

languages. This expert defined segmentation can be based on simple characters in the two alphabets or natural phonetic segments (based on CV, CVC model) - the mapping rules however have to take into account the context (the prior few and the next few segments) of a source word segment (due to pronunciation context, etc.), making the task of defining the rules quite difficult (in fact, this rule-based approach is followed in many English-to-Indic Transliterators available freely today); The rule-set should be such that every possibility can be properly captured in the rules, making the process cumbersome, and language and expert dependent. The other option is to learn the rules from carefully constructed training sets (which is perhaps an easier task) using machine learning approaches- these approaches aim to automate the process of discovering the rule-set from the training data and if the training set has sufficient generalization and if the approach is robust, then this approach can succeed and hope to even better an expert created set of rules: clearly the method must be good at discovering and weighting the multiple and perhaps context-sensitive mappings between source and target segment sets. Where experts at rule-set creation are lacking, the statistical approach may be the only recourse to coming up with a good transliteration, making this language agnostic approach easy to use by non-experts. Especially in a domain like web search where the search terms are used by even native speakers with multiple written variations for the same phonetic intent, it goes without saying that all relevant variations of the source word must be discovered, matched, and ranked based on various criteria including contextual and statistical ones, making the statistical discovery approach the more viable one. In this paper, we take this machine learnt approach (using the maximum entropy method), with the segmentation and alignment during training based on a natural CV, CVC model, and we use the same segmentation during testing.

In contrast to the syllabic method that we follow here, learning the alignment at the individual character or grapheme level purely from data has been studied before [9], and while it does not have the highest accuracy, it is robust, language agnostic, and does not require rule experts. On the other hand, learning the alignment at our phonetic segment level has also been studied for English-Chinese [2]. Our basic question about this syllabic method was "does it help Indic languages?" This question becomes more interesting because Indic languages have a "alpha-syllabary", where vowel inflections of a basic consonant sound in the source word (e.g. the English segments 'ka' and 'ki') are represented as vowel diacritic symbol next to the basic consonant symbol. This is easily mapped into a target Unicode sequence which has the Unicode element for the 'k' sound preceded or succeeded by the Unicode element for the appropriate vowel diacritic (it helps to think of vowels in Latin and generally western scripts as performing the same diacritic symbol role, but in a split way and reconstructed the same way cognitively - that is, the 'ka' would be rendered as two graphemes in English but as a single grapheme in the target Indian language). The question for an Indic language then becomes one of whether learning this mapping at our larger syllable-like segment level would be able to beat the more atomic grapheme level mapping (note that context - that is, previous i and next j segment windows - plays a role in learning the mapping both in [9] and ours, regardless of the difference in training pairs). Where the languages are not as simple and do not have alpha-syllabaries (e.g. the dialect scenario encountered in the prior English-Chinese work [2]), the authors have to use more involved phonetic segmentation to agglomerate the source symbols. This summarizes what is easy about statistically learning a English->Indic mapping. The harder perhaps counter-intuitive part is our main result of this paper: shifting up the learning from grapheme level to a subsyllable-like segment level does not appear to provide significant additional benefit for Indian languages, at least for similar training set sizes. Note that our approach is not based on phoneme or pronunciation models, but capture the net effect of these in a rank ordered text-to-text mapping based on learning from an appropriately segmented training set.

By way of published history of the methods investigated in this area, we note the following. [8] experimented with a Finite State Transducer based model followed by the EM algorithm. [1], [3] used the alignment model used in statistical machine translation systems. [17] proposed a transliteration method without any parallel resources. Other approaches to transliteration include- using bilingual corpora, web-mining and comparing terms of different languages in a phonetic space. [21] compared and evaluated transliteration alignment and its impact on transliteration efficiency. To see in context, our work considers a source-target language alignment for an Indian language by considering the syllable-like segments instead of individual characters. Segmenting word pairs as 'syllabic sequences' instead of characters was mentioned in [10] and [22] using rule based approaches. [18], [16] used monotone search based models and bi-directional character alignment respectively for a substring based transliteration system, which is a form of segmentation. [7] used dynamic programming to learn the segmentation process. There is also another route followed in [24] which solves the easier problem of transliteration as a special case of machine translation by using phrase-based statistical machine translation techniques. This is in effect the closest to our approach, but our easier method is specifically tailored for Indic languages.

