SN Computer Science (2025) 6:844
https://doi.org/10.1007/s42979-025-04377-4

ORIGINAL RESEARCH

Word Level Language Identification from Social Media Code-Mixed Data Leveraging Transformer-Based Models

Supriya Chanda[1,2] · **Sukomal Pal**[1]

Received: 31 May 2024 / Accepted: 5 September 2025

Abstract

Code-mixing is mixing of two or more languages in a statement or a conversation. Multilingual communities all over the world often use this on a regular basis, especially during communication in social media. People mix their mother tongue with other national and international languages, like English. While code-mixing in verbal communication is a serious problem, it is not either easy for written communication as well. In informal written communication, people use multiple languages without changing the script, i.e. words from two or more languages occur next to each other using a single script. For an intelligent system, automatic language identification in such scenarios is an essential task. Language identification is considered here as a token classification problem. Every word in a sentence receives a linguistic tag in a supervised setup. For this task, we leverage pre-trained Bidirectional Encoder Representations of Transformers (BERT) to obtain the contextual representations of sentences. We evaluate several combinations of deep learning models and input representations. Characters, sub-words, and their combination embeddings are primarily considered for CNN and LSTM-based models. Later, BERT with LSTM model is used. Through three different datasets: ICON_POS, ICON_SAIL, and LinCE, we conduct language identification (LID) task in Bengali-English (BN-EN), Hindi-English (HI-EN), and Spanish-English (ES-EN) code-mixed sentences. Our proposed method of the Bi-LSTM model on top of BERT neural representations of code-mixed data outperforms the existing state-of-the-art techniques in terms of F_1 scores. For two datasets of Bengali-English language pairs, ICON_POS and ICON_SAIL, we observe performance gains of 8.12% and 4.23%, respectively. We demonstrate performance gains of 6.41%, 0.68%, and 7.83% for three datasets of Hindi-English language pairs ICON_POS, ICON_SAIL, and LinCE, respectively. We also show an improvement of 6.08% in language identification for the Spanish-English language pair in the LinCE dataset.

Keywords Code-mixing · Language identification · BERT · mBERT · Bengali-English · Hindi-English · Spanish-English

Introduction

With the growing popularity of social media (SM) platforms such as Facebook, Twitter, and WhatsApp, Natural Language Processing (NLP) techniques have become ubiquitous in the last decade. These platforms have also become a quick source of information on local events and global news. Anyone in any corner of the world can share any information and communicate with each other instantly. This user-generated content (UGC) is predominantly informal, without following grammar of any particular language, any standard rules and formats of formal communication. Thus SM text is characterized by non-standard spelling of words, self-created abbreviations, and non-standard grammatical structures. But this ease of communication has thrown a

✉ Supriya Chanda
supriyachanda.rs.cse18@itbhu.ac.in;
supriya.chanda@bennett.edu.in

Sukomal Pal
spal.cse@itbhu.ac.in

[1] Department of Computer Science and Engineering, IIT (BHU) Varanasi, Varanasi, Uttar Pradesh 221005, India

[2] SCSET, Bennett University, Varanasi, Uttar Pradesh 201310, India

Published online: 22 September 2025

series of challenges to the NLP systems. For a substantial number of languages, native keyboards are not available in the computing devices. Even if they are available, sizeable fraction of users are not familiar with their native language keyboard or do not use such keyboard for the sake of convenience, so phonetic typing (transliterated or Romanised text) and code-mixing are frequently observed in social media.

In linguistics, using more than one language or script in a conversation or a discourse is referred to as code-switching. When bilingual speakers face problems during a conversation in a particular language, they often switch to a different language to make the listener better comprehend them, and this may happen several times during the course of their conversation. Hymes [1] defines code-switching as "a common term for alternative use of two or more languages, varieties of a language or even speech styles" while Bokamba [2] defines code-switching is "the blending of words, phrases and sentences from two independent grammatical (sub) systems across sentence boundaries within the same speech event". Here is an example of code-switching (CS) between Bengali and English, where Bengali segments are in italic with their corresponding English gloss on a new line.

- **CS:** *Amar americate thaka char bachor hoye gache* (Bengali), but I still miss my country (English).
- **Gloss:** I have been living in America for four years, but I still miss my country.

The other phenomenon closely associated with code-switching is code-mixing. It is generally observed that when a speaker uses two or more languages simultaneously, she often switches between them. Sometimes, one utilizes more than one language even within a single phrase. Such a person performs code mixing (CM) subconsciously, and it can involve many components of a language, such as phonology, morphology, grammatical structures, or lexical items. Following is an example of code-mixing between Bengali and English. Bengali segments are in italic with their corresponding English gloss on a new line

- **CM:** Generally *valo holeo* proposal *pay na, tora pachchis* at least good
- **Gloss:** Even if it is good, I do not get the proposal, It is good that you get it.

Code-switching and code-mixing are "the embedding of linguistics units such as phrases, words, and morphemes of one language into an utterance of another language" [3]. Code-switching occurs when a speaker switches between languages across sentence borders, whereas code-mixing occurs when a person switches languages inside a phrase. In general, we can say that code-switching occurs intentionally, while code-mixing more often happens unintentionally. Despite this subtle difference, researchers frequently use these terminologies interchangeably. In this paper also, we do not make any distinction between CS and CM and refer to both the instances as code-mixing.

Code-mixing is a common phenomenenon in spoken languages. As a result, code-mixing in speech has been extensively investigated in various areas like linguistics, psycholinguistics, and socio-linguistics for almost a half-century. However, work on processing data of code-mixed text, namely code-mixing in a social media text, was initially carried out in the early 1980s [4] to late 1990s [5]. Automatically identifying the language for monolingual data [6] has been widely used in the industry and is a well-known research problem for a long time. Modern state-of-the-art NLP approaches excel in all downstream tasks such as sentiment analysis, automated language identification, Parts of speech (POS) tagging, Named-entity recognition (NER), question answering, and text summarization for monolingual text. However, these approaches do not perform well on code-mixed text [7]. In the case of code-mixed text, it is thus imperative to determine the language of each word.

Motivation

With the advent of smartphones and ease of content creation in Web 2.0, tremendous growth has been observed in user-generated content. People express themselves in a free-wheeling spree without bothering about the correctness of their spelling, grammar and languages. Code-mixing in social media texts is extremely common [8], especially when the writer is bilingual or multilingual. Although comprehending this is natural for her human counterpart, it is difficult for a language processing system to comprehend the context of the statement.

Furthermore, it is often difficult to distinguish between borrowing from a second language as an acceptable native vocabulary and code mixing [9]. In such scenarios, accurately identifying the language of each word becomes essential for downstream tasks like sentiment analysis, hate speech detection, and other text processing tasks due to the following reasons.

- Various languages might entail distinct preprocessing needs. For instance, tokenization, stemming, or stop word elimination could vary based on the language. Precisely discerning the language of each word aids in implementing the suitable preprocessing techniques for individual language elements within the code-mixed data.
- Numerous NLP resources, such as lexicons, embeddings, or linguistic rules, are tailored to specific languages. By pinpointing the language at the word level, one can harness language-specific resources for each

segment of the code-mixed data, thereby improving the effectiveness of subsequent tasks.

- Tasks like sentiment analysis and hate speech detection frequently depend on models or features designed for specific languages. Detecting the language at the word level enables the application of language-specific models or features to each text segment, thereby enhancing the precision of subsequent tasks. Some research has shown that the integration of a language identification module enhances the effectiveness of downstream code-mixed tasks, including sentiment analysis, hate speech detection, and emotion recognition [10, 11].
- In code-mixed data, ambiguity is common, particularly when a word or a phrase may pertain to more than one language. Identifying the language at the word level aids in clarifying such ambiguity by assigning a distinct language label to each word, thereby mitigating confusion in subsequent analyses.

The following example can give a better understanding about the ambiguity issue of code-mixed data.

- **Original:** *Jani na ajj ki hoyeche. bad day!*
- **Translation:** Don't know what happened today. **bad day! / leave it!**

In the above example, if one assumes it to be a completely Bengali text, then the meaning of this sentence is *Don't know what happened today. leave it!*, but if it is taken to be a Bengali-English code-mixed sentence, then the meaning is *Don't know what happened today. Bad day!*. So, the task of identifying languages of code-mixed text at the word level is complicated but crucial and, therefore, critically necessary for text understanding.

The objective of this study is to identify the language affiliation of individual words within a text that is characterized by seamless integration of two or more languages. Specifically, our focus lies in devising a model capable of accurately assigning language tags to each word in a code-mixed sentence, as exemplified in Fig. 1. In this example, *bn* stands for Bengali, and univ stands for universal.

Many Python tools, like spacy-langdetect[1], Pycld2[2], TextBlob[3], and Googletrans[4], can identify languages at the document or sentence level. At the word level, capturing the contextual language tag is challenging for these libraries. If we use the above mentioned example, the python libraries will identify "bad" and "day" as English words. Addressing the task of word-level language identification enables us to develop NLP tools that are more precise and culturally attuned for handling code-mixed social media content. This advancement ultimately fosters a deeper comprehension of online communication and its societal implications.

Novelty and Authors' Contribution

The main aim of this paper is to find a comparison between non-contextual (Word2Vec, GloVe, FastText) and contextual (BERT) input representation in automatic language identification of CM data. This paper concentrates on three language pairs: English-Hindi, English-Bengali, and Spanish-English. These language combinations are chosen because Hindi (Northern India and certain regions of Pakistan have roughly 342 million native speakers), Bengali (Eastern India and Bangladesh have approximately 229 million native speakers), and Spanish (with approximately 500 million native speakers worldwide) are the world's fourth, seventh and second most spoken languages, respectively.

The novelty and main contributions of this proposed research work are as follows.

1. The problem statement addressed in this work focuses on word-level language identification from code-mixed social media data.
2. A comparative analysis is conducted between non-contextual input representations (Word2Vec, GloVe, FastText) and contextual input representation (BERT) in the automatic language identification of code-mixed data.
3. The proposed approach utilizes a deep learning framework that leverages pre-trained models for embedding, offering a novel solution to the problem statement.
4. The proposed model is evaluated using six distinct datasets encompassing three language pairs: Bengali-English, Hindi-English, and Spanish-English.
5. Additionally, the percentage of code-mixed data within each dataset is calculated, and the model's performance is evaluated under varying conditions, including the utilization of solely code-mixed data and a combination of code-mixed and monolingual data.

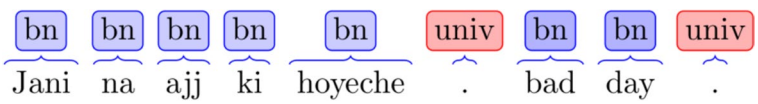


Fig. 1 Example of language identification task

6. An in-depth analysis of the results is conducted, focusing on the performance of the proposed model compared to BERT and BiLSTM methods.

[1] https://pypi.org/project/spacy-langdetect/
[2] https://pypi.org/project/pycld2/
[3] https://textblob.readthedocs.io/en/dev/
[4] https://py-googletrans.readthedocs.io/en/latest/

The remainder of the paper is organized as follows. Section Related Works provides a synopsis of some earlier work. The datasets that we have used are discussed in Sect. Dataset. Section Model Architecture presents our computational methodologies, model descriptions, and evaluation methodology, followed by Sect. Results for results and Sect. Discussion for discussion. We conclude in Sect. Conclusions and Future Works.

