Project Report
Language Detection in Code-mixed Social Media Text

Alefiya Naseem, Paarth Kotak, Siddhesh Acharekar

CS6120 NLP Fall 2018

naseem.a@husky.neu.edu

kotak.p@husky.neu.edu

acharekar.s@husky.neu.edu

I. Introduction

The past few years have seen social media websites and networks penetrate greatly into multilingual societies. The linguistic and syntactic freedom that these environments on social media platforms afford their users lead to many deviations from what is the accepted norm of communication in a language. One of these deviations happens when users switch between languages i.e. code-mixing. Code-mixed text adopts the vocabulary and grammar of two or more languages and often forms new structures based on its users. Studies on the phenomenon can be found in linguistics literature [1] which majorly discuss conversational and sociological motivations behind code-mixing as well as it's linguistic nature. This proves as a roadblock for NLP tasks on social media text like sentiment analysis. Presence of phonetic typing of words leads to error in interpretation and hence mistakes in sentiment detection, translation, and other NLP tasks.

A clean example would be:

Hindi-English code-mixed: *Snell library bohot accha hain, don't you agree?*

Literal meaning: *Snell library is very good, don't you agree?*

An example of a code-mixed sentence from our dataset is:

'millar kuch kar na saka killer ab india is becoming more thriller :P'

It's language tags will be:

millar\\UN *kuch*\\HI *kar*\\HI *na*\\HI *saka*\\HI *killer*\\EN *ab*\\HI *india*\\EN *is*\\EN *becoming*\\EN *more*\\EN *thriller*\\EN *:P*\\EMT

Where the tag legend is: HI: Hindi, EN: English, UN: Universal, EMT: Emoticon

Code-mixing is traditionally distinguished between inter-sentence (two or more languages in the same sentence), intra-sentence (two or more sentences with different languages), and, intra-word (code mixing within a single word) code mixing. This project evaluates previous word-level language identification experiments [2] on an inter-sentence Hindi-English code-mixed data, suggest improvements to those, then builds our own models and finally compare them. We have incorporated more features: both extracted from our data and external sources to better language identification. We will see ahead that the simplest unsupervised dictionary-based approach is outperformed by supervised word-level classification without contextual clues and later that it is crucial to take context into consideration [2] when performance is significantly boosted with probabilistic sequence models. Further, we set forth on using word-level classification algorithms, building multiple n-gram generative language models, conditional random fields and deep learning-based classification algorithms for sequential text data using bidirectional LSTM.

II. Related work

The problem of language identification has been investigated for half a century [3] and that of computational analysis of code switching for several decades [4], but there has been less work on automatic language identification for multilingual code-mixed texts, even less for Indian languages. The figure below is an example from Twitter where we can see that the machine successfully translates the text in authentic Hindi (Devanagari) script but fails to translate text typed in phonetic Hindi.

Prior research related to this task has mainly been focused on monolingual texts [5] due to their large-scale availability. However, in multilingual societies like India, usage of code-mixed languages (among which Hindi-English is most prominent) is quite common for conveying opinions online. Initial studies on Chinese-English code mixing [6] indicated that mainly linguistic motivations were triggering the code mixing in those highly bilingual societies. Reference [7] showed that Facebook users tend to mainly use inter-sentential switching (switched on 2 sentences) over intra-sentential (switched within a sentence), and report that 45% of the switching was instigated by real lexical needs, 40% was used for talking about a topic, and 5% for content clarification.

Turning to the work on automatic analysis of code mixing, there have been some studies on detecting code mixing in speech. Reference [8] tries to predict the points inside a set of spoken Spanish-English sentences where the speakers switch between the two languages. Other studies have looked at code mixing in different types of short texts, such as information retrieval queries and SMS messages. They explore several language identification approaches, including a Naive Bayes classifier for individual word-level classification and sequence labelling with Conditional Random Fields trained with Generalized Expectation criteria [9], which achieved the highest scores.

