ISSN 2007-9737

# An Effective Bi-LSTM Word Embedding System for Analysis and Identification of Language in Code-Mixed Social Media Text in English and Roman Hindi

Shashi Shekhar[1], Dilip Kumar Sharma[1], M.M. Sufyan Beg[2]

[1] GLA University,
Department of Computer Engineering and Applications,
India

[2] Aligarh Muslim University,
Department of Computer Engineering,
India

{shashi.shekhar, dilip.sharma}@gla.ac.in, mmsbeg@cs.berkeley.ed

**Abstract.** The paper describes the application of the code mixed index in Indian social media texts and comparing the complexity to identify language at word level using BLSTM neural model. In Natural Language Processing one of the imperative and relatively less mature areas is a transliteration. During transliteration, issues like language identification, script specification, missing sounds arise in code mixed data. Social media platforms are now widely used by people to express their opinion or interest. The language used by the users in social media nowadays is Code-mixed text, i.e., mixing of two or more languages. In code-mixed data, one language will be written using another language script. So to process such code-mixed text, identification of language used in each word is important for language processing. The major contribution of the work is to propose a technique for identifying the language of Hindi-English code-mixed data used in three social media platforms namely, Facebook, Twitter, and WhatsApp. We propose a deep learning framework based on cBoW and Skip gram model for language identification in code mixed data. Popular word embedding features were used for the representation of each word. Many researches have been recently done in the field of language identification, but word level language identification in the transliterated environment is a current research issue in code mixed data. We have implemented a deep learning model based on BLSTM that predicts the origin of the word from language perspective in the sequence based on the specific words that have come before it in the sequence. The multichannel neural networks combining CNN and BLSTM for word level language identification of code-mixed data where English and Hindi roman transliteration has been used. Combining this with a cBoW and Skip gram for evaluation. The proposed system BLSTM context capture module gives better accuracy for word embedding model as compared to character embedding evaluated on our two testing sets. The problem is modeled collectively with the deep-learning design. We tend to gift an in-depth empirical analysis of the proposed methodology against standard approaches for language identification.

**Keywords.** Language identification, transliteration, character embedding, word embedding, NLP, machine learning.

## 1 Introduction

Humans use natural language as their medium for communication. Natural Language Processing (NLP), is an area of Artificial Intelligence where we train the machine to understand and process the text to make human-computer interactions more efficient. Applications of NLP lies under several fields like machine translation, text processing, entity extraction and so on [1]. A large amount of data is now available on the Web as text. With the emergence of several social media platforms and the availability of a large amount of text data in them, NLP plays a great role in understanding and generating data today. The social media platforms are used widely today by people to discuss the interests, hobbies, reviews on products, movies and so on.

In earlier days, the language used in such platforms was purely English. Today mixing multiple languages together is a popular trend. These kinds of languages are called code-mixed language.

A large amount of textual data is available on the web. With the emergence of several social media platforms and the availability of a large amount of text data in them, NLP plays a great role in understanding and generating data today. The social media platforms are used widely today by people to discuss the interests, hobbies, reviews on products, movies and so on. In earlier days, the language used in such platforms was purely English. Today combining different languages together is a common phenomenon. These kinds of languages are called code-mixed language. An example of Hindi-English code-mixed text is described in the below sentences:

***Sentence 1:*** **GLA University** ka **course structure** kaisa *hai:*

NE/OOV E H E E H H

**Sentence 2:** Aray **Friend**, ek **super idea** hai mere paas:

H E H E E H H H

Here Hindi words are labeled as H and English word are labeled as E and Named entity as NE. We can observe from the example that the Hindi words, tagged as H, were written in Roman Script instead of Unicode characters. The above example has been cited to give an idea of code mixed data.

The paper presents a novel architecture, which captures information at both word level and context level to output the final tag for language identification in context to the word belongs to which language. For word level, we have used a multichannel neural network (MNN) inspired by the recent works of computer vision. Such networks have also shown promising results in NLP tasks like sentence classification [2]. For context capture, we used Bi-LSTM-CRF. The context module was tested more rigorously as in quite a few of the previous work, this information has been sidelined or ignored. We have experimented on Hindi-English (H-E) code mixed data. Hindi is the most popular spoken language of India.

Here Hindi words are written in Roman transliterated form using the English alphabet.

For processing monolingual text, the primary step would be Part-Of-Speech (POS), tagging of the text. However, in the case of social media text, the primary feature to be considered is the identification of the language particularly for code-mixed text [3]. The language identification for code-mixed text proposed in this paper is implemented using word embedding models. The term word embedding refers to the vector representation of the given data capturing the semantic relation between the words in the data. The work is a generalized approach because this system can be extended for other NLP applications since only word embedding features are considered. The work involves features obtained from two embedding models, word-based embedding and character-based embedding. A comparison of the performance of the two models with the addition of contextual information is performed in this paper. The machine learning [4] based classification is used for training and testing the system.