Related Works

Language identification, as a field of study, has a rich historical backdrop that predates the advent of computational methodologies. Initially motivated by the exigencies of translation, early forays into this domain relied on rudimentary manual techniques aimed at swiftly discerning the languages of documents. Over the course of more than five decades, automatic language identification has garnered sustained scholarly attention, marking significant milestones in its evolution.

Mustonen's [12] seminal work marked the genesis of formal inquiry into automated language identification. Rau [13] introduced the concept of gapped bigrams, leveraging relative frequencies of character combinations to discern linguistic patterns. Subsequent advancements by Henrich [14] pioneered the application of positive Decision Rules, relying on unique character and character n-grams for language classification. Vitale [15] further expanded the repertoire of techniques, employing character combinations coupled with decision rules to ascertain etymological groupings of proper names, supplemented by trigram analysis for unresolved cases.

The 1990s witnessed a proliferation of methodologies aiming to refine language identification systems. Souter et al. [16] compiled extensive lists of common words for each language, while Giguet [17] meticulously curated exhaustive lists of function words sourced from dictionaries. Kikui [18] delved into the utilization of word suffixes and their relative frequencies for linguistic classification. Giguet's subsequent work in 1998 explored the efficacy of suffix analysis derived from untagged corpora, expanding the scope of language identification techniques.

Mather [19] marked a departure towards a data-driven approach, collating word frequencies into feature vectors for enhanced categorization. Furthermore, the convergence of language identification with broader text categorization methodologies was underscored by Elworthy [20], laying the groundwork for cross-pollination of techniques between these allied fields. Canvar and Trenkle [6] proposed a character n-gram approach on a newsgroup article corpus which achieved 99.8% accuracy. However, the same method failed drastically with code-mixed social media data. The reason that most of the newsgroup articles follow formal and standard writing, whereas social media is a collection of informal text can be attributed to this anomaly.

There has been a surge of attention in the computational linguistic field in recent years in supporting code-mixing tasks. The first and second workshops on a computational approach to code-switching conducted a shared task on language identification for numerous language pairs co-located with Empirical Methods in Natural Language Processing (EMNLP) 2014 [5] and EMNLP 2016. Furthermore, the Forum for Information in Retrieval Evaluation (FIRE) organized several shared tasks on language identification for Indian language pairs at FIRE 2014, FIRE 2015, and FIRE 2016 [21].

Word-level language identification has received significant attention over the past decade, with various approaches proposed to solve the problem. For better understand we try to group all the related work into three thematic clusters.

- ***Traditional Machine Learning Approaches:*** Prior to the emergence of deep learning models such as Transformers, language identification tasks were predominantly approached using handcrafted linguistic features combined with traditional machine learning algorithms. Nguyen and Dogruoz [22] explored different methods like dictionary lookup, language models, logistic regression classifier, and conditional random fields (CRF) classifier for Turkish-Dutch code-mixed data. For Indian language pairs, the word level language identification has been addressed by Barman et al. [23] at the first workshop on a computational approach to Code-Switching. Different methods like supervised classification (Support Vector Machine (SVM)), Sequence classification (CRF), and dictionary lookup have been applied on Bengali-Hindi-English Facebook comments. Das and Gamback [24] used the same methods with various features on code-mixed chat message corpora (English-Bengali and English-Hindi). They also introduced a Code Mixed Index (CMI) to evaluate the level of blending in a corpus. Vyas et al. [25] created a multi-level annotated corpus of Hindi-English code-mixed text collected from Facebook forums and explored language identification, back-transliteration, normalization, and POS tagging on this data. In this paper [26], a CRF model was used with features such as word (word vector of the current term), spellings, and intra-word features. Chittaranjan et al. [27] used a CRF-based system for word-level language identification. As features, word-based and character-based n-gram (3-gram and 5-gram) embeddings are used and passed through a SVM classifier for training and testing [28]. In addition, for word language identification, data such as modified edit distance, word frequency, character

[5] https://emnlp2014.org/workshops/CodeSwitch/call.html

n-grams, part of speech tag, and language tag of surrounding words were employed by Jhamtani et al [29]. Ansari et al. [30] have used pos tag and several others features like word length and the word itself to perform language identification in code-switched scenarios. SVM, Decision Trees, Logistic Regression and Random Forests have been experimented with these features.

- ***Contextual Embedding-based Methods:*** With the growing adoption of contextual embedding models, several studies have investigated their effectiveness across different language pairs, while some survey papers [31, 32] have reviewed the performance trends of these models. The effectiveness of this paper by Joshi and Joshi [33] for the language identification of Hindi-English code-mixed data has been evaluated using character, sub-word, and word-based representation [34]. This task has been formulated as a token classification task. On top of word representation, the most popular convolutional neural network (CNN) and LSTM networks have been used. Jamatia et al. [34] used pre-trained word embedding (GloVe) with LSTM layer and Character level Recurrent Neural Network (RNN) with CRF classifier. They also demonstrated that when compared to a supervised approach like CRF, the deep learning model achieved comparable accuracy. For text vectorization, the deep learning models stated in their preceding two papers used context-independent embedding. For each word/token, this technique is termed prediction-based vectors. The context is lost when all of the various interpretations of a token are combined (averaged) into a single vector. With dynamic language representation, BERT models have recently boosted the language understanding field. On the basis of the words around it, these models construct a contextual representation of the word. Yasir et al. [35] tackle the challenge of mixed-script identification in a code-mixed dataset comprising Roman Urdu, Hindi, Saraiki, Bengali, and English. The language identification model is trained using word vectorization and various RNN architectures. Through experimental evaluation, they optimize several architectures including LSTM, Bi-LSTM, GRU, and Bi-GRU for the task. Among these, the Bi-GRU model achieves the highest accuracy when using learned word class features combined with GloVe embeddings. Their study also addresses challenges inherent to multilingual environments, such as Romanized words with English characters, generative spelling variations, and phonetic typing. Dutta [36] develops a word-level language identification model for Bangla-English code-mixed social media text using subword embeddings. The process begins with generating an embedding vocabulary using the Google SentencePiece model. The resulting subword embeddings, each representing a timestep, are then passed through a BiLSTM recurrent unit for classification.
- ***Cross-lingual and Code-Mixed Language Models:*** For a few language pairs and tasks, there are two centralized benchmarks. Linguistics Code-switching Evaluation (LinCE) [37] is one of them, and it includes four language pairs: Spanish-English [38], Nepali-English [39], Hindi-English [40], and Modern Standard Arabic-Egyptian Arabic. Another is GLUECoS [41], a framework for evaluating language understanding in Code-Switched NLP. Language pairings such as English-Hindi and English-Spanish are chosen. Language identification still needs to be explored, particularly within contexts of linguistic scarcity. Within this framework, the CoLI-Kanglish [42] shared task emerges as a significant initiative, spearheaded by organizers committed to open-sourcing a dataset characterized by code-mixed content in Kannada and English, transcribed in Roman script. Lakshmaiah et al. [43] delves into this realm, employing machine learning methodologies to address Code-mixed Language Identification at the Word Level in Kannada-English texts. Furthermore, the CoLI-Tunglish shared task, held at the FIRE 2023, extends this exploration to encompass Tulu, a language with its unique linguistic characteristics. This initiative seeks to cultivate learning models tailored for Word-level Language Identification in Code-mixed Tulu Texts. Notably, the CoLI-Tunglish [44] dataset integrates three languages: Tulu, Kannada, and English at both the word and sub-word levels, reflecting the complexity inherent in multilingual communication. Same task also offered at FIRE 2024. Chanda et al. [45] leveraged Multilingual BERT (mBERT) for word embedding and a Bi-LSTM model for sequence representation for this task. Ojo et al. [46] built a system architecture on LSTM neural network and Word2Vec embedding to distinguish Kannada from English at word level. Gundapu and Mamidi [47] proposed CRF model and HMM model for word level language identification in English - Telugu Code Mixed Data. Some study shows that non-textual features like neighbourhood based features are useful to solve the Language Identification task for more than three languages in a text [48].

We follow the footsteps of recent works by Joshi and Joshi [33] and Jamatia et al. [34] on language identification for code-mixed social media data with contextual representation. However, instead of using non-contextual GloVe, we explore BERT, a contextual word embedding that has recently produced good results in other applications.

Dataset

Data collection and annotation procedures have a significant impact on the quality of the dataset. The majority of the datasets were retrieved from social media sites by crawling or scraping. The data was then annotated either by a domain-specific annotator or by crowdsourcing [49]. Some datasets contained few code-mixing, while other datasets contained frequent code-mixing. Due to the lack of a publicly available standardized dataset, we have included two different datasets from each language pair (Bengali-English and Hindi-English) taken from NLP tools contests at ICON 2017[6] and NLP tools contests at ICON 2016[7] respectively to account for some heterogeneity. We have incorporated a dataset of Spanish-English and Hindi-English language pairs collected from Linguistic Code-switching Evaluation (LinCE)[8] in addition to the Indian languages. LinCE is a shared task, and we don't have gold labels on the test dataset. As a result, the outcome of our experiment is based on the validation dataset. Table 1 shows the three datasets that we have used with their statistics. The relevant organizers split each dataset into training (Train), development (Dev), and testing (Test). The same splitting statistics as is on the website were used.

We have used the annotated corpus from the ICON-2017 SAIL [50] tool-contest (In the rest of this paper [50], we will allude to this dataset as `ICON_SAIL`[9]) 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 available from the organizer after signing a copyright statement. `ICON_POS`[10] 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 just 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

Table 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	–	–

[6] https://ltrc.iiit.ac.in/icon2017/

[7] https://ltrc.iiit.ac.in/icon2016/

[8] https://ritual.uh.edu/lince/datasets

[9] http://www.dasdipankar.com/SAILCodeMixed.html

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

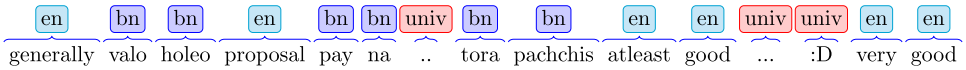


Fig. 2 Example of CM sentence with language switch between English and Bengali (Example 1)

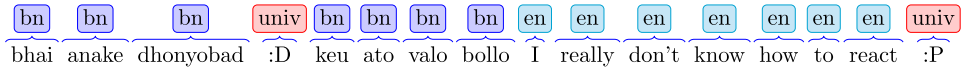


Fig. 3 Example of CM sentence with language switch between English and Bengali (Example 2)

langauge other than *lang1* and *lang2*), *ne* (named entities) and *unk* (unrecongnizable words) tokens. We did not perform any additional preprocessing steps such as removing special characters or handling rare tokens. Let us illustrate this with a few examples from `ICON_POS` dataset.

Illustrative Examples of Code-Mixed Data

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

Multilingual Index (M-Index) [51] was introduced as a tool for calculating the distribution of languages used within a text document that contains at least two languages. The index is bounded between 0 (monolingual corpus) and 1 (terms are uniformly shared among languages in a given text or each language is represented by an equal number of tokens). Non-language tokens are excluded when calculating the M-Index. The M-index is defined as

$$\text{M-Index} = \frac{1 - \sum p_j^2}{(N-1)\sum p_j^2} \tag{1}$$

Where $N > 1$ is the total number of languages present in the corpus and p_j is the probability of presence for language j in the corpus.

However, M-Index does not show the degree of cohesion among the languages (in other words, how the languages are distributed within a sentence). For instance, in Example 1 (Fig. 2) and Example 2 (Fig. 3), we see equal number of Bengali and English words; M-Index of both sentences is, therefore, 1. However, the two languages are integrated (strongly coupled) in Example 1 (Fig. 2), with four switch points while Example 2 (Fig. 3) is having just one switch point.

