Transliterated Search of Hindi Lyrics

Pallavi Verulkar
IIIT Bhubaneswar
Department of Computer Science

Rakesh Chandra Balabantray
IIIT Bhubaneswar
Department of Computer Science

Rohit Arvind Chakrapani
IIIT Bhubaneswar
Department of Computer Science

ABSTRACT
A huge number of Indian languages are written using native scripts. However, usually the websites and the user generated content (such as tweets, chats and blogs) in these languages are written using Roman script due to various phonetic-typing as the users feel comfortable in writing in their native language. Transliteration of many languages into Roman script is used copiously on the web not just for documents but also for user queries that are used to search these documents. A challenge that search engines face while processing transliterated queries and documents is that of extensive spelling variation. The aim of this topic is to systematically formalize several research problems that one must solve to tackle this unique situation prevalent in Web search for users of many languages around the world. We choose to solve the problem of Language identification when the Hindi words are written in Roman script. Then, transliterate the roman scripted Hindi words into Devanagri form. And when a search query is given, results should be retrieved for Hindi song lyrics.

Keywords
Language Identification, Transliteration, Indexing, Search query

1. INTRODUCTION
A huge number of Indian languages are written using native scripts. However, usually the websites and the user generated content (such as tweets, chats and blogs) in these languages are written using Roman script due to various phonetic-typing as the users feel comfortable in writing in their native language[1]. Such content creates a monolingual or multi-lingual space with more than one script which we refer to as the Mixed-Script space. Information retrieval in the mixed-script space, which can be termed as Mixed-Script IR (MSIR), is challenging because queries written in either the native or the Roman scripts need to be matched to the documents written in both the scripts.

Transliteration is a technique of mapping from one script of writing into another, word by word, or alphabet by alphabet. It is to follow the transcribing a word or text written in one writing system of script into another writing system. Transliterations in the narrow sense are used in situations where the original script is not available to write down a word in that script, while still high accuracy is required. One instance of transliteration is the use of an English computer keyboard to type in a language that uses a different alphabet, such as Marathi, Russian, Chinese and Hindi etc.

The process of phonetically representing the words of a language in a non-native script is called transliteration [2]. Transliteration, into Roman script is most widely accepted on the internet not just for files but for the instant messages that users send in the conversation on any social networking sites, or for any query fired on search engines. When the queries are written there is no specific spelling in the transliteration as the user freely converts the word according to the phonetics, user think is correct. That makes it very difficult to identify the language and the context of the query of document. The native word can have extensive spelling variations. For example, the word अच्छा ("good" in Hindi and many other Indian languages) can be also written as aacha, acha, aachha, achha, aachaa and so on.

This observable fact gives as a major matching problem faced by the search engines to match the original or native script to the transliterated script or Roman transliteration, where the query asked need to search in the various documents and having extensive spelling variations because of the transliteration. For many languages, where typing in the native script is not very convenient or popularly used for input of search engine, in such cases, the user types the native language words and sentences (usually) in Roman script, and a transliteration engine automatically converts the Roman input back to the native script. It is important to make a distinction between *forward* and *backward transliteration*. While the former refers to transliterating a word of language A (say Hindi) into the script of language B (say English, in which case the script is Roman), the latter is the reverse process of getting back the word in the native script, given its transliteration in a foreign script[3]. Thus, the process of generating aacha or accha from the word अच्छा, is forward transliteration, whereas the process of generating अच्छा given accha is backward transliteration. For the proper forward and backward transliterations it required to have two different datasets for training. For example, to learn an English-to Hindi backward transliteration engine one would need transliterations pairs such as "*accha*, अच्छा", where the original word is in Hindi, and its transliteration in English is a representation of the sound (/accha/ in IPA) of the Hindi word using the English script. On the other hand, for English-to-Hindi forward transliteration engine, one would need instances like "*accha*, अच्छा", where the original word of English origin –"accha", and the transliterated word is representation of its sound (/accha/ in IPA) in Hindi script, i.e., अच्छा.