Framework for discovering user intend based on Hindi roman transliteration by identifying the word level language identification was addressed here. The remaining section of the paper is organized as follows: An overview of the related works on language identification in the multilingual domain is discussed in section 2. A discussion on the methodology proposed considering word embedding and character embedding method is discussed in section 3.The dataset description is stated in section 4. Section 5 describes the experimental evaluation and results obtained. Section 6, analyses the inferences obtained from the work done and a pointer towards the future work.

## 2 Related Research

In this section, some of the recent techniques regarding to the language transliteration and identification is listed and reviewed as follows: code-switching and mixing is a current research area in the field of language tagging. Language Identification (LID), is a primary task in many text processing applications and hence several

research is going on this area especially with the code-mixed data. King and Abney [5] used semi-supervised methods for building a word level language identifier. Nguyen and Dogru [6] used CRF model limited to bigrams for identifying the language. Logistic regression along with a module, which gives code-switching probability was used by Vyas et al. [7]. Das and Gamback [8] used various features like a dictionary, n-gram, edit distance and word context for identifying the origin of the word.

A shared task on Mixed Script Information Retrieval (MSIR) 2015 was conducted in which a subtask includes language identification of 8 code-mixed Indian Languages, Telugu, Tamil, Marathi, Bangla, Gujarati, Hindi, Kannada, and Malayalam, each mixed with English [9].

The MSIR language identification task was implemented by using machine learning based SVM classifier and obtained an accuracy of 76% [16]. Word level language identification was performed for English-Hindi using supervised methods in [10]. Naive Bayes classifier was used to identify the language of Hindi-English data and an accuracy of 77% was obtained [11].

Language Identification is also performed as a primary step to several other applications. [12], implemented a sentiment analysis system which utilized MSIR 2015 English-Tamil, English-Telugu, English-Hindi, and English-Bengali code-mixed dataset. Another emotion detection system was developed for Hindi-English data with machine learning based and Teaching Learning Based Optimization (TLBO), techniques [13]. Part-of-Speech tagging was done for English-Bengali-Hindi corpus including the language identification step in [14].

Since the code-mixed script is the common trend in the social media text today, many kinds of research are going on for the information extraction from such text. An analysis of the behavior of code-mixed Hindi-English Facebook dataset was done in [15]. POS Tagging technique was performed on code-mixed social media text in Indian languages [16].

A shared task was organized for entity extraction on code-mixed Hindi-English and Tamil-English social media text [17]. Entity extraction for code-mixed Hindi-English and Tamil-English dataset was performed with embedding models [18].

Sapkal et al. [19] have given the approach by the use of SMS, which is meant for communicating with others in minimal words. The regional language messages are printed using English alphabets due to the lack of regional keywords. This SMS language may fluctuate, which leads to miscommunication. The focus was on transliterating short form to full form. Zubiaga et al. [20] had mentioned language identification, as the mission of defining the language of a given text. On the other hand, certain issues like quantifying the individuality of similar languages in multilingualism document and analyzing the language of short texts are still unresolved. The below section describes the proposed methodology to overcome the research gap identified in the area of transliterated code mixed data. Alekseev et al. [29] consider word embedding as an efficient feature and proposed entity extraction for user profiling using word-embedding features.

## 3 Proposed Methodology for Language Identification

The code mixed data include the combination of the native script (familiar language) and the non-native script (unfamiliar language). Due to this combination, a massive number of complications arises while dealing with this mixed code. Language Identification is the main and the foremost problem identified in the mixed code data since every user will not be clear about every language recognized in the globe. The language identification arises when the text is written in different languages. This incorporates problems such as the script specifications, which is the possibility of different scripts between the source and target languages.

The proposed system is comprised of two modules. The first one is a multichannel neural network trained at the word level, while the second one is a simple bidirectional LSTM trained at the context level. The second module takes the input from the first module along with some other features to produce the language tag.

