A Hybrid Approach for Transliterated Word-Level Language Identification: CRF with Post Processing Heuristics

Somnath Banerjee
CSE Department,JU,India
s.banerjee1980@gmail.com

Aniruddha Roy
CSE Department,JU,India
aniruddha@gmail.com

Alapan Kuila
CSE Department,JU,India
alapan.cse@gmail.com

Sudip Kumar Naskar
CSE Department,JU,India
sudip.naskar@cse.jdvu.ac.in

Sivaji Bandyopadhyay
CSE Department,JU,India
sivaji_cse@yahoo.com

Paolo Rosso
NLE Lab,UPV,Spain
prosso@dsic.upv.es

ABSTRACT

In this paper, we describe a hybrid approach for word-level language (WLL) identification of Bangla words written in Roman script and mixed with English words as part of our participation in the shared task on transliterated search at Forum for Information Retrieval Evaluation (FIRE) in 2014. A CRF based machine learning model and post-processing heuristics are employed for the WLL identification task. In addition to language identification, two transliteration systems were built to transliterate detected Bangla words written in Roman script into native Bangla script. The system demonstrated an overall token level language identification accuracy of 0.905. The token level Bangla and English language identification F-scores are 0.899, 0.920 respectively. The two transliteration systems achieved accuracies of 0.062 and 0.037. The system presented in this paper resulted in the best scores across almost all metrics among all the participating systems for the Bangla-English language pair.

Categories and Subject Descriptors

1.2.7 [**Artificial Intelligence**]: Natural Language Processing, Language parsing and understanding

General Terms

Experimentation, Languages

Keywords

Word level language identification, Transliteration

1. INTRODUCTION

In spite of having indigenous scripts, often Indian languages (e.g., Bangla, Hindi, Tamil etc.) are written in Roman script for user generated contents (such as blogs and tweets) due to various socio-cultural and technological reasons. This process of phonetically representing the words of a language in a nonnative script is called (forward) transliteration. Especially the use of Roman script in transliteration for those languages presents serious challenges to understanding, search and (backward) transliteration. These challenges include handling spelling variations, diphthongs, doubled letters, reoccurring constructions, etc.

Language identification for documents is a well-studied natural language problem [3]. King and Abney[9] presented the different aspects of this problem and focussed on the problem of labeling the language of individual words within a multilingual document. They proposed language identification at the word level in mixed language documents instead of sentence level identification.

The last decade has seen the development of transliteration systems for Asian languages. Some notable transliteration systems were built for Chinese [14], Japanese [7], Korean [8], Arabic [1], etc. Transliteration systems were also developed for Indian languages [6, 16].

2. TASK DEFINITION

A query q : $< w_1 w_2 w_3 ... w_n >$ is written in Roman script. The words, $w_1, w_2, w_3, ..., w_n$, could be standard English words or transliterated from Indian languages (IL), e.g., Bangla, Hindi, etc. The objective of the task is to identify the words as English or IL depending on whether it is a standard English word or a transliterated IL word. After labeling the words, for each transliterated word, the correct transliteration has to be provided in the native script (i.e., the script which is used for writing the IL). Names of people and places in IL should be considered as transliterated entries, whenever it is a native name. Thus, the system has to transliterate the identified native names (e.g. *Arundhati Roy*). Non-native names (e.g. *Ruskin Bond*) should be skipped during labeling and are not evaluated.

3. DATASETS AND RESOURCES

This section describes the dataset that have been used in this work. The training and the test data have been constructed by using manual and automated techniques and made available to the task participants by the organizers . The training dataset consists of 800 lines.The testset contains 1000 sentences.

The following resources provided by the organizers were also employed:

• *English word frequency list*[1]: contains standard dictionary words along with their frequencies computed from a large corpus constructed from news corpora.

• *Bangla word frequency list*[2]: contains Bangla words in Roman script along with their frequencies computed from the *Anandabazar Patrika* news corpus.

• *Bangla word transliteration pairs dataset*[15]: contains Bangla-English transliteration pairs collected from different users in multiple setups - chat, dictation and other scenarios.