Integration Index (I-Index) [52] is a metric that includes the probability of switching within a text. Given a corpus of tokens labeled with language l_i, having i varying from 1 to $v-1$, and j ranges from $i+1$ to v (considering a pair of words in a sliding window of two words in a sentence). The presence of any non-language token between the same or different language tags has no effect on the I-Index as per the definition. The I-index is calculated as follows,

$$\text{I-Index} = \frac{1}{v-1} \sum_{1 \le i < j \le v} S(l_i, l_j) \tag{2}$$

where switch point, $S(l_i, l_j) = 1$ if $l_i \neq l_j$ and 0 otherwise. n denotes the total number of tokens in the corpus., The amount of tokens with *other* tags is shown by u. v is the number of tokens given language tags ($v = n - u$). The factor of $1/(v-1)$ reflects the fact that there are maximum $(v-1)$ possible switch points in a corpus of size n.

The I-index is a simple way to count the number of Code-Switches in a document. In this case, value 0 indicates a monolingual text with no switching, and value 1 represents the highest amount of code switches, i.e., a text in which every word switches language, which is an implausible real-world circumstance.

Code-Mixing Index (CMI) [24] helps to understand the amount of multi-linguality in a given text based on number of words coming from different languages.

CMI is calculated as follows,

$$\text{CMI} = \frac{\sum_{i=1}^{N}(|\ W_i\ |) - max\{|\ W_i\ |\}}{n-u} \text{ where } n > u \tag{3}$$

where $\sum_{i=1}^{N}(|\ W_i\ |)$ is the sum of number of words in all N laguages present in an utterance. $max\{|\ W_i\ |\}$ is the maximum number of words found in any language. n is the total number of tokens. u is the number of tokens given other tags (NE (Named Entity), UNIV (Universal), and ACRO (acronym) etc).

CMI values are expressed in percentages (0 to 100), where 0 represents that the utterance contains only language-independent tokens or it is a single language utterance or it might be a mixture of single language and language independent tokens, and for others, CMI is a positive number $<= 100$. Thus, it can be also expressed as

$$CMI = \begin{cases} 100 \times \left[1 - \frac{max\{|W_i|\}}{n-u}\right] & : n > u \\ 0 & : n = u \end{cases} \tag{4}$$

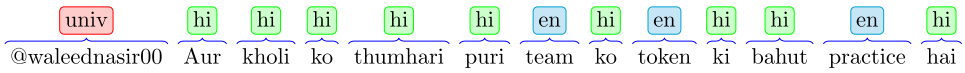


Fig. 4 Example of CM sentence with language switch between English and Hindi (Example 3)

Fig. 5 Example of monolingual sentence with Hindi language (Example 4)

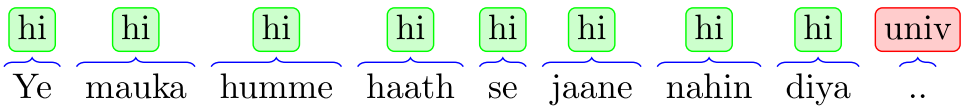
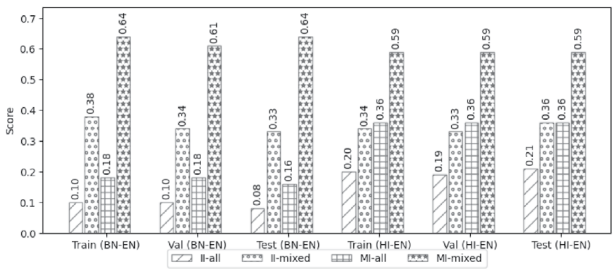
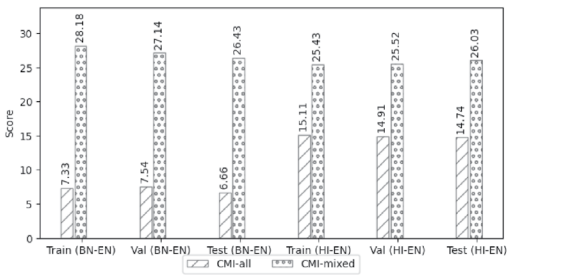


Table 2 Spans for examples

Language pairs	BN-EN		HI-EN	
Example	1	2	3	4
# Total tokens	15	16	13	9
# Lang. 1(en) tokens	6	7	3	0
# Lang. 2(bn/hi) tokens	6	7	9	8
# Switch Point	4	1	6	0
CMI	50	50	25	0
M-Index	1	1	0.6	0
I-Index	0.36	0.07	0.55	0



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



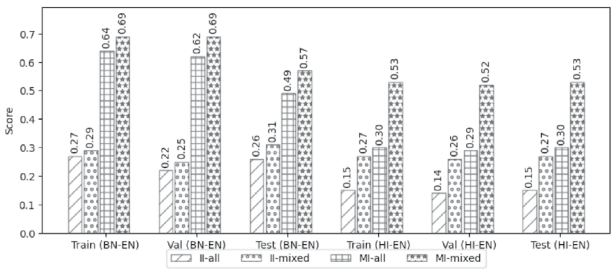
(b) Code Mixed Index (CMI)

Fig. 6 Graphical comparison of Code-Mixing metrics for the ICON_POS dataset

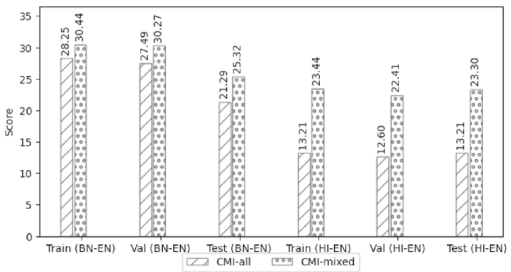
Example 3 (Fig. 4) contains unbalanced Hindi and English words and thus its CMI value is less. Also, the two languages are much more integrated with six switch points, leading to high I-Index value. Example 4 (Fig. 5) is a case for single language utterance only (Table 2).

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

Figures 6, 7 and 8 show the balanced distribution of different pair of languages in datasets.



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



(b) Code Mixed Index (CMI)

Fig. 7 Graphical comparison of Code-Mixing metrics for the ICON_SAIL dataset

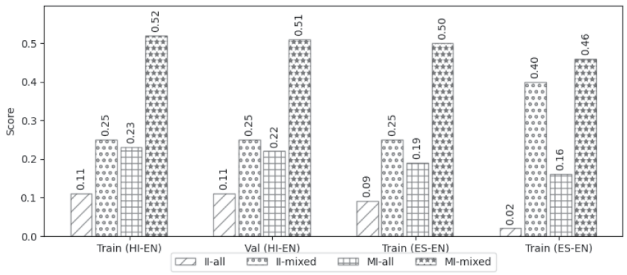
Model Architecture

We use a layered architecture in the proposed system for all our experiments. The model architecture comprises of the following three layers.

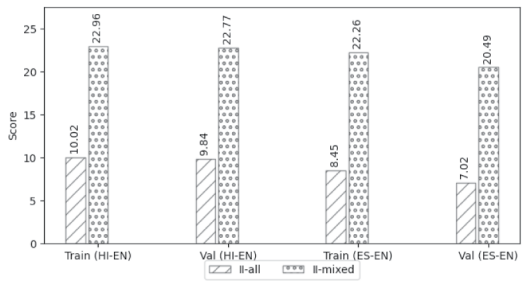
Embedding Layers

Word Embeddings are learned representations of individual words in a text that consider the meaning of the word along with other words with which the individual word appears. This layer converts each word in the input sequence into a vector which is an essential step as the model needs to take input in vectors. Different embedding models can be used in this layer. The models used in our study are listed below.

- **Word2vec**: Word2vec is a method to create word embeddings efficiently and has been around since 2013 [53]. It allows users to compute continuous vector representations of words from large datasets in less time



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



(b) Code Mixed Index (CMI)

Fig. 8 Graphical comparison of Code-Mixing metrics for the LinCE dataset

and improves the semantic regularities of learned words using word offset technique.

- **CharRNN**: Character-based models for word embedding learn word representations using the corresponding character representations, and these models learn representations corresponding to the task at hand. Since we generate representations from the characters, out-of-vocabulary words can be elegantly handled using this technique. Each character is initialized with a random representation vector in the character look-up table. For each word in a sentence, the characters form a sequence and are given in direct and reverse order to the forward and backward pass of the BiLSTM. The final embedding for the word obtained from the model is the concatenation of character-level representation of the word as given by the concatenation of forward and backward representations of the BiLSTM and the word level representations stored in the word look-up table. A representation vector corresponding to the UNK token is used for words that are not present in the word look-up table. Since RNNs and LSTMs have a bias towards their most recent inputs, the forward LSTM generates an accurate representation of the word suffix, and similarly, backward LSTM generates an accurate representation of the word prefix.
- **FastText**: FastText [54] is an open-source library that combines concepts from other successful models such as bag-of-words, N-gram with the help of a new extension called subword information. FastText is an extension of the word2vec model. Each word is represented by an n-gram of letters. This permits the embeddings to recognize suffixes and prefixes, as well as capture the meaning of shorter words. FastText can outperform other deep learning-based methods in terms of training times by significantly decreasing it from several days to just a few seconds while increasing the accuracy.
- **GloVe**: Global vectors for word representation (GloVe) [55] is a word embedding approach that captures both global and local data of a corpus. It's an unsupervised learning approach that aggregates global word-word co-occurrence matrices from a corpus to generate word embeddings. In vector space, the resultant embeddings reveal fascinating linear substructures of the word. It is based on word-context matrices and matrix factorization algorithms.
- **BERT**: BERT uses a transformer, an attention mechanism that learns contextual relationships between words (or sub-words) in a text, is used by BERT. In its most basic version, the transformer consists of two distinct mechanisms: an encoder that reads the text input and a decoder that provides a prediction for the task. The models are pre-trained on large text corpora such as Wikipedia and produce state-of-the-art results with necessary fine-tuning on several downstream tasks. The contextual language representation model BERT [56] has been used for the downstream task of code-mixed language identification. We have used two models sizes for BERT from huggingface library[11], `BERT-BASE` and `BERT-LARGE`. There are a number of Transformer blocks in both BERT model sizes (also known as encoder blocks). The base version has 12 encoder blocks, whereas the large version contains 24 encoder blocks. Both the base and large versions include larger hidden units (feedforward networks) - 768 and 1024, respectively - as well as more attention heads - 12 for the base version and 16 for the Large version. The variants for the base and large versions are listed below.

  – **bert-base-cased**: The `bert-base-cased` is pre-trained on English text and has a total 110M parameters.
  – **bert-large-cased**: The `bert-large-cased` is pre-trained on English text and has a total 335M parameters. |Cased| indicate that the model traned on cased english text.
  – **bert-base-uncased**: The |bert-base-uncased| is pre-trained on English text and has a total 109M parameters. |Uncased| indicate that the model traned on lower-cased english text.

[11] https://huggingface.co/transformers/pretrained_models.html

– **bert-base-multilingual-cased**: The |bert-base-multilingual-cased|[12] (also known as Multilingual BERT or mBERT) is pre-trained on cased text in the top 104 languages with the largest wikipedias and has a total 179M parameters with 12 transformers blocks, 768 hidden layers and 12 attention head.

