Chapter 4

Code-Mixed Language Identification in Word Level

As outlined in the research objectives (Section 1.8) of this thesis, it is evident that word-level language identification stands out as a critical task prerequisite for initiating any form of text processing with code-mixed data. We explore the task of word-level language identification within social media code-mixed data in this chapter. The structure of this chapter unfolds as follows: Section 4.1 furnishes a comprehensive delineation of the problem statement to be addressed here. Subsequently, Section 4.2 discusses the datasets employed. The experimental setup is expounded upon in Section 4.3, followed by the presentation of results in Section 4.4, and subsequent discussion in Section 4.5. Finally, Section 4.6 encapsulates our findings drawn from the observations.

4.1 Problem Statement

We will provide a code-mixed sentence as a input and the model will assign language tags to each word as exemplified in Figure 4.1. This is a Bengali-English code-mixed sentence. meaning *"generally, although they are good, still they do not get proposal.. You are getting, at least good. very good"*. The output that we want to get is the tag

associated with each word. *bn* stands for Bengali, *en* stands for English. The tag *univ* stands for universal that is not specific to any languages and often corresponds to a punctuation or an emoji.

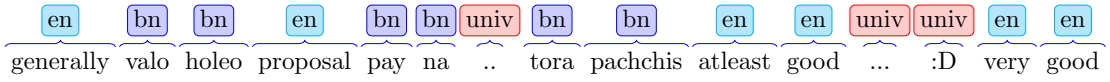


Figure 4.1: Example of a code-mixed sentence along with its language and non-language tag.

4.2 Dataset

Data collection methodology and annotation principle have significant impacts on the quality of a dataset. Majority of the datasets that we have used were retrieved from different social media sites by actually crawling or scraping. The data was then annotated either by a domain-specific annotator or by crowdsourcing [118]. Some datasets contained only a few instances of code-mixing, while others contained frequent code-mixing. Due to paucity of publicly available standardized dataset, we have considered two datasets from each language pair (Bengali-English and Hindi-English) taken from NLP tools contests at ICON 2017[1] and NLP tools contests at ICON 2016[2] respectively to account for some heterogeneity. We have also used a dataset of Spanish-English and Hindi-English language pairs collected from Linguistic Code-switching Evaluation (LinCE)[3] in addition to the Indian languages. LinCE is a shared task, and we do not have gold labels on the test dataset. As a result, the outcome of our experiment is based on the validation dataset. Table 4.1 shows the three datasets with their statistics. The task organizers split each dataset into training (Train), development (Dev), and testing (Test). We reproduce here the same splitting statistics as available on the respective websites.

[1] `https://ltrc.iiit.ac.in/icon2017/`
[2] `https://ltrc.iiit.ac.in/icon2016/`
[3] `https://ritual.uh.edu/lince/datasets`

Table 4.1: Corpus Statistics

Language Pair	Corpus		Train	Dev	Test	All
Bengali-English	ICON_SAIL	Sentences	2002	500	3038	5540
		Tokens	35470	8825	57410	101705
		Monolingual	144	46	483	673
		code-mixed	1858	454	2555	4867
	ICON_POS	Sentences	1852	618	618	3088
		Tokens	20998	7505	7200	35703
		Monolingual	1344	440	454	2238
		code-mixed	508	178	164	850
Hindi-English	ICON_SAIL	Sentences	10347	2587	5527	18461
		Tokens	156335	39564	85494	281393
		Monolingual	4516	1132	2392	8040
		code-mixed	5831	1455	3135	10421
	ICON_POS	Sentences	1578	526	527	2631
		Tokens	23920	8747	8475	41142
		Monolingual	640	217	228	1085
		code-mixed	938	309	299	1546
	LinCE	Sentences	4823	744	1854	7421
		Tokens	95224	15446	36052	146722
		Monolingual	2702	419	-	-
		code-mixed	2121	325	-	-
Spanish-English	LinCE	Sentences	21030	3332	8289	32651
		Tokens	253221	40391	97341	390953
		Monolingual	12851	2182	-	-
		code-mixed	8179	1150	-	-

We have used the annotated corpus from the ICON-2017 SAIL [119] tool-contest (In the rest of this paper, we will allude to this dataset as `ICON_SAIL`[4]) dataset, which was initially created for the sentiment analysis task on English-Hindi and English-Bengali code-mixed data. Every word of this dataset is tagged with its corresponding language labels i.e., *BN* (Bengali language), *HI* (Hindi language), *EN* (English language), *UN* (universal), *EMT* (emoticon) and *MIX* (mixing of two languages) . The datasets were extracted from Twitter via Twitter4j API. These datasets are not open to access but are

[4]`http://www.dasdipankar.com/SAILCodeMixed.html`

available from the organizers after signing a copyright statement. ICON_POS[5] dataset is also structurally similar. The main purpose of the dataset was to predict POS tags at the word level, whereas language tags (en, hi/bn, univ) at word level are also given i.e, *bn* (Bengali language), *hi* (Hindi language), *en* (English language), *univ* (universal), *ne* (named entities), *undef* (undefined) and *mixed* (mixing of two languages). In our experiments, sentiment and POS tags are omitted, and only language tags are considered. LinCE LID dataset uses the Computational Approaches to Linguistic CodeSwitching (CALCS) LID label system where the sentences are split in *lang1*, *lang2*, *other*, *mixed* (partially in both languages), *ambiguous* (either one or the other languages), *fw* (a diffrent langauge other than *lang1* and *lang2*), *ne* (named entities) and *unk* (unrecongnizable words) tokens. Let us illustrate this with a few examples from ICON_POS dataset.

4.2.1 Examples

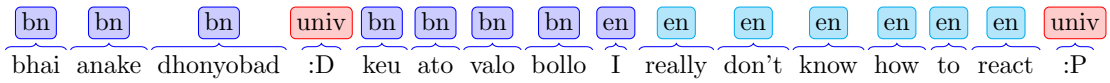


Figure 4.2: Example 2: example of a code-mixed sentence along with its language and non-language tag.

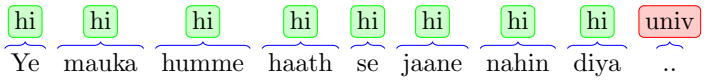


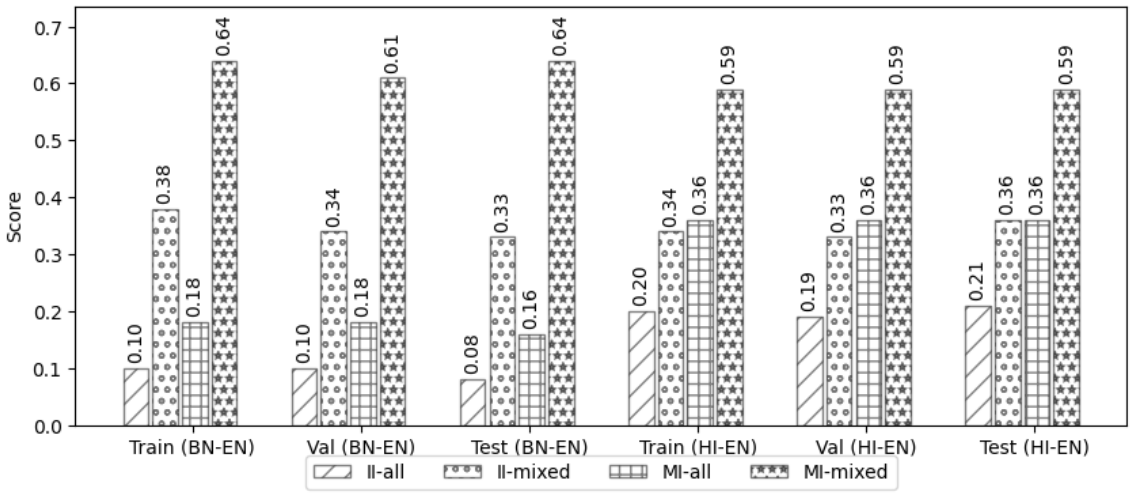
Figure 4.3: Example 3: example of a code-mixed sentence along with its language and non-language tag.

As we use datasets from different sources, with each having a diverse code-mixing pattern, to know the level of code-mixing between languages, we compute three different metrics (M-Index, I-Index, CMI) for the datasets. **CMI-all** is an average over all the

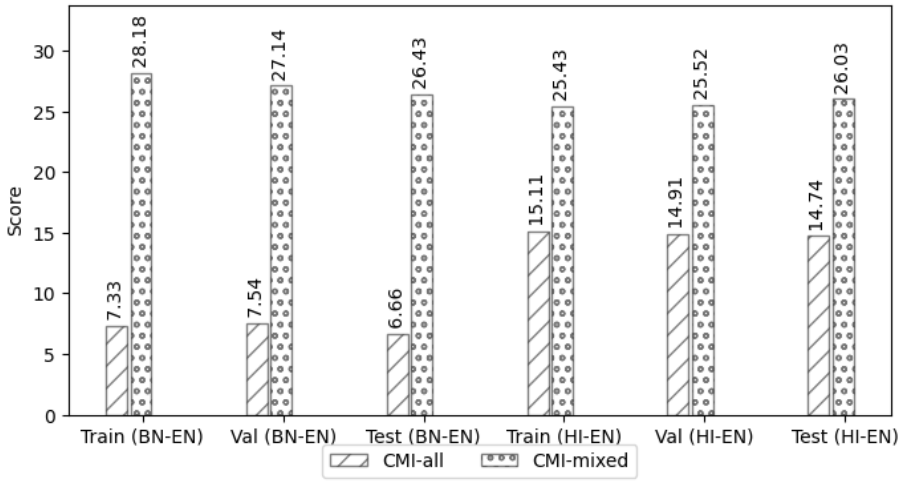
[5] http://www.amitavadas.com/code-mixing.html

utterances in a corpus, and **CMI-mixed** is an average over only code-mixed utterances in a corpus.

