Question Classification in Code Mixed Languages

Archit Rathore
Prabuddha Chakraborty

Indian Institute of Technology, Kanpur
Department of Computer Science and Engineering

Abstract. In this project we try to address the problem of question classification in Hindi-English code-mixed sentences. The unavailibility of parsers, pos-taggers and due to loose grammer rules and individual variations, the task can't be solved using conventional language processing pipelines. We also explore the word embeddings in such a language using an albeit smaller dataset to get these embeddings. We have also created a dataset of code-mixed questions annotated with their expected answer type and tested our method on this dataset.

Key words: natural language processing, question classification, code-mixing, Hindi-English

1 Introduction

Code-Mixing or Code-Switching is defined as the embedding of linguistic units such as phrases, words and morphemes of one language into an utterance of another language[1]. Code-mixing is prevalent amongst bilingual and multilingual individuals. A bilingual user usually code mixes for a variety of reason: to make up for lack of terms in the native langauge, substituting less frequent words of native language with those of other words. For example "visham paristhiti" would be substituted by "odd situation" .Following are some examples of code mixing in Hindi-English and Bangla-English:

- "Mein soch raha tha ki we should atleast once isko discuss kar lena chahiye"
- "Annual report mein jo old wala team structure decide kiya tha wahi rehne do."
- "DOTA ta install kor agee then suvradeep-er team-e dhuke jabi."

The above example makes it evident that code mixing still mostly follows the underlying semantic and grammatical structure of the dominant language, but this cannot be generalized due to presence of phrase level mixing. The task for processing code mixed questions becomes a significant challenge due to the unavailability of tools such as parsers, pos-taggers and other methods that exploit

[1] source: Wikipedia

the semantic structure of the language. Every person mixes languages in a different way which is hard to generalize. Thus we need a way to address these problems while performing language processing on a corpus that contains code mixed text.

2 Related Work

Question classification problem is well explored for English language and Xin Li *et al*[2] have achieved 92.5% accuracy in course grain answer type classification and 85% for fine grained answer type. They have employed language specific techniques such as POS tagging and head word detection to improve the classifiers. Successive to this, Zhang and Lee[6] proposed their variant of tree kernel for question classification.Work that deals with question classification task in code-mixed language is done by Raghavi *et al* in their work [5]. They have used translations and adjacency features based on the question word of the sentence. Among the unsupervised feature learning approaches, sentence modelling using deep convolutional neural networks proposed by Kalchbrenner *et al* [1] has shown great results for question classification.

3 Dataset

Question type	**# of questions**	**% of questions**
DESC	111	19.30
HUM	102	17.75
LOC	86	14.95
NUM	67	11.65
ENTY	124	21.56
ABBR	85	14.79

Table 1. Distribution of various question types

Number of questions	574
Number of course categories	6
Average length of questions	9
Average degree of code mixing	4

Table 2. Dataset statistics

A supervised question classification task requires a dataset containing questions annotated with corresponding answer types. For our task we required

questions that were posed by bilingual users so that these questions had sufficient amount of code mixing. There is no such dataset openly available for code-mixed English-Indic languages. We have therefore created our own dataset containing English-Hindi code-mixed questions annotated with their corresponding coarse grained and fine grained answer types. In order to keep our dataset consistent with standard answer type classification hierarchy we have converted 574 questions from Li and Roth's English Question dataset [2] to code-mixed English-Hindi question dataset annotated with corresponding answer types. The conversion of an English question to a code-mixed question was carried out in 3 steps:

- Rough translation of the English question to Hindi Question using Goslate API[2]
- Manually replace a few appropriate Hindi words with corresponding English words.
- Transliterating the remaining Hindi words in the sentence into Romanized script using our own transliteration script.

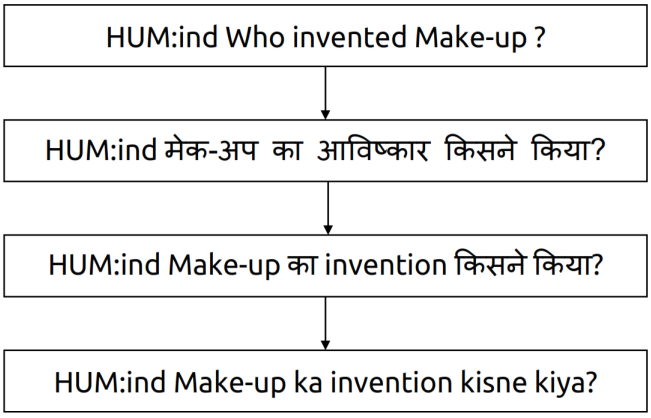

Fig. 1. Workflow for dataset creation

Also for training a Word2Vec[3] and a GloVe [4] model we have obtained 150,000 sentences of Hindi-English code-mixed personal messages from Hangout, Facebook and Whatsapp.

4 Methodology

In this project we have tried two different approaches for solving the question classification problem in Code-Mixed language. Before we start using the ques-

[2] https://pypi.python.org/pypi/goslate

tion dataset and the 150,000 chat sentences, we must first normalize them. The need for normalization arises because any Hindi word may be written in several different ways in Romanized script.After our normalization step is carried out, we guarantee that every Hindi word in our dataset will have an unique representation in Romanized script.