This model accepts |[CLS]| as a first special input token, followed by a sequence of words as input. The input is subsequently sent to the next layer. |[CLS]| here stands for Classification. Each layer applies self-attention and communicates the outcome to the next encoder using a feedforward network.

Word Sequence Layer

This layer is used to generate a word sequence representation. It takes the sequence of embedding vectors of the text sequence as input and captures the context of the surrounding words to generate new vectors. The different sequence processing models used in this layer are:

- **Bi-LSTM**: Bi-LSTM stands for bidirectional LSTM, which indicates that the signal propagates both forward and backward in time. A Bi-LSTM converts the input sequence from the opposite direction to a hidden forward sequence and a backward hidden sequence. The encoded vector is formed by the concatenation of the final forward and backward hidden unit outputs. In this study, we employ a simple LSTM model. A single Bi-LSTM with 300 hidden units is used in the model. This layer processes word embedding in 300 dimensions. The output is subjected to two dense layers of size 100 and 1 at each time step. A dropout of 0.5 is used in the recurrent layer.
- **CNN**: The CNN component is used to capture character-level features. The model uses a convolution and a max-pooling layer for each word to extract a new feature vector utilising per-character feature vectors such as character embeddings and (optionally) character type. After that, the output is treated in two thick layers. The amount of characters in the word correlates to the time axis.
- **Multi-CNN**: This combines four layers of the CNN layer. The word embeddings generated by this multi-CNN network are then concatenated with 300-dimension learnable word embeddings from prior models. A Dense layer processes the result of concatenating the output from these layers into a single vector.
- **CRF**: CRF [57] utilizes undirected graphical models to calculate the conditional probability of values of output nodes when the values of the input nodes are known. Conditional probability is defined for a state sequence $S = \{S_1, S_2, \ldots, S_n\}$ given a corresponding observation sequence $O = \{O_1, O_2, \ldots, O_n\}$, is defined as:

$$P(S \mid O) = \frac{1}{NF} exp\left(\sum_{i=1}^{n}\sum_{k}\lambda_k f_k\left(S_{i-1}, S_i, O, i\right)\right) \tag{5}$$

here, $f_k\left(S_{i-1}, S_i, O, i\right)$ is real-valued and represents the feature function whose weights are denoted by λ_k, and the aim of the training is to effectively learn these weights. In order to make the conditional probability over all state sequences S normalized, the exponent term are divided by a normalization factor (NF) which is the sum of conditional probabilities corresponding to all the state sequences and is calculated through dynamic programming,

$$NF = \sum_{S} exp\left(\sum_{i=1}^{n}\sum_{k}\lambda_k f_k\left(S_{i-1}, S_i, O, i\right)\right) \tag{6}$$

The objective function for the training process is given as:

$$L = \sum_{i=1}^{n} log\left(P\left(S^{(i)}, O^{(i)}\right)\right) - \sum_{k}\frac{\lambda_k^2}{2\sigma^2} \tag{7}$$

The aim of the training process is to maximize this penalized log-likelihood function. Here, the second term helps in optimizing the algorithm, by making the log-likelihood function strictly convex. In light of the sequence tagging task, the sentences represent the state sequence and the corresponding tags for that sentence represent the observation sequence.

Inference Layer

This final layer is made up of softmax feedforward networks that generate the probability distribution corresponding to each word of the sequence over the list of all the tags. We take the tag with the highest probability as the predicted tag during the prediction stage for each word.

Experimental Setup

Here we talk about the different approaches and evaluation metrics. We start with the baseline approaches followed by our proposed approaches. We then describe the metrics

[12] https://github.com/google-research/bert/blob/master/multilingual.md

used to measure and compare the performances of different approaches.

Baseline Approaches

We included baseline models from [33, 34]. Jamatia et al. [34] used word embedding (GloVe) with Bi-LSTM layer and character level RNN with 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 author. For learning sequential data, here we have utilized the open-source neural sequence labeling toolkit NCRF++ [13].

Joshi and Joshi [33] used a character, word, and sub-word representation with CNN, Multi CNN, and Bi-LSTM layer. We have applied 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 was transmitted through a dense layer to obtain the final prediction.

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 must be fed into the Softmax layer, and the label is output with the highest probability via the Softmax layer, yielding each word category. Figure 9 shows the model architecture.

Input Representation Layer or Embedding layer: BERT

BERT is a contextual language model introduced by Devlin 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 which is absent in the dictionary 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.

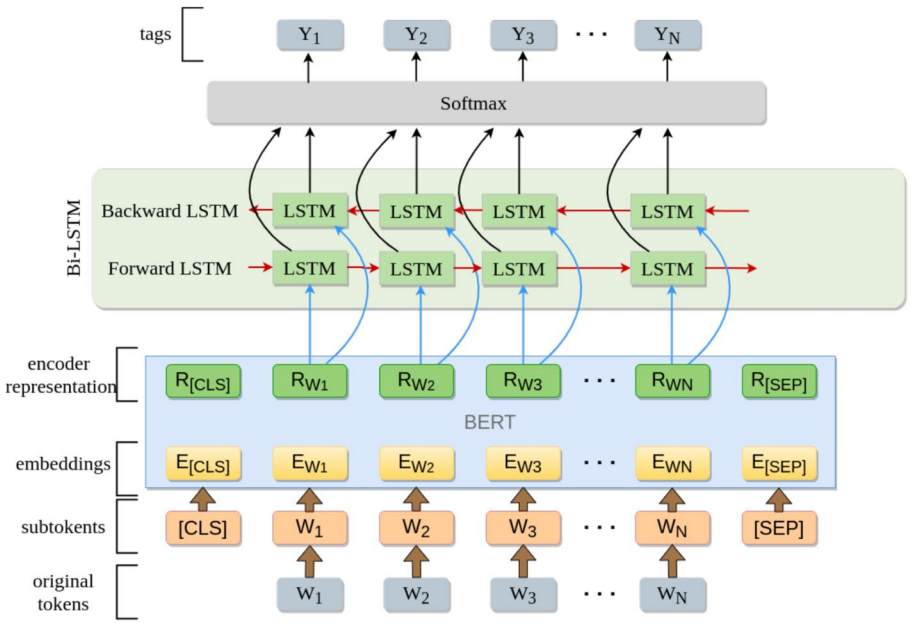


Fig. 9 Model architecture of our proposed system

[13] https://github.com/jiesutd/NCRFpp

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{8}$$

In equation (8), Q, K, and V represent the word vector matrix of natural hazard text, and d_k is the embedding dimension.

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. Token embedding refers to the embedding of the current word, Segment Embedding refers to the index embedding of the phrase in which the current word is positioned, and Position Embedding refers to the index embedding of the current word position.

Context Encoder Layer or Word Sequence Layer: BiLSTM

The BiLSTM is an LSTM extension that combines a forward and backward LSTM network to process sequences and connects the network to the output layer. The BiLSTM structure provides the output layer to acquire contextual information from the past (backward) and future (forward) at the same time. Furthermore, the BiLSTM 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. These gates take the input *x* and the past hidden state $h^{(t-1)}$ in the following equations.

$$i^{(t)} = \sigma(W^{ix} \cdot x^{(t)} + W^{ih} \cdot h^{(t-1)} + b_i) \tag{9}$$

$$f^{(t)} = \sigma(W^{fx} \cdot x^{(t)} + W^{fh} \cdot h^{(t-1)} + b_f) \tag{10}$$

$$o^{(t)} = \sigma(W^{ox} \cdot x^{(t)} + W^{oh} \cdot h^{(t-1)} + b_o) \tag{11}$$

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:

$$g^{(t)} = tanh(W^{gx} \cdot x^{(t)} + W^{gh} \cdot h^{(t-1)} + b_g) \tag{12}$$

$$c^{(t)} = f^{(t)} * c^{(t-1)} + i^{(t)} * g^{(t)} \tag{13}$$

$$h^{(t)} = tanh(c^{(t)}) * o^{(t)} \tag{14}$$

$h^{(t)}$ is a hidden state that is now utilized to determine what the cell should input, forget, and output in the next time step. In the above equations, the input gate is represented by $i^{(t)}$. The Forget gate is represented by $f^{(t)}$. The output gate is denoted by $o^{(t)}$. The cell state is denoted by $c^{(t)}$. The weight matrices are W^{ix}, W^{ih}, W^{fx}, W^{fh}, W^{ox}, W^{oh}, and the biases are b_i, b_f, and b_o. σ have used as the sigmoid function, whereas the hyperbolic tangent function is abbreviated as $\tanh$.

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

$$H_i = \left[\overrightarrow{h_{ft}} \cdot \overleftarrow{h_{bt}}\right] \tag{15}$$

Inference or Tag Prediction Layer: Softmax Function

The prediction layer generates a tag sequence corresponding to the input word sequence. After obtaining the hidden state for each word from the BiLSTM layer, we map each hidden state to a *k*-dimensional vector, where *k* is the number of possible language tags. This results in a matrix of shape $n \times k$, where *n* is the number of words in the input sequence. The Softmax function is then applied to each row to convert these scores into a probability distribution over the language tags.

Given the fixed maximum input size constraint of 512 tokens for BERT, we kept the original sentence length within this limit. The BERT model outputs contextual embeddings of size 768 for BERT-base and 1024 for BERT-large, depending on the variant used. To further refine the representations and capture contextual dependencies, we incorporated a BiLSTM layer, which yielded embeddings of size 512. Finally, a linear layer was utilized to map the BiLSTM outputs to the class level, producing the desired output size corresponding to the number of classes in the classification task. In our experiment, different transformer-based models are used with fine-tuning enabled to adapt the pretrained language model to the specific task. The hidden layer size of the sequence tagger is set to 256. The model is trained using a learning rate of 0.0001, a mini-batch size of 32, and for a maximum of 20 epochs. The models are trained on the training dataset and evaluated by predicting the labels for the test dataset.

Evaluation Metrics

We evaluate and compare every model's performance in terms of precision, recall, F_1-score, and accuracy for highly imbalanced label distribution, the definitions of which are listed below.

