Language identification of short text in transliterated space.

Harsha Kokel
DA-IICT, Gandhinagar
200901199@daiict.ac.in

Abstract— **Huge amount of data available on social media services like twitter, Facebook etc. comprises of short text in transliterated space. For such transliterated text there is a need to identify the language in order to access information. Language Identification is a well studied problem. Various classifiers are used for language identification but it is not evident how well they function with short text in Romanized Indian languages. We generated an annotated collection of Tweets in Roman script for three Indian languages: Bengali, Gujarati and Hindi. Three classifiers: Naive Bayes, Logistic Regression and Support Vector Machine were compared for language identification of transliterated text. It was found that SVM performs the best with 4-gram features. Hence integrated SVM with the Terrier-3.5 IR platform to detect language of the query provided.**

I. INTRODUCTION

India is a multilingual and multiscript country. There are 23 officially listed Indian languages and 12 officially listed Indian scripts. Most of the Indian languages have a script of their own. For e.g. Hindi and Marathi use Devanagari Script, Gujarati uses Gujarati Script, Bangla and Assamese use Bengali Script, Kannada uses Kannada Script, Tamil uses Tamil Script, Telugu uses Telugu Script etc. There are multiple standard keyboards available for using Indian scripts like in-script keyboard, phonetic keyboard, unicode keyboard etc. But layman abstain from using such keyboards and rather use Transliteration. It has been observed that in most of the social media and discussion forums of Indian context, Indian languages have been written in Roman alphabets. It is evident from twitter and facebook that good amount of Hindi text is written in transliterated Roman script. Majority portion of the Indian song lyrics found online are in Roman Script and to search these lyrics people use search engine like Google or Bing. It becomes inevident for such search engines to identify the language of query to provide Cross Script information like lyrics of the Hindi song in Devanagari Script as well as Roman Script when query is in Roman Script. Query logs of search engines are full of transliterated texts and if only I can identify the language of these Transliterated text. In this paper I focus on the language identification of Indian languages transliterated in Roman script only.

Language identification in such a transliterated space is a problem as

1) Indian languages have large character set, which is mapped to character set of size 26 in Roman Script while transliteration. This mapping is nor one to one neither many to one. for e.g.,
   "श" and "ष" both are represented as sha.
   "त" and "ट" both are represented as ta.
   "ज" can be represented as j or z.
   like "मजुमदार" (Majumadāra) can be written as "Majumder" or "mazumdar".
2) This transliteration do not abide by any standard rule and hence it has many spell variations according to the corresponding sound of the alphabets in both languages. Vowels in Hindi can be mapped to single or multiple alphabets in Roman Script.
   for e.g.,
   "भूल" (Bhūla) can be written as "bhul" or "bhool".
   "है" can be written as "hai" or "hain".
3) Humans have tendency to remove redundancies. Like in SMS Language, online users also skip vowels and replace alphabets with numbers
   for e.g.,
   "नही" (Nahī̃) is written as "nhi" instead of "nahii",
   "पीछे" (Pīchẽ) is written as "pe6y" instead of "piche".
4) This transliterated text is multilingual i.e., it is comprised of words from an Indian language and a few from English or sometimes from a third language.
   for e.g.
   Bengali "shon besi deri hole Producer er office giye unedited film ta dekhe chole asbo"
   Gujarati "*Jab we met*, movie na joie hoye to aaje jao, Must Watch!!! "
   Hindi : "Hey, kahan hai?? kabse wait kar rahe hai, come soon."

   Here underlined words are in English.

Nowadays considerable amount of communication happens via SMS in transliterated space in the form of short text, [1]. Also I deal with lot of short texts such as statuses, updates, comments, tweets, reviews, snippets, chats etc. Twitter proves to be a good sample for short texts. So I used tweets for our experiment. The problem of language identification on Twitter have been addressed in [2] and [3]. But these studies are based on text written in the script of the language itself and not the transliterated Roman text. Here, I processed transliterated text on Twitter. Also, in [4] language identification for transliterated text was carried out but the set of challenges were different. They analysed 5 Slavic languages which use Cyrillic alphabets while I analysed 3 languages using 3 different scripts. They developed different transliteration tables to transliterate documents to Roman Script while I used human transliterated text in form of Tweets. Also their documents were monolingual while I used bilingual short texts.