4. SYSTEM DESCRIPTION

We divided the overall task into two sub-problems: (a) word-level language (WLL) classification, and (b) transliteration of identified IL words into native script.

4.1 WLL classification Features

4.1.1 Character n-grams

Few studies [9, 5] successfully used the character n-gram feature and they obtained reasonable results. Therefore, following them, we also used this feature from character unigrams up to five-grams. After empirical study on the development set, we decided on the maximum length of a word to be 10 for generating the character n-grams. Therefore, if the length of the word is more than 10, then due to the fixed length vector constraint the system generates 10 unigrams and the last two characters are skipped. Thus the system always generates a total of 40 n-grams, i.e., 10 unigrams, 9 bigrams, 8 trigrams, 7 four-grams and 6 five-grams. The entire word is also considered as a feature.

4.1.2 Symbol character

A word might start with some symbol, e.g. *#*, *@*, etc. It has also been observed from the training corpus that symbols appear within the word itself, e.g. *a***a*, *kankra-r*, etc. Sometimes the entire word is built up of a symbol, e.g. *“*, *?*.

$$has\_symbol(word) = \begin{cases} 1 & \text{if } word \text{ contains any symbol} \\ 0 & \text{otherwise} \end{cases}$$

4.1.3 Links

This feature is used as a binary feature. If a word is a link, then it is set to 1, otherwise it is set to 0.

$$is\_link(word) = \begin{cases} 1 & \text{if } word \text{ is a link} \\ 0 & \text{otherwise} \end{cases}$$

4.1.4 Presence of Digit

The use of digit(s) in a word sometimes means different in the chat dialogue. For example, *‘gr8’* means *‘great’*, *‘2’* could mean *‘to’* or *‘too’*. This feature is also used as binary feature. Therefore,

$$has\_digit(word) = \begin{cases} 1 & \text{if } word \text{ contains any digit} \\ 0 & \text{otherwise} \end{cases}$$

4.1.5 Word suffix

Any language dependent feature increases the accuracy of the system for a particular language. [2] successfully used the fixed length suffix feature in the Bangla named entity recognition task. To include this feature, we have prepared a small suffix-list (10 entries) under human supervision from the archive (10 documents) of an online Bangla newspaper. This feature is also used as a binary feature.

$$has\_suffix(word) = \begin{cases} 1 & \text{if } word \text{ contains any suffix} \\ 0 & \text{otherwise} \end{cases}$$

4.1.6 Contextual Probability

This feature is very much crucial to resolve the ambiguity in the WLL identification problem. Let us consider examples given below.

• Mama *take* this badge off of me.

• Ami *take* boli je ami bansdronir kichu agei thaki.

The word *‘take’* exists in the English vocabulary. However, the backward transliteration of *‘take’* is a valid Bangla word. Words like *‘take’*, *‘are’*, *‘pore’*, *‘bad’* are truly ambiguous words with respect to the WLL identification problem as they are valid English words as well as backward transliterations of valid Bangla words. In this regard, context of the word can be used to correctly identify the language for such an ambiguous word. Therefore, we have considered this very useful feature.

As in the Bangla-English language identification task the label should be one from the tag-list: {English, Hindi, Bangla, Others}, we calculate the probability of the previous word being English, Hindi, Bangla and Others. Thus, four probabilities have been calculated for the previous word. In a similar way, the labeling probabilities for the next word have also been calculated.

The system calculates the respective probabilities as $P_{tag}(W) = \frac{F_{tag}(W)}{F(W)}$, where, *tag* is any one from the list: {E, O, H, B}; $F_{tag}(W)$ = frequency of the word W belonging to *tag*; $F(W)$ = Frequency of word W. These frequencies are counted from the training corpus. However, for few words in the testset the respective probabilities are 0. Since we do not want assign zero probability to those words, we need to assign some probability mass to those words using smoothing. We use the simplest smoothing technique, Laplace smoothing, which adapts the empirical counts by adding a fixed number (say, 1) to every count and thus eliminates counts of zero. For simplicity, we use *add-one smoothing*. Therefore, the adjusted formula is: $P_{tag}(W) = \frac{F_{tag}(W)+1}{F(W)+N}$, where, N = total number of words in the training corpus.