Due to lack of standardized data sets for Transliteration into Indic languages, our experiments were done on an artificially created and partially validated dataset from a monolingual word-list, by generating multiple possible Romanized spellings for every given Telugu word in a word-list (this portion of the paper can be thought of as engineering to create a dataset, and not in the main flow). We believe that the results would be the same if we had standardized data sets. Also we note that we did not use any post-processing based on a speller for better ranking but used bigram-trigram weights to rank order, and we believe that this did not have significant effect on our results. Also since our method is extensible to most Indic and also other language groups, we believe similar results would apply there.

2. LEARNING A TRANSLITERATION FROM SYLLABLE-LIKE ALIGNMENT

Say we are given a sufficiently large set of training pairs (s,t) where s and t are from the source language S and target language T. The problem is to learn a general transliteration mapping from the S to T such that given an unseen word s from S, we can construct a reasonable transliteration t from T. The basic idea is to learn this by enumerating and estimating the mapping by segmenting each two-word pair <s,t> in the training set into disjoint concatenated segments < s=(s_1 ,s_2,…, s_k), t=(t_1,t_2,…, t_l)>, such that k==l, and each s_i aligning with t_i for all1 <= i < k. Instead of a brute force segmentation which is intractable, we can use a character level segmentation [9] or choose a segmentation of the source and target words in some consistent and natural way – one clear choice is a fixed syllable and sub-syllable based segmentation grounded in phonology and phonotactics. This is the method we adopt. The question now is whether this approach can learn the Transliteration more accurately than the character level approach, with the same or similar training set. Intuitively the answer seems to be yes, the reason why we started this work, but however, our results seem to indicate otherwise.

Here, we learn the mapping with the maximum entropy method used in [9], which is this: given a sequence pair < s=(s_1 ,s_2,…, s_p), t=(t_1,t_2,…, t_q)>, and the next segments < s_{p+1} , t_{q+1} >, the joint probability of this occurring is determined by how many times this event occurred in the training set (subject to window length constraints): if it did not occur at all, then we assume that all such

events have equal likelihood, which is the assumption that maximizes the entropy of the state (hence the name). So the algorithm that does this will simply keep track of the probabilities in a table as training proceeds: during test time, the input sequence is parsed from left to right (subject to window constraints) and the probability sequence unwound from the table - when we reach the right end, we would have computed a probability of every target sequence thus encountered- by keeping only the larger probability target sequences (a process we term as beam forming where the beam represents the higher probability expansions from left to right), we can keep the complexity under check while not losing the top candidate target sequences. By looking at transliteration units (TU) that are k windows to the left and right of the current TU, when looking for a proper alignment, we maintain a context (note that the context is learnt not enforced) and also build the probability beam.

We borrow from the Speech domain the syllable or sub-syllable as the unit of segmentation: the simple (V, VC, CV, CVC) model of syllable-formation comes to mind, but defining the syllable boundary consistently for a single language as well as from source to target is needed to make training in this context possible. We adopt a simple model (explained in Section 3) to segment the words of both languages during Training. This model is designed to enforce phonetic consistency (of the source and the target segmentations) by appealing to vowel sequences as markers (we have also tried a consistent syllable/sub-syllable segmentation but the results were not significantly different and training sets need to be larger in this case to learn the larger and more varied segments in addition to larger training times). The Indic language of our choice during training was Telugu - however, the methods would work identically for other Indic languages as well, and for many other languages as well.

3. OUR APPROACH TO WORD SEGMENTATION AND ALIGNMENT

For details on syllabic and phonetic structure of Indic languages and scripts, we refer the reader to [12]. We explored a couple of approaches to word segmentation during both Training and Testing, but here we focus on the details of only one. One can construct other methods based on the (V, VC, CV, CVC) model also (we in fact did this), but we believe that the specific method used here does not have a significant bearing on the conclusions we reach.

3.1 Word Segmentation during Training Phase

The English word is first broken in to segments in a multi stage process:

1. Aligning the English-Indic pair character by character
2. Segmentation of the Indic word
3. Using the segmented Indic word and the character alignment to generate Indic script based segmentation of the English word.