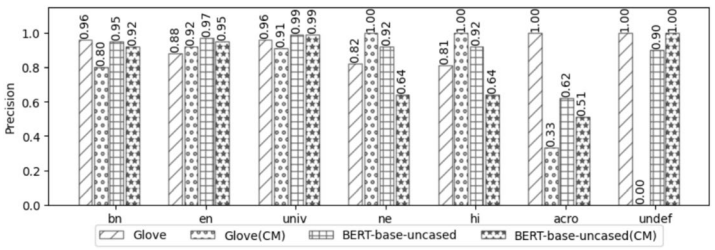


Fig. 10 Graphical comparison of Precision for baseline and best performance model on ICON_POS (BN-EN) dataset

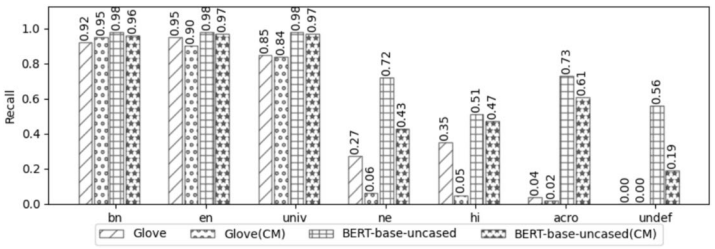


Fig. 11 Graphical comparison of Recall for baseline and best performance model on ICON_POS (BN-EN) dataset

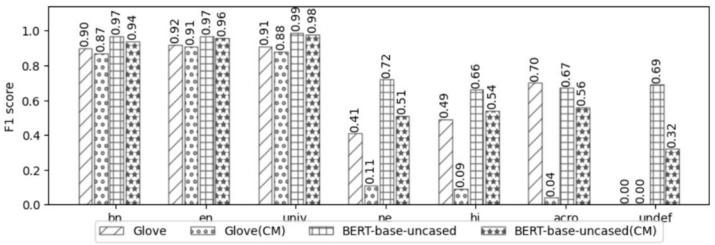


Fig. 12 Graphical comparison of F_1 score for baseline and best performance model on ICON_POS (BN-EN) dataset

1. Precision is the ratio of true-positives (TP) to the sum of TP and false-positives (FP).

$$Precision(P) = \frac{TP}{TP + FP} \tag{16}$$

2. Recall is the ratio of true-positives (TP) to the sum of TP and false-negatives (FN).

$$Recall(R) = \frac{TP}{TP + FN} \tag{17}$$

3. F_1-score is the balanced harmonic mean of precision and recall and used to have a composite idea of precision and recall.

$$F_1 = \frac{2 * R * P}{R + P} \tag{18}$$

4. *Weighted_*F_1: It is the weighted version of the average F_1-scores where each class is weighted by the number of samples from that class.
5. Accuracy is the ratio of no. of correct predictions to the total number of original entities i.e.

$$Accuracy = \frac{\#\ \text{correct}\ \ \text{predictions}}{\#\ \text{test-instances}} \tag{19}$$

Results

The dataset contains both monolingual and code-mixed sentences (See the corpus statistics, Table 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.

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. BERT models are trained for maximum 150 epochs depending on the learning rate 0.001. The architectural diagram of BERT model is shown in Fig. 9. BERT based experiments are performed on Google's Colab[14]. PyTorch deep learning library is used to implement the models. We also use HuggingFace's transformers to fine-tune pre-trained BERT based models.

We use accuracy, precision, recall, and F_1 scores to analyze the performance of the baseline model and the top-performing model on the entire dataset and the code-mixed dataset (CM). Figures 10, 11 and 12 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, omit them in the above figures. Table 3 shows

[14] https://colab.research.google.com

Table 3 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)	

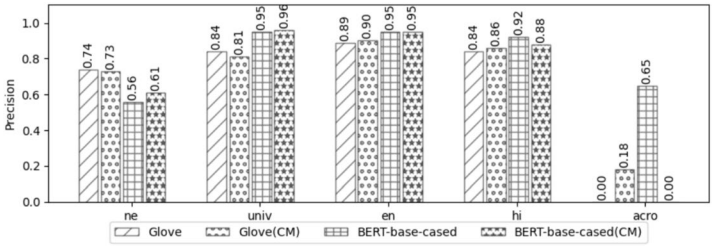


Fig. 13 Graphical comparison of Precision for baseline and best performance model on ICON_POS (HI-EN) dataset

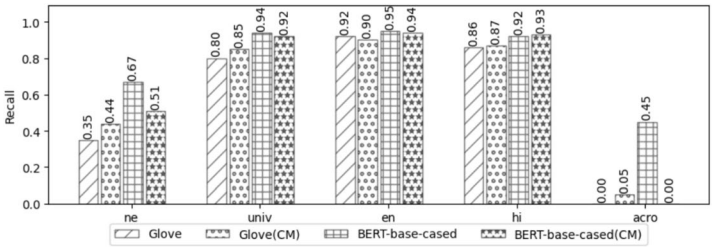


Fig. 14 Graphical comparison of Recall for baseline and best performance model on ICON_POS (HI-EN) dataset

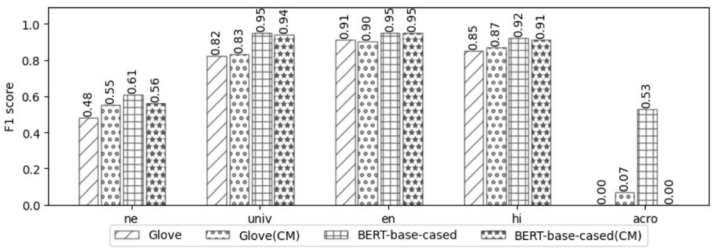


Fig. 15 Graphical comparison of F_1 score for baseline and best performance model on ICON_POS (HI-EN) dataset

Table 4 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)	

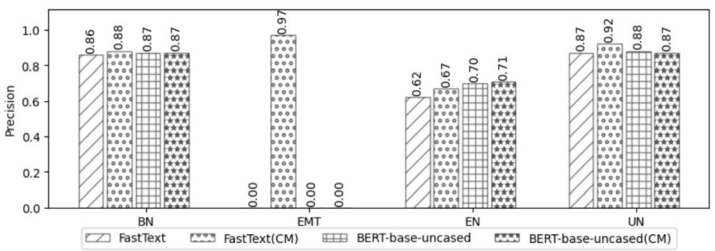


Fig. 16 Graphical comparison of Precision score for baseline and best performance model on ICON_SAIL (BN-EN) dataset

the class distributions for the 7200 tokens contained in the test data.

Overall class level system performance on the ICON_POS (HI-EN) dataset is shown in Figs. 13, 14 and 15. The mixed tags in the test dataset are too few to be identified by any model and, therefore, are omitted in the figures above. Table 4 shows the class distributions for the 8475 tokens contained in the test data.

Overall class level system performance on the ICON_SAIL (BN-EN) dataset is shown in Figs. 16, 17 and 18. The number of MIX tags in the test dataset is insufficient for any model to recognize them. As a result, they have been removed from the above figures. Table 5 shows the class distributions for the 57,410 tokens contained in the test data.

Overall class level system performance on the ICON_SAIL (HI-EN) dataset is shown in Figs. 19, 20 and 21. Table

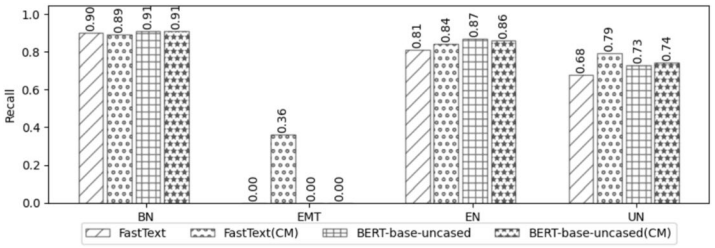


Fig. 17 Graphical comparison of Recall score for baseline and best performance model on ICON_SAIL (BN-EN) dataset

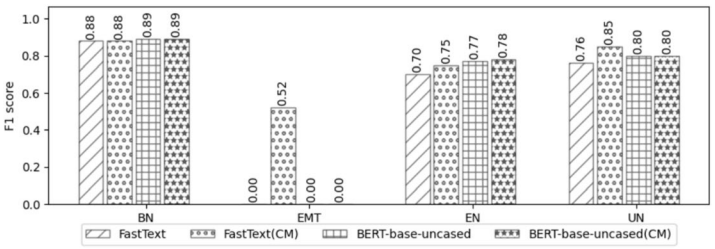


Fig. 18 Graphical comparison of F_1 score for baseline and best performance model on ICON_SAIL (BN-EN) dataset

Table 5 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)	

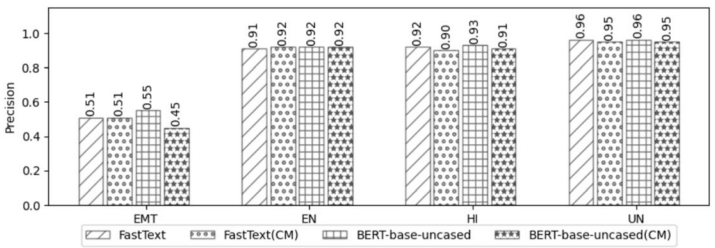


Fig. 19 Graphical comparison of Precision score for baseline and best performance model on ICON_SAIL (HI-EN) dataset

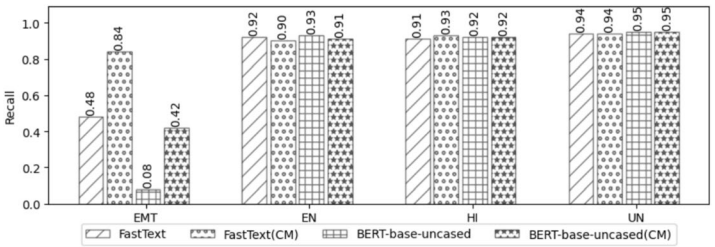


Fig. 20 Graphical comparison of Recall score for baseline and best performance model on ICON_SAIL (HI-EN) dataset

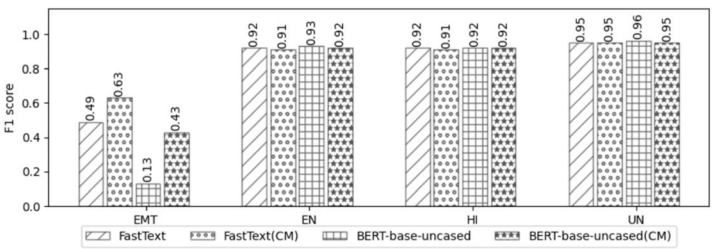


Fig. 21 Graphical comparison of F_1 score for baseline and best performance model on ICON_SAIL (HI-EN) dataset

Table 6 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)	

6 shows the class distributions for the 85,494 tokens contained in the test data.

Figures 22, 23 and 24 show the overall system performance of class level on the LinCE (HI-EN) dataset. The development dataset has inadequate foreign words (fw), unknown (unk), mixed, and ambiguous tags for any model

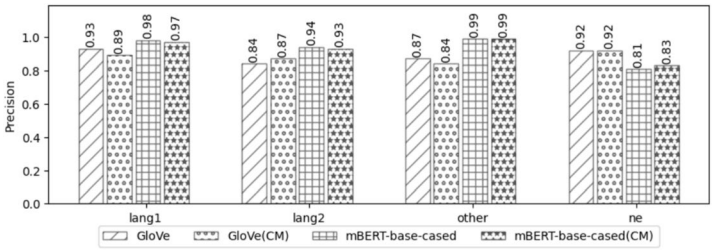


Fig. 22 Graphical comparison of Precision score for baseline and best performance model on LinCE (HI-EN) dataset

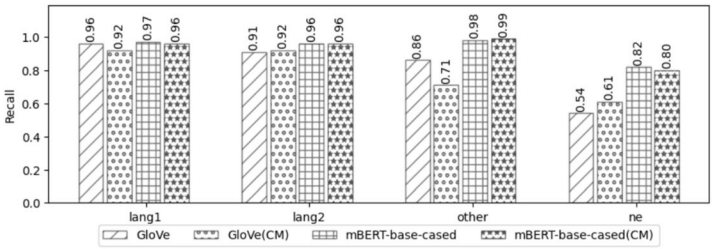


Fig. 23 Graphical comparison of Recall score for baseline and best performance model on LinCE (HI-EN) dataset

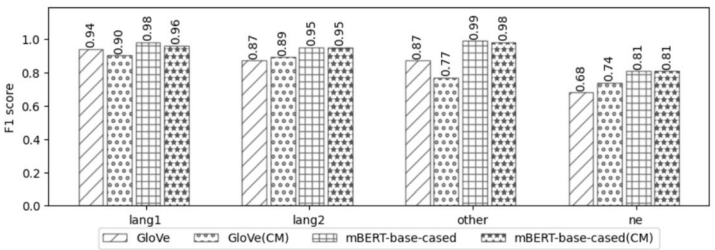


Fig. 24 Graphical comparison of F_1 score for baseline and best performance model on LinCE (HI-EN) dataset

Table 7 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)	

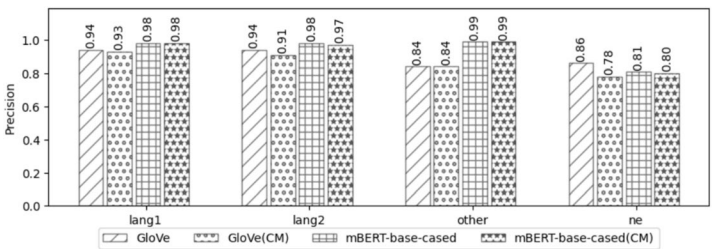


Fig. 25 Graphical comparison of Precision score for baseline and best performance model on LinCE (ES-EN) dataset

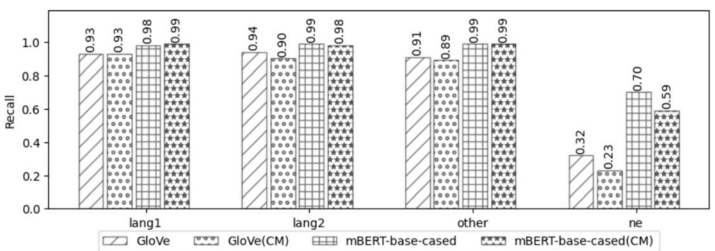


Fig. 26 Graphical comparison of Recall score for baseline and best performance model on LinCE (ES-EN) dataset

to recognise. As a result, they have been eliminated from the figures listed above. Table 7 shows the class distributions for the 15,446 tokens contained in the development data.

