Word-level Language Identification Using Subword Embeddings for Code-mixed Bangla-English Social Media Data

Aparna Dutta
Brandeis University
aparnadutta@brandeis.edu

Abstract

This paper reports work on building a word-level language identification (LID) model for Bangla-English social media data using subword embeddings, with an ultimate goal of using this LID module as the first step in a modular part-of-speech (POS) tagger in future research. In this work, I report preliminary results of a word-level LID model that uses a single BiLSTM with subword embeddings trained on very limited code-mixed resources. As far as I know, there are no previous reported results of code-mixed Bangla-English subword embeddings being used for language identification. I also present a labeled resource for word-level language identification by correcting 85.7% of labels from an existing resource. The trained model was evaluated on a test set of 4,015 tokens compiled from the 2015 and 2016 ICON datasets, and achieved a test accuracy of 93.61%.

1 Introduction

Code-mixing refers to the phenomenon when people communicate using two or more languages interchangeably within the same phrase, and is widely-observed in areas with significant multilingual populations. India has 22 official native languages, but its usage of English also contributes to its linguistic diversity. English has widespread usage in India in both informal and official contexts, with it being the main language used in schools and for education. As such, bilingualism is very common in India, and people are used to speaking in a mix of English and other Indian languages.

Bangla is the second most-spoken native language in India and is frequently mixed with English and Hindi on social media, as code-mixing is particularly common in social media communication. Although Bangla and Hindi each have their own native scripts with accompanying digital keyboards, speakers often switch between multiple languages within one social media post, and it is more convenient to transliterate into Roman characters than to switch back and forth between keyboards.

The automatic understanding of social media text has become a key research area in recent years, and being able to identify the language for individual words in code-mixed text is considered a prerequisite for more complex downstream NLP tasks such as part-of-speech (POS) tagging, named entity recognition, and sentiment analysis. In many cases, language identification can allow us to reuse existing monolingual models that have already been trained rather than resorting to re-training models for each new code-mixed language pair. For this reason, one of the foundational problems for NLP on code-mixed data is language identification at the word level. Transfer learning is another approach to addressing low-resource NLP, but is not explored in the current work.

This paper's main contribution is providing baseline results for a Bangla-English word-level LID model with subword embeddings, using very limited data and no external resources. Although subword embeddings have been used for language identification with other code-mixed language pairs, at the time of this writing I have not been able to find reported results of subword embeddings applied to Bangla-English code-mixing.

The next section of this paper discusses some specific challenges of language identification with Bangla-English social media data. Section 3 then describes the dataset that is used for training and evaluating the model. The methodology used for word-level language identification is discussed in Section 4, while the results are reported and discussed in section 5. Finally, section 6 wraps up the overall findings of the paper and suggests a future direction for this research.

Reproducibility: Source code and data are available at: https://github.com/aparnadutta/code-mixed-lid.

2 Challenges of Bangla-English Language Identification

This section introduces some of the challenges that were encountered during the development of the system, specifically with respect to the data.

Transliteration is one of the main challenges of code-mixed Bangla-English data. Although there are formalized systems for transliterating Bangla into Roman or Latin script such as IAST (International Alphabet of Sanskrit Transliteration) and ITRANS (Indian languages Transliteration), these have not been widely adopted by social media users. Additionally, conversion from Bangla into Roman script is not one-to-one, since Bangla has more sounds than English, and even traditional Bangla orthography does not accurately reflect pronunciation due to its strict adherance to the Sanskrit writing system. All of these issues result in the same word often being transliterated in multiple different way by social media users. For example, the Bangla word *shaathey* meaning 'together', is also transliterated as *sathei* and *shatey* within the data.

Language ambiguous words are also common in the data. There are many words between Bangla and English that appear orthographically identical, such as *to* meaning 'so', *choke* meaning 'eyes', and *dish* meaning 'give' in Bangla. In these cases, the text of the word itself cannot be used to identify the language of the token, as the word would be broken into the same subwords and mapped to the same embedding spaces. Instead, an accurate language classification would rely entirely on the context of the surrounding words and the grammatical structure of the sentence.

One final difficulty is caused by **abbreviations and misspellings**. Some examples of this are *ka6e* used for *kachey* (meaning 'near'), *hbe* for *hobe* (meaning 'it will happen'), and *j* for *jey* (meaning 'that'). The shortening of words in this way can make it extremely difficult to figure out the language of a token, especially when there are English abbreviations that may have the same form. Similarly, words on social media are often misspelled, both accidentally and on purpose for exaggeration or effect, in cases like 'plssssssss' and *vishoooooooooooooon* meaning 'very'.