This paper is divided into sections: section 2 describes the work that has already being done and the work which is related. Section 3 describes the problem statement of the paper while section 4 gives all the links for the dataset that were used for the work done in this paper. Section 5 gives the

proposed method and the challenges. Section 6 describes the results and Section 7 gives the conclusion and future work that can be carried on.

2. RELATED WORK

Ben King and Steven Abney have worked in labelling the languages of words in mixed language documents [6]. The focus of the paper is labeling the languages of individual words within a those documents that have more than one language or in general term a multi-lingual document. They have explores the possibilities of language identification in multi-lingual documents. After their evaluation they found that language identification of words is difficult if only single word is considered while if the context of the word is considered the results are better. This paper attempts to address three of the ongoing issues specifically- supporting minority languages, sparse or Impoverished training data, and multilingual documents. For evaluation of the data they created a corpus from BootCat tool and decided to consider 30 languages. They addressed the problem at the word level classification. They used the logistic regression method considering the n-gram approach with the use of tagging tool like MALLET(A Machine Learning for Language Toolkit). For further evaluation this problem is addressed in weakly supervised fashion, where the sequence of the word is also considered while in the independent word level identification it was not considered. Among the approaches evaluated, they found a conditional random field (CRF) model trained with generalized expectation criteria was the most accurate and performed consistently since the amount of training data was varied.

Barman, Utsab, Amitava Das, Joachim Wagner, and Jennifer Foster [7] have discussed the problem of automatic language identification of the social media content is solved. They have used three methods first a simple heuristic-based approach which uses a combination of our dictionaries to classify the language of a word. Second word-level classification using supervised machine learning with SVMs but no contextual information. Third Word-level classification using supervised machine learning with SVMs and sequence labeling both employing contextual information. A dictionary-based language detector predicts the language of a word based on its frequency in multiple language dictionaries. In their data the Bengali and Hindi tokens are phonetically typed. To predict the language of a word, dictionaries with normalized frequency were considered first (BNC,SemEvalTwitter, Training Data), if not found, word list look-up was performed. The predicted language is chosen based on the dominant language(s) of the corpus if the word appears in multiple dictionaries with same frequency or if the word does not appear in any dictionary or list. They have got two errors firstly Named entity errors - when a named entity is given a label that does not match the label it was given in the original annotation. Second Shared word errors - when a word that could belong to either language is classified incorrectly.

Kanika Gupta, Monojit Choudhury and Kalika Bali have worked on Mining Hindi-English Transliteration Pairs from Online Hindi Lyrics [3]. They have done the work on Hindi-English language but the work can be applied to any other native language if the transliteration is present in Roman form. In paper describes a technique to mine Hindi-English transliteration pairs from online Hindi song lyrics. The mining task is non-trivial as the Hindi lyrics which are usually present in websites are different as their transliterations differ. It hence creates the problem of matching the lyrics to the correct transliteration of the song and makes it a challenging task. The problem increases as the data acquired from the web may contain some form of noise. They mined the web and collected data both for forward and backward transliteration pairs and the crawling of different websites for making the corpus of lyrics for aligning. After the corpus collection the lyrics noises (repetitions, line breaks or transcriptions) are removed. Then they align the song using cosine similarity for Devanagri corpus and for roman corpus, vector space representation. Again they have used the word level alignment using an edit distance based approximate string matching algorithm, using HMM classifier. With this work done the results are of high quality and noise free.

3. PROBLEM STATEMENT

The problem statement was taken from the Shared task on Transliterated Search at Forum for Information Retrieval Evaluation (FIRE) in 2013[4]. Language detection of individual words in the corpus or search query - Suppose that q: w_1 w_2 w_3 ... w_n, is a query which is written in Roman script. The words, w_1 w_2 etc., could be Standard English words or transliterated from Hindi**.** The task is to label the words as E or H depending on whether it an English word, or a transliterated **Hindi** word. For each transliterated word, provide the correct transliteration in the native script (i.e., the script which is used for writing **Hindi**). Then the second task after the language identification is transliteration of the search query entered by the user. Retrieval of appropriate song lyrics file based on the input search query.