4.2 WLL Classifier

In this work, Conditional Random Field (CRF) is used to build the model for WLL identification classifier. We used *CRF++ toolkit*[3] which is a simple, customizable, and open source implementation of CRF.

4.3 Post Processing

After CRF classifier labels each word, post-processing heuristics are applied to make a rule-based decision over the outcome of the classifier. The following heuristics are employed:

Rule-1: Many English words end with *‘ed’* (e.g. *decided, reached, arrested, looked*, etc.), but we have not found any occurrences of any Bangla word ending with that suffix in

[1]http://cse.iitkgp.ac.in/resgrp/cnerg/qa/fire13translit/English%20-%20Word%20frequencies.txt

[2]http://cse.iitkgp.ac.in/resgrp/cnerg/qa/fire13translit/Bangla-Word%20frequencies.txt

[3]http://crfpp.googlecode.com/svn/trunk/doc/index.html

the given corpus. Therefore, an word ending with *'ed'* and having no symbol inside it is tagged as an English word. In the test corpus we found 306 such occurrences.
R1: H-Tag(w)=E ; if C-Tag(w)= B or O, has_suffix(w, 'ed')= true and w $\notin S$
Where, C-Tag(w)=Classifier's output, H-Tag(w)=Heuristic based output, has_suffix(w, s)= word ends with suffix s, and S = set of special character , E = English tag, B = Bangla tag, O = Others tag.

Rule-2: An English word may end with *'ly'* suffix also, e.g. *thoughtfully, anxiously, unfriendly*, etc. It has been observed in the test dataset that few English words were not written in correct spelling and they were mis-classified as Bangla words, e.g. *lvly, xactly, physicaly*, etc. These words are corrected by applying this rule.
R2: H-Tag(w)=E ; if C-Tag(w)= B or O, has_suffix(w, 'ly')= true and w $\notin S$

Rule-3: It was also observed that unlike English words (e.g. *evening, kissing, playing*, etc.) no Bangla words end with *'ing'* suffix in the training corpus. We found 316 such occurrences in testset, but some occurrences are not tagged as English because those words start with '#' (e.g. *#engineering*). This rule was able to correct some spelling errors such as *luking, nthing, njoying*, etc.
R3: H-Tag(w)=E ; if C-Tag(w)= B or O, has_suffix(w, 'ing')= true and w $\notin S$

Rule-4: The use of apostrophe s (i.e., *'s*) is very common in English words, e.g. *women's, uncle's* etc. In the test dataset, we found 73 use of it.
R4: H-Tag(w)=E ; if C-Tag(w)= B or O, has_suffix(w, ''s')= true and w $\notin S$

Rule-5: Another very common use of apostrophe is apostrophe *t* (i.e., *'t*), e.g., *don't, isn't, wouldn't*, etc. Even it is used in different way such as *rn't, cudn't*, etc.
R5: H-Tag(w)=E ; if C-Tag(w)= B or O, has_suffix(w, ''t')= true and w $\notin S$

Rule-6: A few users prefer to use words ending with *'ll*, e.g., *I'll, It'll, he'll, you'll*, etc. We found 20 such occurrences in the test set.
R6: H-Tag(w)=E ; if C-Tag(w)= B or O, has_suffix(w, ''ll')= true and w $\notin S$

Rule-7: The use of words like o'clock, O'Keefe etc. are very uncommon in Bangla social media users. But we found 16 such occurrences in the test dataset.
R7: H-Tag(w)=E ;if C-Tag(w)= B or O, starts_with(w, 'o")= true and w $\notin S$

Rule-8: This rule is very much straightforward. If a word contains a special symbol, then the word is tagged as O.
R8: H-Tag(w)=O ; if C-Tag(w)=B or O or E or H and w $\in S$

Rule-9: Although a few ambiguities are discussed in 4.1.6, there is a high chance of a word being English if it is in the English dictionary. Considering the ambiguity, we also consider the probability of the word to be in Bangla language.
R9: H-Tag(w)=E ; if C-Tag(w)=B and probability_Bangla(w) < 0.08 (this threshold was set empirically.)