Figure 4.4, 4.5 and 4.6 show the balanced distribution of different pair of languages in datasets.

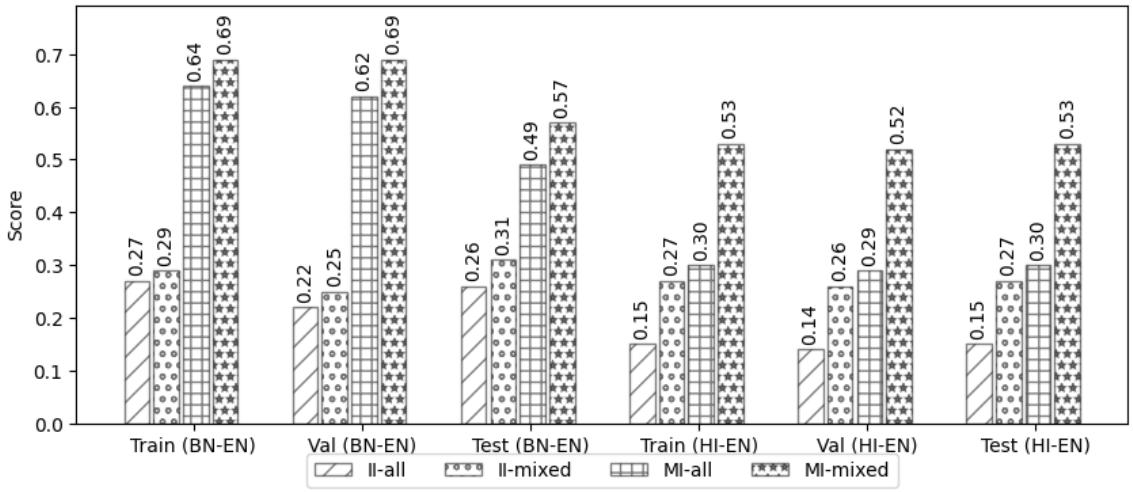


(a) Integration Index (II) and Multilingual index (MI)

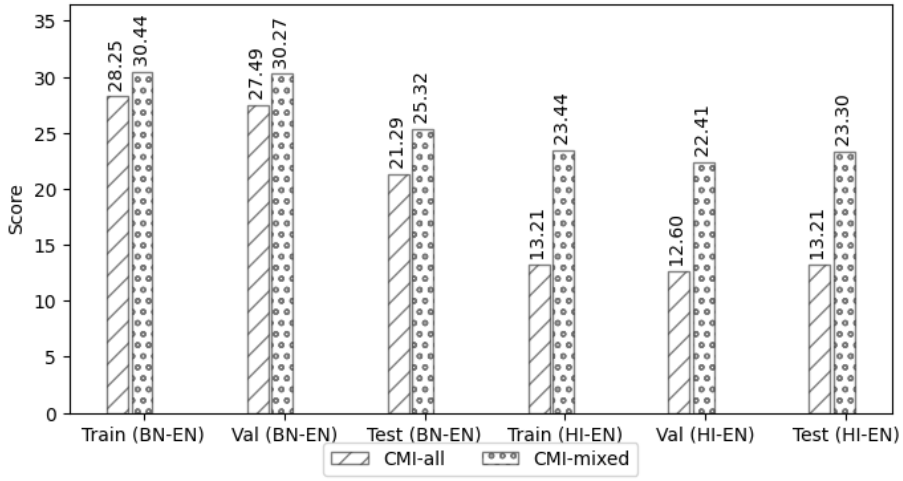


(b) Code Mixed Index (CMI)

Figure 4.4: Graphical comparison of Code-Mixing metrics for the ICON_POS dataset



(a) Integration Index (II) and Multilingual index (MI)

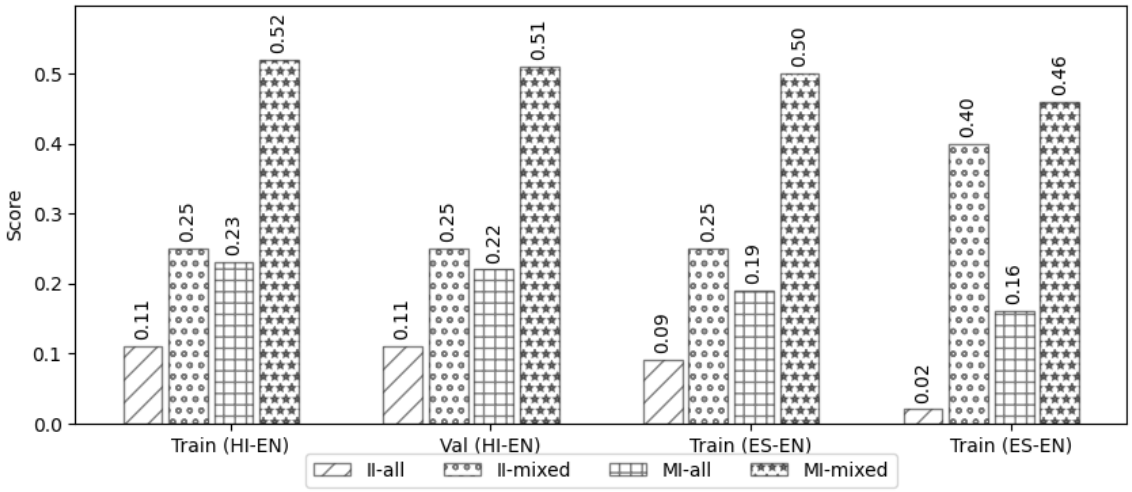


(b) Code Mixed Index (CMI)

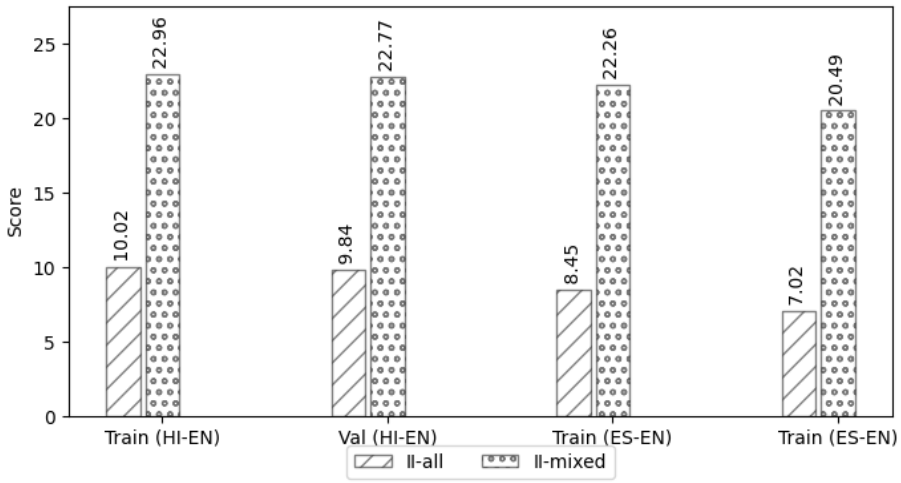
Figure 4.5: Graphical comparison of Code-Mixing metrics for the ICON_SAIL dataset

4.3 Experimental Setup

We discuss different approaches, beginning with baseline methods and then presenting our proposed approaches.



(a) Integration Index (II) and Multilingual index (MI)



(b) Code Mixed Index (CMI)

Figure 4.6: Graphical comparison of Code-Mixing metrics for the LinCE dataset

4.3.1 Baseline Approaches

We include baseline models from Joshi et al. [63] and Jamatia et al. [64]. Jamatia et al. [64] used word embedding (GloVe) with Bi-LSTM layer and character level RNN with a CRF classifier. The models read a phrase from the dataset word by word, assigning the word ID to their embedding representations (GloVe) before passing the representation through a Bi-LSTM layer. After reading all of the words as input to

a dense layer with a softmax layer, the hidden state of the recurrent unit estimates the probability of the language identification tag. Another supervised model with a CRF classifier was implemented by the same authors [63]. For learning sequential data, however, here we utilize the open-source neural sequence labeling toolkit NCRF++ [6].

Joshi et al. [63] used a character, word, and sub-word representation with CNN, MultiCNN, and Bi-LSTM layer. We apply Word2vec for word representation and fastText embedding for sub-word representation. The sentence can be viewed as a sequence of word vectors. Following that, this representation is passed into CNN and LSTM based models. The output is also a sequence of words vector encoded with contextual information. This information is transmitted through a dense layer to obtain the final prediction.

4.3.2 Proposed Approaches

Our proposed model consists of three main layers. The first layer produces the input representations using a BERT module. The sequence of the word vectors generated by the BERT layer is then fed into the second Bi-LSTM module for semantics encoding after the vector representation of each word in the sentence is achieved. Finally, the Bi-LSTM results are fed into the Softmax layer, which outputs the label with the highest probability for each word category. Figure 4.7 shows the model architecture.