Another very recent work on this topic is [10]. They report on language identification experiments performed on Turkish and Dutch forum data. Experiments have been carried out using language models, dictionaries, logistic regression classification and Conditional Random Fields. They find that language models are more robust than dictionaries and that contextual information is helpful for the task.

III. Dataset

Considering the claim that code-mixing is most frequent among young, multilingual speakers [1], we have decided to use an Indian student community in the 20-30 age group as our data source. To save data collection and annotation time we have taken data (with permission) from previous researchers on this topic [2] who scraped a Facebook group and 11 of its users posts and comments. This data was then annotated by a team of 4 who were proficient in all 2 languages of our corpus (English and Hindi). Annotators were supplied with 4 basic language tags viz. English (EN), Hindi (HI), Universal (UN) and Emoticon (EMT). The attribute UN is associated with symbols, numbers, names, and universal expressions (e.g. hahaha, lol). The attribute EMT is specified for a phrase or a word for which no language tags can be attributed or cannot be categorized as UN and include punctuations. In addition, annotators were instructed to annotate named entities separately. Posts having words in any language other than Hindi or English were discarded, leaving us with 12936 code-mixed posts in total.

III.A Preprocessing

Our approaches fall into 2 categories overall: non-contextual and contextual. So, for the experiments we set out to try our data pre-processing approaches also had 2 variations.

For the non-contextual approach, we went with a bag-of-words technique where every sentence with word-language tag pairs was reduced to a list of words and tags by splitting on space. Then, all the word lists and tag lists corresponding to each sentence were concatenated leading to one list of words, and one list of corresponding tags. For the example shown above this would be:

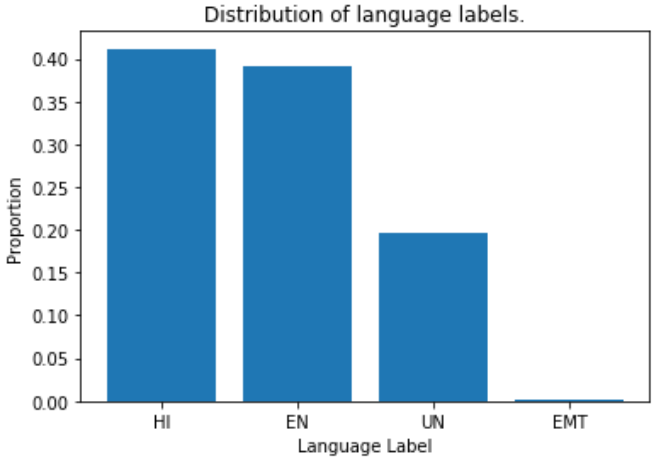
Word	Tag
millar	UN
kuch	HI
kar	HI
na	HI
…	…

On the other hand, for the contextual approach we have retained sentence structure by adding start-of-sentence(SOS) and end-of-sentence(EOS) tags for every sentence and then parsing them as mentioned above. This gave us 2 lists, one of just words with SOS and EOS tags and the other of just tags with SOS and EOS tags. For the example this would be as:

> SOS\\SOS *millar*\\UN *kuch*\\HI *kar*\\HI *na*\\HI *saka*\\HI *killer*\\EN *ab*\\HI *india*\\EN *is*\\EN *becoming*\\EN *more*\\EN *thriller*\\EN *:P*\\EMT EOS\\EOS