II. Data

We generated a corpus of manually annotated bilingual tweets (henceforth called Twitter Corpus) in three languages viz., Bengali, Gujarati and Hindi. These tweets are from different domains and are bilingual i.e., a mixture of English and the Roman transliteration. The main problem in collecting this data is that there are no dedicated users on Twitter who tweet only in the Roman transliteration. So I made a list of unique and frequently used words in all the three languages and searched for tweets containing these words. Similar approaches have been adopted in [2] and [5]. We appointed three native speakers as annotators. They manually filtered and annotated the tweets. These tweets were then cleaned by removing all punctuations, URLs and digits. We also replaced multiple space and tabs by single space. We considered tweets containing more than 30 characters for our experiment. Each tweet was considered as individual document while indexing. The median document length was 73 characters for Bengali, 61 characters for Gujarati and 104 characters for Hindi. Average word length was 6 characters. Average character set size was 51 characters. Table 1 contains the statistics of the Twitter Corpus.

TABLE I
Statistics of annotated bilingual Tweets in Transliterated space

Language	No of tweets	Total terms	Unique terms	Character set $(vowels, consonants)$
Bengali	1532	21162	6447	$46_{(11,35)}$
Gujarati	1529	20046	4745	$43_{(15,28)}$
Hindi	1592	32292	6895	$64_{(14,50)}$
TOTAL	**4653**	**73500**	**15744**	

III. Language identification systems

We now describe the system implemented and tested on the Twitter corpus. All the experiments were conducted with 5-fold cross validation. The 70% data was used for training and 30% for testing.

A. Features

Two methods to generate the feature vector were used. First one is the basic Information retrieval feature, i.e. unique terms in the corpus and the tf-idf as its value. Second is based on Character N-gram as described in [6]. We generated list of overlapping n-grams and treated each unique n-gram as a different feature. Eight different sets of feature vectors using 2 grams, 3 grams, 4 grams, 5 gram, 6 gram, 2+3 gram (i.e. union of 2 gram and 3 gram), 3+4 gram and 2+3+4 gram were generated. Values of these feature vectors were calculated as mentioned in [7] using formula 1. Table 2 shows lengths of different feature vectors.

$$feature\_value = \frac{\textit{freq of the n-gram in the document}}{\textit{total n-grams in the document}} \quad (1)$$

TABLE II
Lengths of Feature Vectors.

	Feature Vector	Number of Features
1	Term	15396
2	2 gram	1685
3	3 gram	12624
4	4 gram	50205
5	5 gram	121807
6	6 gram	200084
7	2+3 gram	14309
8	3+4 gram	62829
9	2+3+4 gram	64514

B. Classifiers

For all the feature vectors mentioned in Table 2 three different classifiers were used .

1) Naive Bayes: : It is a very important text classification algorithm and it is based on Bayes Rule (equation 2). It is very fast, requires less storage and is robust to irrelevant features. It assumes that position of a word in the document doesn't matter and the probability of different features is independent given a class. We used Naive Bayes implementation in weka,[8].

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \quad (2)$$

2) LogR: : Logistic regression has shown to perform well on a range of NLP tasks so I use L2-regularized logistic regression (primal) classifier implemented in LibLinear Package,[9]

3) SVM linear: : The Support Vector Machine (SVM) is a state-of-art algorithm for classification. We used linear L2-regularized L2-loss support vector classification (dual) implemented in LibLinear Package,[9]. We only report results for multi-class SVM with linear kernels, as they were found to perform best over the corpus.

TABLE III
Confusion matrix.

K \ P	ben	guj	hin	Precision	Recall	F1 measure
ben	434	8	8	0.9886	0.9644	0.9764
guj	3	439	8	0.9712	0.9756	0.9734
hin	2	5	443	0.9651	0.9844	0.9747
Average				0.9749	0.9748	0.9749

K: Known Class, P: Predicted Class

IV. Experiments and Analysis

At first I used 4500 tweets from the Twitter corpus, 3150 tweets for training and 1350 tweets for testing to analyse the influence of various classifiers and the feature vectors. All the experiments were conducted with 5 fold cross validation. The Confusion Matrix as shown in Table 3 was generated. From this matrix individual Precision, Recall and F1-measure were calculated using the formulae 2, 3 and 4. These values were then averaged out to obtain final Precision, Recall and F1-measure.

