Proceedings of the 1st Workshop on Dataset Creation for Lower-Resourced Languages (DCLRL) @LREC2022, pages 76–82
Marseille, 24 June 2022

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

Aparna Dutta
Brandeis University
415 South St, Waltham, MA 02453, United States
aparnadutta@brandeis.edu

Abstract

This paper reports work on building a word-level language identification (LID) model for code-mixed 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. This work reports preliminary results of a word-level LID model that uses a single bidirectional LSTM with subword embeddings trained on very limited code-mixed resources. At the time of writing, there are no previous reported results available in which subword embeddings are used for language identification with the Bangla-English code-mixed language pair. As part of the current work, a labeled resource for word-level language identification is also presented, by correcting 85.7% of labels from the 2016 ICON Whatsapp Bangla-English dataset (ICON, 2016). 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%.

Keywords: Bangla, Bengali, code-mixing, code-switching, language identification, subword embeddings

1. Introduction

Code-mixing refers to the phenomenon to communication 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 educational contexts. Bilingualism is very common in India and people are accustomed 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 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 the reuse of existing monolingual models rather than re-training models for each new code-mixed language pair. For this reason, one of the foundational problems for NLP with code-mixed data is language identification at the word level.

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, there are no reported results of subword embeddings applied to Bangla-English code-mixing to the best of our knowledge.

The next section discusses specific challenges of language identification with Bangla-English social media data. Section 3 then describes the dataset that was used for training and evaluating the model. The methodology used for word-level language identification is discussed in section 4, and 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. Related Work

This section provides an overview of past research into word-level LID for code-mixed data, focusing on the Bangla-English code-mixed pair.

Research into language identification for code-mixed data first began with Solorio and Liu (2008)'s initial research into predicting code-switching points. Solorio and Liu's work used a spoken Spanish-English corpus and tested both Naive Bayes and VF1 (Value Feature Interval) models. They found that Naive Bayes performed best, gaining an F1-score of 28%.

Das and Gambäck (2014) introduced the first Indian social media text corpus for the task of language-identification, and achieved an F1-score of 76.37% on Bangla-English, using an SVM trained with ngrams with weights, dictionary, minimum edit distance, and

a 7-word context window. In this research, Das and Gambäck also introduce the code-mixing index (CMI) to evaluate the level of code-mixing across corpora. This is a metric used to quantify the amount of code-mixing that is present in a corpus, and can allow researchers to better compare results using different corpora.

One of the most well-known approaches to word-level language identification for code-mixed data was conducted by Sristy et al. (2017). Their best-performing Bangla-English model achieved an F1-score of 86.15% using Naive Bayes EM with CRF (Lafferty et al., 2001).

Barman et al. (2014) reported one of the highest metrics on the task of word-level language identification, with an accuracy 95.14% using a CRF model. However, they excluded named entities and word-level mixing from their training and test sets. They also noted that there was a substantial token-level overlap between their cross-validation and test sets, such that their baseline dictionary approach already achieves an accuracy of 93.65%. This makes it difficult to compare these results to other studies that include these more difficult to handle labels in their testing sets.

More recently, advanced deep-learning approaches have proven to be very successful at the task of word-level LID. Jamatia et al. (2019) compared the performance of CRFs to LSTMs (long short term memory) and BiLSTMs (bidirectional long short term memory) (Huang et al., 2015), both deep-learning based approaches. Their research utilized a Bangla-English corpus, and is most similar to the dataset being used in the current work. They found that the LSTM and BiLSTM significantly outperformed the baseline CRF (81.57% F1 and 83.93% accuracy) on the Bangla-English data, with a slight improvement between the LSTM (88.19% F1 and 88.27% accuracy) and BiLSTM (88.23% F1 and 87.57% accuracy) approaches.

LSTMs and BiLSTMs have shown to be successful at this task because they are able to capture contextual relationships and long-distance dependencies between words. The next section details various input representations that have been explored for LSTMs, along with their benefits and drawbacks.