4.3.2.1 Input Representation layer or Embedding layer: BERT

BERT is a contextual language model introduced by Devlin [27] in 2019. The design is made up of encoder and decoder components that take into account both left and right contexts. BERT was trained on the masked language model and the next sentence prediction, two unsupervised tasks. To address the out-of-vocabulary (OOV) issue, BERT employs the WordPiece tokenizer, which splits any term absent in the dictio-

[6] `https://github.com/jiesutd/NCRFpp`

nary into sub-words. The encoder block, which includes a multi-head attention layer and a feed-forward layer, receives the contextual information vector. The attention layer is a technique that generates an attention vector for each word in the phrase, capturing contextual interactions between words in the sentence. The attention equation is described below.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4.1}$$

where, Q, K, and V represent different matrices formed from the raw text as described below, and d_k is the embedding dimension.

Query-matrix(Q) : $[seq\_len\_q \times d_k]$, where $seq_len_q$is the length of the raw query text in number of words, d_k is the dimension of each query

Key-matrix (K) : $[seq\_len\_k \times d_k]$, representing the key for a given a given word in Q

Value-matrix(V) : $[seq\_len\_k \times d_v]$, where for each key, there is a value and d_v is the dimension of each value.

The attention function maps a query (Q) to a vector (called the attention vector) using key (K)-value (V) pairs, indicating the importance of each term in Q.

The feed-forward neural network is then applied to all the attention vectors. It modifies the attention vectors for the following encoder or decoder block. The sum of token embedding, segment embedding, and position embedding yields the final embeddings of BERT.

4.3.2.2 Context Encoder Layer or Word Sequence Layer: Bi-LSTM

The Bi-LSTM is an LSTM extension that combines a forward and backward LSTM network to process sequences and connects the network to the output layer (Section 2.9). Here, the Bi-LSTM structure provides the output layer to acquire contextual informa-

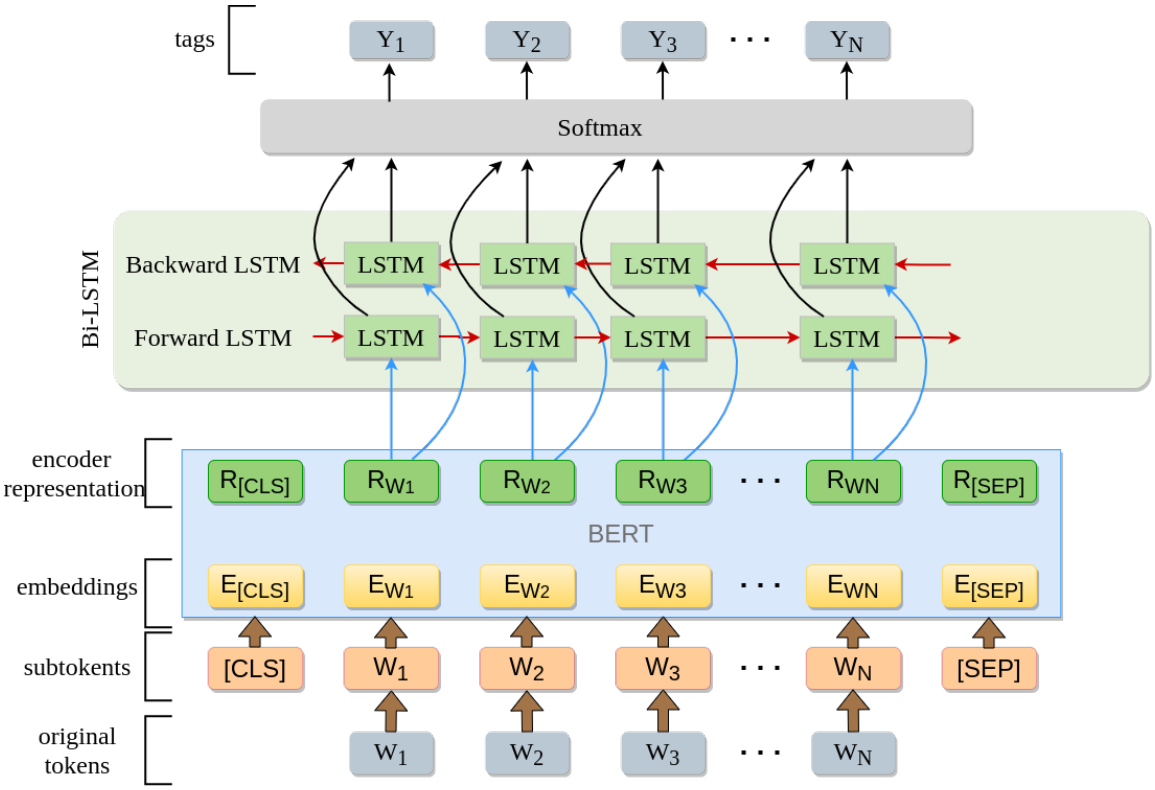


Figure 4.7: Model Architecture of our proposed system

tion from the past (backward) and future (forward) at the same time. Furthermore, the Bi-LSTM incorporates LSTM properties that prevent the vanishing gradient problem. In LSTM, both forward and backward LSTM networks employ the same equations. The LSTM unit protects and regulates the neural network unit's memory (or forgetting) state by using three gates: input, forget, and output, indicated as i, f, and o, respectively (see Figure 4.8). At time step t, these gates take the input $x^{(t)}$ and the past hidden state $h^{(t-1)}$ in the following equations.

$$i^{(t)} = \sigma(W^{ix} \cdot x^{(t)} + W^{ih} \cdot h^{(t-1)} + b_i) \tag{4.2}$$

$$f^{(t)} = \sigma(W^{fx} \cdot x^{(t)} + W^{fh} \cdot h^{(t-1)} + b_f) \tag{4.3}$$

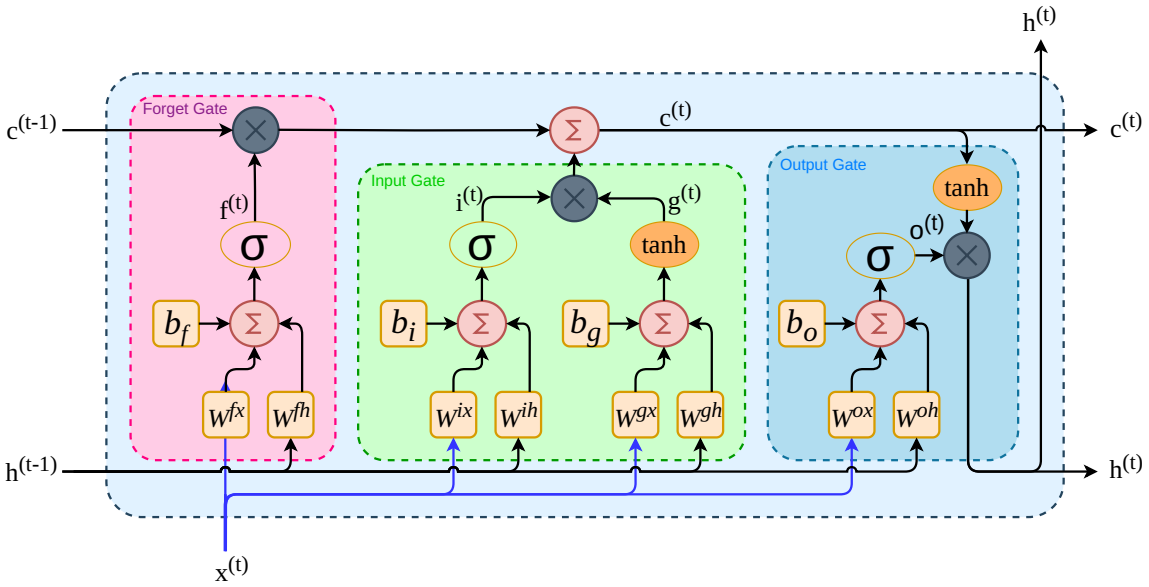


Figure 4.8: Architecture of a LSTM block

$$o^{(t)} = \sigma(W^{ox} \cdot x^{(t)} + W^{oh} \cdot h^{(t-1)} + b_o) \tag{4.4}$$

We can notice from the above equation that the gates are dependent on h and x. We must calculate what might enter the cell (memory) state. This candidate value is calculated as follows:

$$g^{(t)} = tanh(W^{gx} \cdot x^{(t)} + W^{gh} \cdot h^{(t-1)} + b_g) \tag{4.5}$$

$$c^{(t)} = f^{(t)} * c^{(t-1)} + i^{(t)} * g^{(t)} \tag{4.6}$$

$$h^{(t)} = tanh(c^{(t)}) * o^{(t)} \tag{4.7}$$

$h^{(t)}$ is a hidden state that is now utilized to determine what the cell should input,

forget, and output in the next time step. The cell state is denoted by $c^{(t)}$. W^{pq} corresponds to the weight matrix between layer p and q and the bias b_m corresponds to the gate m. σ represents sigmoid function and tanh, the hyperbolic tangent function.

For extracting sentence features, the Bi-LSTM uses the output of the BERT embeddings as an input vector. As the final output H_i, the hidden state of Bi-LSTM will concatenate the forward LSTM $\overrightarrow{h_{ft}}$ and backward LSTM $\overleftarrow{h_{bt}}$.

$$H_i = \left[\overrightarrow{h_{ft}} \cdot \overleftarrow{h_{bt}}\right] \tag{4.8}$$

4.3.2.3 Inference or Tag Prediction Layer: Softmax Function

The prediction layer generates a tag sequence that corresponds to the input sequence. After obtaining the entire sequence of hidden states from the Bi-LSTM layers, we map each hidden state vector to an m-dimensional vector, where m is the number of potential language tags for each word.

