Department of Computer Science and Engineering
IIT Kanpur

Romanagari Detection in Twitter

Hrishikesh Terdalkar - 14111265 { `hrishirt@` }
Shubhangi Agarwal - 14111268 { `sagarwal@` }

November 13, 2015

CONTENTS

1 INTRODUCTION

Twitter is an online social networking service that enables users to send and read short 140-character messages called "tweets". Twitter has been the target platform for various research studies due to its free APIs giving access to a small percentage of all public tweets.

In location specific studies, pertinent to Indian continent, people write various native languages in Roman (English) script instead of the conventional scripts of respective languages. Hindi is the most spoken language in Indian continent (40% of the population). Romanagari is writing Devanagari-script-based-language in Roman script. We often see examples of these codemixed texts in social media such as twitter.
e.g. *"kya kar rahe ho, i dont even see you these days".*
In majority of twitter based studies, such tweets have to be discarded as noise. We intend to tackle this problem, by attempting detection and tagging of such tweets.

1.1 PROBLEM

Romanagari is Devanagari-script-based-language written in Roman script.
Given random collection of roman-script tweets, we want to find out tweets that are English-Hindi codemixed (or pure Hindi), tag the individual words as well as entire tweet with language prediction.

1.2 CHALLENGES

First big challenge of course is lack of good datasets, for training and testing. Previous works have used synthetically created codemixed datasets [1]. We have attempted to collect codemixed tweets, as well as manually tagged around 600 lines of Hindi and Marathi sentences (tweets, chats etc).

Some of the challenges in this are, words that are part of both "Romanagari" vocabulary, and English. For example, word "*log*" in a sentence "**log** *in to our website*" is English, however when used in "*tum* **log** *kab aaoge*" is Hindi.

Same Hindi word can be written in various ways, such as "*kyon*", "*kyun*", "*kyu*". Some of the inflections of a word might clash with English words.

1.3 RELATED WORKS

Recently there have been a number of works in code-mixed language detection. Gella et al deal with a wide set of languages for word labelling using existing language classifiers for language pairs [1] [2]. This achieve 80-90% accuracy on synthetically generated datasets. Das et al. have used n-gram pruning with SVM for word level language identification [3]. Barman et al. is an extension of their work using very similar strategies with addition of CRF [4], achieving better accuracy.

However all of these works do not assume the amount of word variations and ambiguity that come in due to the nature of twitter.

1.4 Phonetics

When someone writes *Romanagari*, they do so based on how they perceive the pronunciation of that word, and while the spelling people associate with a word might differ, pronunciation generally remains same. So applying a phonetic algorithm to take care of inflections of the word seemed like a valid step.

Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English [5]. The goal of soundex is for homophones to be encoded to the same representation so that they can be matched despite minor differences in spelling Every soundex code consists of a letter and three numbers, such as `W252`. The letter is always the first letter of the word. The numbers are assigned to the remaining letters of the word according to the soundex guide. Zeroes are added at the end if necessary to produce a four-character code. Additional letters are disregarded.
eg:
Washington is coded `W252` (W, 2 for S, 5 for N, 2 for G, remaining letters disregarded).

An alternative of this gives a six character output, with similar rules, just carrying more character information. We use this version of soundex.
A sample sentence looks as follows:

Original: `['kya', 'kar', 'rahe', 'ho', 'kyaa', 'hua', 'tumhe', 'tumhein', 'kuch', 'chaiye', 'ky']`
Soundex: `['K00000', 'K60000', 'R00000', 'H00000', 'K00000', 'H00000', 'T50000', 'T50000', 'K20000', 'C00000', 'K00000']`

As we can see, "*kya*", "*kyaa*" and "*ky*" get mapped to same output. Similarly, "*tumhe*" and "*tumhein*" also is identified with each other.

There are other alternatives to Soundex, such as Metaphone, NYSIIS. We have presented the performance comparison of these algorithms in handling Hindi word-variations in Sec 3.1.

1.5 Approach

After applying the chosen phonetic algorithm on English and Hindi (Romanagari) corpora separately, we transform corpora into n-grams. After that we learn probabilistic language models based on n-grams, to predict in a given query, the probabilities of individual words and thus query itself of being Hindi or English. We apply some post-filters like word ratio considerations to increase accuracy.

2 DATASETS

Datasets for Romanagari are very rare. We have generated, collected and used (existing in literature with some processing) various datasets. This section lists details of these datasets as well as our collection and cleaning methods.