2.1. Input Representations for LSTMs

Word-level, character level, concatenated word and character, and more recently subword embeddings have been used as input representations for LSTMs. Each of these embedding levels indicates the granularity that is be used to map a given sentence into a group of embeddings.

2.1.1. Word embeddings

Word embeddings are considered the default input representations for text processing due to their logical nature, but are more likely to encounter out-of-vocabulary (OOV) issues when faced with unknown words, or noisy or misspelled data. These are dense representations of words that exhibit similarity between words with similar meanings or contexts. When an unknown word is encountered in a word-based representation, it is by default mapped to the embedding for an unknown token, leaving the neural network to rely on only the contextual information from surrounding words. As mentioned earlier, Jamatia et al. (2019) used word embeddings with LSTMs on Bangla-English data and trained their embeddings on both code-mixed social media data and monolingual Wikipedia data, achieving an F1-score of 88.23%.

2.1.2. Character embeddings

An alternate input representation that is used to address the OOV problem present with word embeddings is the character-level embedding. Character embeddings are generated on a character-by-character basis, and then pooled using a CNN (convolutional neural network) (Kim, 2014) to achieve word-level representations. The pooling layer prevents misspellings and abbreviations from causing OOV problems. Pooled character embeddings can either be used alone or concatenated to the original word embeddings to capture additional context and make up for OOV tokens and noisy data.

Mandal et al. (2018) used character embeddings along with phonetic-based character embeddings with an LSTM to build an ensemble model for language tagging. Their phonetic model alone achieved the best results, with an F1-score of 91.71% on Bangla-English code-mixing, but they also explored an ensemble threshold model that took the mean of the outputs of both models and used a brute force technique to select between the two. With this ensemble model they achieved an F1-score of 92.35%. However, they discarded all words with lengths less than 3, numeric characters, or word-level mixing from their training and test datasets. These adjustments make the task less transferable to data in the wild, and less comparable to the current study since this noisier data is included in the present study. Additionally, the ensemble model that performed best requires two different models to be trained, which is complex and computationally expensive.

2.1.3. Subword embeddings

The final input representation explored here is subword embeddings. A subword is a unit smaller than a word but larger than a character. Subwords can be generated through multiple approaches including unigram and byte-pair encoding (BPE), but in general, result in a vocabulary consisting of character groupings that appear most commonly in the data. This type of representation falls in between word-level and character-level embedding and can be especially useful for code-mixed data because unknown words will be broken into smaller subwords until they can be recognized by the vocabulary, while more common words and affixes can be recognized and build up individual embedding rep-

resentations over time.

More recently, Joshi and Joshi (2021) evaluated word, character, and subword level representations for language identification in Hindi-English code-mixed data. They experimented with CNN, multi-CNN, BiLSTM, CNN+BiLSTM, and character CNN+BiLSTM models, each with word-level and subword-level input. They found that word embeddings performed worst, with word+character embeddings performing slightly better. Across all combinations, the plain BiLSTM model with the subword input representation performed best, with F1-scores of 95.64% and 92.60% for English and Hindi respectively.

By comparing all input representations, Joshi and Joshi showed that subword embeddings work remarkably well for code-mixed social media data by vastly reducing the number of OOV tokens, because the vocabulary is specifically broken into the most common chunks. While their results show the success of subword embeddings on the task of language identification with Hindi-English data, this is an area yet to be explored within the Bangla-English code-mixing literature. With this motivation for the current task, the next section will describe some particular challenges of dealing with code-mixed Bangla-English data.

3. 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 adherence to the Sanskrit writing system. All of these issues result in the same word often being transliterated in multiple different ways 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, a more accurate language classification would rely wholly 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 'plsssssss' and *vishoooon* meaning 'very'.

Overall, transliteration is noisy, as social media users use shorter length words and incorporate more abbreviations than are present in standard text. 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. The next section goes into more detail on the dataset used for training and testing the model, along with any preprocessing that has been done to address the challenges mentioned here.

4. Dataset

The dataset used for training, development, and evaluation of the model was 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 made publicly 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.