Given the fixed maximum input size constraint of 512 tokens for BERT, we maintain the original sentence length within this limit. The BERT model outputs embeddings of size 768. To further refine the representations and capture contextual dependencies, we incorporate a Bi-LSTM layer, which yields embeddings of size 512. Finally, a linear layer is utilized to map the Bi-LSTM outputs to the class level, producing the desired output size corresponding to the number of classes in the classification task. The models are trained on the training dataset and evaluated by predicting the labels for the test dataset. BERT models are trained for maximum 150 epochs depending on the learning rate 0.001. BERT based experiments are performed on Google's Colab[7]. PyTorch deep learning library is used to implement the models. We also use HuggingFace's transformers to fine-tune pre-trained BERT based models.

[7] `https://colab.research.google.com`

4.4 Results

The dataset contains both monolingual and code-mixed sentences (See the corpus statistics, Table 4.1). We, therefore, train our model in 2 steps. First we train our model with monolingual and code-mixed sentences, and then we train it solely on code-mixed sentences. We evaluate and compare every model's performance in terms of precision, recall, F_1-score, and accuracy.

Character level RNN and CRF-based model obtain a F_1 score of 85.1% and 86.30% on the language pairs Bengali-English and Hindi-English, respectively. BERT and mBERT are used as pre-trained embedding models.

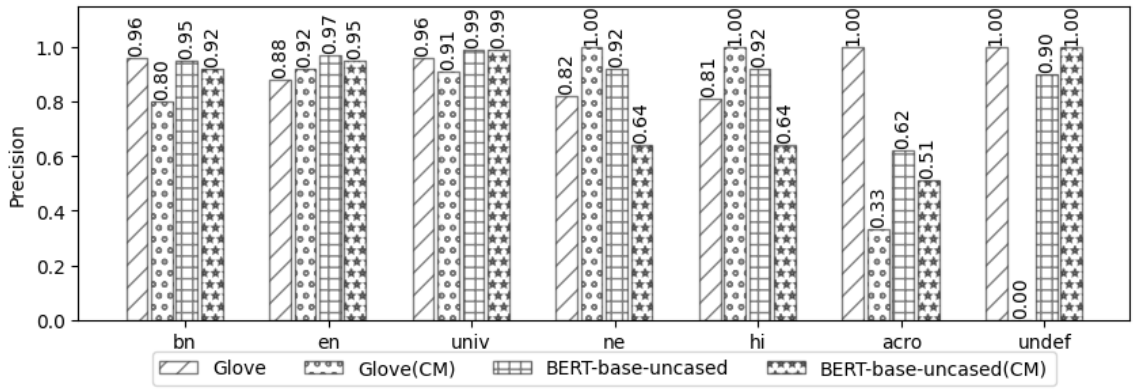


Figure 4.9: Graphical comparison of Precision for baseline and best performing models on ICON_POS (BN-EN) dataset

Figure 4.9, 4.10 and 4.11 show the overall system performance of class level on the ICON_POS (BN-EN) dataset. In the test dataset, the mixed language tags (ne+bn, en+bn, acro+en) are too few to be detected by any model. We, therefore, omitted them in the above figures. The test data contains 7200 tokens with the following class distributions (Table 4.2).

Table 4.2: Class distributions of ICON_POS (BN-EN) test data

Tokens Class	bn	ne	univ	en	hi	acro	undef	ne+bn	en+bn	acro+en	Total
# tokens	2814	204	1366	2608	133	49	16	4	3	3	7200
(%)	(39.08)	(2.83)	(18.97)	(36.22)	(1.84)	(.68)	(.22)	(.05)	(.04)	(.04)	

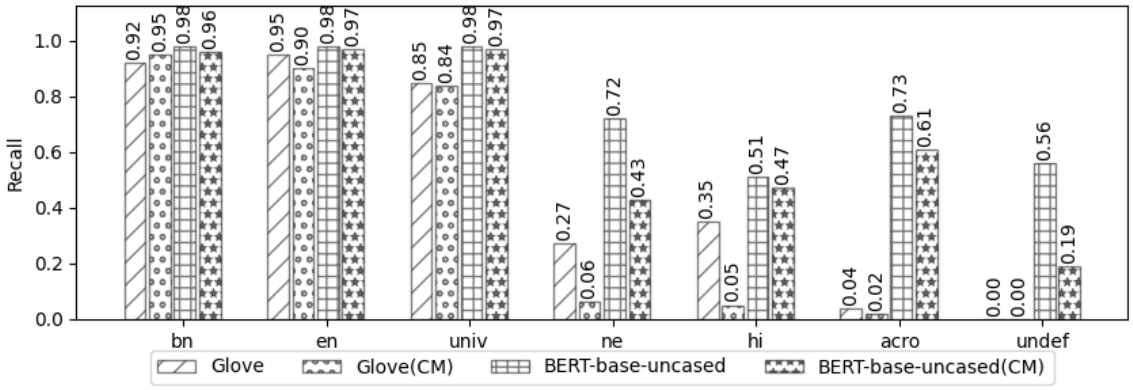


Figure 4.10: Graphical comparison of Recall for baseline and best performing models on ICON_POS (BN-EN) dataset

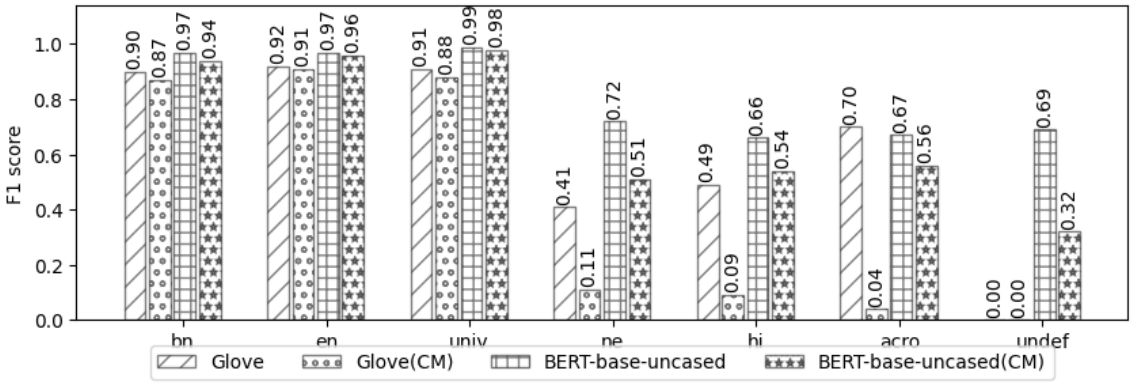


Figure 4.11: Graphical comparison of F_1 score for baseline and best performance model on ICON_POS (BN-EN) dataset

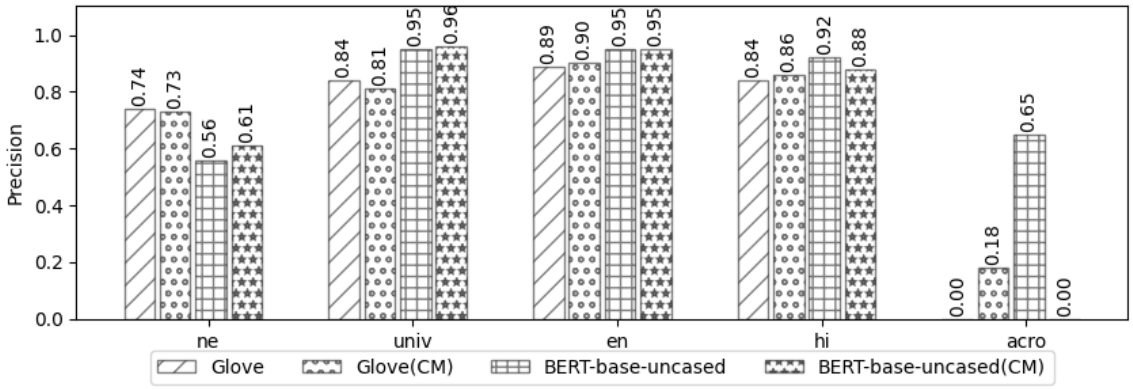


Figure 4.12: Graphical comparison of Precision for baseline and best performing models on ICON_POS (HI-EN) dataset

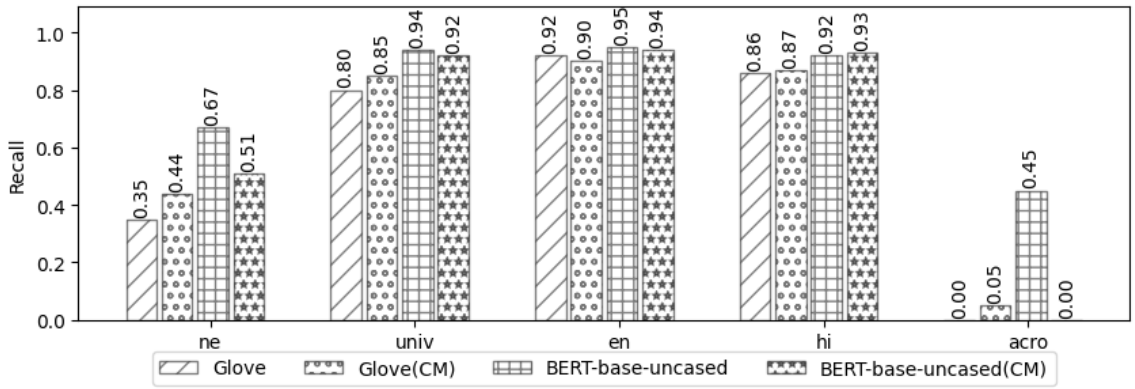


Figure 4.13: Graphical comparison of Recall for baseline and best performing models on ICON_POS (HI-EN) dataset