The proposed architecture illustrated in Figure 2 is inspired by [30, 31, 21] where the recent deep

ISSN 2007-9737



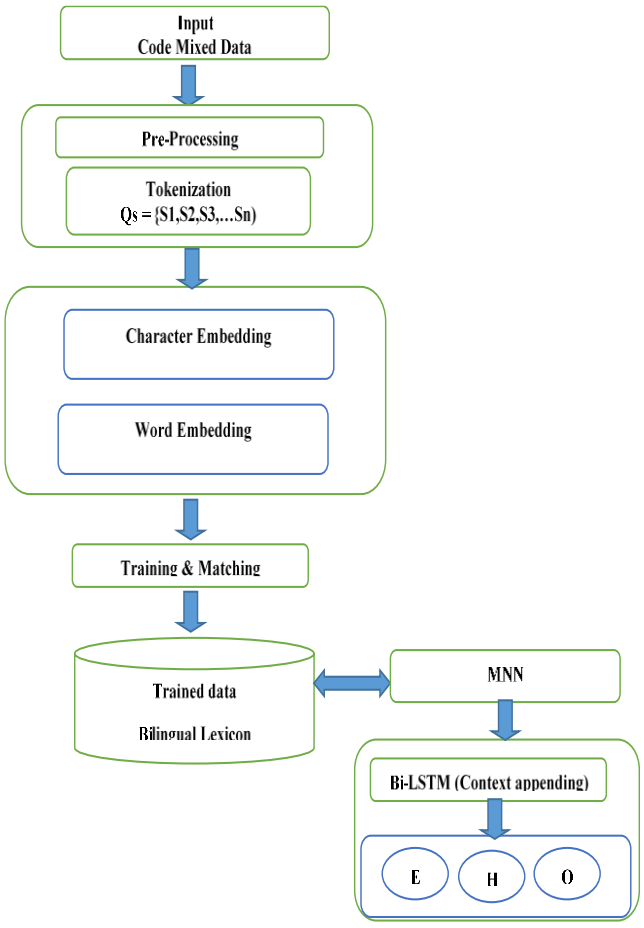


**Fig. 1.** Framework for word origin detection

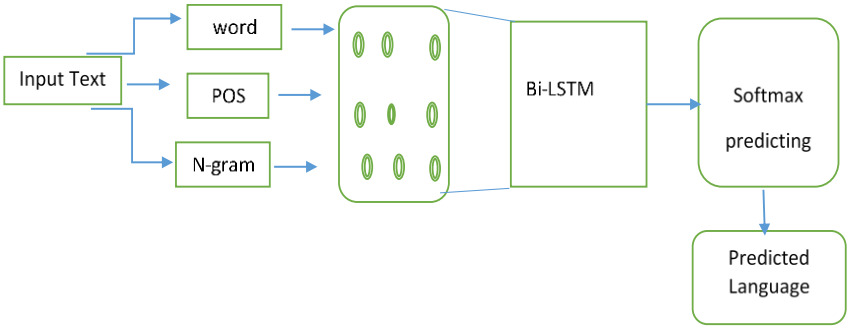


**Fig. 2.** Methodology of the proposed system

neural architectures developed for image classification tasks.

The proposed model uses a very similar concept for learning the language at the word level. This is because the architecture allows the network to capture representations of different types, which can be really helpful for NLP tasks for identifying the origin of a word in context to the language used

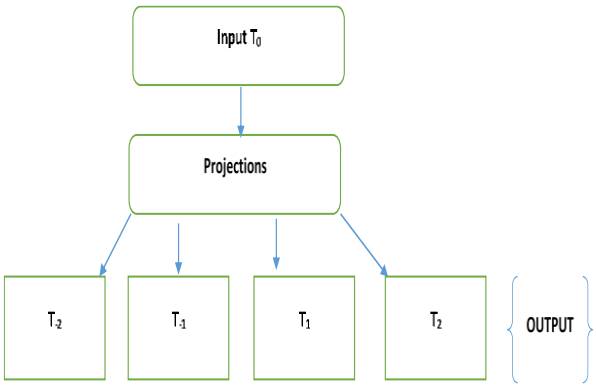


**Fig. 3.** Skip gram model

in code mixed data. The network we developed has 4 channels, the first three enters into a Convolution 1D (Conv1D) network [22], while the fourth one enters into a Long Short Term Memory (LSTM) network [23].

In this work, two systems were developed based on word-based embedding features and character-based context features.

For the character-based system, the same procedure as that of word-based is done except that the vectors are character vectors. The methodology of the proposed system is illustrated in Figure 2.

For the embedding to capture the word representation more effectively, additional data apart from the train and test data must be provided to the embedding model. The additional data used here is also a code-mixed Hindi-English social media data collected from other shared tasks.

The input for the word embedding will be the train data and the additionally collected dataset. The embedding model generates the vector of each vocabulary (unique), words present in the data. Along with extracting the feature vectors of the train data, its context information is also extracted.