2.1 COLLECTION

We mention the existing datasets, as well as ones we collected. (with processing) as our training datasets.

2.1.1 DATASETS FROM EXISTING LITERATURE

Rovereto Twitter n-gram Corpus (RTC) is an n-gram dataset enriched with meta-data such as gender and time of posting. The n-gram corpus is based on 75 million English tweets extracted from a larger sample of 240 million tweets collected from the public stream of Twitter, between December 2010 and July 2011.[6]

NLTK tweet_samples is English tweets collection, part of NLTK Corpora containing 20,000 tweets.

IITB Hindi Devanagari Corpus is Devanagari script Hindi corpus containing around 1200 files.[7] It has roughly 220,000 lines (2.85 million words). We converted this to Roman script to use for training.

In addition, we got list of top Hindi terms with their inflections from FIRE2013 dataset.[8]

2.1.2 COLLECTED DATASETS

Hindi-English Tweets Corpus (Code-mixed)
We attempted collecting Hindi tweets by unigram-search on most frequent Hindi terms, however this yielded on very poor results, giving less than 5% useful tweets.
We then used some of the "pairs of words" that occur in a same sentence frequently. By manual selection, we chose 49 such terms that we perceived "most frequent". Few examples of such terms (skip-grams) are "*mujhe-hai*", "*tumhe-jab*", "*aur-nahi*". We searched these terms as with 'AND' condition (both must occur), in Twitter's REST API. This gave us 38,264 tweets of rich code-mixed quality.
Based on success of skip-gram we combined 94 most-frequent Hindi words, computed all combinations of these words, and for these 4,371 words, we ran our tweet collection. After 16 hours of collection, we obtained 335,672 tweets from this.

Social Media: gchat, WhatsApp, Facebook (Code-mixed)
We collected various handpicked lines of codemixed text from social media such as Google-talk, WhatsApp, Facebook. Overall 297 lines of Hindi and 390 lines of Marathi were collected.

2.2 Preprocessing

We employed various cleaning methods depending on nature of the data. This section will give details of pre-processing performed on data.

2.2.1 Tools and Scripts

Depending on need, we wrote and used a number of bash, awk, sed, grep, tr, python, js scripts.

collect.py collect tweets using REST API for provided terms with 'AND' condition.

mass_collect compute ${}^nC_2$ on most-frequent Hindi terms, and collect tweets in bulk and put them in labelled files

basic_cleaning is a repeated used cleaning script. It removes accents, turns corpus into lower-case, removes punctuations, removes extra spaces/empty lines, and add start-end markers.

tweet_cleaning clean tweets, remove tweets with links, remove retweets, replace handles by HANDLE, remove hashtags, uniq and then apply `basic_cleaning`

tagger was made to tag codemixed lines interactively on terminal efficiently. Provide input file, language tags and start tagging with hotkeys for language tags, *skip* and *back*

devToRom.js convert Devanagari script to Latin for Romanagari corpus collection. we have modified this script, mainly doing character-wise replacement following certain rules. [9] This can be run on terminal using `node.js`.

nltk various nltk tools, including tweet collection framework

SRILM Tools SRILM is a toolkit for building and applying statistical language models (LMs) [10] It contains set of C++ class libraries implementing language models, supporting data structures and miscellaneous utility functions, various n-gram related supporting tools/scripts.

2.2.2 Cleaning Details and Statistics

We have used aggressive cleaning methods to reduce ambiguity from corproa. Following are methods applied on each individual corpora to clean.

Rovereto (RTC) corpus contains lot of noise. It has demographic information about each n-gram, a 1200 dimension vector, as well as junk characters. We only took n-grams that *do not* contain *any* special characters, and added up appropriated demographic information to obtain frequency of n-grams for $n = 1, 2, \ldots, 6$. This reduced total size of corpus from 250 GB to 1.2 GB.