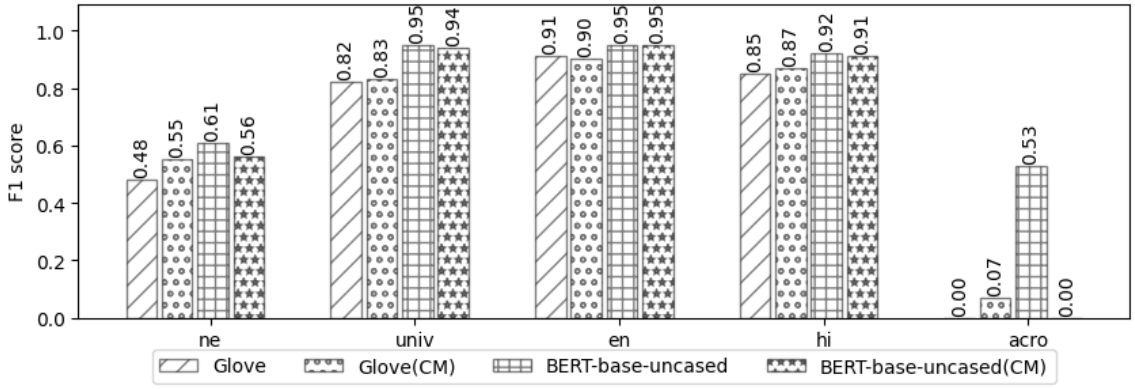


Figure 4.14: Graphical comparison of F_1 score for baseline and best performing models on ICON_POS (HI-EN) dataset

Overall class level system performance on the ICON_POS (HI-EN) dataset is shown in Figures 4.12, 4.13 and 4.14. Like before, the mixed tags in the test dataset are too few to be identified by any model and, therefore, are omitted from the figures above. The test data contain 8475 tokens with the following class distributions (Table 4.3).

Table 4.3: Class distributions of ICON_POS (HI-EN) test data

Tokens Class	**ne**	**univ**	**en**	**hi**	**acro**	**mixed**	**Total**
# tokens	192	1500	3804	2924	53	2	8475
(%)	(2.26)	(17.69)	(44.88)	(34.50)	(.62)	(.02)	

Overall class level system performance on the ICON_SAIL (BN-EN) dataset is

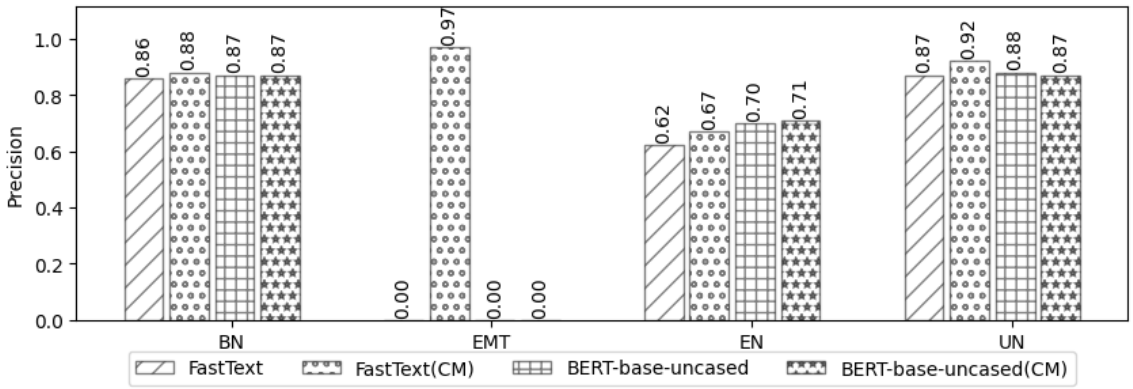


Figure 4.15: Graphical comparison of Precision score for baseline and best performing models on ICON_SAIL (BN-EN) dataset

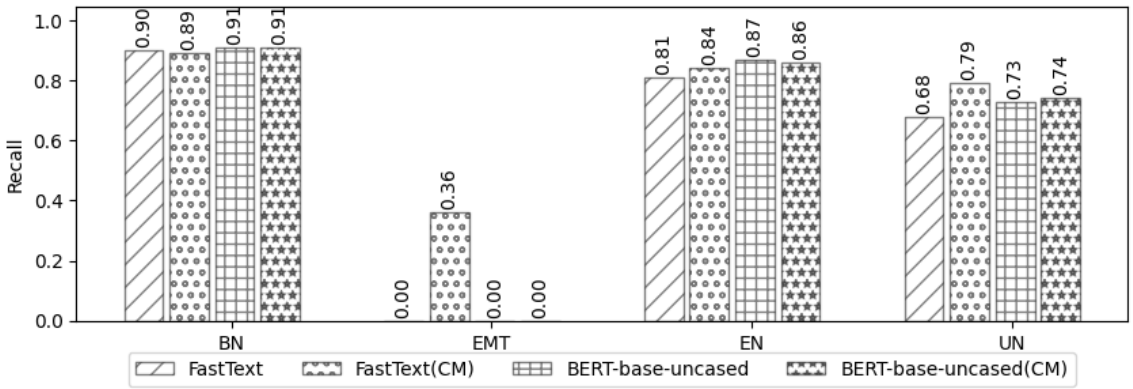


Figure 4.16: Graphical comparison of Recall score for baseline and best performing models on ICON_SAIL (BN-EN) dataset

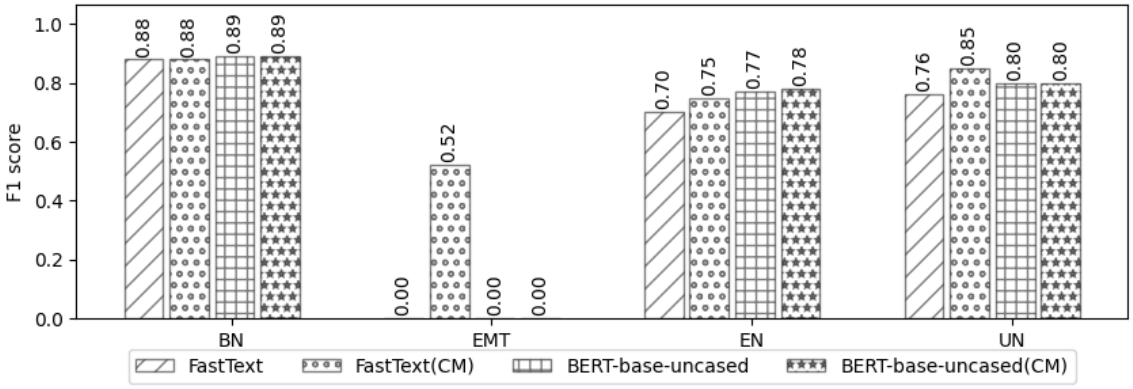


Figure 4.17: Graphical comparison of F_1 score for baseline and best performing models on ICON_SAIL (BN-EN) dataset

shown in Figures 4.15, 4.16 and 4.17. Here, the test data contained 57,410 tokens with the following class distributions (Table 4.4):

Table 4.4: Class distributions of ICON_SAIL (BN-EN) test data

Tokens Class	BN	UN	EN	EMT	MIX	Total
# tokens	29775	15255	10899	1479	2	57410
(%)	(51.86)	(26.57)	(18.98)	(2.57)	(.003)	

Overall class level system performance on the ICON_SAIL (HI-EN) dataset is shown in Figures 4.18, 4.19 and 4.20. The test data contained 85,494 tokens with the following class distributions (Table 4.5).

Table 4.5: Class distributions of ICON_SAIL (HI-EN) test data

Tokens Class	HI	EN	UN	EMT	Total
# tokens	35396	33228	16728	142	85494
(%)	(41.40)	(38.86)	(19.56)	(.16)	

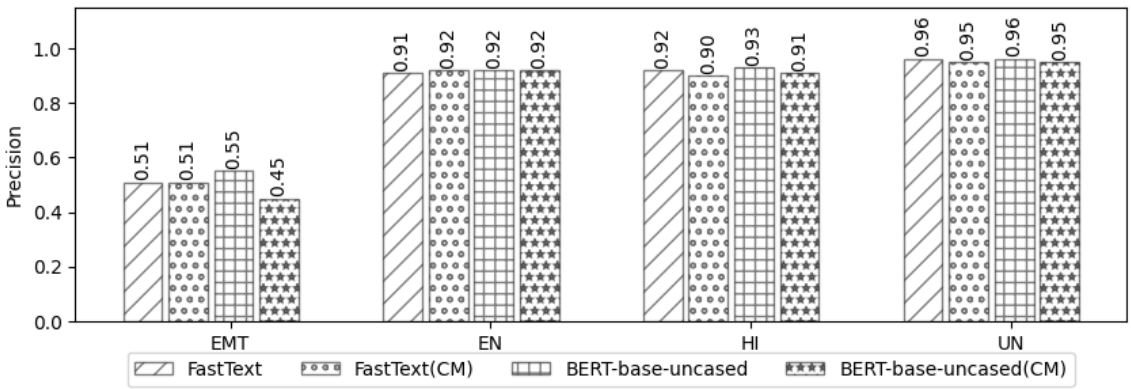


Figure 4.18: Graphical comparison of Precision score for baseline and best performing models on ICON_SAIL (HI-EN) dataset

Figure 4.21, 4.22 and 4.23 show the overall system performance of class level on the LinCE (HI-EN) dataset. Like before, few tags are eliminated from the figures because of very small numbers. The development data contained 15446 tokens with the following class distributions (Table 4.6).