The incorporation of the immediate left and right context features with the features of the current word is called 3-gram context appending. 5-gram features were also extracted, which is the extraction of features from two neighboring words before and after the current word. So if the vocabulary size of the training data is |V |, and the embedding feature size generated is 100 for each word, then after context appending with 3-gram features, a matrix of size |V| x 300 is obtained. 5-gram appending will result in a matrix of size |V| x 500.

The test data was also given to the embedding models. The data were then appended with the 3-gram and 5-gram context information. These are then fed to a machine learning based classifier, to train and test the system.

### 3.1 Word-Based Embedding Model

The word-based embedding model is used to find the feature vectors that are useful in predicting the neighboring tokens in a context. The feature vector for this model is generated using Skip-gram architecture of popular Word2vec package proposed by Mikolov et al. [24]. Apart from the skip-gram model, another architecture continuous Bag of Words (cBoW), is also present [24].

Word2vec is a predictive model that is used to produce word embeddings from raw text. It exists in two forms, the continuous Bag-of-Words model (cBoW) and the Skip-Gram model. Algorithmically, these two are similar, except that cBoW forecasts target words from source context words, whereas the skip-gram forecasts source context-words from the target words.

This gives the flexibility to use skip gram when we are having a large dataset and one can use cBoW for the smaller dataset. We focused on the skip-gram model for language identification at word level in the multilingual domain to answer (word belongs to which language) in the rest of this paper. The illustration of Skip-gram model is shown in Figure 3.

Here the input token is T0, which is fed to a log-linear classifier to predict the neighboring words. $T_{-2}$, $T_{-1}$, $T_1$ and $T_2$ are the words that are before and after the current word.

When the data is given to the Skip-gram model, it maximizes the average log probability, given by L, which is formulated as in Equation 1. In the equation, N is the total number of words in the train data and x is the context size. p is the softmax probability which is given using Equation 2:

$$L = \frac{1}{N} \sum_{n=1}^{N} \sum_{-x \le i \le x} logP(Tn + i | Tn) \,, \qquad (1)$$

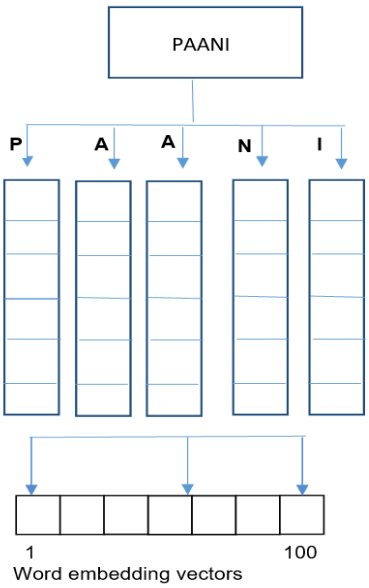


**Fig. 4.** Embedding Model

$$P(Tj|Tk) = \frac{\exp(V'T_j(VT_k))}{\sum_{w=1}^{w} \exp(V'T_j(VT_k))}, \quad (2)$$

where w is the vocabulary size, p(Tj |Tk), is the probability of occurrence of the next word. V' is the output vector representation. The dataset along with the additional dataset collected was given to the skip-gram model.

The vector sizes to be generated were fixed as 100. The skip-gram model generates a vector of size 1 × 100 for each vocabulary word available in the dataset.

From this, the vectors for the training data were extracted. The context appending features were then extracted from this file. The final training file for the classifier will consist of the tokens in the train data, their language tag and the 3-gram and 5-gram context feature vectors extracted.

Thus, three training files are generated with |V| x 101, |V| x 301 and |V| x 501 dimension. The test data with its corresponding context appended vectors are fed to the classifier for testing the system.

### 3.2 Character-Based Embedding Model

The procedure for character embedding is the same as that of skip-gram based word embedding. Each token in the train data gets splitted into characters and fed to the system. This will generate a vector for each character. The vector size to be generated was fixed as 100. The vectors generated for each character is used to create vectors for each token as per equation 3:

$$Y = x + S_h(W, C_{t-k,\ldots}C_{t+k,}C). \quad (3)$$

In regard to above, equation softmax parameters are denoted by x and S where h is the embedding features of character and word. C is the character vectors and W is the word vectors. $c_{t-k}$,... $c_{t+k}$, are the characters in the train data.

The figure 4 suggests that the word *PAANI* gets splitted into characters and given to the system to produce an embedding feature vector. The vectors are generated for each character in the word. These are then transformed to produce the character-based embedding vector of the word *PAANI* using Equation 3. The vectors for each token are then used to extract the context feature vectors. The feature vector with context features is appended along with the language tag and is fed to the classifier for training the system. The similar procedure is done for the test file. The vectors generated from character embedding model is then transformed as a context matrix for the test data. This context matrix with the test words is fed to the classifier for testing the system.