III.B Cleaning

Considering this was a human annotated data there were bound to be some errors in the language labelling of words. 25 True ‘EN’ labels for ‘English’ were found to be assigned as ‘E’, ‘EM’, ‘EN,’, ‘EN:’, ‘ENEOS’, ‘HEN#’, ‘en’, ‘NE’, ‘RN’, ‘MIX’, ‘ENC’. These were changed to ‘EN’ for English. 3 True ‘HI’ labels for ‘Hindi’ were found to be assigned as ‘HII’, ‘HIS’, and ‘USN’. 12 True ‘UN’ labels for ‘Universal’ were assigned as ‘U’, ‘UI’, ‘UN.’, ‘UNEOS’, ‘_’, ‘uN’, ‘un’. These were changed to ‘UN’ for ‘Universal’ (names, symbols, numbers and punctuations). Finally, our data had 4 Language tags: ‘EN’: English, ‘HI’: Hindi, ‘EMT’: Emoticons, and ‘UN’: Universal. Its distribution is as shown



Further, the data was divided into an 80% train – 10% dev – 10% test split for model training, hyperparameter tuning and model evaluation on unseen data respectively.

IV. Methodology

With 12936 posts split into training, development and test sets we tried the following approaches for word level language detection:

1) Unsupervised dictionary based language detection using the *WordNet synsets feature and a custom dictionary created from Twitter users modelled on the Twitter SemEval 2013 dataset.*

2) Supervised classification algorithms *Naïve Bayes, SVM and RandomForests* without contextual features.

3) Supervised classsification algorithms *SVM and Condtional Random Fields (CRF)* with contextual features.

4) Deep Learning classification with *LSTM's.*

IV.A Feature Selection

Given the methods we planned to use, and the 2 contextual and non-contextual paths to solve the task of word level classification we evaluated several features and debated their intuitive and empirical validity. The features we selected were:

Non-contextual:

1) Char n-grams (n = 2 to 5): We start with a character n-gram approach [11], which is most common and followed by many language identification researchers as they are always highly correlated to the language. We select character n-grams (n = 2 to 5) as the word features in our experiments. This involved creating char n-gram profiles for every language label and then assigning 4 features to each word indicating the count of the words char n-grams in each language profile.

2) Length: Raw length of a word is used as opposed to the technique used in the previous work as average length of an English word was at least 1 alphabet more than an average word of a Hindi word.

3) Capitalization: Positional capitalization usually emphasizes on the importance of a word.

4) Number of capitalization: Another feature based on capitalization that we use is the number of alphabets capitalized in a word. We use this as a feature as sometimes in social media text random words might be fully capitalized to show it's importance.

5) Suffixes: The affix placed after the stem of the word is used as another feature. These can be indicative of a language as the English language has suffixes mostly not found in phonetic-typed Hindi such as 'ing', 'ed', 'ly.'

6) Presence of digits: Another feature that we used was presence of digits in a word.

Contextual:

7) POS tags: The assigned part of speech tag for each word is used as a feature. These are obtained through NLTK's POS tagger.

8) Next word: The immediate next and second next word is a feature.

9) Previous word: The immediate previous and second previous word is a feature.

10) Next POS Tag: The immediate next POS tag is a feature.

11) Previous POS Tag: The immediate previous POS tag is a feature.

Meanwhile, the features we rejected were:

1) Count Vectorizer/ Tf-Idf: The count vectorizer was not effective as the count of how many times a word occurs in a document does not help us identify a language.

2) Soundex or phonetic matching: The values returned by soundex for phonetic matching of English and phonetically typed Hindi weren't very different.

3) Readability scores: Scores such as Flesch scores weren't helpful to predict the language because it measures the complexity of a text which were always low due to the language changes.

4) Maximum repetition of char unigrams and bigrams: As the task is language identification on word level it is very difficult to find unigram or bigram occurrences of a character more than two or three times in a word.

IV.B Evaluations and Baseline

This problem is a multi-class classification one where the classes whose classification we are most concerned with are the 2 languages (EN and HI). The distribution of classes as seen in part III.B shows that the classes are not skewed. The EMT class is highly unrepresented but that is not a concern because emoticons do not need translation and can be removed by eliminating punctuations or special characters, but we retain them to understand how robust our training algorithms can be when faced with unfiltered social media text. Judging by the distribution we have chosen *accuracy, precision, and recall* as our evaluation metrics.