Tweets For English and Hindi code-mixed tweets that we collected, we first removed duplicates, retweets and tweets containing URLs. We also removed accents and lower-cased

the entire corpus. After this we removed every "mention" of user (such as @narendramodi, @OfficeOfRG) by word "HANDLE". This step was taken to preserve semantic information of a "person" being target of certain actions. Basically, these tweets were passed through `tweet_cleaning` which resulted in final corpus 59,287 Hindi tweets and 3,187 English tweets, tagged with start-end markers `<s>` and `</s>`.
Example of a cleaned tweet:

```
<s> HANDLE sorry bolne se kuch nai hota huh </s>
```

Social Media (Handpicked)
This data did not have too much noise as it was handpicked. We passed it through `basic_cleaning`. This served as training corpus. We made an interactive `tagger` script with which we tagged 297 Hindi and 300 Marathi lines.
Example of a tagged tweet:

```
<s> <hi>bhaisaab itna mazaa kafi</hi> <en>time</en> <hi>baad aya</hi> <en>a
lot of catching up</en> <hi>bhi ho gayi</hi> </s>
```

IITB Hindi (Devanagari) For this large Devanagri corpus we ran `devToRom.js` using `node.js` and converted it to Roman text. Applied `basic_cleaning` on that. There was not much data loss.

3 Model

We have considered various n-gram based models to predict language of a tweet. From our cleaned Hindi and English training corpora, we computed 1-grams, 2-grams, 3-grams, 4-grams for both languages with frequencies. We used these n-grams for Training and Tagged datasets for testing.

Following table lists statistics of these datasets.

N-grams in Training Set		
	English	**Hindi**
1-grams	1,168,077	120,546
2-grams	10,644,439	998,300
3-grams	17,353,446	2,027,733
4-grams	14,007,551	65,186,143

Tagged Test Data	
Language (type)	**Count**
English (lines)	3187 tweets
Hindi (lines)	3000 tweets
Hindi (words)	297 lines
Marathi (words)	390 lines

For each method, we need to train *two* language models, one for *English* and other for *Codemixed-Hindi*.
For English Training we use RTC n-grams, and for Codemixed Hindi training, we use mixed dataset of transliterated-Hindi and collected code-mixed tweets. Phonetic conversion of n-grams is computed after this, preserving the frequencies.

3.1 Comparison of Phonetic Algorithms

Intuitively, since the method of writing Romanagari is by phonetic interpretation, it was clear we need to choose some phonetic Algorithm. We tried out Soundex, Metaphone and NYSIIS.

To evaluate Soundex's output, we used FIRE 2013 data [8], which has over 30,000 transliteration pairs of Hindi words and their inflections. (18,000 unique Hindi words). We first computed soundex on the transliterated words, and computed the amount of inflections that were handled by soundex. 6500 words had 2 or more inflections, out of which average 53.6% inflections were taken care of by soundex. However this number went up to 57% for top 1000 frequent Hindi words, and 62.3% of inflections of top 100 words were handled by soundex. Double Metaphone, yielded lesser inflection handling capacity on our sample experiments. While NYSIIS even lesser.

Following is the comparison of some phonetic algorithms, based on which *Soundex* was the obvious choice.

Variations handled by Phonetic Algorithms			
In top .. terms	5000	1000	100
Soundex	53.96%	57.01%	62.3%
Double Metaphone	48.05%	50.52%	51.83%
NYSIIS	12.43%	19.39%	28.26%

At this point we would like to remark that Double Metaphone and NYSIIS are supposed to work better than Soundex. They have high accuracy, but that is tested on English corpus, and they fail to perform on Hindi code-mixed text and social media context.

3.2 Simple ngram-model

In this model, the training and learnt sets is simply 6 sets, one corresponding to soundex-ed n = 2, 3, 4-grams each, in both the languages. In evaluation, it iterates over the query, finds all possible n-grams for a word in that query, n = 2, 3, 4. Assign the word the maximum value of n as score, one for Hindi and one for English. The language for which it has a higher score is deemed as the language of the word. Further score for the query is calculated by summing over the square of this score for each word in the query, and tagged with the language of sentence that gives higher score.

$q = w_1, w_2, \ldots, w_n$

Model v1

$w\_score(w, L) = \max_n$ (ngram containing $w \in q) \in L$

$q\_score(q, L) = \sum_{w \in q}$ [$w\_score(w, L)$]^2

Model v2

$w\_score(w, L) = \sum_n \sum_j$ n .. where j is an ((ngram containing $w) \in q) \in L$

$q\_score(q, L) = \sum_{w \in q}$ [$w\_score(w, L)$]

3.3 Probabilistic ngram-model (SRILM)

We model conditional probabilities based on n-gram frequencies. This language model, supplied with many options, learns on the TrainData of user-specified n-gram length, by learning probabilities of words in vocabulary conditioned on containing 1-gram, 2-gram,..., n-gram and calculating backoff weights [optionally]. Evaluation of the TestData is done with varied level of outputs.