Step 1: Character by Character Alignment of the English-Indic pair:

The English characters are aligned to Indic character sequences, by making use of a mapping between English and Indian alphabets to decide on the alignment. The English character alignment array is of the length equal to the English string and its elements can take values of (0,1,2) depending on how many Indic characters does the English character map to. The process of character alignment proceeds as follows:

1. Iterate through the English word character by character
2. At each character, check the respective Indic character. If the English character can map to the current and next Indic characters (which can be understood from the mapping table) assign 2 to the corresponding alignment array element and increment the Indic character index. Else, If the English character can map to the current Indic character, assign 1 to the corresponding alignment array element. Else, if the English character does not map to the current Indic character at all, assign 0 to the corresponding alignment array element

This will give our English character alignment array by the end of iterations through the English string. A sample mapping for English-Telugu alignment ("#" indicates null mapping) is in the Figure below:

English Letter	Telugu Mappings
a	అ,ఆ,#,క ా,క ి
c	చ,చ్,క,క్,ఛ్,చ్
g	గ, గ్, ఘ, ఘ్
h	హ,హ్,#

Step 2: **Generating Indic Segment Array:**

Here, each half vowel that combines with a consonant (preceding it in the character sequence) is treated as a separate unit. Therefore, the word **Australia** ఆస్ట్రేలియా splits into ఆ-స్-ట్-రే-లి-యా (in Telugu Script). While, splitting by character, the word has 11 characters- ఆ, స, ్, ట, ్, ర, ే, ల, ి, య, ా.

The Indic segmentation array is an array of length equal to the number of segments of the Indic string (6 in this case). Each element of this array indicates the number of characters in that segment. Hence, the Indic segment array here is (1 2 2 2 2 2). Just like diphthongs in English, certain Indic languages (like Hindi) allow juxtaposition of two full vowels, and our method can be suitably modified to take care of this.

Step 3: **Generating English Segment Array:**

The English segment array is generated by combining the English character alignment array generated from Section \ref{step1} and Indic segmentation array generated previously as follows: For the example considered, English character Array: (1 0 2 2 1 1 1 1 2) and the Indic segmentation Array (1 2 2 2 2 2).

The English segments Array is equal in length to the Indic segments Array and each element in it will be equal to the number of English character array elements that should be combined to form an Indic segmentation array element. For example in the word-pair, (Australia, ఆస్ట్రేలియా), the English segmentation becomes (au-s-t-ra-li-a). Note that this is based on simple sequential addition of the English character array to match the number in the Indic segment array, and iterating - the number of iterations determines the number of English segments generated. For example, the English segment array here is (2 1 1 2 2 1), where the first 2 represents the number

of elements from the English character array to sum up to the first number (that is, 1) in the Indic segment array. We have verified the phonetic consistency of this scheme arising from positioning on the vowel markers. Some sample alignments are shown here, including non-Indic origin words

and source words with silent characters (might–మైట్) : (mi–gh–t – మై-∗-ట్) (write – రైట్) : (w–ri–te – ∗-రై-ట్) (jefferson – జెఫర్సన్) : (je–f–fe–r–so–n – జె-∗-ఫ-ర్-స-న్) .

3.2 Word Segmentation during Testing Phase

During testing phase, the absence of a word pair makes the segmentation process slightly tricky, since the Segmentation of English word is based on that of a putative Indic word. This is done in two steps:

1. An approximate Unicode (Indic word) representation of the Roman word is obtained
2. Word segmentation algorithm explained in previous section is applied on this word pair English word-Its approximate Unicode representation.

Obtaining an approximate Unicode representation of the Roman word is done by maintaining an approximate one-to-one mapping between the English alphabet and Indic alphabet. This need not cover all the Indian alphabet, since it is only an intermediate stage. For example, a sample mapping will be similar to that of the alignment in the figure in Step 1 of 3.1: a one-one mapping between consonants and one-to-two mapping for vowels to enable choosing one of them depending on their presence after consonants or independent existence. Thus, for (Australia), the Telugu equivalent in this mapping will be ఆస్ట్రేలియ. Now, from the previous section: English alignment Array: (1 0 2 2 1 1 1 1 1) (Refer: Step 1 of previous section), and the Indic segment Array: (1 2 2 2 2 1) (Refer: Step 2 of previous section). Hence, the English segments array: (2 1 1 2 2 1) is (au-s-t-ra-li-a). Thus the segmentation is performed on English words during the Testing phase, in the absence of parallel word pairs. Note the slight difference in the array elements between training and testing phase, which does not lead to any loss in accuracy. Again, we have manually verified the segmentation of innumerable test words to ensure the results of this paper are not affected by errors in this section.