4. DATASET USED

The complete dataset for training and testing was taken from-
1. Training data for language identification task: http://tinyurl.com/m6yyw48
2. English word frequency list: http://tinyurl.com/lyck5js
3. Hindi word frequency list: http://tinyurl.com/laxahsk
4. Hindi words transliteration pairs: http://tinyurl.com/lnx4vha, http://tinyurl.com/oewsyx7
5. Song lyrics document collection: http://www.dsic.upv.es/~pgupta/data/msir-doc-collection.zip

5. PROPOSED METHOD

The problem is divided into three major subtasks and they have different approaches and methods.

5.1 Language Identification

Language identification is a qualification for all other task that is needed to perform. Hence, having a good language classifier is necessary and increases the accuracy of the results. This portion describes the algorithm used and the functioning of the language classifier. The major problem that is needed to identify is the words from two languages first English and second Hindi. If the words are written in Devanagri script for Hindi and Roman script for English it is then a simpler task to identify the language. This becomes one of the important and non-trivial tasks as the Hindi words are also written in roman script rather than Devanagri script. In the web this problem is often faced as the users who are firing the query know English as their second language and use phonetic writing for convenience.

In this method several dataset has been used, the data which is used for training the classifier consists of bilingual documents containing annotated English and Hindi words in Romanized script along with the transliterated form for the Hindi words. In this method auxiliary English and Hindi dictionary datasets

are used. In the English word dictionary, 207856 English words are stored with frequency. The Hindi dictionary which consists of Hindi words written in roman script is taken into account. In the Hindi word dictionary 15,000 roman scripted Hindi words are stored with frequency. The words present are with various spelling variations as the user can phonetically write different spellings.

In this approach the system designed is purely lookup based. The dataset taken contains of 30824 English Hindi pair's words, 207856 English words with frequency. But the dataset given contained noise and therefore reduction of noise is a necessary step. We reduce the given data by taking out all words having frequency 1. As the words which have frequency 1 means their chance of occurring in a document is 1 in 10000. So we consider it as negligible. From the above reduction the dataset renaming to us is now containing 1 lakh words.

Table 1 - Word variations in Roman Script

Word Variations
Pradhanmanti, pradhaanmantri, pradhaanmaantri, pradhanmantree, pradhaanmantree, pradanmantri
Dheemi, dhimi, dhemi, dhiimii, dhiimi, dheemee
Maloom, malum, maaloom, maalum, mallum
Zindagi, zendagi, zindagee, zindhagi

Based on first character, these English words are splitted and stored into 26 different files. Initially system reads a term (*word*) from text file and compares first character of *word* with files name, and then search the *word* in matched file (e.g. suppose *Airplane* is input word and first character is *A*, so this term will be searched in file *a.txt*). If word found in respective English file we temporarily assume it is English term, but there is possibility that it may be a Hindi term. For this reason without taking final decision we search term in EH pair file. If the term is found here also we assume it may be Hindi term. Hence we assign H to word as label. If the word is found in both the dictionaries, we check the frequencies with respect to the word in both the dictionary and finally label the word with either E or H whose frequency is more.

Algorithm-

1. Input *term from Test Document*

2. Check first letter of *word {a-z}*

3. Search *word in corresponding Document*

4. *if match found*

4.1 Search *word in Hindi* Document

4.2 if *found*

4.3 Check the frequencies in both the documents

4.4 Label the word as either English or Hindi based on the frequency .

5. else

5.1 Label it as Hindi word.

We have used NetBeans IDE 7.2.1 **Java:** 1.7.0; Java HotSpot(TM) 64-Bit Server VM 21.0-b17 as the simulator to run and check all the proposed algorithms.