Algorithm 1 Normalization

```
procedure NORMALIZATION(Code–mixed corpus C)
    for each word w in the C do
        if w ∈ H_t then
            return w
        else if w ∈ E_t then
            return w
        else
            return T(H_h(T^{-1}(w))
```

where H_t is transliterated Hindi vocabulary, H_h is the devanangri Hindi vocabulary, E_t is the english vocabulary and $T : H_h \rightarrow H_t$ is the transliteration mapping.
Following is the description of the approaches we have tried.

4.1 Approach 1

- Obtain the vector representation of each questions from the normalized question dataset using bag of n-gram.
- Divide the Question dataset into training and test sets.
- Train an SVM and tuned using grid search over parameter space and a Logistic classifier using the Training Set.
- Verify the accuracy using the Test Set.

```
>>> model.most_similar('bhot')
[('jyada', 0.96732563825026041), ('zyaada', 0.93778812849866489), ('itna', 0.91608035380807096), ('waisa', 0.91299409862481473)]
>>> model.most_similar('galat')
[('achcha', 0.95868523267208094), ('inka', 0.94901687051595385), ('humara', 0.94789504218748499), ('theek', 0.94270970534028276)]
>>> model.most_similar('bhai')
[('saale', 0.929751305233396), ('ye', 0.92069538967575437), ('oye', 0.91229835767620793), ('yaar', 0.91171201300615545)]
```

Fig. 2. Most similar word as found by the GloVe vectors

4.2 Approach 2

- Train a GloVe and a Word2Vec model using the 150,000 normalized code-mixed chat sentences.
- Using the GloVe and the Word2Vec model we obtain the vector representation of each question sentences from the normalized question dataset.

– Based on a gaussian curve having the mean position on the question word (such as “What”,“When”, “kaise”), we have multiplied corresponding weights to the word vectors.Then the average of these weighted word vectors is taken as the sentence vector for a given question sentence.
– Divide the Question dataset into training and test sets.
– Train an SVM and a Logistic classifier using the Training Set.
– Verify the accuracy using the Test Set.

Note: We have computed the accuracy over 100 iterations in each iteration 10% questions were randomly picked for test set and 90% were kept for training.

5 Results

The accuracy and confusion matrices for the different approaches that we have tried are presented here:

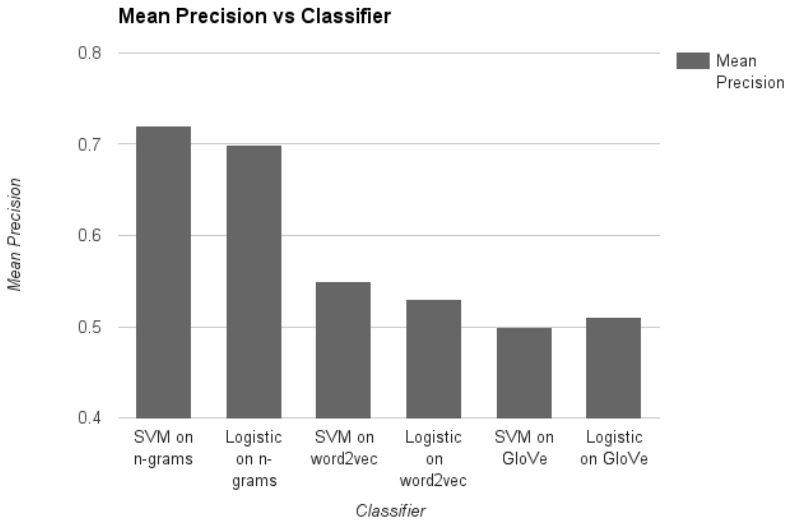

Fig. 3. Plot of classifier and their mean precision

5.1 Approach 1

– obtained 0.72 mean precision using 574 question data with 10% test to train ratio over 100 iterations using SVM on unigram features.
– obtained 0.69 mean precision using above same with logistic regression (maximum entropy) classifier on unigram features.

5.2 Approach 2

– obtained 51% mean precision using 574 questions with 10% test to train ratio over 100 iterations using SVM on 100 dimensional GloVe vectors
– obtained 55% mean precision using 574 questions with 10% test to train ratio over 100 iterations using SVM on 100 dimensional word2vec vectors

Method	Precision	Recall	f-1 score
SVM with bag of n-grams	**0.72**	**0.70**	**0.71**
Logistic classifier with bag of n-grams	0.69	0.69	0.69
SVM with word2vec vectors	0.55	0.54	0.54
Logistic classifier with word2vec vectors	0.53	0.53	0.52
SVM with GloVe vectors	0.50	0.50	0.50
Logistic classifier with GloVe vectors	0.51	0.52	0.51

Table 3. Classification benchmarks

6 Conclusion

From the results obtained, it is clear that our method surpasses the approach of Raghavi et al, mainly because of the reduction of noise by normalization. Also, even though our approach 2 did not give good results, we have observed that the word clusters formed by the word2vec and GloVe models when trained with code-mixed data is decent considering that the small data size. With dense enough data for training these models we expect them to work better.