Rule-10: The use of character repetition in the word is observed not only in English and Hindi, but in Bangla as well. The following observations have been noticed:
(1) Repetition of a character more than twice at the end of a word has the higher chance of the word being an English/Hindi word than Bangla. E.g. *torengeee, plzzzzzz*, etc.
(2) Repetition of a character more than twice in the middle of a word has the higher chance of the word being a Bangla word than English. E.g. *kisssob, oneeek*, etc.
(3) If a word satisfies both condition (1) and (2), then the word is more likely to be an English word. E.g. *muuuuaaahhhhhhh.*
The following rules are employed:
Case-1: R10a: H-Tag(w) = E ; if C-Tag(w) = B or O or H, end_repeat(ch) >= 3 and w $\notin S$
Case-2: R10b: H-Tag(w) = B ; if C-Tag(w) = E or O or H, middle_repeat(ch) >= 3 and w $\notin S$
Case-3: R10c: H-Tag(w) = E ; if C-Tag(w) = B or H or O, end_repeat(ch) >= 3 and middle_repeat(ch) >= 3 and w $\notin S$

Rule-11: This rule is also very much straightforward. If a word contains any substring from the list: {*www., http:, https::*}, then the word is tagged as Others.
R11: H-Tag(w) = O ; if C-Tag(w) = B or E or H, and contains(w) = *www.|http:|https::*

5. TRANSLITERATION SYSTEM

For transliterating the detected Romanized Bangla words, we built our transliteration system based on the state-of-the-art phrase-based statistical machine translation (PB-SMT) model [13] using the Moses toolkit [12]. PB-SMT is a machine translation model; therefore, we adapted the PB-SMT model to the transliteration task by translating characters rather than words as in character-level translation. For character alignment, we used GIZA++ implementation of the IBM word alignment model [4]. To suit the PB-SMT model to the transliteration task, we do not use the phrase reordering model. The target language model is built on the target side of the parallel data with Kneser-Ney smoothing [10] using the SRILM tool [11]. The PB-SMT model was trained on the English-Bangla word transliteration pairs dataset [15] provided by the task organizers. In a bid to simulate syllable level transliteration we also built a transliteration model by breaking the English and Bangla words to chunks of consecutive characters and trained the transliteration system on this chunked data. The chunk-level transliteration system is supposed to perform better than the character-level transliteration system since a chunk contains more context than a character. While decoding, we first apply the chunk-level transliteration system on the detected Bangla words. If the chunk-level transliteration system is able to transliterate a word only partially (i.e., it still contains roman characters), the untranslated parts are decoded using the character-level transliteration system. For breaking the English and Bengali words into chunks, we take two approaches. In the first approach (Run-1) we simply break words into chunks of consecutive 2/3 characters. In the other approach (Run-2), we break words into transliteration units (TU) following the heuristic used in [6]. The TU-level transliteration system was trained on named entities.

6. RESULTS

Table-1 presents the obtained results. Our system achieved an overall accuracy of 0.905 for the language labeling task

which is the best among the participating teams.

Table 1: Results

Token level language accuracy			
Language	Precision	Recall	F-Measure
Bangla	0.866	0.935	0.899
English	0.944	0.899	0.920
Token level Transliteration			
Run	Precision	Recall	F-Measure
Run-1	0.033	0.572	0.062
Run-2	0.019	0.338	0.037
Other Performance Metrics			
EQMF All(No Translit.)			0.444
EQMF without NE(No Translit.)			0.548
EQMF without MIX(No Translit.)			0.444
EQMF without NE&MIX(No Translit.)			0.548
EQMF All Run-1			0.005
EQMF All Run-2			0.004
EQMF without NE: Run-1			0.007
EQMF without NE: Run-2			0.004
EQMF without MIX: Run-1			0.005
EQMF without MIX: Run-2			0.004
EQMF without NE&MIX: Run-1			0.007
EQMF without NE&MIX: Run-2			0.004
ETPM: Run-1			227/364
ETPM: Run-2			134/364
Language Identification Accuracy			**0.905**