As we run the first segment we get the desired results and the terms are classified based on the input query given. The user can give either all the terms of the query in English or Hindi. But there are fair chances that it can be mixed or roman scripted transliteration. Therefore for clarification we give labels \E for English and \H for Hindi so that we can transliterate the terms which are only in roman scripted format.

5.2 Transliteration

A corpus of Hindi song lyrics in Roman and Devanagri script was collected along with the frequency lists of Hindi and English words. Necessary filtering was done, and then a mapping from Roman Words to a list of their Devanagri counterparts was created. Now the frequency lists were used to remove lower frequency Devanagri words from this one-to-many mapping. In this process, we also get mapping for Roman syllables. Thus, a many-to-one mapping of transliterated word pairs and transliterated syllable pairs was created to directly transliterate words. All the words are identified as English or Hindi. If the word is identified as Hindi, the word is transliterated to Devanagri script. The English words remain same and no changes are done to those words.

Table 2- Variations for Hindi words written in Roman script

Roman Scripted Hindi word	Hindi Transliteration
achchhi	अच्छी
Achchi	अच्छी
achcchi	अच्छी
achhi	अच्छी
achi	अच्छी

In the transliteration of Hindi words written in roman script, the lookup dictionary is created which consist of 14,000 words with their corresponding transliterated word. Those words are considered for transliterations which are identified as Hindi language and English words are not transliterated. In the process of transliteration all the possible combination of words are considered. As the table 2 gives the idea that all the spelling variations are considered in transliteration and this mainly occur due to phonetic typing, for the speakers whose second language is English.

If the search query consists of both Hindi and English words, all the Hindi words are transliterated and all the English words remain same as they are. For example if the query is as follows "main kya karu". In this query the first term is found in both English and Hindi dictionaries. But word main(Roman Scripted Hindi) has more frequency than the word found in English dictionary. Therefore all the three terms will be tagged by \H. And then the transliteration of the words will be displayed as the result as follows, for main – मैं, kya - क्या, and for karu - करूँ.

5.3 Searching of Hindi Lyrics

This module is for the query processing part. The query consists of either Devanagri script or its transliterated Roman text or a combination of both and this module outputs ranked list of song lyrics, with results in both Devanagri and Roman scripts of the songs list with their respective document ids to locate the result. The mining task is tough and challenging because the Hindi lyrics and its transliterations are usually available on the web but are very different, and often unrelated on several websites. Therefore, it is a non-trivial task to match the Hindi lyrics to their transliterated counterparts. Moreover, there are various types of noise in lyrics data that needs to be appropriately handled before songs can be aligned at word level [3].

Note that, after pre-processing of the lyrics, different Devanagri versions of the same song are expected to be identical. On the other hand, their Roman counterparts can still be significantly different from each other due to natural spelling variations generated during forward transliteration. It is important as well as useful for us to capture these variations. Therefore, we divide the problem of song alignment into the following two sub-tasks: First, we identify all the Devanagri versions of the same song. Since they are expected to be identical, we only take one of the versions for further processing. The second sub-problem is to align the Roman songs to one of the unique Devanagri songs discovered in the previous step.

- Complete database of songs consist of 63,000 documents in form of text file.
- Indexing is created and then classified based on scripts.
- Input query is being searched in indexing files.
- Based on the input query, decision is taken whether to search in Roman corpus or Devanagri corpus.

For the index creation complete database of songs is considered. The index creation is an important task because we cannot open and check if a particular search query matches the contents of the lyrics or not. Therefore, we create index for database and search the query in the formed index. The index creation is also divided into two sections- Roman based lyrics and Devanagri based. New files are created to store the indexes for songs. The roman based index files are again categorized into 26 files based on the roman alphabets. And rest all are Devanagri script indexes and stored in separate text files.