### 3.3 Design Consideration and Proposed Algorithm

– Each document must consist of words from two languages.

– All the documents must be in a single script. The chosen script, in this case, is ROMAN Script.

– In the Indian scenario, code-mixing is applicable between English and other Indian languages.

– The language used in the proposal is English and Hindi, where Hindi is represented using Roman, not Devanagari.

If the Hindi words are written in Devanagari script, it is then a simpler task to identify the language. This becomes non-trivial tasks to

identify the language as both Hindi and English are written using the same character set.

**Algorithm 1.** Proposed Algorithm for Language Identification

**Input:** Code Mixed Data

**Output:** Language of the input word.

***Algorithm steps***

**1.** Input term from Test Document

Let D=$W_1$, $W_2$….$W_n$ be a document where

$W_i$s are the words

**2.** Letter of the words{a-z}

**3.** // word level tagging task based on vectors

3.1 $L_b(W_i)$ chosen from Language L

Where L= {$L_E$,$L_H$,$L_O$)

//Check the frequencies of character in $W_i$ and $W_j$.

3.2 Generate Vectors for characters for $W_i$ and $W_j$

3.3 Apply Similarity metrics

$$Sim(X,Y) = \frac{\sum_{i=1}^{n} Xi\,Yi}{\sqrt{\sum_{i=1}^{n} Xi^2}\sqrt{\sum_{i=1}^{n} Yi^2}}$$

**4.** Label the word E- English or H-Hindi.

**5.** Check the Conf_Score of the classifier for Language $L_j$ on input $W_i$ as *0 ≤ Conf_Score ≤ 1*

Where Conf_Score is similarity metrics

sim($W_x$,$W_y$) x and y can be word in string

sim(x,y) ∈[0,1] for Normalization

sim(x,y) = 1 : exact match

sim(x,y) = 0 : “completely different“ x and y.

0 < sim(x,y) < 1 :approximate similarity

Threshold value =1 for exact match

1 matches $L_E$

< 1 matches $L_H$ OR $L_o$ Based on List condition $L_o$W

If $L_E$ matches LoW

$L=L_o$

**6.** Classify the Word as E, H or O.

**Table1.** Dataset ICON 2016 [25]

| Data | No. of Sentences | | No. of Tokens | |
|---|---|---|---|---|
| | Training data | Testing data | Training data | Testing data |
| Facebook | 772 | 111 | 20,615 | 2,167 |
| Twitter | 1,096 | 110 | 17,311 | 2,163 |
| WhatsApp | 763 | 219 | 3,218 | 802 |

**Table 2.** Sample data

| Data sample |
|---|
| amir se **hoti** hai, garib se **hotii** hai |
| door se **hotee** hai, qarib se ***hot***i hai |
| magar jahaan bhi ***hoti*** hai, ai mere dost |
| shaadiyaan to naseeb se ***hoti*** hai |
| **Mixed Script Data sample** |
| Party abhi baaki hai…….. |
| Party abhee baaki hai…….. |

## 4 Dataset Descriptions

The dataset used for this work is obtained from POS Tagging task for Hindi-English code-mixed social media text conducted by ICON 2016 [25].

The dataset contains the text of three social media platforms namely Facebook, Twitter and WhatsApp. The train data provided contains the tokens of the dataset with its corresponding language tag and POS tag.

The dataset used here for language identification is Indian language corpora used in the FIRE2014 (Forum for IR Evaluation) shared task on transliterated search. Data used for training the classifier consists of bilingual documents containing English and Hindi words in Romanized script for Bollywood Song Lyrics.

Complete database of songs consists of 63,000 documents in form of text file. (Dataset of FIRE MSIR). The below table shows the sample dataset showing various transliterated variations for non-English word and a second sample for mixed script data having words as English and transliterated Hindi words.