4.1. Data Correction and Pre-processing

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 mis-labeled as *undef* or *univ* in this dataset. 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.

The data was also minimally pre-processed 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, la-

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

bels for words that included word-level mixing such as *en+bn_suffix* or *be+en_suffix* were collapsed into a single *mixed* label because there were too few examples of word-level mixing in the data to enable accurate classification.

4.2. Language Tag Breakdown

Table 1 shows the token-level distribution of languages from each data source. The Facebook, Twitter and WhatsApp sources are come 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.

4.3. 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 researchers to understand when results are comparable from research using different code-mixed datasets. Since some corpora may contain monolingual sentences as well as code-mixed sentences, the CMI of a test dataset can also be used 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. 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, the overlap in tokens between the validation and test sets is reported here, 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. This is significantly less overlap than was reported by Barman et al., and is adequate for the current purposes.

5. 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 (2021)'s experimental setup, with modifications made to account for a smaller dataset with more labels. The dataset used by Joshi and Joshi contained only Hindi and English language labels and was a binary task, while the current dataset contains at least three languages (Bangla, English, and Hindi) as well as mixed language labels, making the current task a problem of multiclass classification. To address the multiclass problem, a softmax activation is used rather than a sigmoid activation in the current work.

Figure 1 shows a general architecture of the full model with the sentence *toder college ta* meaning '*your college*' being split into subwords. Each subword embedding passes through the model to generate the final output. The model is a single-layer BiLSTM that uses unigram-based subword embeddings as the input representation.

5.1. Vocab Generation Using SentencePiece

The first step in the task involves generating an embedding vocabulary for the text input. Following Joshi and Joshi (2021), 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 (2021) was 12k tokens, but due to the smaller amount of avail-

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

Source	# tokens	Language Label							
		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

able data for the current work, a vocab size of 3k tokens was chosen. The final model is tested using both a unigram and BPE based subword tokenizer.
The next step after generating the subword vocabulary is to complete the language identification task using the BiLSTM, as is described in the next section.

5.2. 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 details all of the steps involved in outputting the final tagged sentence at the word level.

1. Model input is a single social media post (or utterance). The full utterance is segmented into a flat list of subwords.
2. Each subword is mapped to an index, which is later used to retrieve embeddings. Embeddings are initialized randomly and trained over time.
3. Each subword embedding (representing one timestep) passes through the BiLSTM recurrent unit. The first subword of each token is assigned the real language label while the remaining subwords are assigned a dummy label. Masks are created to track 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, dummy labels for non-initial subwords are masked from cross-entropy loss calculations. Predictions are generated for only the first subword in each token. This is accomplished by predicting a label for each subword, and masking the non-initial subwords so that the length of the final predictions made is equal to the number of original tokens in the utterance.
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 implementation of the steps described above can be found here[3]

6. 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 (2021)'s 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 (Srivastava et al., 2014) 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 (Loshchilov and Hutter, 2017) is used and the loss function is cross-entropy with the dummy label index ignored. The model is trained for