7 Future Work

– We currently have the forward transliteration tool and are using the Google Transliterate API (deprecated) for back transliteration. This web-based API becomes a performance bottleneck. To make a complete efficient pipeline from this system we would need to create a back-transliteration tool.
– The word vector embedding were created from a small dataset of personal messages. We intend to mine a larger code-mixed dataset from social platforms and create code-mixed word embeddings which may improve the performance of these models and also allow us to use deep CNNs as well for sentence modelling.
– The transliteration tool is currently deterministic and single output. Transliterational variations can be incorporated into it so that it predicts transliterations over a probability space.
– Extend the question dataset with more number of questions and make it openly available for evaluations of code-mixed language models.

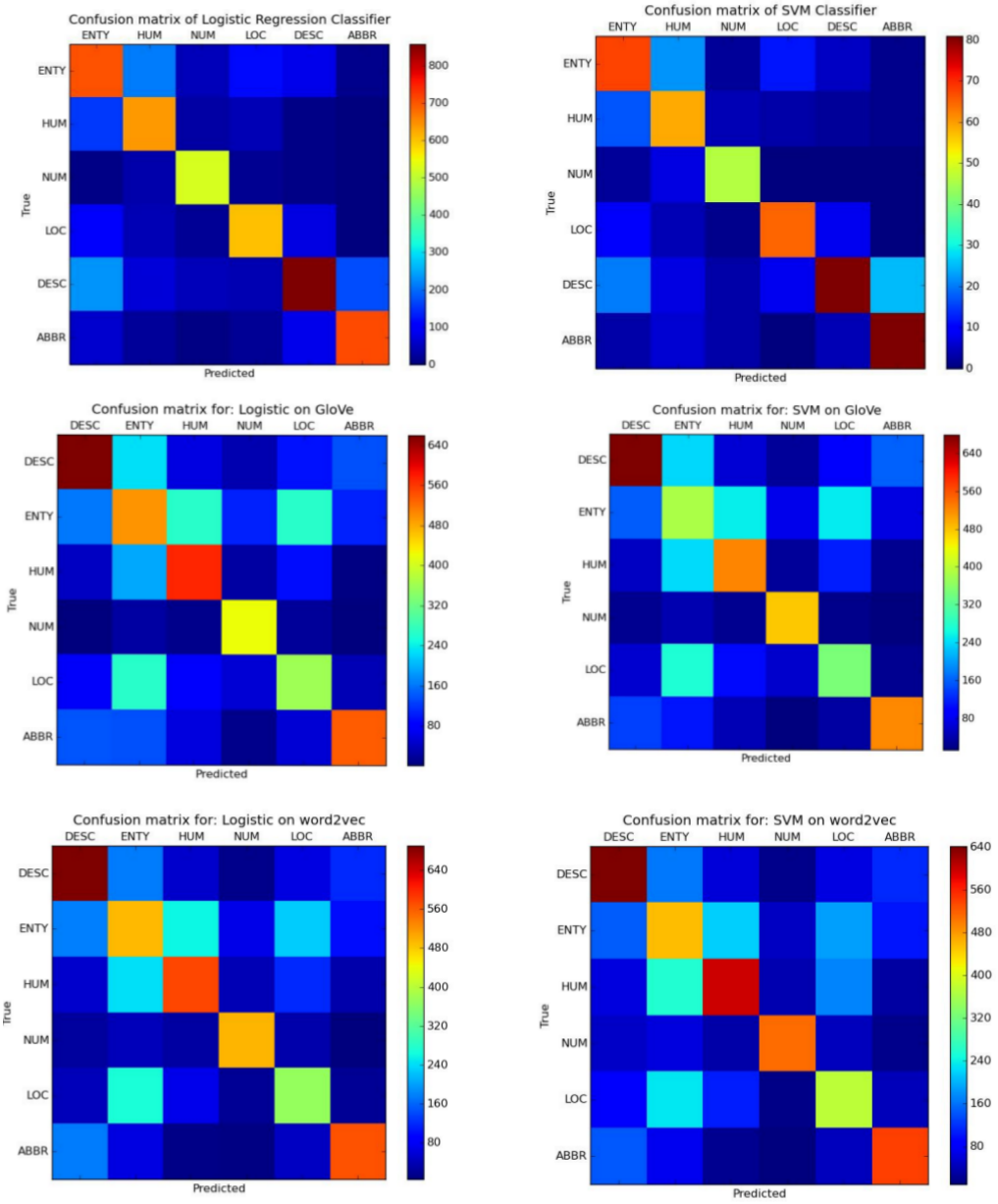

Fig. 4. Confusion matrices for various classifiers

References

1. Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.
2. Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.
3. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
4. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.
5. Khyathi Chandu Raghavi, Manoj Kumar Chinnakotla, and Manish Shrivastava. Answer ka type kya he?: Learning to classify questions in code-mixed language. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 853–858. International World Wide Web Conferences Steering Committee, 2015.
6. Dell Zhang and Wee Sun Lee. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM, 2003.