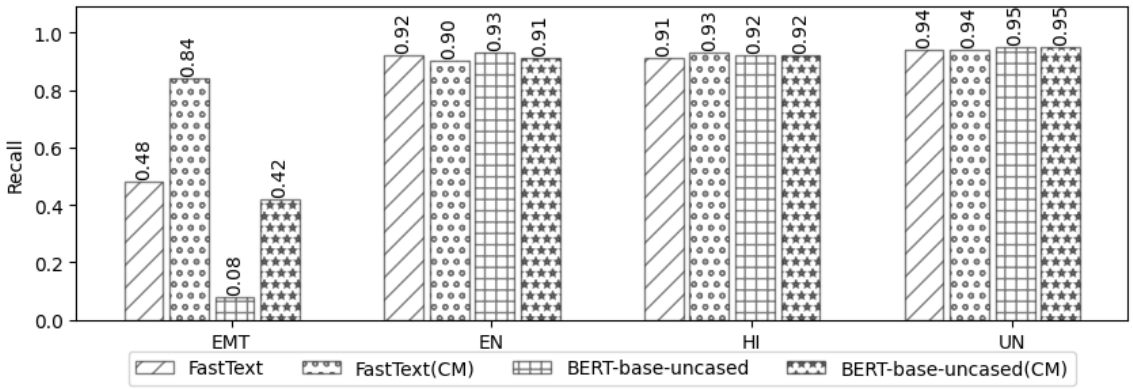


Figure 4.19: Graphical comparison of Recall score for baseline and best performing models on ICON_SAIL (HI-EN) dataset

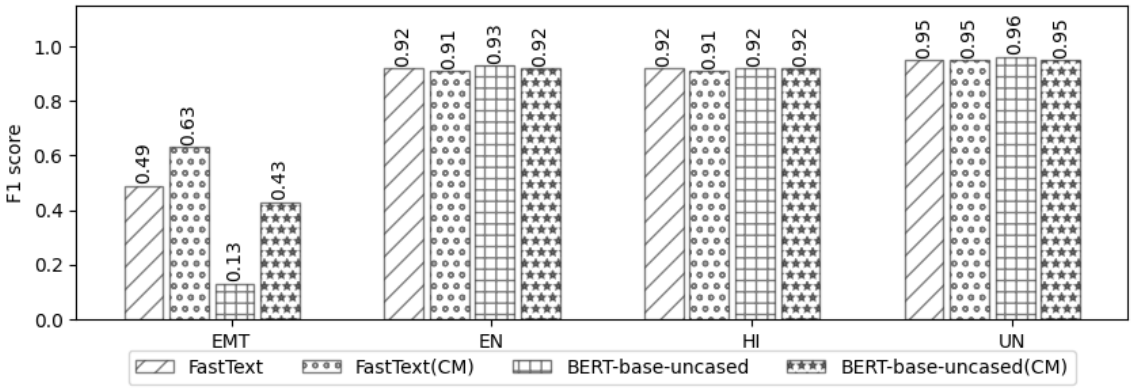


Figure 4.20: Graphical comparison of F_1 score for baseline and best performing models on ICON_SAIL (HI-EN) dataset

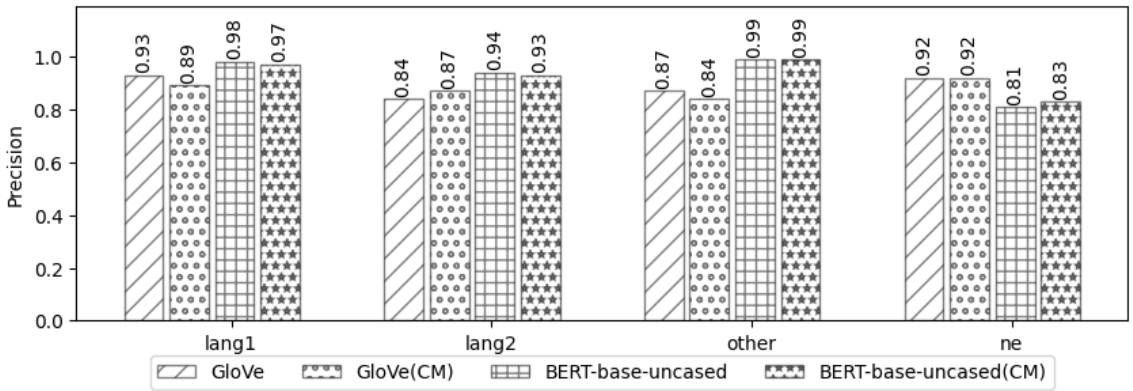


Figure 4.21: Graphical comparison of Precision score for baseline and best performing models on LinCE (HI-EN) dataset

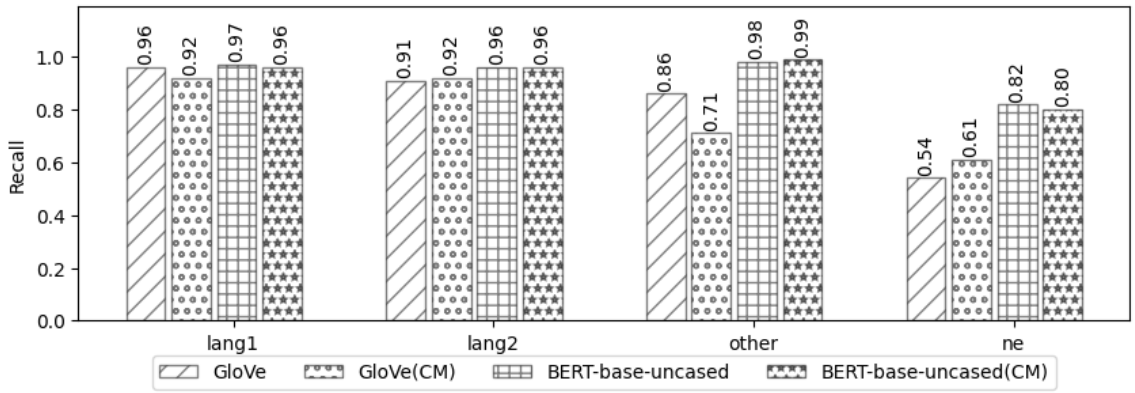


Figure 4.22: Graphical comparison of Recall score for baseline and best performing models on LinCE (HI-EN) dataset

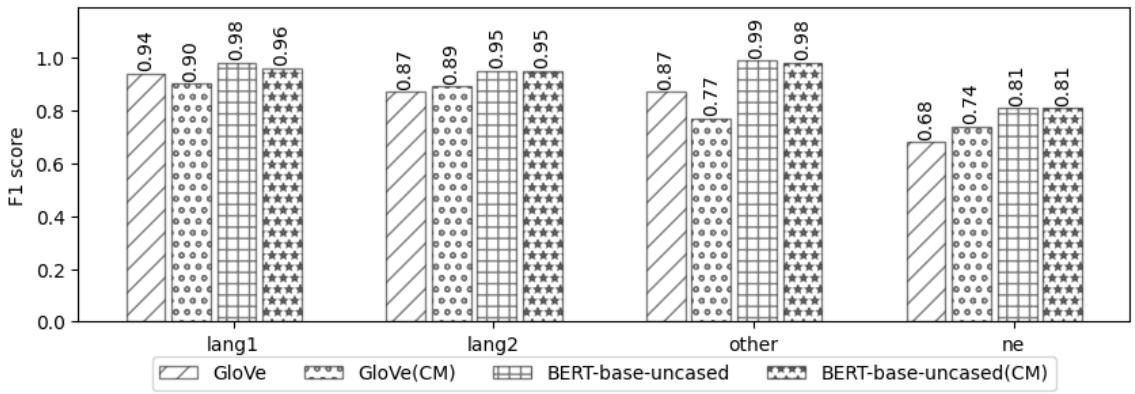


Figure 4.23: Graphical comparison of F_1 score for baseline and best performing models on LinCE (HI-EN) dataset

Table 4.6: Class distributions of LinCE (HI-EN) development data

Tokens Class	**lang1**	**lang2**	**other**	**ne**	**fw**	**unk**	**mixed**	**ambiguous**	**Total**
# tokens	8997	3306	2230	875	29	2	5	2	15446
(%)	(58.97)	(21.40)	(14.43)	(5.66)	(.18)	(.01)	(.03)	(.01)	

Figure 4.24, 4.25 and 4.26 show the overall system performance of class level on the LinCE (ES-EN) dataset. The development data contained 40391 tokens with the following class distributions (Table 4.7).