Figures 25, 26 and 27 show the overall system performance of class level on the LinCE (ES-EN) dataset. The quantity of foreign words (fw), unknown (unk), mixed, and ambiguous tags in the development sample are minimal for any model to recognize. As a result, they have been removed from the above figures. Table 8 shows the class distributions for the 40,391 tokens contained in the development data.

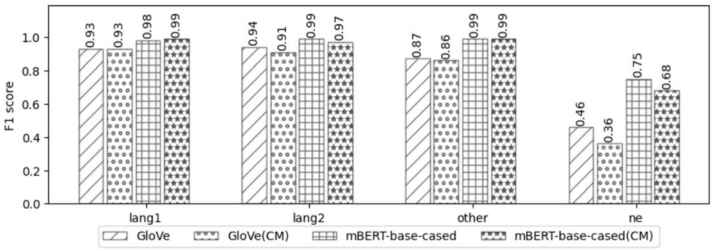


Fig. 27 Graphical comparison of F_1 score for baseline and best performance model on LinCE (ES-EN) dataset

Table 8 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)	

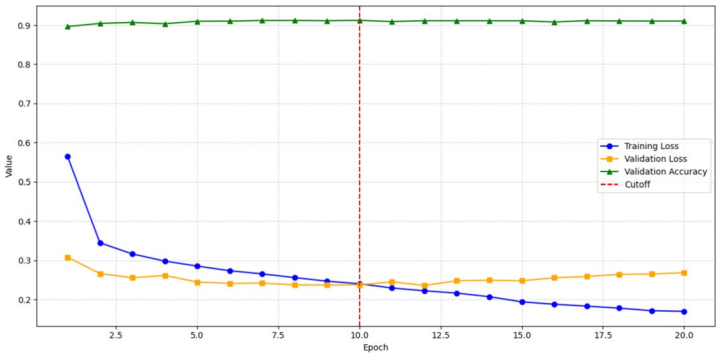


Fig. 28 Training and validation loss, and validation accuracy across epochs for ICON_SAIL (HI-EN) dataset

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 Figs. 10, 11 and 12, 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 cased 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 **93.02**% F_1 score on BERT base uncased embedding with Bi-LSTM layer (Table 11). Initially, we experimented with various epoch settings, including 10, 20, 30, 40, and 50 epochs. However, we observed that after a certain point, the validation loss started to increase, indicating potential overfitting. To address this, we implemented an early stopping criteria, which enabled us to identify the optimal number of training epochs and save the best-performing model accordingly. Figure 28 shows the training and validation loss along with validation accuracy across each epochs. BERT base uncased model with Bi-LSTM layer has a F_1 score of **82.75**% and emerges as our best performing model for ICON_SAIL (BN-EN) dataset. In Figs. 16, 17 and 18, 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 **96.72**% F_1 score on mBERT base cased embedding with Bi-LSTM layer (Table 13). Here also, we experimented with various epoch settings, including 10, 20, 30, 40, and 50 epochs. And we find the same trend observed earlier. So, we implemented an early stopping criteria save the best-performing model accordingly. The findings are acquired after 11 epochs of training on the data sets. Increasing the number of epochs to 30, 40, and 50 had no effect. Figure 29 shows the training and validation loss along with validation accuracy across each epochs. Multilingual BERT base cased model with Bi-LSTM layer has a F_1 score of **98.34**% 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

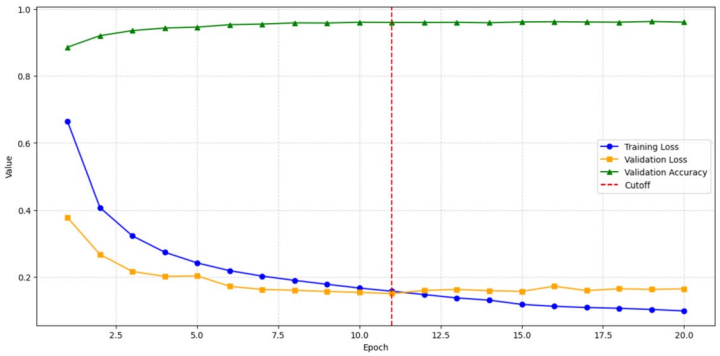


Fig. 29 Training and validation loss, and validation accuracy across epochs for LinCE (ES-EN) dataset

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 (Fig. 30) for both the models show that many tokens which are wrongly classified in the GloVe model (Fig. 30a), are not present in the BERT model (Fig. 30b). Our model does not predict any intra-word code-mixed data like *Facebooke* (a Bengali term that means “in Facebook" and contains the nameentity *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 employs the WordPiece tokenization technique to effectively reduce the occurrence of unknown tokens, which likely contributed to its success in handling the out-of-vocabulary (OOV) problem.

The inability of the baseline system to capture context information was 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* and *bar* 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*. As native speakers, 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*. All of these 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. Below is an example illustrating the model’s output for a code-mixed Hindi-English sentence:

- **Input:** r8 esa hmari eco wali mam krti hai kbi usne kisi b student ko 10 se above nh die
- **Output:** EN HI HI HI HI EN HI HI HI HI HI HI EN HI UN HI EN HI HI

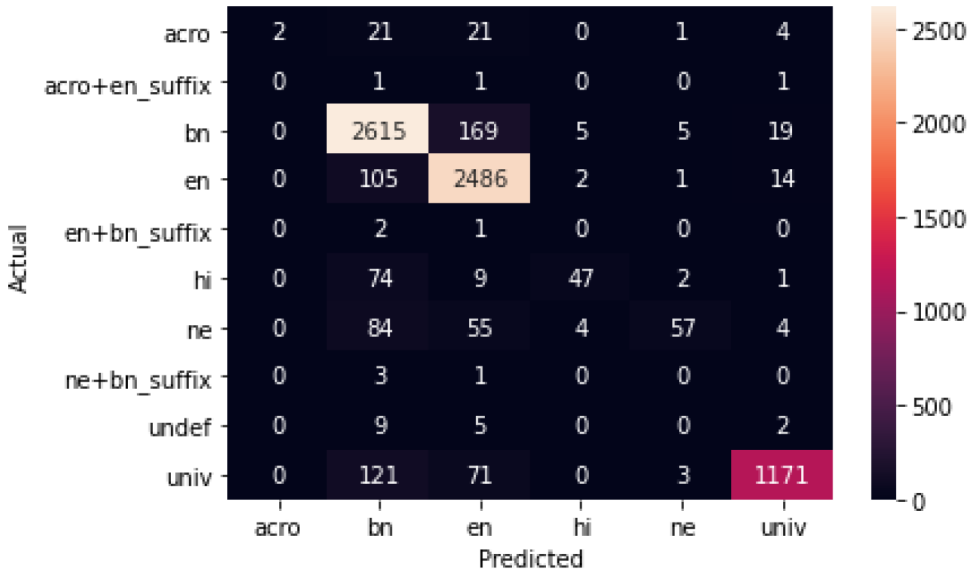
In this example, the model correctly identifies the language for almost all tokens. The only misclassification occurs for the word “eco", which should be labeled as EN (English), but the model predicts it as HI (Hindi). Such cases are primarily due to ambiguous context or the influence of neighboring Hindi tokens.

We also applied different models but only XLM-Roberta show some promising results. rest of all models performs well better than baseline but could not outperform.

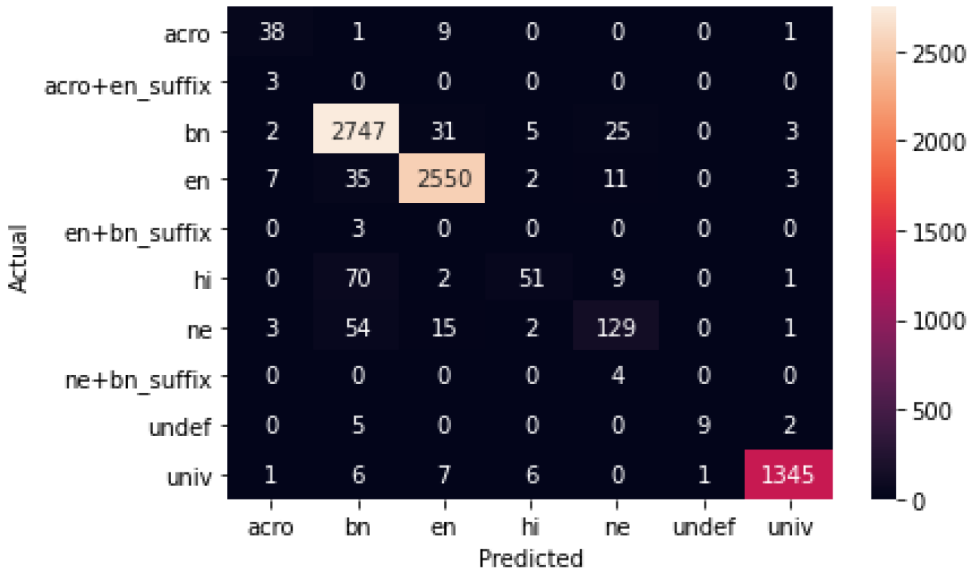
Comparison with Baseline

We compared the performance of our proposed models with that of baseline [33] and [34] models in Tables 11, 12 and 9, 10 respectively. We observed that our proposed models performed better for both the datasets. The overall performance margin for ICON_POS (BN-EN) data was 8.12% and for ICON_POS (HI-EN) data was 6.41%. On the other hand, for ICON_SAIL (HI-EN) and ICON_SAIL (BN-EN) dataset, we have received 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



(a) GloVe model



(b) BERT model

Fig. 30 Confusion Matrices for the baseline and proposed models on the ICON_POS (BN-EN) Corpus Test Set

monolingual sentences from the dataset and train only on code-mixed sentences, a performance deterioration was observed across all models and language pairs during the evaluation.