Overall, transliteration is noisy, as social media users use shorter length words and incorporate more abbreviations than standard text. In addition to the challenges mentioned, we also exhibit These issues can all lead to OOV errors (when the system sees a new word it hasn't previously encountered) and make it more difficult to complete language identification. In the next section, we will go into more detail on the dataset used for training and testing the model, along with any pre-processing that has been done to address the challenges mentioned here.

3 Dataset

The dataset used for training, development, and evaluation of the model has been compiled from the 2015 and 2016 ICON shared tasks. Although both corpora consist primarily of Bangla-English code-mixing, there are also Hindi words present in the data. For both shared tasks, the training data was publicly released online[1] but the validation and testing sets were not available.

The 2016 ICON data consists of English-Bangla code-mixed data that was scraped from Facebook, Twitter, and WhatsApp, and was manually and automatically tagged at the word level. The 2015 data also consists of social media data, but is not broken into separate files based on source. For the current research, only the word-level language tags are used.

3.1 Data Preprocessing

The language tags in the 2016 WhatsApp dataset were manually corrected for the purposes of this research. A large majority of the tokens that were clearly Bangla or English were originally tagged as *undef* or *univ* in this dataset for unknown reasons. As such, 85.7% of the original language tags were manually corrected by a native speaker of Bangla and English, with a background in linguistics and annotation. The tags in the Facebook and Twitter datasets were also examined but did not appear to have these issues. The corrected dataset is released with this project for future usage.

There was also minimal preprocessing of the data to address the challenges described in the previous section. All words were lowercased, and words with >2 consecutive identical characters were normalized to 2 consecutive characters (Mandal et al. 2018). Finally, all labels that included word-level mixing, such as *en+bn_suffix* or *be+en_suffix* were collapsed down to a single *mixed* label.

[1] http://www.amitavadas.com/Code-Mixing.html

Source	#	Language Label							
	tokens	bn	en	univ	ne	acro	hi	mixed	undef
Facebook 2016	7,392	48.55	29.76	17.06	2.91	0.54	1.16	0.00	0.01
Twitter 2016	3,680	48.72	26.60	19.84	2.99	0.27	0.68	0.22	0.68
WhatsApp 2016	3,510	52.99	34.25	10.11	2.28	0.00	0.14	0.09	0.14
ICON 2015	24,547	33.94	40.60	19.03	2.80	2.51	0.80	0.19	0.12
Total	39,129	39.80	36.67	17.94	2.79	1.70	0.80	0.15	0.16

Table 1: *Token-level language distribution from all sources (%). The language tags are Bangla, English, Universal (punctuation and numbers), Named Entity, Acronym, Hindi, Mixed (word in one language and suffix in another), and Undefined (things that can't be classified, or non-Unicode).*

Source	Number of		CMI		Code-mixed
	tokens	utterances	all	mixed	(%)
Facebook 2016	7,392	147	31.63	31.63	100.00
Twitter 2016	3,680	172	33.50	33.50	100.00
WhatsApp 2016	3,510	304	28.17	29.63	95.07
ICON 2015	24,547	2,828	4.88	25.14	19.41
Total	39,129	3,451	9.50	28.33	33.53

Table 2: *Average Code-Mixing Index (CMI) for all data sources*

This was done because there were too few examples of word-level mixing in the data to enable accurate classification.

Since the datasets came from a shared ICON task, they were already formatted in a standard way, with one word per line, followed by a language tag and a POS tag. A line of space separated each instance within the data. As such, tokenization and sentence-boundary detection were not issues that specifically came up during my work, since the dataset came pre-tokenized.

However, I did originally struggle with the decision of sentence-splitting, as I needed to decide whether I wanted to follow the instance-level splits defined in the dataset, or if I alternatively wanted to try and detect actual sentence boundaries. I eventually did not end up pursuing sentence-boundary splitting, since this is quite a difficult task to complete with social media data, and there often is no correct answer as to where a sentence boundary lies.

3.2 Language Tag Breakdown

Table 1 shows the token-level distribution of languages from each data source. The Facebook, Twitter and WhatsApp sources are all from the 2016 data, while the 2015 data is grouped together into one source. The 2015 data makes up over half of the tokens in the overall dataset, and is also the only source that is majority English rather than Bangla. For a more in-depth comparison of the various sources, the code-mixing index of the dataset is discussed next.