Our biggest motivation to work on this project was [2] where Barman, Das et. al worked with code mixed data for 3 languages; Hindi-English-Bengali. They ran an unsupervised dictionary-based model where their training data words labelled Hindi and Bengali were part of a dictionary along with the British National Corpus [12] and Semeval Twitter 2013 [13] for English. Then under Supervised Learning they implemented SVM and CRF both with and without contextual features the findings of which are shown below.

System	Precision (%)				Recall (%)				Accuracy
	EN	*BN*	*HI*	*UNIV*	*EN*	*BN*	*HI*	*UNIV*	(%)
Baseline (Dictionary)	92.67	90.73	80.64	99.67	92.28	94.63	43.47	94.99	93.64
SVM-GDLC	92.49	94.89	80.31	99.34	96.23	94.28	44.92	97.07	95.21
SVM-P_1N_1	93.51	95.56	83.18	99.42	96.63	95.23	55.94	96.95	95.52
CRF-GDC	94.77	94.88	91.86	99.34	95.65	96.22	55.65	97.73	95.76

V. Experiments and Results

1) Unsupervised dictionary learning: We tested a Naïve approach where every word was labelled a language based on its presence and frequency in a language dictionary. Since all Hindi words were phonetically typed and not in devnagri-script, there is no dictionary, yet which would directly help our purpose. So, the Hindi dictionary comprised of all words labelled 'HI' from the training set. For English, the initial dictionaries used for [2] (British National Corpus: BNC [12], and SemEval 2013 Twitter Corpus [13]) carried a lot of computational overhead which after careful consideration of time and resources we decided not to go ahead with. BNC is a 100-million-word corpus which needed to be parsed and cleaned. SemEval 2013 Twitter Corpus scraped a certain set of users whose tweet's contained words in non-English scripts as well which we had to parse out. Instead we opted for WordNet's inbuilt language checking feature with synsets

(synonym sets including original word) [14]. This returned all synsets with POS tags for each word but None if the word was not English, providing a simple check. But, WordNet is trained on a formal English dictionary. For 'EMT' and 'UN', we classified words that were not classified as 'EN' or 'HI' by checking for presence of punctuations for 'EMT' else classifying as 'UN.'

For the task of identifying words we need a dictionary that reflects social media word patterns. For this we scraped our own dictionary from specific Twitter accounts and avoided the noise previously present in the SemEval 2013 corpus [13]. The evaluations on test data were as follows:

Model	Precision (%)				Recall (%)				Accuracy (%)
	EN	HI	UN	EMT	EN	HI	UN	EMT	
WordNet	66	78	10	1	69	69	3	100	52.33
Twitter	83	83	6	1	58	91	3	99	56.66

As expected, the dictionary-based approached did not perform too well. However the Twitter based dictionary performed marginally better since the words in its corpus came from the same domain (Social media) as that of the train and test data.

2) Supervised classification Algorithms with non-contextual features: For this approach we used the following features:

We perform experiments with Naïve Bayes, SVM classifier (poly kernel, C=1.0, L=0.001) and Random Forests (batch_size = 100, max_depth = 10) for the features specified above. The results obtained on each of the classifiers is as shown in the table below:

Model	Precision (%)				Recall (%)				Accuracy (%)
	EN	HI	UN	EM	EN	HI	UN	EMT	
NB	84.6	81.5	97.6	0.0	81.0	87.8	90.9	0.0	87.36
SVM	83.4	83.5	97.7	42.3	83.7	85.8	91.4	37.9	86.05
RF	85.6	85.8	95.2	53.8	85.9	86.4	93.2	58.3	88.7

3) Supervised classification Algorithms with contextual features: Contextual clues can play a very important role in word-level language identification. As our goal is to apply contextual clues we first use Conditional Random Fields which takes history in account in predicting the optimal sequence of labels. We use a linear chain CRF(algorithm = 'lbfgs', c1 = 0.0183, c2 = 0.0256).