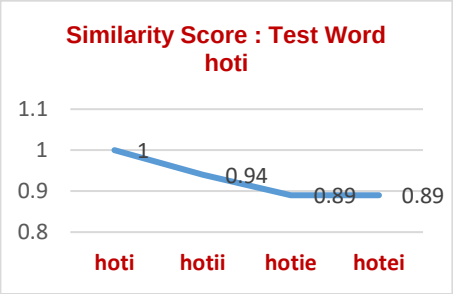


**Fig. 5 (a).** Word level similarity

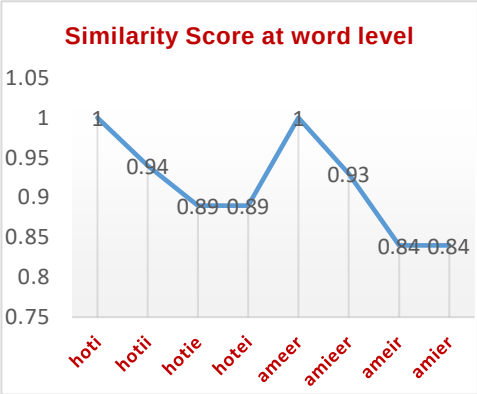


**Fig. 5 (b).** Word level similarity

**Table 3.** MI and CMI values

| Language set | MI | CMI |
|---|---|---|
| Hindi-English | 0.582 | 22.229 |

## 5 Evaluation and Experimental Results

The next section discusses the complete experimental part along with results and consequent discussions.

### 5.1 Experimental Results

The proposed algorithm for retrieving language of the word in code mixed data is evaluated on the basis of statistical measures and also evaluated using the machine learning approach. The below section provides the complete evaluation based on the statistical model. We performed two separate experiments on the code mixed data to rationalize the performance of the language, we have computed code-mixing patterns in the dataset on two metrics. This is being used to know the mixing patterns in the dataset.

The proposed system is analyzed and evaluated based on the following code mixing metrics.

**MI:** Multilingual index is a measure for word-count that quantifies the distribution variations of the language tags in a corpus of languages. Equation 4 defines the MI (Multilingual Index) as:

$$MI = x = \frac{1 - \sum P^2 J}{(k-1)\sum P^2 J}, \tag{4}$$

where k denotes the number of languages, Pj denotes the number of words in the language j over the number of words in the corpus. The value of MI resides between 0 and 1. Value of 0 relates monolingual corpus and 1 relates to the equal number of tokens from each language in a corpus.

**CMI:** Code-Mixing Index: At the phonetic level, this is calculated by discovering the most frequent language in the utterance and then counting the frequency of the words belonging to all other languages present. It is calculated using equation (5):

$$CMI = \frac{\sum_{i=1}^{n}(w_i) - max(w_i)}{n - u}, \tag{5}$$

where $\sum_{i=1}^{n} w_i$ is the sum of all languages present in the utterance, max{wi} is the maximum number of words exists from any language.(considering the case more than one language can have same maximum word count), n denotes total number of tokens, and u denotes the number of tokens for other language independent tags.

If an utterance only contains u (i.e., N=u) language independent tokens. Its index is considered to be zero. For other utterances, we use the normalization (multiply the value by 100) to acquire the digits in the range of 0 to 100.

The next wi are the tagged language words and max(wi) is the most prominent language words. Applying this equation we will get CMI=0 for monolingual utterances because max(wi = n − u). Equation 5 is normalized as below in equation (6):

$$CMI = \begin{cases} 100 \times \left[1 - \frac{max\{wi\}}{n-u}\right] & : n > u, \\ 0 & : n = u, \end{cases} \tag{6}$$

where wi are the words labelled with each language tag, max{wi} is the most prominent language words. By applying the above equation we will get a value of CMI as 0 for monolingual and a higher value of CMI designates high mixing of languages.

To understand the model, consider the following scenario, sentence S1 contains ten words. Five words are from Language L1 and remaining 5 words are from Language L2.

Applying equation 6 the CMI will be 100 × (1 – 5/10) = 50. However, another sentence S2 contains 10 words and each word is from a different language. The CMI = 100 × (1 – 1/10) = 90. It rightly reflects that S2 is highly mixed as every word belongs to a different language. This CMI value helps us to understand the level of code mixing available in the dataset. The table below describes the values obtained for MI and CMI for the corpus.

Secondly, we computed the similarity score based on the proposed algorithm on the dataset using the equation (7).It gives significance in labeling the word as either English or Hindi based on the frequency of the word. The proposed algorithm checks the Conf_Score of the classifier for Language Lj on input Wi as 0 ≤ Conf_Score ≤ 1, where Conf_Score is similarity metrics, sim(Wx,Wy) x and y can be the word in a string. The below section describes the different results obtained on the code mixed dataset for calculating the similarity score at word level and sentence level. Figure 5(a) and 5(b) describes the result obtained at word level for Hindi roman transliterated words in the corpus. Figure 6 plots the similarity at the sentence level. Figure 7 describes the sentence level language identification based on the proposed design and algorithm discussed in section 3.3:

$$Sim(X,Y) = \frac{\sum_{i=1}^{n} Xi\,Yi}{\sqrt{\sum_{i=1}^{n} Xi^2}\sqrt{\sum_{i=1}^{n} Yi^2}}, \qquad (7)$$