We also conducted a paired-sample t-test to determine whether the performance difference between our approach and the baseline approach is significant. We perform one-sided t-test on class wise F_1-score to validate the significance of our proposed method against the baseline methods. The statistical significant results are denoted by the symbol $*$ at level of significance $(\alpha) = 0.05$.

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 (Tables 9, 10). HI-EN data had comparatively higher CMI, M-index, and I-index (Fig. 6).

On the other hand, for the ICON_SAIL dataset, BN-EN performs much below its HI-EN counterparts (Tables 11 and 12). This observation can be attributed to the fact that training data was comparatively less and the test data was comparatively more in case ICON_SAIL BN-EN data.

Table 9 Comparison of baseline with best proposed model on ICON_POS (BN-EN) Data

Embedding	Model	F1-score$_{ALL}$(%)	F1-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
XLM-roberta-base	Bi-LSTM	94.70	**94.12**
XLM	Bi-LSTM	91.20	90.75
t5	Bi-LSTM	92.34	91.68
roberta	Bi-LSTM	93.10	91.45
gpt2	Bi-LSTM	93.87	91.58
distilbert	Bi-LSTM	91.90	90.12

Table 10 Comparison of baseline with best proposed model on ICON_POS (HI-EN) Data

Embedding	Model	F1-score$_{ALL}$(%)	F1-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
XLM-roberta-base	Bi-LSTM	93.40	91.32
XLM	Bi-LSTM	90.18	89.41
t5	Bi-LSTM	91.70	90.16
roberta	Bi-LSTM	92.04	91.10
gpt2	Bi-LSTM	92.37	91.29
distilbert	Bi-LSTM	91.50	89.78

LinCE dataset shows the best figures for both HI-EN and ES-EN (Tables 13, 14). For HI-EN, LinCE has the least code-mixing (CMI, M-index and I-index) among the three datasets (Fig 8). 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've already shown that our proposed model often predicts the proper language tag, yet the gold standard annotation data sometimes predicts it incorrectly.

Table 11 Comparison of baseline with best proposed model on ICON_SAIL (HI-EN) dataset

Embedding	Model	F1-score$_{ALL}$(%)	F1-score$_{CM}$(%)
Word2vec	Bi-LSTM	92.39	92.18
Word2vec	CNN	91.90	91.85
Word2vec	Multi CNN	92.04	91.92
FastText	BiLSTM	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**
XLM-roberta-base	Bi-LSTM	92.80	91.47
XLM	Bi-LSTM	91.15	90.43
t5	Bi-LSTM	91.50	90.12
roberta	Bi-LSTM	91.94	90.59
gpt2	Bi-LSTM	91.78	90.49
distilbert	Bi-LSTM	91.63	90.71

Table 12 Comparison of baseline with best proposed model on ICON_SAIL (BN-EN) dataset

Embedding	Model	F1-score$_{ALL}$(%)	F1-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	76.67
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**
XLM-roberta-base	Bi-LSTM	82.60	81.41
XLM	Bi-LSTM	81.19	80.76
t5	Bi-LSTM	81.32	80.82
roberta	Bi-LSTM	81.64	80.59
gpt2	Bi-LSTM	81.43	80.81
distilbert	Bi-LSTM	81.13	80.78

During evaluation, we observed that different BERT variant models provide similar performance in the language identification task. However, the 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 is 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

Table 13 Comparison of baseline with best proposed model on LinCE (HI-EN) development data

Embedding	Model	F1-score$_{ALL}$(%)	F1-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
XLM-roberta-base	Bi-LSTM	96.70	94.44
XLM	Bi-LSTM	96.10	94.03
t5	Bi-LSTM	95.80	95.43
roberta	Bi-LSTM	95.91	95.53
gpt2	Bi-LSTM	96.06	95.62
distilbert	Bi-LSTM	95.81	95.23

Table 14 Comparison of baseline with best proposed model on LinCE (ES-EN) development data

Embedding	Model	F1-score$_{ALL}$(%)	F1-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
XLM-roberta-base	Bi-LSTM	98.32	97.86
XLM	Bi-LSTM	98.00	97.54
t5	Bi-LSTM	97.32	96.85
roberta	Bi-LSTM	97.90	97.48
gpt2	Bi-LSTM	97.86	97.22
distilbert	Bi-LSTM	97.95	97.37

Table 15 Effect of BiLSTM layer

Dataset	Without BiLSTM	With BiLSTM	Improvement
ICON_SAIL (BN-EN)	81.03	81.88	.85
ICON_SAIL (HI-EN)	92.42	93.02	0.60
ICON_POS (BN-EN)	94.55	94.85	0.30
ICON_POS (HI-EN)	92.58	92.79	0.21
LinCE (HI-EN)	96.40	96.72	0.32
LinCE (ES-EN)	98.20	98.34	0.14

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 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.

In Table 15, our experiments demonstrated that incorporating BiLSTM 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 BiLSTM 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 pre-processing techniques to optimize performance in similar natural language processing tasks.

Conclusions and Future Works

In this paper, we attempted 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 BiLSTM model on top of BERT neural representations of code-mixed data emerged as the best performing model. For the language pairs analysed, the deep learning model with fine-tuned pre-trained word embedding model outperformed the classic CRF approach. Modifying the input representation helps to obtain the highest results. The collection consisted of monolingual and code-mixed sentences. Thus, we initially trained our model with both monolingual and code-mixed sentences, and then proceeded to code-mixed sentences. Though there was an obvious decline in performance across all models and language pairs when only code-mixed sentences were considered compared to a combination of mono-lingual and code-mixed sentences, our models provided better performances over the state-of-the-art models. Our work opens several exciting avenues for future exploration as given below.

- While most existing research focuses on code-mixing involving two languages, it is imperative to investigate the impact of incorporating three or more languages into the mix, which presents unique challenges due to the increased complexity of language interactions and potential ambiguities.
- We aim to delve deeper into studying the influence of code-mixing levels on the performance of language identification models. We can gain valuable insights

Table 16 Examples of incorrect predictions

Word	Actual tag	Predicted tag	Word	Actual tag	Predicted t
salman	NE	HI	h	HI	EN
door	HI	EN	me	HI	EN
is	HI	EN	aap	NE	HI
gun	EN	HI	hi	HI	EN
be	EN	HI	rahulgandhi	NE	UN
q	HI	EN	yr	HI	EN
to	HI	EN	komal	NE	HI
panch	HI	EN	haiiiiiiiiiii	EN	HI

into the model's robustness and adaptability by analyzing varying degrees of code-mixing (e.g., word-level, phrase-level, sentence-level).

- We plan to expand our research to include more low-resource Indian language pairs with high linguistic similarity, which will necessitate addressing the challenges of limited training data and potential morphological overlap between languages. By tackling these diverse yet critical aspects, we hope to contribute to advancing robust and language-agnostic code-mixed language identification systems.
- We plan to explore the use of more recent transformer-based multilingual models, which have demonstrated strong performance on cross-lingual and low-resource language tasks. These models, especially when fine-tuned on code-mixed or social media data, have the potential to significantly enhance the performance of language identification systems. Incorporating such models will not only strengthen the benchmark but also allow for a deeper understanding of their effectiveness in handling the complexities of code-mixed language in real-world scenarios.
- While our BiLSTM model achieved promising results, investigating other architectures like ensemble or LLM could yield further improvements and hence worth exploring.
- We plan to extend our study to include additional Indian language pairs, including those from non-Indo-Aryan language families such as Dravidian or Tibeto-Burman. This will help assess the adaptability and scalability of our approach across linguistically diverse and typologically distinct language combinations. Exploring unseen or low-resource language pairs will also provide insights into the model's robustness and generalization capabilities.
- One can also analyze how language identification performance varies across domains (e.g., social media, technical documents) where code-mixing patterns might differ.

Appendix

See Table 16.

Funding This study was funded by IIT (BHU), Varanasi, INDIA.

Data Availability The data used in this study can be accessed upon request to the owner.

Declarations

Conflicts of Interest The authors declare that they have no conflict of interest.

References

1. Hymes D. Foundations in sociolinguistics: an ethnographic approach. Lang Soc. 1975;4(3):352–61. https://doi.org/10.1017/S0047404500006758.
2. Bokamba EG. Are there syntactic constraints on code-mixing? World Englishes. 1989;8(3):277–92. https://doi.org/10.1111/j.1467-971X.1989.tb00669.x.
3. Myers-Scotton C. Common and uncommon ground: social and structural factors in codeswitching. Lang Soc. 1993;22(4):475–503.
4. Joshi AK. Processing of sentences with intra-sentential code-switching. In: Proceedings of the 9th International Conference on Computational Linguistics, COLING '82, Prague, Czechoslovakia, July 5-10, 1982, pp. 145–150. ACADEMIA, Publishing House of the Czechoslovak Academy of Sciences, ??? (1982). https://aclanthology.org/C82-1023/
5. Paolillo JC. Language choice on soc.culture.punjab. In: Electronic Journal of Communication/La Revue Electronique de Communication, vol. 6 (1996). https://eric.ed.gov/?id=EJ550464
6. Cavnar WB, Trenkle JM. N-gram-based text categorization. In: In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, pp. 161–175 (1994)
7. Çetinoglu Ö, Schulz S, Vu NT. Challenges of computational processing of code-switching. In: Diab MT, Fung P, Ghoneim M, Hirschberg J, Solorio T (eds) Proceedings of the Second Workshop on Computational Approaches to Code Switching@EMNLP 2016, Austin, Texas, USA, November 1, 2016, pp. 1–11. Association for Computational Linguistics, ??? (2016). https://doi.org/10.18653/V1/W16-5801.
8. Rijhwani S, Sequiera R, Choudhury M, Bali K, Maddila CS. Estimating code-switching on Twitter with a novel generalized word-level language detection technique. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1971–1982. Association for Computational Linguistics, Vancouver, Canada (2017). https://doi.org/10.18653/v1/P17-1180. https://aclanthology.org/P17-1180
9. Bali K, Sharma J, Choudhury M, Vyas Y. "I am borrowing ya mixing ?" an analysis of English-Hindi code mixing in Facebook. In: Proceedings of the First Workshop on Computational Approaches to Code Switching, pp. 116–126. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/W14-3914. https://aclanthology.org/W14-3914
10. Takawane G, Phaltankar A, Patwardhan V, Patil A, Joshi R, Takalikar MS. Language augmentation approach for code-mixed text classification. Nat Lang Process J. 2023;5:100042. https://doi.org/10.1016/j.nlp.2023.100042.