3.3 Code-mixing Index

CMI is a metric that was introduced by Das and Gambäck (2014) to measure the amount of code-mixing present in a corpus. This is a useful metric because it allows us to better compare results from studies using different code-mixed datasets. Since some corpora may contain monolingual sentences as well as code-mixed sentences, we can use the CMI of a test dataset to evaluate the performance of the tool on different real-world cases. CMI is calculated for each utterance using the following formula:

$$CMI = \begin{cases} 100 \times [1 - \frac{max\{w_i\}}{n-u}] & : n > u \\ 0 & : n = u \end{cases}$$

Where w_i is the words tagged with a language tag such as: $bn, en, hi, mixed$, while excluding non-language tags such as $univ, acro, ne, undef$. Therefore, $max_w i$ refers to the count of the most common language tag in the post. So, a monolingual utterance of Bangla would have a CMI of 0, since the number of Bangla tokens would be equal to the number of overall language tokens minus the number of non-language tokens. Similarly, a post with only non-language tokens would also have a CMI of 0.

The CMI of each data source is presented here in Table 3. The 'all' column describes the average utterance-level CMI for all utterances in the dataset, while 'mixed' refers to the average utterance-level CMI for utterances that have any code-mixing at all. This provides a better picture of the amount of code-mixing at the utterance-level. Finally, the last column shows the percentage of overall utterances from each dataset that are at all code-mixed, meaning utterances that have a non-zero CMI. We can see that the 2015 dataset exhibits far less code-mixing than the 2016 sources, which are almost entirely code-mixed.

Since the datasets are significantly different from one another in terms of code-mixing and the majority of the data comes from ICON 2015, the decision was made to shuffle the full dataset and then divide it into train, validation, and test splits using a 60%: 20%: 20% ratio. This allowed us to have a final dataset that is not entirely code-mixed or monolingual. However, this also means that there is a risk of overfitting since the test data may be too similar to the training and validation data. In order to address this concern, we note the overlap in tokens between the validation and test sets, as per Barman et al. (2014). 33.47% of the Bangla tokens in the test set were also present in the validation set, while 40.73% of the English tokens in the test set were also present in the validation set.

4 System Design

This section describes the architecture of the model built for Bangla-English word-level language identification. The task of language identification in code-mixed text can be defined as a joint sequence-labeling and classification task. The language of each word in a given utterance must be individually labeled, but incorporating the context of surrounding words is also crucially important to account for challenges like orthographic similarity between words of different languages, and out-of-vocabulary tokens.

To address these challenges, a BiLSTM model is used following Joshi and Joshi's (2020) experimental setup, with certain modifications made to account for a smaller dataset. In addition, while their task was a binary classfication between Hindi and English, ours is multiclass classification, since there are more than two language tags. As such, a softmax activation is used rather than a sigmoid activation in the current work. A general architecture of the full model is provided in Figure 1.

This shows the sentence *toder college ta* meaning '*your college*' being split into subwords, with each subword embedding passing through the model to generate the final output. The model built as a part of this project is a single-layer BiLSTM that uses unigram-based subword embeddings as the input representation.

4.1 Vocab Generation Using SentencePiece

The first step in the task involves generating an embedding vocabulary for the text input. Following Joshi and Joshi (2020), the subword model is trained using Google SentencePiece (Kudo and Richardson 2018)[2]. All of the unlabeled training data is used to train the SentencePiece model, which is then able to split each word in a sentence into smaller subwords.

The subword vocab size used by Joshi and Joshi (2020) was 12k tokens, but due to the smaller amount of available data, a vocab size of 3k tokens was used in the current work. Additionally, the final model is tested using both a unigram and BPE based subword tokenizer.

Figuring out how to use SentencePiece was one area of the project that required more work than initially expected, as there was very limited documentation available for the PyTorch package of SentencePiece, and the input to model training was unclear. In the end, I realized that I needed untagged data for training the SentencePiece model, and I had to fiddle for quite a bit to convert my original data into shuffle train, development and test splits, before rewriting the training data into an unlabeled file for SentencePiece training.

The next step after generating the subword vocabulary is to complete the actual language identification task using the BiLSTM, as is described in the next section.