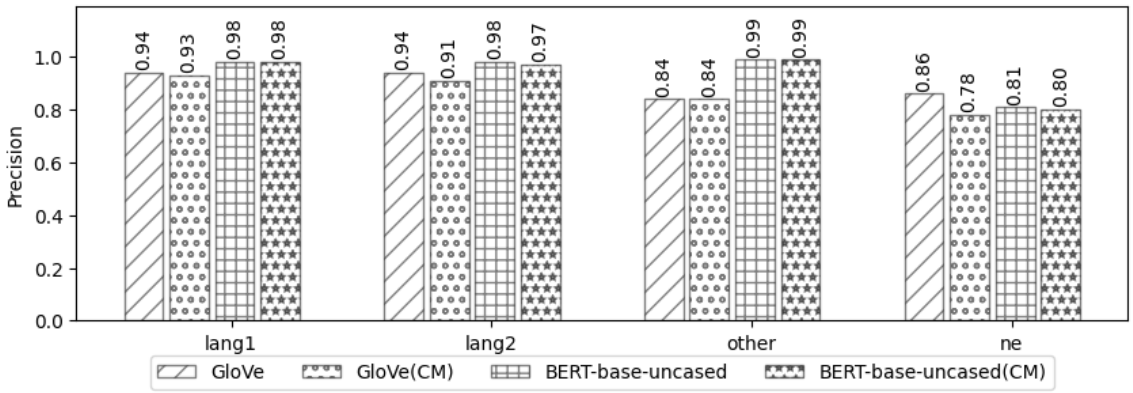


Figure 4.24: Graphical comparison of Precision score for baseline and best performing models on LinCE (ES-EN) dataset

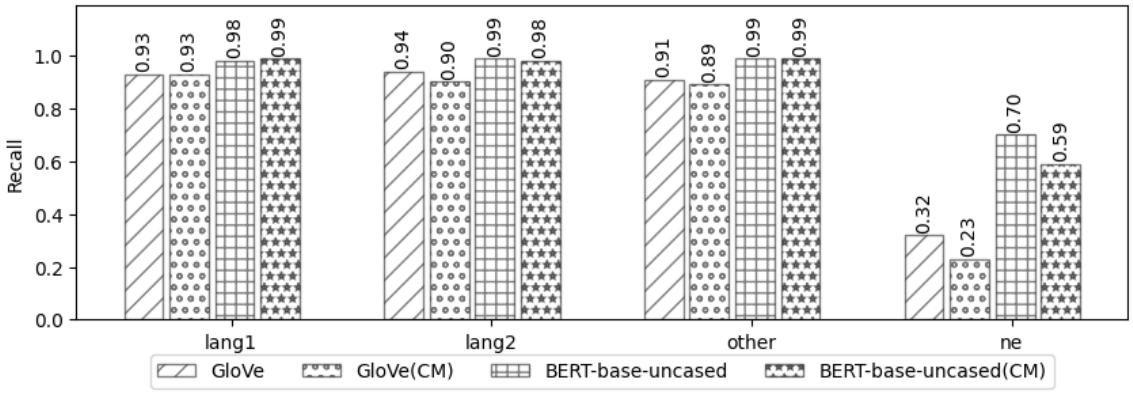


Figure 4.25: Graphical comparison of Recall score for baseline and best performing models on LinCE (ES-EN) dataset

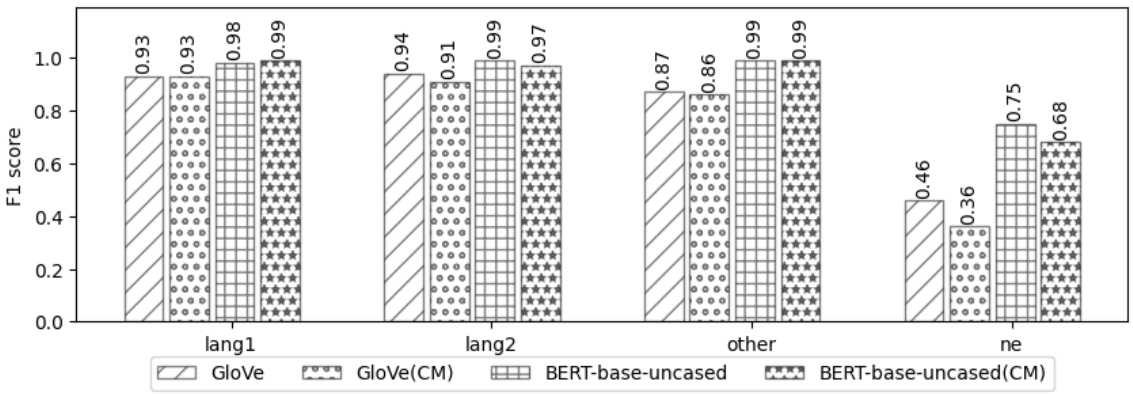


Figure 4.26: Graphical comparison of F_1 score for baseline and best performing models on LinCE (ES-EN) dataset

Table 4.7: Class distributions of LinCE (ES-EN) development data

Tokens Class	lang1	lang2	other	ne	fw	unk	mixed	ambiguous	Total
# tokens	16712	14955	7830	815	2	32	6	39	40391
(%)	(41.37)	(37.02)	(19.38)	(2.01)	(.004)	(.07)	(.01)	(0.09)	

4.5 Discussion

For ICON_POS (BN-EN) dataset, we obtain a F_1 score of **94.85**% on BERT-base-uncased embedding with Bi-LSTM layer with the other variants of BERT model performing similarly. The GloVe based baseline model is able to secure F_1 score of **87.72**%. In Figure 4.9, 4.10 and 4.11, we observe that, our model performs well on almost all language tags. Not only for the language tags, even for detecting acronyms, our models exhibit promising results, the baseline models are incapable of detecting acronyms. Our best performing model for the ICON_POS (HI-EN) dataset is the BERT-base-uncased model with Bi-LSTM layer, which has a F_1 score of **93.42**%. We see a performance drop after removing monolingual sentences from both the HI-EN and BN-EN datasets, but our models still surpass the baselines.

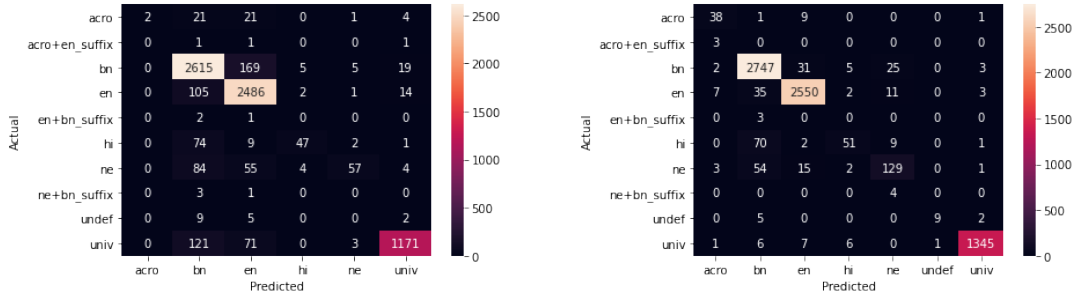
For ICON_SAIL (HI-EN) dataset, we obtain F_1 score of **93.02**% on BERT-base-cased embedding with Bi-LSTM layer (Table 4.10). The results are obtained after training for 25 epochs on the data sets. Increasing the number of epochs to 30 and 50 made no differences. Multilingual BERT model with Bi-LSTM layer has a F_1 score of **82.88**% and emerges as our best performing model for ICON_SAIL (BN-EN) dataset. In Figures 4.15, 4.16 and 4.17, we observe that our models are not able to detect EMT (emoticon) tags. In training dataset some emoticon are labeled as UN (Universal) tag which mislead our models to detect those symbol as a UN tag.

For LinCE (HI-EN) dataset, we obtain an F_1 score of **95.9**% on BERT-base-uncased embedding with Bi-LSTM layer (Table 4.12). The findings are acquired after 20 epochs of training on the data sets. Increasing the number of epochs to 30, 40, and 50 had no effect. BERT-base-uncased model with Bi-LSTM layer has an F_1 score of **98.09**% and

emerges as our best performing model for LinCE (ES-EN) dataset. We observe a loss in performance after deleting the monolingual sentences from both the datasets HI-EN and ES-EN, but our models still outperform the baseline ones.

Despite the far smaller training dataset, the contextual embedding on the code-mixed corpus has better results. BERT model outperforms on the ICON_POS (BN-EN), ICON_POS (HI-EN) and ICON_SAIL (HI-EN) datasets but shows a poorer performance on the ICON_SAIL (BN-EN) dataset. Also, number of monolingual sentences in the ICON_SAIL (BN-EN) dataset is comparatively low and thus training points are very few, which lead to the low F_1 score in both *All* and *CM* data.

The reduction in the number of token mismatches utilizing the BERT word embeddings can be attributed to the gains in F_1 score. The confusion matrices (Figure 4.27) for both the models show that many tokens which are wrongly classified in the GloVe model (Figure 4.27a), are not present in the BERT model (Figure 4.27b). Our model does not predict any intra-word code-mixed data like *Facebooke* (a Bengali term that means "in Facebook" and contains the named entity *Facebook* and the Bengali suffix *e*. So the structure looks like ne+bn_suffix). This is one type of code-mixed data where mixing occurs within a word itself. BERT uses WordPiece technique to diminish unknown token occurrences significantly, which must have played a role in the success and dealing with the out-of-vocabulary issue.



(a) GloVe model

(b) BERT model

Figure 4.27: Confusion Matrices for the baseline and proposed models on the ICON_POS (BN-EN) Corpus Test Set

The inability of the baseline system to capture context information is one of its major flaws. Some words with identical spellings in different languages retain the same tag, as seen in the results. For example,

- **Original:** Ota *cmplt hole* 1 ti montro jop korte hobe 25 *bar*.
- **Desired translation:** When it is complete, you have to chant 1 mantra 25 times.