3.3 Learning a Good Segment Mapping

We trained a segmentation based transliterator using a language-agnostic learner which works as follows:

1. Take the training pair and segment and align as mentioned in the previous section. (The datasets are explained in Section 4).
2. Run the Maximum Entropy method of [9], suitably modified to learn on segments instead of characters. This gives us the mapping probabilities of (English -> Indic) segments learned from the context of the Training set
3. On a test input, we produce a beam as in [9], with the additional step of further re-ranking using language bigrams and trigrams. The weights assigned to the bigrams and trigrams are chosen on a trial and error basis. A weighted sum of (3*bigram + 2*trigram) was found to give optimal performance.

4. DATA SETS : TRAINING AND TESTING

As explained before, due to lack of standardized training sets for Indic languages, the dataset for training was generated automatically while the test set was obtained from a series of user experiments

Training Set Collection: The parallel data has been automatically generated applying a set of transliteration rules (as explained in Section 3.1), created heuristically, on the monolingual word list, obtained from a search engine index. These rules mapped each alphabet from the Indian language word to possible Roman equivalents. We have obtained 200K word pairs through this process. However, a subset of this collection was chosen for training purpose.

Optimal Training Set Collection: We applied a greedy algorithm based approach followed in speech corpus creation ([20] and [11]), to select optimal training data. Thus, a dataset of around 80K words were chosen to train the transliterator.

Test Data: The test transliteration data was obtained through a series of data collection experiments with live users and it consisted of a general conversational text. A data of 20K word pairs was collected through these experiments, for the language pair - English-Telugu.

5. EXPERIMENTS AND RESULTS

We evaluated the segmentation as well as character alignment based transliteration, using the metrics defined by [6]. The table figure here shows the results on the optimal training set, of both the character alignment and segmentation approaches with different feature sets. Here, ACC-i stands for accuracy at position i or less, and MRR stands for mean reciprocal rank. We use a three dimensional feature vector (a,b,c), where {a} and {b} refers to window of source units before and after the current transliteration unit, {c} refers to the window of the target units before the current transliteration unit. This models the local contextual component of the learner, and long range contextual dependence was not modeled. Perhaps as a result of this, the learner converged quite rapidly indicating that the local context was being identified quickly in the learning process.

Character level alignment

Alignment Without Reranking			
Feature Set->	(330)	(331)	(332)
ACC	0.239	0.228	0.197
ACC-2	0.346	0.329	0.281
ACC-5	0.47	0.449	0.386
Mean F-score	0.819	0.810	0.799
MRR	0.337	0.322	0.2775
Alignment with Reranking			
Feature Set->	(330)	(331)	(332)
ACC	0.347	0.275	0.272
ACC-2	0.447	0.395	0.366
ACC-5	0.531	0.499	0.442
Mean F-Score	0.827	0.777	0.800
MRR	0.423	0.368	0.343

Syllable level alignment

Segmentation without Reranking		
Feature Set->	(111)	(110)
ACC	0.2546	0.276
ACC-2	0.358	0.367
ACC-5	0.446	0.456
Mean F-score	0.799	0.804
MRR	0.337	0.352
Segmentation with Reranking		
Feature Set->	(111)	(110)
ACC	0.329	0.302
ACC-2	0.408	0.398
ACC-5	0.476	0.478
Mean F-score	0.779	0.763
MRR	0.391	0.376

It can be observed that our approach improved the Top-1 transliteration accuracy. However we can also see that the original character alignment model outperformed our subsyllable-like model when the Top-10 results of the system are re-ranked using simple and common techniques,

including bigram frequencies (these experiments were repeated on the full training dataset of 200K, with similar numbers, which proved our intuition of selecting the optimum training set to be correct). A possible explanation of the above results: the Indic phonetic/grapheme/unicode matching and unicode concatenation form a natural training "window" making the previous simple unicode based character level alignment method at least as powerful compared to our syllable based method.

6. CONCLUSIONS AND FUTURE DIRECTIONS

Based on the methods here and the results from our experimentation, we did not see any gains from the syllable-like approach of the transliteration. We think that the conclusions would be the same regardless of the learning method used. We think that if we repeated our experiments on other Indic languages, the findings would not change, and would still favor the atomic letter level alignment, in spite of the seeming attractiveness of syllable-like segmentation and alignment. We suspect that the syllable-like method is not learning any increased contextual discrimination between a compound symbol (e.g. "ta") as opposed to their broken-down components (e.g. "t" + "a"), either because that increased contextual discrimination does not exist in the training set (or in the language meaning that the alternate phonetic contextual sub-sequences are equally likely, yielding maximum entropy anyway) or the method is being overwhelmed by the much larger set of compound symbols. While this is not the last word on the utility of our syllable-like transliteration method, it looks like it is hard to beat the learning of a traditional letter-level or atomic-level mapping (by the way, this method can also learn compound symbol combinations when needed by varying window sizes) and its most significant advantage is that it is mostly language agnostic.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Kumaran for extensive help and suggestions and Rahul Maddimsetty for collaborating on rule-based approaches