4.2 Sequence Labeling Using BiLSTM

This section describes the step-by-step sequence labeling task for word-level language identification, as illustrated in Figure 1. This represents the bulk of the code produced as a part of this project, and details all of the steps involved in outputting the final tagged sentence at the word level.

[2] https://github.com/google/sentencepiece

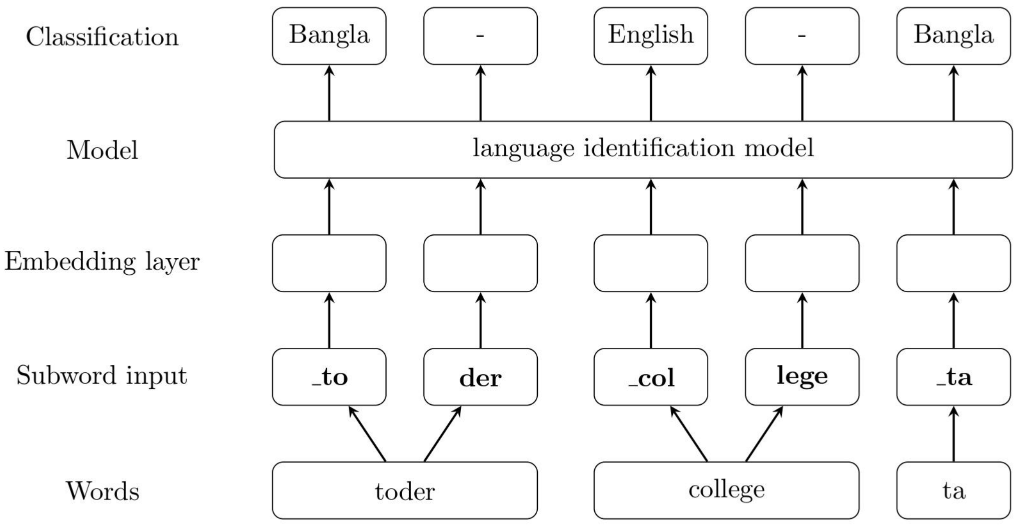

Figure 1: *Outline of subword embedding language identification module using a single layer BiLSTM*

1. The input to the model is a single social media post (or utterance). Each word in the utterance is broken down into subwords, and this is flattened into a single list of subwords.
2. Each subword is mapped to an index, which is used to retrieve the embedding for that subword. The subword embeddings are initialized randomly and trained over time.
3. With each subword representing one time-step, subword embeddings pass through the BiLSTM recurrent unit one-by-one. The first subword of each token is assigned the real language label while the remaining subwords are assigned a dummy label. A mask is also created to keep track of the indices of the first subword of each token.
4. The hidden state of the recurrent unit after reading all of the subwords is used as input to a dense layer, which outputs features of shape (S, V), with S being the number of subwords in the utterance, and V being the number of possible language tags.
5. A softmax activation function is applied to the resulting scores which results in a probability distribution over all possible language labels for each subword.
6. During training, the dummy label is masked from cross-entropy loss calculations, while predictions for non-initial subwords are also masked. This allows us to output language predictions for only the first subword in each token, while the subsequent subwords still contribute to their parent token through the hidden layer.
7. The argmax of the masked scores is taken for each word, resulting in a single language prediction for each original word in the sentence.

The full code implementation of the steps described above can be found here[3]

5 Evaluation

The system was evaluated using precision, recall, and F1-score, on 20% of the data set aside before training. The hyperparameters selected follow Joshi and Joshi's (2020) work on Hindi-English code-mixed data as closely as possible, with the only change being a reduced subword vocab size due to the lack of data. The results on the validation set were used to determine the optimal number of epochs for running.

The recurrent unit has a hidden dimension of 300. The subword embedding dimension is 300, and is passed through the recurrent unit after a dropout of 0.4. The output of the recurrent unit is passed through one dense layer, with the final output dimension being equal to the number of language labels. An AdamW optimizer is used and the loss function is cross-entropy with the dummy label index ignored. The model is trained for 40 epochs with a batch size of 64. The same hyperparameters

[3]https://github.com/aparnadutta/code-mixed-lid/tree/main/src

Subword Model	Metric (%)	bn	en	univ	ne	hi	acro	mixed	undef
Unigram	Precision	92.99	93.27	98.07	61.17	79.12	48.81	25.00	37.50
	Recall	94.58	93.83	98.37	45.63	60.00	64.06	18.18	75.00
	F1-Score	**93.78**	**93.56**	98.22	52.27	68.25	55.41	21.05	50.00
BPE	Precision	91.60	93.89	98.21	62.31	59.83	56.72	27.27	100.00
	Recall	94.58	92.73	97.62	49.21	58.33	59.38	27.27	75.00
	F1-Score	**93.07**	**93.31**	97.91	54.99	59.07	58.02	27.27	85.71

Table 3: *Metrics on test data with unigram and BPE-based subword encodings (%)*

are used when evaluating both unigram and BPE encoding on the test set.