Here, the baseline model predicts *cmplt* as a *bn* tag, *hole*, and *bar* as an *en* tag, but *hole*(meaning, if it happens) and *bar* (meaning, times) are Bengali words as well with different meaning than in English. The baseline models could not understand the context and failed. Our model distinguishes between spelling variations of *complete* keywords and predicts *en* tags, as well as analyzes the context of two words and predicts the correct tags.

Another aspect may be the inconsistency of the labels. Some of the words like *upokrito* (benefitted), *golata* (the throat), and *janle* (if you know) are wrongly labelled as *en* in the gold standard dataset. Our model correctly marked them as *bn*. Based on the domain knowledge, we are confident that our predicted tags are correct, as these all are Bengali words and have been used in a Bengali context. On the other hand, it is also noticed that sometimes our model predicts a word as *en*, but the gold standard dataset marked as *bn* like *gender, are, on, change, to, be, expectation*. These all are English words and have been used in an English context. We find that the LinCE dataset is the most correctly annotated among the three datasets - a fact that helped our models obtain a high score above the other two datasets.

4.5.1 Comparison with the baselines

We compare the performance of our proposed models with that of baseline [63] and [64] models in Table 4.10, 4.11 and 4.8, 4.9 respectively. We observe that our proposed models performed better for both the datasets. The overall performance margin for ICON_POS (BN-EN) data is 8.12% and for ICON_POS (HI-EN) data is 6.41%. On the

other hand, for ICON_SAIL (HI-EN) and ICON_SAIL (BN-EN) dataset, we receive comparatively less performance gain of 0.68% and 4.23% respectively.

For LinCE (HI-EN) and LinCE (ES-EN) dataset our models outperform the baseline models. When we remove monolingual sentences from the dataset and train only on code-mixed sentences, a performance deterioration is observed across all models and language pairs during the evaluation.

Table 4.8: Comparison of baseline with best proposed model on ICON_POS (BN-EN) Data

Embedding	**Model**	F_1-**score**$_{ALL}$(%)	F_1-**score**$_{CM}$(%)
Character Level RNN	CRF	85.10	82.68
GloVe	Bi-LSTM	87.72	83.05
BERT Base Cased	Bi-LSTM	94.31	93.72
BERT Base Uncased	**Bi-LSTM**	**94.85**	93.80
mBERT Base Cased	Bi-LSTM	94.30	93.95
BERT Large Cased	Bi-LSTM	93.52	91.87

Table 4.9: Comparison of baseline with best proposed model on ICON_POS (HI-EN) Data

Embedding	**Model**	F_1-**score**$_{ALL}$(%)	F_1-**score**$_{CM}$(%)
Character Level RNN	CRF	86.30	79.69
GloVe	Bi-LSTM	87.79	85.28
BERT Base Cased	**Bi-LSTM**	**93.42**	91.80
BERT Base Uncased	Bi-LSTM	92.79	91.03
mBERT Base Cased	Bi-LSTM	92.79	91.43
BERT Large Cased	Bi-LSTM	91.80	91.41

Table 4.10: Comparison of baseline with best proposed model on ICON_SAIL (HI-EN) Dataset

Embedding	**Model**	F_1-**score**$_{ALL}$(%)	F_1-**score**$_{CM}$(%)
Word2vec	Bi-LSTM	92.39	92.18
Word2vec	CNN	91.90	91.85
Word2vec	Multi CNN	92.04	91.92
FastText	Bi-LSTM	92.25	92.15
FastText	CNN	92.01	91.26
FastText	Multi CNN	92.15	91.98
BERT Base Uncased	**Bi-LSTM**	**93.02**	92.31
mBERT Base Cased	Bi-LSTM	92.88	92.64

When we see the results across the datasets, there are some interesting observations. Between the two ICON_POS data (BN-EN vs HI-EN) F_1 figures are marginally better for BN-EN (see Table 4.8, 4.9). HI-EN data is having comparatively higher CMI, M-index and I-index (Figure 4.4).

Table 4.11: Comparison of baseline with best proposed model on ICON_SAIL (BN-EN) Dataset

Embedding	**Model**	F_1-**score**$_{ALL}$(%)	F_1-**score**$_{CM}$(%)
Word2vec	Bi-LSTM	78.65	77.23
Word2vec	CNN	78.73	78.88
Word2vec	Multi CNN	77.82	75.71
FastText	Bi-LSTM	79.39	83.67
FastText	CNN	74.43	77.85
FastText	Multi CNN	75.21	77.12
BERT Base Uncased	**Bi-LSTM**	**82.75**	81.70
mBERT Base Cased	Bi-LSTM	81.88	81.86

Table 4.12: Comparison of baseline with best proposed model on LinCE (HI-EN) development data

Embedding	**Model**	F_1-**score**$_{ALL}$(%)	F_1-**score**$_{CM}$(%)
CharRNN	CRF	84.80	79.67
GloVe	Bi-LSTM	89.69	87.16
BERT Base Cased	Bi-LSTM	95.79	94.48
BERT Base Uncased	Bi-LSTM	96.23	94.16
mBERT Base Cased	**Bi-LSTM**	**96.72**	94.78

Table 4.13: Comparison of baseline with best proposed model on LinCE (ES-EN) development data

Embedding	**Model**	F_1-**score**$_{ALL}$(%)	F_1-**score**$_{CM}$(%)
CharRNN	CRF	92.02	91.02
GloVe	Bi-LSTM	91.07	89.95
BERT Base Cased	Bi-LSTM	98.02	97.52
BERT Base Uncased	Bi-LSTM	98.09	97.80
mBERT Base Cased	**Bi-LSTM**	**98.34**	97.80

On the other hand, for the ICON_SAIL dataset, BN-EN performs much below its HI-EN counterparts (Table 4.10 and 4.11). This observation can be attributed to the fact that training data is comparatively less and the test data is comparatively more in case ICON_SAIL BN-EN data.

LinCE dataset shows the best figures for both HI-EN and ES-EN (Table 4.12, 4.13). For HI-EN, LinCE has the least code-mixing (CMI, M-index and I-index) among the three datasets (Figure 4.6). For LinCE EN-ES data, code-mixing is seen to be the least among all the language pairs and hence puts up the best show in terms of F_1 score.

Thus we can infer that language identification depends upon the percentage of code-mixing between languages. If the code-mixed metric value increases, then the performance of language identification decreases. Language identification performance is also affected by sample quantity and annotation quality in the dataset. We have already shown that our proposed model often predicts the proper language tag, yet the gold standard annotation data sometimes predicts it incorrectly.

During evaluation, we observe that different BERT variant models provide similar performance in the language identification task. However, BERT-base model provides the best performance among them. The primary reason is that the BERT-base model was trained on English data written in the Roman script, and our code-mixed data are also written in the Roman script, whereas Multilingual BERT was trained in several languages using their native language script.

While our proposed method demonstrates superior performance across all language pairs, several limitations persist. Specifically, challenges arise in effectively handling emoticons and the mixing of two languages within a single word. Our model exhibits low performance in these scenarios. Furthermore, there are instances where our model's performance in assigning universal (UNIV) tags closely resembles that of alternative methods. Within the dataset, numerous words represent Bengali named entities, leading to occasional confusion between Bengali tags and named entity tags. Addressing

these limitations is crucial for enhancing the robustness and accuracy of our model in multilingual processing tasks.

Table 4.14: Effect of Bi-LSTM layer

Dataset	Without Bi-LSTM	With Bi-LSTM	Improvement (%)
ICON_SAIL (BN-EN)	81.03	**81.88**	1.04
ICON_SAIL (HI-EN)	92.42	**93.02**	0.64
ICON_POS (BN-EN)	94.55	**94.85**	0.31
ICON_POS (HI-EN)	92.58	**92.79**	0.22
LinCE (HI-EN)	96.40	**96.72**	0.33
LinCE (ES-EN)	98.20	**98.34**	0.14

In Table 4.14, our experiments demonstrate that incorporating Bi-LSTM on top of BERT embedding led to a modest improvement in model performance across all three datasets. While BERT's bidirectional nature theoretically eliminates the need for additional bidirectional processing, our findings suggest that leveraging Bi-LSTM can still enhance the model's ability to capture complex linguistic patterns. However, it is important to note that the observed performance improvements were relatively small, indicating potential diminishing returns with additional model complexity. Future research could further explore alternative architectures or preprocessing techniques to optimize performance in similar natural language processing tasks.

4.6 Summary of this Chapter

In this chapter, we attempt to tackle the most fundamental problem of any code-mixed downstream task: the problem of language identification. The difficulty of identifying languages in code-mixed data is determined by the data type and code-mixing behavior. We have studied the problem with extensive experiments using multiple models and architectures with different representations of the input texts. Our Bi-LSTM model on top of BERT neural representations of code-mixed data emerges as the best performing model. For the language pairs analysed, the deep learning model with fine-tuned pre-trained word embedding model outperforms the classic CRF approach. Simply

modifying the input representation assists in obtaining the highest results. The collection here consists of both monolingual and code-mixed sentences. Thus, we initially train our model with both monolingual and code-mixed sentences, and then proceed to code-mixed sentences. Though there is an obvious decline in performance across all models and language pairs when only code-mixed sentences are considered, compared to a combination of mono-lingual and code-mixed sentences, our models provide better performances over the state-of-the-art models.