The next section describes the experimental evaluation based on applying BLSTM neural model. The dataset used for this work is obtained from POS Tagging task for Hindi-English code-mixed social media text conducted by ICON 2016 [25]. The dataset contains the text of three social media platforms namely Facebook, Twitter and

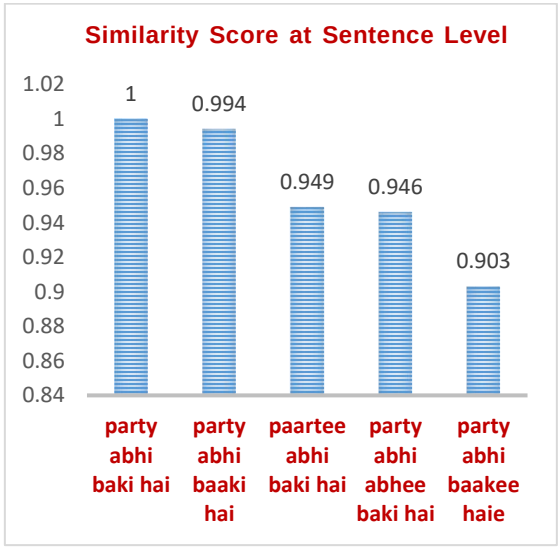


**Fig. 6.** Sentence level similarity

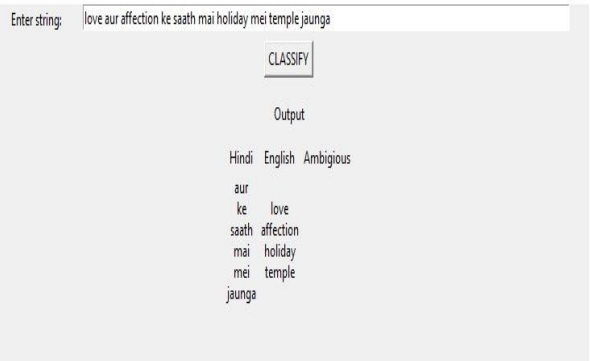


**Fig. 7.** Visualization of word level language identification by the statistical model

WhatsApp. We use the Hindi-English dataset for the experimental evaluation.

The labels used are summarized in Table 4**.** The training data contains the tokens of the dataset with its corresponding language tag and POS tag.

1. **E** indicates English words, for example: This, and, there.
2. **H** indicates Hindi words, for example: aisa, mera, tera.
3. **NE** indicates named entities like Person, Location and Organization, for example: Narendra Modi, India, Facebook.
4. **Other** indicates tokens containing special

characters and numbers, for example: @,#,0- 9.

5. **Ambiguous** indicates words used ambiguously in Hindi and English, for example: is, to, us.
6. **Mixed** indicates words of Hindi-English and Number combination, for example: MadamJi Sirji.
7. **Unk** indicates unrecognized words, for example: t.M , @.s,Ss.

All the seven tags are present in the Facebook dataset, where 'E', 'H', 'NE', 'Other' are the tags present in Twitter and Whatsapp data. The size of the training and testing data is summarized in Table 4. From the table, it can be observed that the average tokens per comment of Whatsapp training and testing data are very less than Facebook and Twitter data. This may be due to the fact that Facebook and Twitter data mostly contains news articles and comments which make the average tokens per comment count to be more while Whatsapp contains conversational short messages.

For generating the embedding vectors, more dataset has to be provided to obtain efficiently the distributional similarity of the data. The additional dataset collected along with the training data will be given to the embedding model. The Hindi-English additional code-mixed data were collected from Shared task on Mixed Script Information Retrieval (MSIR), conducted in the year 2016 [26] & 2015 [27] and shared task on Code-Mix Entity Extraction task conducted by Forum for Information Retrieval and Evaluation (FIRE), 2016 [28]. Most of the data collected for embedding is Hindi-English code-mixed Twitter data. The size of the dataset used for embedding is given in below table.

Context appending was done for each Facebook, Twitter and WhatsApp train as well as test data. These were given to the learning model for training and testing. The cross-validation accuracies obtained for Facebook, Twitter, and WhatsApp with 1-gram, 3-gram and 5-gram features for character-based embedding model and word-based embedding model is presented in below section.