5.1 Results and Discussion

To understand the effect of both subword encoding models, the final trained model tuned on the validation set was evaluated on blind test set, 20% of the overall dataset. The results on the test set for both the unigram and BPE models are provided in Table 2 above.

We can see that our model performed very well on the test dataset and comparably to past results. The unigram model performed best, achieving F1-scores of 93.22% and 93.56% on Bangla and English respectively. We also see that the unigram and BPE-based encodings performed very similarly to one another. Due to the nature of the model training and encoding, it is not possible to say whether or not this difference is statistically significant. This is because re-training the SentencePiece model with the same encoding multiple times results in a slightly different vocabulary each time.

Looking back to past research, Mandal et al.'s (2018) best ensemble model achieved an F1-score of 92.35%, but it is difficult to compare the present results to other works since they have been tested on different datasets that have different amounts of code-mixing present. Regardless, the F1-scores exhibited by the unigram model are impressive within the landscape, and it would be worthwhile gain access to an existing test set to re-test the fully trained model.

6 Conclusion

The first section of this paper introduced code-mixed social media data and the various approaches that have been taken to it in the past. Then, the importance of word-level language identification was discussed with a description of various input representations, ending with Joshi & Joshi's (2020) findings that subword embeddings perform best for language identification on Hindi-English code mixed data.

After this, section 3 described the features of the ICON 2016 dataset that are used for building the LID module and evaluation of the entire model. In section 4, the overall model architecture is described, followed by the results of each experiment in section 5. The major findings were that subword embeddings appear to perform very well on Bangla-English with F1-scores of 93.22% and 93.56% for Bangla and English respectively. While it is difficult to compare directly with previous studies due to differences in code-mixed corpora and test sets, these results are very promising given the simplicity of the current model.

In the future, I would like to explore how to better handle mixed-language labels. As mentioned, here I collapsed all mixed-language labels into one category due to the small number of them present in the data. However, with a a larger dataset, it would be interesting to experiment with collapsing word-level mixes into one of the two categories present in the mixing, or to keep them as their own category. Another future direction of the work is exploring other strategies for combining subword predictions for each word. In the current work, the approach used by Joshi and Joshi (2020) is used, in which non-initial subwords are assigned a dummy label which is later masked out. One approach that could be explored in the future is assigning the parent label to all subwords, and utilizing masking in a different way to combine all subwords back into one parent label. Finally, the original goal of this LID module was to test subword embeddings applied to Bangla-English. Given the success of the model, I would like to experiment with adding a CRF for the last step of decoding, and eventually integrate the module into a pipeline POS tagger.

References

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code Mixing: A Challenge for Language Identification in the Language of Social Media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 13–23, Doha, Qatar, October. Association for Computational Linguistics.

S. Nagesh Bhattu, N. Satya Krishna, B. Shiva Krishna, and Vadlamani Ravi. 2017. Language Identification in Mixed Script. In *Proceedings of the 9th Annual Meeting of the Forum for Information Retrieval Evaluation*, pages 14–20, New York, NY, USA, December. Association for Computing Machinery.

Amitava Das and Björn Gambäck. 2013. Code-mixing in social media text: the last language identification frontier?. In *Traitement Automatique des Langues*, 41–64.

Anupam Jamatia, Amitava Das, and Björn Gambäck. 2019. Deep Learning-Based Language Identification in English-Hindi-Bengali Code-Mixed Social Media Corpora. In *Journal of Intelligent Systems*, 28(3):399–408, July.

Ramchandra Joshi and Raviraj Joshi. 2020. Evaluating Input Representation for Language Identification in Hindi-English Code Mixed Text. arXiv:2011.11263 [cs], November.

Soumil Mandal, Sourya Dipta Das, and Dipankar Das. 2018. Language Identification of Bengali-English Code-Mixed data using Character & Phonetic based LSTM Models. arXiv:1803.03859 [cs], June.