REFERENCES

[1] Abdul Jaleel and Larkey: Statistical Transliteration for English Arabic Cross-Language Information Retrieval, Proceedings of CIKM, 2003, p. 139-146
[2] W. Gao, K Wong, and W. Lam: Phoneme-based Transliteration of Foreign Names for OOV Problem, Proceedings of the IJCNLP, 2004
[3] P. Virga, and S. Khudanpur: Transliteration of Proper Names in Cross Lingual Information Retrieval, Proceedings of the ACL Workshop on Multi-lingual Named Entity Recognition, 2003
[4] Ari Pirkola, Jarmo Toivonen, Heikki Keskustalo, Kari Visala, and Kalervo Järvelin: Fuzzy Translation of Cross-Lingual Spelling Variants, Proceedings of SIGIR, 2003
[5] D.Goldwasser and D.Roth: Active sample selection for named entity transliteration, Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL), 2008
[6] Haizhou Li, A Kumaran, Min Zhang and Vladimir Pervouchine: Whitepaper of NEWS 2009 Machine Transliteration Shared Task, Proceedings of Named Entities Workshop, ACL 2009, p. 19-26
[7] Jeff Pasternack and Dan Roth, Learning Better Transliterations}: Proceedings of CIKM, 2009
[8] Kevin Knight and Jonathan Graehl: Machine Transliteration, Computational Linguistics, 24(4)
[9] Kumaran A. and Kellner T: A generic framework for machine transliteration, Proceedings of SIGIR, p. 721-722

[10] Long Jiang, Ming Zhou, Lee-Feng Chien and Cheng Niu: Named Entity Translation with Wed Mining and Transliteration, Proceedings of IJCAI 2007
[11] Rahul Chitturi, Sebsibe H Mariam and Rohit Kumar: Rapid Methods for Optimal Text Selection, Proceedings of RANLP
[12] Richard Sproat: A formal computational Analysis of Indic Scripts, The International Symposium on Indic Scripts: Past and Future
[13] Richard Sproat, Tao Tao and ChengXiang Zhai: Named entity transliteration with comparable corpora, Proceedings of ACL 2006
[14] Sandeva G, Hayashi Y, Itoh Y and Kishino F., Srishell Primo: A Predictive Sinhala Text Input System, Proceedings of IJCNLP workshop on NLP for Less Privileged Languages, 2008
[15] Scott McCarley J., Cross: Language Name Matching, Proceedings of ACM-SIGIR, 2009
[16] Sravana Reddy and Sonji Waxmonsky: Substring-based Transliteration with Conditional Random Fields, Proceedings of Named Entities Workshop, ACL 2009, p. 96-99
[17] Sujith Ravi and Kevin Knight: Learning Phoneme Mappings for Transliteration without Parallel Data, Proceedings of the 47th Annual Meeting of the ACL and the 5th IJCNLP
[18] Tarek Sherif and Grzegorz Kondrak, Substring-Based Transliteration: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, 2007}. p. 944-951
[19] U. Hermjakob, K. Knight, and H. Daume III: Name translation in statistical machine translation - learning when to transliterate, Proc. of the Annual Meeting of ACL, 2008[
[20] Van Santen J.P.H. and Bushsbaum A.L.: Methods for optimal text selection, Proceedings of Eurospeech, p. 553-556
[21] Vladimir Pervouchine, Haizhou Li and Bo Lin, Transliteration Alignment, Proceedings of the 47th Annual Meeting of the ACL and the 5th IJCNLP, p. 136-144.
[22] Xue Jiang, Le Sun and Dakun Zhang: Syllable-based Name Transliteration System, Proceedings of Named Entities Workshop, ACL 2009, p. 96-99
[23] Yo Ehara and Kumiko Tanaka-Ishii: Multilingual Text Entry using Automatic Language Detection, Proceedings of IJCNLP, 2008
[24] Li Haizhou, Zhang Min, and Su Jian: A Joint Source-Channel Model for Machine Transliteration. Proceedings of ACL, 2004