11. Chanda S, Mishra A, Pal S. Sentiment analysis of code-mixed dravidian languages leveraging pretrained model and word-level language tag. Nat Lang Process. 2025;31(2):477–99. https://doi.org/10.1017/nlp.2024.30.
12. Mustonen S. Multiple discriminant analysis in linguistic problems. Stat Methods Linguist. 1965;4:37–44.
13. Rau MD. Language identification by statistical analysis. PhD thesis, Naval Postgraduate School (1974)
14. Henrich P. Language identification for the automatic grapheme-to-phoneme conversion of foreign words in a german text-to-speech system. In: Proceedings of EUROSPEECH (1989)
15. Vitale T. An algorithm for high accuracy name pronunciation by parametric speech synthesizer. Comput Linguist. 1991;17(3):257–76.
16. Souter C, Churcher G, Hayes J, Hughes J, Johnson S. Natural language identification using corpus-based models. HERMES-J Lang Commun Bus. 1994;13:183–203.
17. Giguet E. Categorization according to language: A step toward combining linguistic knowledge and statistic learning. In: International Workshop of Parsing Technologies (IWPT'95) (1995)
18. Kikui GI. Identifying the coding system and language of on-line documents on the internet. In: COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics (1996)
19. Mather LA. A linear algebra approach to language identification. In: International Workshop on Principles of Digital Document Processing, pp. 92–103 (1998). Springer
20. Elworthy D. Language identification with confidence limits. arXiv preprint cs/9907010 (1999)
21. Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge University Press, ??? (2008). https://doi.org/10.1017/CBO9780511809071. https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf
22. Nguyen D, Doğruöz AS. Word level language identification in online multilingual communication. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A Meeting of SIGDAT, a Special Interest Group of The ACL, pp. 857–862. ACL, ??? (2013). https://aclanthology.org/D13-1084/
23. Barman U, Das A, Wagner J, Foster J. 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, pp. 13–23. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/W14-3902. https://aclanthology.org/W14-3902
24. Das A, Gambäck B. Identifying languages at the word level in code-mixed indian social media text. In: Sharma, D.M., Sangal, R., Pawar, J.D. (eds.) Proceedings of the 11th International Conference on Natural Language Processing, ICON 2014, Goa, India, December 18-21, 2014, pp. 378–387. NLP Association of India, ??? (2014). https://aclanthology.org/W14-5152/
25. Vyas Y, Gella S, Sharma J, Bali K, Choudhury M. POS tagging of english-hindi code-mixed social media content. In: Moschitti A, Pang B, Daelemans W (eds) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A Meeting of SIGDAT, a Special Interest Group of The ACL, pp. 974–979. ACL, ??? (2014). https://doi.org/10.3115/V1/D14-1105.
26. Xia M.X. Codeswitching language identification using subword information enriched word vectors. In: Diab MT, Fung P, Ghoneim M, Hirschberg J, Solorio T (eds.) Proceedings of the Second Workshop on Computational Approaches to Code Switching@EMNLP 2016, Austin, Texas, USA, November 1, 2016, pp. 132–136. Association for Computational Linguistics, ??? (2016). https://doi.org/10.18653/V1/W16-5818.
27. Chittaranjan G, Vyas Y, Bali K, Choudhury M. Word-level language identification using CRF: code-switching shared task report of MSR india system. In: Diab MT, Hirschberg J, Fung P, Solorio T (eds.) Proceedings of the First Workshop on Computational Approaches to Code Switching@EMNLP 2014, Doha, Qatar, October 25, 2014, pp. 73–79. Association for Computational Linguistics, ??? (2014). https://doi.org/10.3115/V1/W14-3908.
28. Veena PV, Kumar MA, Soman KP. Character embedding for language identification in hindi-english code-mixed social media text. Computación y Sistemas **22**(1) (2018). https://doi.org/10.13053/CYS-22-1-2775
29. Jhamtani H, Bhogi S.K, Raychoudhury V. Word-level language identification in bi-lingual code-switched texts. In: Aroonmanakun W, Boonkwan P, Supnithi T (eds) Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation, PACLIC 28, Cape Panwa Hotel, Phuket, Thailand, December 12-14, 2014, pp. 348–357. The PACLIC 28 Organizing Committee and PACLIC Steering Committee / ACL / Department of Linguistics, Faculty of Arts, Chulalongkorn University, ??? (2014). https://aclanthology.org/Y14-1041/
30. Ansari MZ, Khan S, Amani T, Hamid A, Rizvi SRA. Analysis of part of speech tags in language identification of code-mixed text. In: Proceedings of International Conference on Advancements in Computing and Management (ICACM) (2019). https://doi.org/10.2139/ssrn.3462976. https://ssrn.com/abstract=3462976
31. Thara S, Poornachandran P. Code-mixing: A brief survey. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2382–2388 (2018)
32. Jauhiainen T, Lui M, Zampieri M, Baldwin T, Lindén K. Automatic language identification in texts: a survey. J Artif Int Res. 2019;65(1):675–82. https://doi.org/10.1613/jair.1.11675.
33. Joshi R, Joshi R. Evaluating input representation for language identification in hindi-english code mixed text. CoRR **abs/2011.11263** (2020) 2011.11263
34. Jamatia A, Das A, Gambäck B. Deep learning-based language identification in english-hindi-bengali code-mixed social media corpora. J Intell Syst. 2019;28(3):399–408. https://doi.org/10.1515/JISYS-2017-0440.
35. Yasir M, Chen L, Khatoon A, Malik MA, Abid F. Mixed script identification using automated dnn hyperparameter optimization. Comput Intell Neurosci. 2021;2021(1):8415333. https://doi.org/10.1155/2021/8415333.
36. Dutta A. Word-level language identification using subword embeddings for code-mixed Bangla-English social media data. In: Sälevä J, Lignos C (eds) Proceedings of the Workshop on Dataset Creation for Lower-Resourced Languages Within the 13th Language Resources and Evaluation Conference, pp. 76–82. European Language Resources Association, Marseille, France (2022). https://aclanthology.org/2022.dclrl-1.10/
37. Aguilar G, Kar S, Solorio T. LinCE: A centralized benchmark for linguistic code-switching evaluation. In: Proceedings of the 12th Language Resources and Evaluation Conference, pp. 1803–1813. European Language Resources Association, Marseille, France (2020). https://aclanthology.org/2020.lrec-1.223
38. Molina G, AlGhamdi F, Ghoneim M, Hawwari A, Rey-Villamizar N, Diab M, Solorio T. Overview for the second shared task on language identification in code-switched data. In: Proceedings of the Second Workshop on Computational Approaches to Code Switching, pp. 40–49. Association for Computational Linguistics, Austin, Texas (2016). https://doi.org/10.18653/v1/W16-5805. https://aclanthology.org/W16-5805
39. Solorio T, Blair E, Maharjan S, Bethard S, Diab M, Ghoneim M, Hawwari A, AlGhamdi F, Hirschberg J, Chang A, Fung P. Overview for the first shared task on language identification in code-switched data. In: Proceedings of the First Workshop on

Computational Approaches to Code Switching, pp. 62–72. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/W14-3907. https://aclanthology.org/W14-3907
40. Mave D, Maharjan S, Solorio T. Language identification and analysis of code-switched social media text. In: Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching, pp. 51–61. Association for Computational Linguistics, Melbourne, Australia (2018). https://doi.org/10.18653/v1/W18-3206. https://aclanthology.org/W18-3206
41. Khanuja S, Dandapat S, Srinivasan A, Sitaram S, Choudhury M. GLUECoS: An evaluation benchmark for code-switched NLP. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 3575–3585. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.329. https://aclanthology.org/2020.acl-main.329
42. Balouchzahi F, Butt S, Hegde A, Ashraf N, Shashirekha Hl, Sidorov G, Gelbukh A. Overview of CoLI-kanglish: Word level language identification in code-mixed Kannada-English texts at ICON 2022. In: Chakravarthi BR, Murugappan A, Chinnappa D, Hane A, Kumeresan PK, Ponnusamy R (eds) Proceedings of the 19th International Conference on Natural Language Processing (ICON): Shared Task on Word Level Language Identification in Code-mixed Kannada-English Texts, pp. 38–45. Association for Computational Linguistics, IIIT Delhi, New Delhi, India (2022). https://aclanthology.org/2022.icon-wlli.8
43. Lakshmaiah SH, Balouchzahi F, Anusha MD, Sidorov G. Coli-machine learning approaches for code-mixed language identification at the word level in kannada-english texts. Acta Polytech Hungar. 2022;19(10):123–41.
44. Hegde A, Balouchzahi F, Coelho S, H L S, Nayel HA. Coli@fire2023: Findings of word-level language identification in code-mixed tulu text. In: Proceedings of the 15th Annual Meeting of the Forum for Information Retrieval Evaluation. FIRE '23, pp. 25–26. Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3632754.3633075. https://doi.org/10.1145/3632754.3633075
45. Chanda S, Mishra A, Pal S. Advancing language identification in code-mixed tulu texts: Harnessing deep learning techniques. (2023). https://ceur-ws.org/Vol-3681/T4-6.pdf
46. Ojo O, Gelbukh A, Calvo H, Feldman A, Adebanji O, Armenta-Segura J. Language identification at the word level in code-mixed texts using character sequence and word embedding. In: Chakravarthi BR, Murugappan A, Chinnappa D, Hane A, Kumeresan PK, Ponnusamy R (eds) Proceedings of the 19th International Conference on Natural Language Processing (ICON): Shared Task on Word Level Language Identification in Code-mixed Kannada-English Texts, pp. 1–6. Association for Computational Linguistics, IIIT Delhi, New Delhi, India (2022). https://aclanthology.org/2022.icon-wlli.1/
47. Gundapu S, Mamidi R. Word level language identification in English Telugu code mixed data. In: Politzer-Ahles S, Hsu Y-Y, Huang C-R, Yao Y (eds) Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation. Association for Computational Linguistics, Hong Kong (2018). https://aclanthology.org/Y18-1021/
48. Sarma N, Singh SR, Goswami D. Switchnet: learning to switch for word-level language identification in code-mixed social media text. Nat Lang Eng. 2022;28(3):337–59. https://doi.org/10.1017/S1351324921000115.
49. Navya J, Bharathi Raja C, Shardul S. A survey of current datasets for code-switching research, 136–141 (2020). https://doi.org/10.1109/ICACCS48705.2020.9074205
50. Patra B.G, Das D, Das A. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task @ icon-2017. CoRR **abs/1803.06745** (2018) 1803.06745
51. Barnett R, Codó E, Eppler E, Forcadell M, Gardner-Chloros P, van Hout R, Moyer M, Torras MC, Turell MT, Sebba M, Starren M, Wensing S. The lides coding manual: A document for preparing and analyzing language interaction data version 1.1-july, 1999. International Journal of Bilingualism **4**(2), 131–132 (2000) https://doi.org/10.1177/13670069000040020101
52. Guzmán G.A, Serigos J, Bullock B.E, Toribio A.J. Simple tools for exploring variation in code-switching for linguists. In: Diab MT, Fung P, Ghoneim M, Hirschberg J, Solorio T (eds.) Proceedings of the Second Workshop on Computational Approaches to Code Switching@EMNLP 2016, Austin, Texas, USA, November 1, 2016, pp. 12–20. Association for Computational Linguistics, ??? (2016). https://doi.org/10.18653/V1/W16-5802.
53. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space (2013)
54. Mikolov T, Grave E, Bojanowski P, Puhrsch C, Joulin A. Advances in pre-training distributed word representations. In: Calzolari N, Choukri K, Cieri C, Declerck T, Goggi S, Hasida K, Isahara H, Maegaard B, Mariani J, Mazo H, Moreno A, Odijk J, Piperidis S, Tokunaga T (eds) Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA), ??? (2018). http://www.lrec-conf.org/proceedings/lrec2018/summaries/721.html
55. Pennington J, Socher R, Manning C.D. Glove: Global vectors for word representation. In: Moschitti A, Pang B, Daelemans W (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A Meeting of SIGDAT, a Special Interest Group of The ACL, pp. 1532–1543. ACL, ??? (2014). https://doi.org/10.3115/V1/D14-1162.
56. Devlin J, Chang M, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C, Solorio T (eds) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, ??? (2019). https://doi.org/10.18653/V1/N19-1423.
57. Lafferty JD, McCallum A, Pereira FCN. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. ICML '01, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)