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

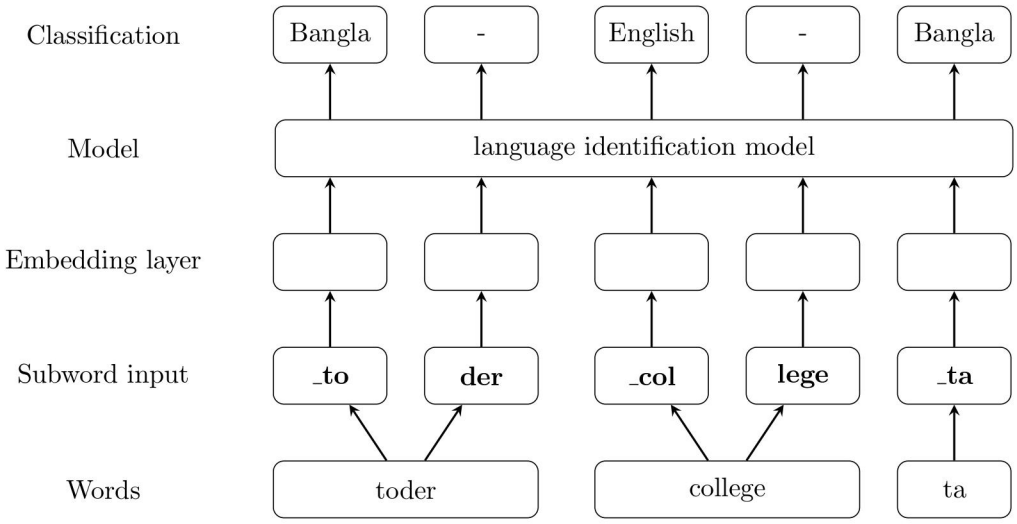

Figure 1: Outline of subword embedding language identification module using a single layer BiLSTM

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 (%)

40 epochs with a batch size of 64. The same hyper-parameters are used when evaluating both unigram and BPE encoding on the test set.

6.1. Results and Discussion

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

The performance achieved by the model on the test set is comparable to that of previously best-performing models. The unigram model performed best, achieving F1-scores of 93.22% and 93.56% on Bangla and English respectively. 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. (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 varying amounts of code-mixing. Regardless, the F1-scores exhibited by the unigram model are very good within the landscape, and it would be worthwhile gain access to an existing test set to re-test the fully trained model.

7. Ethical Considerations and Broader Impact

The main impact of this work is that of providing a large corrected dataset of code-mixed Bangla-English data. The usage of an incorrectly labeled dataset can in many ways hinder the progress of research for lower-resourced languages. This corrected data will allow further research into Bangla-English code-mixing, and will also enable us as a research community to understand a wider variety of people through their language use.

8. 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 and Joshi (2021) findings that subword embeddings are the best-performing input representa-

tion for language identification on Hindi-English code mixed data.

After this, section 3 described the features of the ICON 2016 dataset that were used for building the LID module and evaluation of the entire model. In section 4, the overall model architecture was described, followed by the results of each experiment in section 5. The major findings were that subword embeddings 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, it would be worthwhile to explore how mixed-language labels can be better handled. As mentioned, in the current research all mixed-language labels are collapsed into one category due to the small number present in the data. However, with 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, Joshi and Joshi (2021)'s approach of assigning a dummy label to and masking out non-initial subwords is used. 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.

9. Bibliographical References

Barman, U., Das, A., Wagner, J., and Foster, J. (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.

Das, A. and Gambäck, B. (2013). Code-mixing in social media text. the last language identification frontier? *Trait. Autom. des Langues*, 54:41–64.

Das, A. and Gambäck, B. (2014). Identifying languages at the word level in code-mixed Indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387, Goa, India, December. NLP Association of India.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

ICON. (2016). Icon 2016: Iit (bhu), varanasi ltrc, iiit, hyderabad.

Jamatia, A., Das, A., and Gambäck, B. (2019). Deep learning-based language identification in english-hindi-bengali code-mixed social media corpora. *Journal of Intelligent Systems*, 28:399 – 408.

Joshi, R. and Joshi, R. (2021). Evaluating input representation for language identification in hindi-english code mixed text. In *Lecture Notes in Electrical Engineering*, pages 795–802. Springer Singapore, nov.

Kim, Y. (2014). Convolutional neural networks for sentence classification.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization.

Mandal, S., Das, S. D., and Das, D. (2018). Language identification of bengali-english code-mixed data using character & phonetic based LSTM models. *CoRR*, abs/1803.03859.

Solorio, T. and Liu, Y. (2008). Learning to predict code-switching points. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 973–981, Honolulu, Hawaii, October. Association for Computational Linguistics.

Sristy, N. B., Krishna, N. S., Krishna, B. S., and Ravi, V. (2017). Language identification in mixed script. In *Proceedings of the 9th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE'17, page 14–20, New York, NY, USA. Association for Computing Machinery.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, jan.