Table 3- Roman Scripted Lyrics stored in Index

After creation of Index
Gayab Hoke doc-id-22573.txt
Gayab Hoke Sab Dikhta Hai doc-id-22567.txt
Gayab Hoke Sab Dikhta Hai doc-id-22578.txt
Gayab Hoke Sab Dikhta Hai Lyrics doc-id-22583.txt
Gayatri Mahamantra doc-id-22587.txt
Gayatri Mantra doc-id-2825.txt
Gayatri Mantra Lyrics doc-id-2849.txt
Gaye Dinon Ka Suraag Lekar Kidhar Se doc-id-40225.txt
Gazab doc-id-678.txt

After transliteration of the word, there are two possible cases. The searches is carried for the given query in roman scripted indexes and transliterate it to Devanagri script and search the transliterated query in Devanagri scripted indexes.

Table 4 - Devanagri form of lyrics Index

After creation of Index
दम मारो दम ... हरे कृष्ना हरे राम doc-id-25180.txt
दम मारो दम ॥। हरे कृष्ना हरे राम doc-id-25168.txt
दम मारो दम, मिट जाये ग़म, बोलो सुबह शाम doc-id-25164.txt
दम मारो दम, मिट जाये ग़म, बोलो सुबह शाम doc-id-25181.txt
दम से गये हमदम के लिये doc-id-39251.txt
दम से गये हम्दम के लिये doc-id-39219.txt
दम है बाकी तो ग़म नहीं doc-id-26470.txt
दम है बाकी तो ग़म नहीं doc-id-26473.txt

If the query is in roman script, then firstly then the language identification of every word is done and then its corresponding transliterations if found. And then the query is searched through the lyrics index from roman lyrics. We then again generate new query through the result we got from subtask-2 of transliteration. The words are joined and the new transliterated query is searched in the Devanagri index text files. Suppose the user entered query was "Aa bhi ja" So the two query formed in the below example would be

1. Aa bhi ja
2. आ भी जा.

Some of the results for the first query are:

"Aa Bhi Ja doc-id-21982.txt

Aa Bhi Ja Aaise Na Tarsa doc-id-21936.txt

Aa Bhi Ja Aye Mere Hamdam doc-id-21961.txt"

It gives the songs retrieved with their corresponding document id if the user wishes to open the document. Some of the results for second query are:

"आ भी जा आ भी जा ऐ सुबह आ भी जा doc-id-55364.txt

आ भी जा दिलरुबा ... सीने में भी तूफ़ाँ है doc-id-55840.txt

दिल तड़प तड़प के कह रहा है आ भी जा doc-id-38032.txt"

6. RESULTS

The results are evaluated with different parameters and the parameters are defined below. Correct label pairs imply E−E and H−H, while incorrect label pairs include E−H and H−E, where E is for English and H stands for the Hindi language [5]. TP, TR and TF are based on the overall factor, while EP,

ER and EF are based on the factor of language English precision, recall and F-score, similarly for Hindi language.

Transliteration Precision –

$$Transliteration\ precision\ (TP) = \frac{\#(Correct\ transliterations)}{\#(Generated\ transliterations)}$$

Transliteration Recall –

$$Transliteration\ recall\ (TR) = \frac{\#(Correct\ transliterations)}{\#(Reference\ transliterations)}$$

Transliteration F-Score –

$$Transliteration\ F-score\ (TF) = \frac{2*TP*TR}{TP+TR}$$

LA- Language Accuracy –

$$Labelling\ Accuracy(LA) = \frac{\#(Correct\ label\ pairs)}{\#(Correct\ label\ pairs) + \#(Incorrect\ label\ pairs)}$$

English Precision –

$$English\ Precision(EP) = \frac{\#(E-E\ pairs)}{\#(E-H\ pairs) + \#(E-E\ pairs)}$$

English Recall –

$$English\ Recall(ER) = \frac{\#(E-E\ pairs)}{\#(H-E\ pairs) + \#(E-E\ pairs)}$$

English F-Score –

$$English\ F-score(EF) = \frac{2*EP*ER}{EP+ER}$$

Hindi Precision –

$$Hindi\ Precision(HP) = \frac{\#(H-H\ pairs)}{\#(H-E\ pairs) + \#(H-H\ pairs)}$$

Hindi Recall –

$$Hindi\ Recall(HR) = \frac{\#(H-H\ pairs)}{\#(E-H\ pairs) + \#(H-H\ pairs)}$$

Hindi F-Score –

$$Hindi\ F-score(EF) = \frac{2*HP*HR}{HP+HR}$$

The task that we have taken as a problem statement is mentioned by Shared task on Transliterated Search at Forum for Information Retrieval Evaluation (FIRE) in 2013[4]. We have taken all the dataset from the provided link. And the results that we obtained from completing the task are compared to the results that were already submitted to this track submission.