The most detailed output gives the conditional probability of each word in the best ngram context. This output also gives the perplexity to each sentence, both including and excluding the start end markers. In addition to this the model successfully marks unseen words as *Out of Vocabulary* (OOV).
For example,

```
p( B15200 | A35200 ..  )  = [1gram] 0 [ -1.0867 ]
```

In this, log of 1-gram-probability of word `B15200` (in soundex format), given that preceding word was `A35200` is -1.086

For each sentence we also get overall probability and perplexity of it on each model.
For example

```
3 zeroprobs, logprob= −81.4182, ppl= 33360.6 ppl1= 61561.1
```

The sentence has 3 unknown words, log of probability of this sentence being in current language model is -81.41 with a perplexity of 33k. More the perplexity, less the certainty of model about containing a particular sentence. The model is renormalized (by recomputing backoff weights).

3.4 Wordratio Model

The Wordratio model can use the output of previous tagging methods. We take the output of probabilistic model on TestData and process the text-output to tag each word of raw query text with language tags. Such as `<hi>Hindi</hi>`, `<en>English</en>` or `<oth>Other</oth>`. If output of probabilistic model, tags a word as OOV then tag the word in raw query as other. Otherwise calculate hindi-ness and english-ness score of the word based on the output of two language models (one learnt on Soundex of English Corpora and one on Soundex of Hindi Corpora) by multiplying the conditional probability of the word with the length of best ngram context. The word was tagged Hindi if the score for the language was higher than that for English, and same way for English. The length of best ngram context being the tie breaker.
These tagged query strings were further processed and were classified as Hindi, English and Other based on the ratio of the words of these classes present in the query and setting some manual threshold values.

```
    $hi_{freq} = hi_{count} / total\_words$
$en_{freq} = en_{count} / total\_words$
if $(hi_{freq} + en_{freq}) < M$ then mark query as **Other**
else if $(hi_{freq} > N)$ then mark query as **Hindi**
else mark query as *English*
```

This model, acts as a filter on top of individual word tagging.

4 Experimental Results

For testing, we cut our collected twitter corpus in 2, test set of 3000 tweets manually tagged to have Hindi codemix, and training set of 56,287 tweets. We merged 3000 Code-mixed tagged files with 3187 English tweets file (out of which around 200 were non-English noise). We tested out 3 models on these. Simple n-gram model in both its versions failed the task. SRILM ngram-model performed fairly well on chosen parameters, while word-ratio based n-gram model gave the best results on chosen parameters.

In following individual tables,
(**Correct**, *L*): number of tweets of language *L* that are marked correctly (*True Positives*).
(**Wrong**, *L*): number of tweets of language *L* that were marked wrong (*False Negatives*).
(**Wrong as**, *L*): number of tweets that were marked wrongly as language *L* (*False Positives*).

4.1 Simple n-gram Model

Results of the proposed simple models were disappointing. Almost everything was marked as English. This seems to happen because out of all the codemixing used, percentage of codemixing is low. Therefore there are always more n-grams found in English corpus than Hindi corpus.

Simple ngram-model v1			
	English	Hindi	Total
Correct	3165	352	3517
Wrong	22	2648	2670
Wrong as ..	2648	22	

Simple ngram-model v2			
	English	Hindi	Total
Correct	3176	97	3273
Wrong	11	2903	2914
Wrong as ..	2903	11	

4.2 Probabilistic ngram-model (SRILM)

Perplexity is, in some sense, measure of "confusion" in the model. After computing the perplexity of each tweet, we can get how much "confidence" does a particular language model has about a sentence. Based on this, queries were tagged.

Probabilistic (SRILM) (ppl_thr = 20k)			
	English	Hindi	Total
Correct	2052	1777	3829
Wrong	1135	1223	2358
Wrong as ..	787	495	

Probabilistic (SRILM) (ppl_thr = 25k)			
	English	Hindi	Total
Correct	1920	1936	3856
Wrong	1267	1064	2331
Wrong as ..	628	406	

4.3 Wordratio Filter

Following two tables present results of filtering with WordRatio. Cells indicate number of queries tagged.
Parameters (40,20) imply that $M = 40$ and $N = 20$ for the model described above, i.e., if percentage of English and Hindi words combined is less than 40, mark the tweet as of "other" language. Else, if percentage of Hindi words is more than 20, mark the query as Hindi (i.e. as codemixed), English otherwise. This is very parameter dependent however which in turn depends on nature of data.