We also add contextual clues to our SVM classifier. We include the previous and next word as features in our SVM classifier. The SVM classifier is ran with these parameters C = 1.0 and L = 0.00.

LSTM's are expected to perform well for language identification since language text often has long term dependencies. We use LSTM in our case as it is a similar case of sequential language tagging task.

Model	Precision (%)				Recall (%)				Accuracy (%)
	EN	HI	UN	EM	EN	HI	UN	EMT	
CRF	91.4	90.6	96.6	53.3	91.5	92.0	93.7	26.7	92.10
SVM	79.8	88.0	97.6	30.8	89.9	80.3	90.7	21.1	87.32
LSTM	88.7	89.6	96.4	43.2	89.5	88.1	92.8	32.6	90.31

VI. Conclusion

In this project, we have used [2] as our baseline to implement an automatic language identification system with English-Hindi code-mixed data. Our experimental results lead us to conclude that contextual features, character n-gram features and probabilistic sequence modelling play a pivotal role in word-level language identification. The number of 'EMT' tokens are very low. Hence, there are not enough 'EMT' samples to train the data on which explains the low precision and recall values for the tag. With non-contextual features the best accuracy achieved is 88.7% using a Random Forest classifier. As for contextual clues we get the best accuracy of 92.10 while using Conditional Random Fields.

References

[1] Lesley Milroy and Pieter Muysken, editors. 1995. One speaker, two languages: Cross-disciplinary perspectives on code-switching. Cambridge University Press.

[2] Code Mixing: A Challenge for Language Identification in the Language of Social Media Utsab Barman, Amitava Das, Joachim Wagner and Jennifer Foster CNGL Centre for Global Intelligent Content, National Centre for Language Technology Proceedings of The First Workshop on Computational Approaches to Code Switching, pages 13–23, October 25, 2014, Doha, Qatar, 2014.

[3] E Mark Gold. 1967. Language identification in the limit. Information and control , 10(5):447–474.

[4] Aravind K. Joshi. 1982. Processing of sentences with intra-sentential code-switching. In J. Horeck´y, editor, Proceedings of the 9th conference on Computational linguistics - Volume 1 (COLING'82) , pages 145–150. Academia Praha, North-Holland Publishing Company.

[5] Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media , pages 1–10.

[6] David C. S. Li. 2000. Cantonese-English codeswitching research in Hong Kong: a Y2K review. World Englishes , 19(3):305–322.

[7] Taofik Hidayat. 2012. An analysis of code switching used by facebookers: a case study in a social network site. Student essay for the study programme "Pendidikan Bahasa Inggris" (English Education) at STKIP Siliwangi, Bandung, Indonesia

[8] Thamar Solorio and Yang Liu. 2008b. Part-of-speech tagging for English-Spanish code-switched text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing , pages 1051–1060. Association for Computational Linguistics.

[9] Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In Proceedings of ACL-08: HLT , pages 870–878, Columbus, Ohio. Association for Computational Linguistics.

[10] Dong Nguyen and A. Seza Do˘gru˘oz. 2013. Word level language identification in online multilingual communication. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), pages 857–862, Seattle, Washington, USA. Association for Computational Linguistics.

[11] William B. Cavnar and John M. Trenkle. 1994. Ngram-based text categorization. In Theo Pavlidis, editor, Proceedings of SDAIR-94,

Third Annual Symposium on Document Analysis and Information Retrieval, pages 161–175.

[12] British National Corpus, XML Edition. http://ota.ox.ac.uk/desc/2554

[13] Semeval 2013 Twitter Corpus. https://www.cs.york.ac.uk/semeval-2013/task2/data/uploads/datasets/readme.txt

[14] WordNet Lexical database for English. http://www.nltk.org/howto/wordnet.html

APPENDIX

Code: https://github.com/SiddheshAcharekar/hien-codemixed