$$PRECISION_{ben} = \frac{K_{ben}P_{ben}}{K_{ben}P_{ben} + K_{guj}P_{ben} + K_{hin}P_{ben}} \quad (3)$$

$$RECALL_{ben} = \frac{K_{ben}P_{ben}}{K_{ben}P_{ben} + K_{ben}P_{guj} + K_{ben}P_{hin}} \quad (4)$$

$$F1\text{-}MEASURE_{ben} = \frac{2 * PRECISION_{ben} * RECALL_{ben}}{PRECISION_{ben} + RECALL_{ben}} \quad (5)$$

TABLE IV
F1-MEASURE FOR ALL CLASSIFIERS FOR VARIOUS FEATURE VECTORS.

	Naive Bayes	LogR	SVM
term	0.9595	0.9311	0.9379
2 gram	0.8366	0.8776	**0.9166**
3 gram	0.9267	0.9368	**0.9746**
4 gram	0.9414	0.9629	**0.9911**
5 gram	-NA-	0.9781	**0.9801**
6 gram	-NA-	**0.5515**	0.5457
2+3 gram	0.9249	0.9168	**0.9611**
3+4 gram	0.9338	0.9588	**0.9898**
2+3+4 gram	-NA-	0.9316	**0.9783**

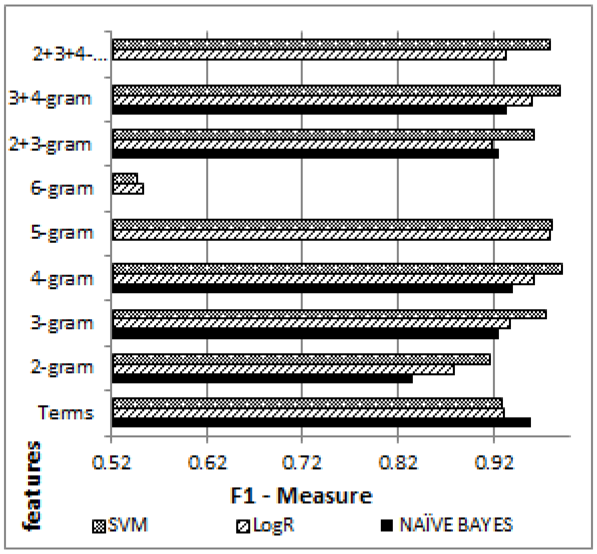

Fig. 1. F1-measures for different classifiers

Table 4 shows the F1-measure of all these Systems obtained on our Twitter Corpus for various Feature Vectors. Figure 1 provides graphical representation of this table. Experiment revealed that while SVM performs better than LogR and Naive Bayes for Character N-gram features, Naive bayes outperforms SVM and LogR for term features. Best results were obtained by SVM with 4-gram features. In [2] studies showed that 5-gram works best for Indian languages for Retrieval, however here 4-gram proved to be the most appropriate for Language Identification for the Twitter corpus. We also notice that the #tags and @tags of the tweet increase the accuracy by 1%. So here I only represent the results including these tags. On analysing the classifiers for individual language I found that for all languages 4-gram features achieved best results.

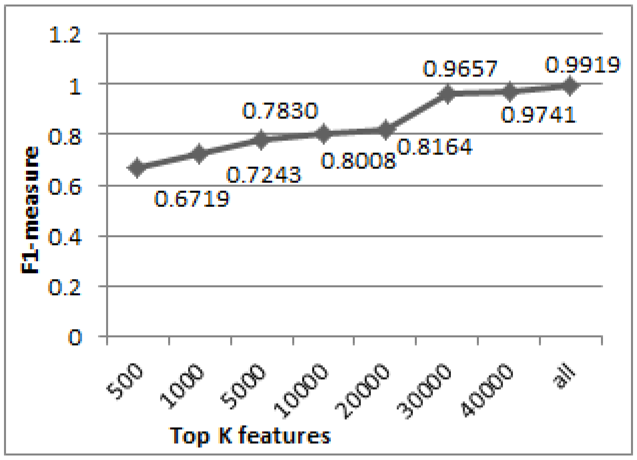

Fig. 2. F1 measure for Top k 4-grams Features

TABLE V
F1 MEASURE OF DIFFERENT DATA SIZE.