6.1 Error Analysis

It was observed that the WLL classifier based on CRF wrongly predicted due to the small training data. Moreover, some words were predicted correctly by the classifier, however, due to the heuristics the final prediction went wrong; e.g., the word *Wannna* is re-classified by (R10b) wrongly as Bangla. R10a also mis-classified Hindi words having character repetition at the end, such as *torengeee*, *Arehhh*, etc. R10a also mis-classified Bangla words such as *jahhh*, *jetooooo*, etc. Rule-8 re-classified some words due to tokenization errors in the provided test dataset *am!", back!", goin', ekjon-eri*, etc. Some words in the testset were of the form *word1/word2*, such as *isharay/nirupay, samanyo/8B* etc., which were simply classified as O (i.e., Others) using Rule-8 in our system.

The TU-level transliteration system was trained over named entities; hence it performed well for NEs, but the overall performance was affected because majority of the detected Bangla words were non-NE words.

7. CONCLUSIONS

In this paper, we presented a brief overview of our hybrid approach to address the automatic WLL identification problem. We found that the use of simple post-processing heuristics enhances the overall performance of the WLL system. Two variants of the transliteration systems were developed based on the segmentation of the transliteration data, i.e., at chunk-level and syllable-level. As future work we would like to explore more features for the machine learning model and better post-processing heuristics for the WLL identification task and try to increase the efficiency of our transliteration system.

8. ACKNOWLEDGMENTS

We acknowledge the support of the Department of Electronics and Information Technology (DeitY), Government of India, through the project "CLIA System Phase II".

9. REFERENCES

[1] Y. Al-Onaizan and K. Knight. Named entity translation: Extended abstract. In *HLT*, pages 122–124. Singapore, 2002.

[2] S. Banerjee, S. Naskar, and S. Bandyopadhyay. Bengali named entity recognition using margin infused relaxed algorithm. In *Text, Speech and Dialogue*, pages 125–132. Springer International Publishing, 2006.

[3] K. R. Beesley. Language identifier: A computer program for automatic natural-language identification of on-line text. In *American Translators Association*, page 54, 1988.

[4] P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. Mercer: The mathematics of statistical machine translation: parameter estimation. In *Computational Linguistics*, pages 263–311, 1993.

[5] G. Chittaranjan, Y. Vyas, K. Bali, and M. Choudhury. Word-level language identification using crf: Code-switching shared task report of msr india system. In *EMNLP*, page 73, 2014.

[6] A. Ekbal, S. Naskar, and S. Bandyopadhyay. A modified joint source channel model for transliteration. In *COLING-ACL*, pages 191–198. Australia, 2006.

[7] I. Goto, N. Kato, N. Uratani, and T. Ehara. Transliteration considering context information based on the maximum entropy method. In *MT-Summit IX*, pages 125–132. New Orleans, USA, 2003.

[8] S. Y. Jung, S. L. Hong, and E. Paek. An english to korean transliteration model of extended markov window. In *COLING*, pages 383–389, 2000.

[9] B. King and S. Abney. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *NAACL-HLT*, pages 1110–1119, 2013.

[10] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *ICASSP*, pages 181–184. Detroit, MI, 1995.

[11] R. Kneser and H. Ney. Srilm-an extensible language modeling toolkit. In *Intl. Conf. on Spoken Language Processing*, pages 901–904, 2002.

[12] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: open source toolkit for statistical machine translation. In *ACL*, 2007.

[13] P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *HLT-NAACL*, 2003.

[14] H. Li, Z. Min, and J. Su. A joint source-channel model for machine transliteration. In *ACL*, 2004.

[15] V. Sowmya, M. Choudhury, K. Bali, T. Dasgupta, and A. Basu. Resource creation for training and testing of transliteration systems for indian languages. In *LREC*, 2010.

[16] H. Surana and A. K. Singh. A more discerning and adaptable multilingual transliteration mechanism for indian languages. In *COLING-ACL*, pages 64–71. India, 2008.