The last column of the result is the results obtained through the implementation of the technique that we proposed earlier. It shows better results than the earlier methods defined in the paper shown in figure given below with comparison.

Table 5 - Comparison of results obtained from language identification and transliteration

	ISM[8]	**NTNU[9]**	**GU[11]**	**Results**
TP	.72	.29	.41	.85
TR	.64	.22	.36	.80
T F-Score	.68	.25	.40	.82
LA	.81	.80	.81	.83
EP	.68	.55	.56	.918
ER	.91	.97	.97	.723
E F-Score	.78	.70	.71	.819
HP	96	.98	.99	.96
HR	.86	.74	.75	.945
H F-Score	.91	.85	.85	.95

The results for the third task is given below where if we give a query we get the result correct .81 times. This checked based on the accuracy calculated. The accuracy is measured in terms of the relevant and non-relevant documents. The list of relevant and non-relevant documents is already present in the given query list with the training and test data set.

Table 6 - Result comparison of searching of Hindi lyrics

	NTNU [9]	**Valencia [10]**	**GU [11]**	**Result**
Accuracy	.56	.805	.563	.813

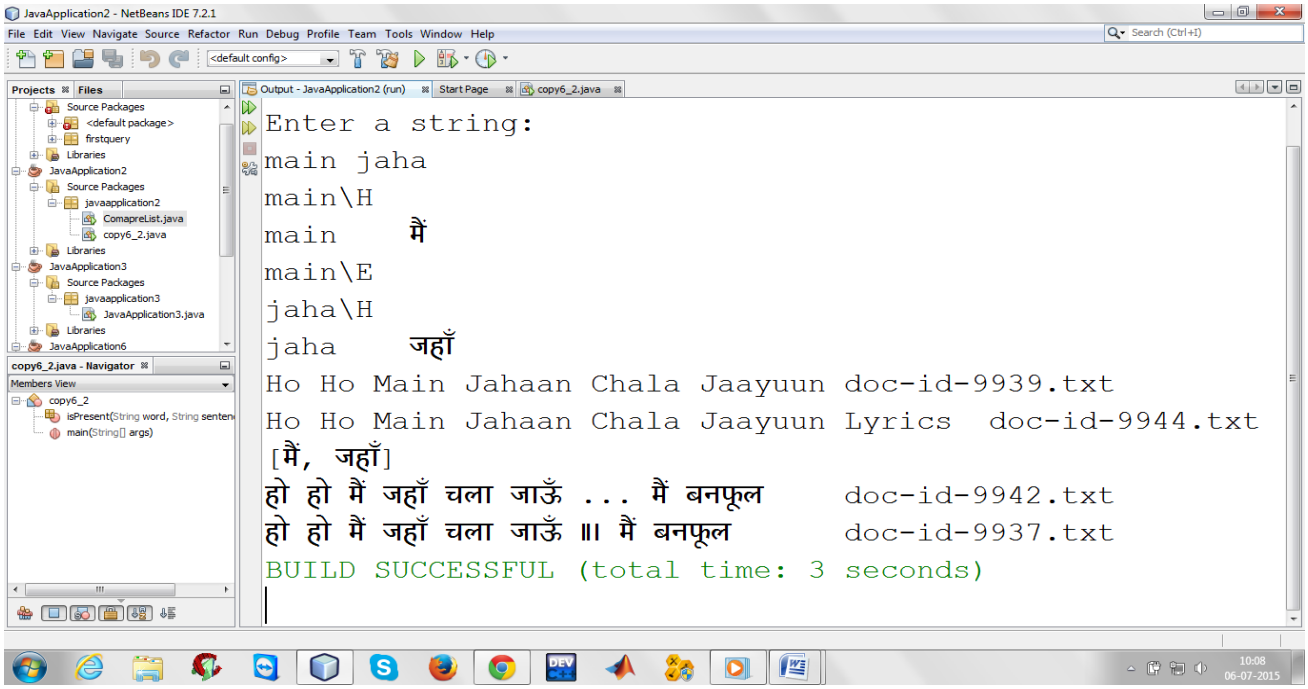

Fig 1: The result showing the search results after entering the query.

7. CONCLUSION AND FUTURE WORK

The results obtained from the above proposed method were slightly better than the earlier proposed works. But there are some points where results are lacking. The point which needed attention is recall portion where the improvement is required. But it is difficult because of the fact that we are using two different languages and using the same script makes it more difficult to recognize and correctly the words because of the different and large spelling variations used by the speakers. There is a lot of scope of improvement in the proposed method. We have not used the already present indexer for indexing of the songs as we are using two different languages and that can create problem while indexing as the two different languages have different semantics and way of writing. The indexing is done accordingly.

For the further work in language identification different machine learning techniques can be used such as Hidden Markov Model, Support vector machines, Conditional Random Field. For this we can also use the context based information, depending on the before and after words of the search query. The context based information also helps in improving accuracy of the language identification. For the transliteration of the words every word can be referred to another word using the editex algorithm.