No of Tweets(Training, Testing)	F1-Measure
100(70,30)	0.8676
200(140,60)	0.9727
500(350,150)	0.9801
1000(700,300)	0.9679
1500(1050,450)	0.9911

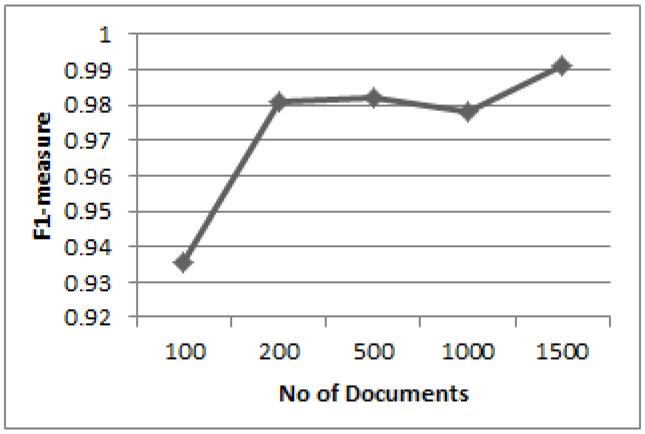

Fig. 3. F1 measure of varying Data size

With the increasing length of feature vectors it was unmanageable to classify using Naive Bayes. This led us to analyse how the classifiers perform when I reduce the length of feature vector. So I did feature selection by considering top K frequent features from the 4-gram feature vector where K $\in$ { 500, 1000, 5000, 10000, 20000, 30000, 40000, ALL }. F1-measure of 4-gram feature vector with SVM classifier is presented in Figure 2. While the results are more or less persistent after top 30000 frequent n-grams, best outcome is obtained when all the n-grams are considered.

To investigate the influence of training data size I decided to use several quantity of Tweets. We here report only the 4-gram feature vector classified using SVM. Table 6 presents the F1-measure for varying data size and Figure 3 shows the graphical representation. Although best results were obtained for maximum number of Tweets, results were more or less persistent above 200 Tweets.

V. Implementation

I implemented a Langauge identification module in Terrier-3.5 IR Platform. As the best results were found with SVM I trained SVM with the whole corpus and used that trained SVM to identify the language of the query in the Terrier. I realised that along with identifying the language a transliteration engine was also required which was out of the scope for this paper and hence I have only implemented this module for FIRE 2010 queries i.e. on inserting a FIRE 2010 transliterated query the Language Identification module will identify the language and turn the query into detected language and display the results.

VI. Conclusions

Large amount of online text is written in Roman scripts for its convenience. Keying Indian languages are also not an exception. Twitter, Facebook, blogs etc. have considerable amount of Bengali, Hindi, Gujarati texts transliterated in Roman script. For such transliterated text there is a need to identify the language in order to access information. Language Identification is a well studied problem but very few works are reported on informal Romanized short text in Indian languages. In this paper, I conducted series of experiments and observed that SVM classifier with 4-gram features perform the best.

References

[1] S. K. P. M. Khushboo Singhal, Gaurav Arora, “Sms normalization for faq retrieval,” in *Working Notes of the Forum for Information Retrieval Evaluation*, 2011, pp. 78–88.

[2] P. McNamee, “N-gram tokenization for indian language text retrieval,” in *Working Notes of the Forum for Information Retrieval Evaluation*, 2008, pp. 12–14.

[3] S. Bergsma, P. McNamee, M. Bagdouri, C. Fink, and T. Wilson, “Language identification for creating language-specific twitter collections,” in *Proceedings of the Second Workshop on Language in Social Media*. Association for Computational Linguistics, 2012, pp. 65–74.

[4] A. Chepovskiy, S. Gusev, and M. Kurbatova, “Language identification for texts written in transliteration.”

[5] R. Ghani, R. Jones, and D. Mladenic, “Automatic web search query generation to create minority language corpora,” in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 432–433.

[6] W. B. Cavnar, J. M. Trenkle *et al.*, “N-gram-based text categorization,” *Ann Arbor MI*, vol. 48113, no. 2, pp. 161–175, 1994.

[7] D. Cavar, “Lid-language identification in python: Practical use of n-gram models and simple statistics,” 2008.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A library for large linear classification,” *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.