Word Ratio Score (40, 20)			
	English	Hindi	Total
Correct	2076	2687	4763
Wrong	1111	313	1424
Wrong as ..	313	1110	

Word Ratio Score (50, 10)			
	English	Hindi	Total
Correct	890	2921	3811
Wrong	2297	79	2376
Wrong as ..	79	2294	

4.4 Comparison of Methods

$$Precision = \frac{TP}{TP + FP}$$

$$Rcall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Following table contains comparison of above methods, and as is clear, WordRatio on top of Conditional Probabilistic Model performs best overall, however other models can be used depending on need of Precision on particular language.

Comparison of Tagging Methods

	English			**Hindi**		
	Precision	Recall	F1	Precision	Recall	F1
Simple v1	54.45%	99.31%	0.70	94.12%	11.73%	0.21
Simple v2	52.25%	99.65%	0.69	89.81%	3.23%	0.06
Prob (ppl=20k)	72.28%	64.39%	0.68	78.21%	59.23%	0.67
Prob (ppl=25k)	75.35%	60.24%	0.67	82.66%	64.53%	0.72
WordRatio(40,20)	86.90%	65.14%	**0.74**	70.77%	89.57%	**0.79**
WordRatio(50,10)	91.85%	27.93%	0.43	56.01%	97.37%	0.71

5 Conclusion

Romanagari Detection is a complex task with various facets, and constantly changing nature of data affects in this. Probabilistic n-gram models seem to achieve acceptable level of accuracy in the detection. From the performance increased obtained due to usage of soundex, it is apparent that soundex played a very vital role in taking care of 50-60% of the inflections. Methods are however very much dependent on nature and statistics of data.

5.1 Disclaimer

While these results give us certain ordering of which methods perform well in the task, this is still subject to the volatile nature of data. For example, if in certain corpus, percentage of Hindi text was higher than English, the Simple Models v1 and v2 will perform better.

5.2 Future

Computing performance of these models without application of soundex , as well as with application of other similar algorithms to soundex, is an interesting branch to explore.

While our predictions about Simple Model might hold true for a more favourable corpus, we suspect there might be a favour for Hindi and thus more errors in the other direction. So, in general "goodness" of a model should be measured on "stability" of results over various corpora with different statistics.

Mixing corpora in controlled different percentages, and then testing output of these models on each can be performed to get variations of percentage accuracy and other measures.

In general, the number of variations possible to handle the task are is large, and it is unclear whether a particular method works definitely better than another without a good measure of comparing methods. Developing one such *goodness measure* is a hopeful future direction.

ACKNOWLEDGEMENTS

We are grateful to Professor Amitabha Mukerjee and our senior M S Ram for valuable insights. We would like to also thank family and friends for the help in data collection, tagging, as well as providing constructive criticism.

6 REFERENCES

[1] S. Gella, K. Bali, and M. Choudhury, "'ye word kis lang ka hai bhai?' testing the limits of word level language identification."

[2] S. Gella, J. Sharma, and K. Bali, "Query word labeling and back transliteration for indian languages: Shared task system description," *FIRE Working Notes*, 2013.

[3] A. Das and B. Gambäck, "Identifying languages at the word level in code-mixed indian social media text," in *Proceedings of the 11th International Conference on Natural Language Processing, Goa, India*, 2014, pp. 169–178.

[4] U. Barman, A. Das, J. Wagner, and J. Foster, "Code mixing: A challenge for language identification in the language of social media," *EMNLP 2014*, p. 13, 2014.

[5] Soundex indexing system. [Online]. Available: www.archives.gov/research/census/soundex.html

[6] A. Herdagdelen. Rovereto twitter n-gram corpus. [Online]. Available: http://clic.cimec.unitn.it/amac/twitter_ngram/

[7] Iitb hindi corpus. [Online]. Available: http://www.cfilt.iitb.ac.in/Downloads.html

[8] Datasets of fire 2013. [Online]. Available: http://cse.iitkgp.ac.in/resgrp/cnerg/qa/fire13translit/

[9] Latin-to-roman transliteration. [Online]. Available: http://www.hindidevanagari.com/transliteration/

[10] A. Stolcke *et al.*, "Srilm-an extensible language modeling toolkit." in *INTERSPEECH*, 2002.

[11] B. Han and T. Baldwin, "Lexical normalisation of short text messages: Makn sens a# twitter," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 368–378.

[12] Twitter package - nltk. [Online]. Available: http://www.nltk.org/howto/twitter.html

[13] Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Main_Page