8. REFERENCES

[1] U. Z. Ahmed, K. Bali, M. Choudhury, and S. VB. Challenges in designing input method editors for indian lan-guages: The role of word-origin and context. In Proceedings of the WTIM, pages 1–9, November 2011.

[2] K. Knight and J. Graehl. Machine transliteration. Comput. Linguist., 24(4):599–612, Dec. 1998.

[3] Gupta, Kanika, Monojit Choudhury, and Kalika Bali. "Mining Hindi-English Transliteration Pairs from Online Hindi Lyrics." In *LREC*, pp. 2459-2465. 2012.

[4] FIRE Shared Task on Transliterated Search http://research.microsoft.com/en-us/events/fire13_st_on_transliteratedsearch/default.aspx

[5] R. Saha Roy, M. Choudhury, P. Majumder, and K. Agarwal. "Overview and Datasets of FIRE 2013 Track on Transliterated Search." In Fifth Forum for Information Retrieval Evaluation, 2013.

[6] Ben King and Steven Abney "Labeling the languages of words in mixed-language documents using weakly supervised methods." Conference of the North American Chapter of the Association for Computational Linguisics: Human Language Technologies, pages 1110-1119, June-2013

[7] Barman, Utsab, Amitava Das, Joachim Wagner, and Jennifer Foster. "Code Mixing: A Challenge for Language Identification in the Language of Social Media." *EMNLP 2014* (2014): 13.

[8] Dinesh Kumar Prabhakar, Sukomal Pal "ISM@FIRE2013 shared task on Transliterated Search" http://research.microsoft.com/en-us/events/fire13_st_on_transliteratedsearch/fire14st.aspx

[9] Partha Pakray, Pinaki Bhaskar "Transliterated Search System for Indian Languages" http://research.microsoft.com/en-us/events/fire13_st_on_transliteratedsearch/fire14st.aspx

[10] Parth Gupta, Paolo Rosso, and Rafael E. Banchs Encoding transliteration variation through dimensionality reduction: FIRE Shared Task on Transliterated Search http://research.microsoft.com/en-us/events/fire13_st_on_transliteratedsearch/fire14st.aspx

[11] Hardik Joshi , Apurva Bhatt, Honey Patel "Transliterated Search using Syllabification Approach" http://research.microsoft.com/en-us/events/fire13_st_on_transliteratedsearch/fire14st.aspx