When comparing the overall accuracy obtained for Facebook, Twitter, and WhatsApp, we can see

**Table 4.** Description of the labels for Hindi-English dataset

| Label | Description | Hindi-English % |
|---|---|---|
| E | English words only | 57.76 |
| H | Hindi words only | 20.41 |
| NE | Named Entity | 6.59 |
| Other | Symbols, Emoticons | 14.8 |
| Ambiguous | Can't determine whether Hindi or English | 0.27 |
| Mixed | Word of Hindi English in combination | 0.08 |
| Unk | Unrecognized word | 0.09 |

**Table 5.** Embedding dataset

| Number of sentences in the dataset used for embedding (Facebook, Twitter and WhatsApp) | |
|---|---|
| ICON2016 | 2631 |
| MSIR 2015 | 2700 |
| MSIR 2016 | 6139 |

**Table 6.** F measure obtained for Twitter

| Embedding Type | | E | H | NE |
|---|---|---|---|---|
| Character Embedding | 1 gram | 84.95 | 93.31 | 78.38 |
| | 3 gram | 85.34 | 93.44 | 77.12 |
| | **5 gram** | **85.38** | **93,49** | **80.27** |
| Word Embedding | 1 gram | 65.86 | 82.96 | 62.22 |
| | **3 gram** | **85.71** | **93.97** | **83.94** |
| | 5 gram | 85.42 | 93.16 | 78.15 |

**Table 7.** F measure obtained for Facebook

| Embedding Type | | E | H | NE |
|---|---|---|---|---|
| Character Embedding | 1 gram | 85.65 | 92.92 | 64.95 |
| | 3 gram | 86.45 | 93.36 | 65.02 |
| | **5 gram** | **85.47** | **92.55** | **65.05** |
| Word Embedding | 1 gram | 85.02 | 92.03 | 62.80 |
| | **3 gram** | **86.99** | **93.51** | **67.21** |
| | 5 gram | 85.15 | 92.47 | 61.03 |

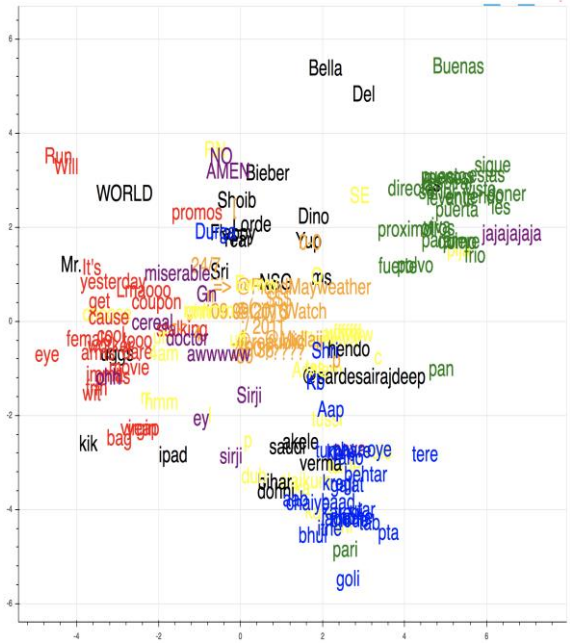


**Fig. 8.** Visualization of word representation learned by the Bi-LSTM model for Hindi-English

**Table 8.** F measure obtained for WhatsApp

| Embedding Type | | E | H | NE |
|---|---|---|---|---|
| Character Embedding | 1 gram | 52.42 | 80.15 | 28.57 |
| | **3 gram** | **54.99** | **80.26** | **37.70** |
| | 5 gram | 54.39 | 80.91 | 31.58 |
| Word Embedding | 1 gram | 50.42 | 79.62 | 40.00 |
| | **3 gram** | **60.88** | **81.98** | **40.27** |
| | 5 gram | 53.70 | 80.19 | 40.12 |

that the accuracy obtained is more with the word-based model as compared to character-based embedding model.

It can also be observed that in the word-based embedding model, 3-gram-based features give more accuracy than 1-gram and 5-gram context feature model while in character-based model 5-gram gives more accuracy than 1-gram and 3- gram.

When observing Table 6,7, and 8 shows the performance of Facebook, Twitter and WhatsApp Hindi-English code-mixed data, we can see that the F-score for language labels E-English, H-Hindi, NE-Named Entity is better using word embedding.

From the performance of data tabulated in Table 6, 7 and 8, it is clearly seen that the word embedding 3-gram based model gives a better score than other models. Table 6 holds label wise accuracy for Twitter data, Table 7 holds label wise accuracy for Facebook data and Table 8 holds label wise accuracy for WhatsApp data.

It can be observed from the table that 3-gram word embedding model gives significant accuracy in comparison to 1-gram and 5-gram word embedding and to character embedding model whereas in case of character gram model accuracy is better in 5-gram model except for WhatsApp accuracy where 5 gram shows better accuracy. This is because the system needs more context information to identify the language. That is why the 5-gram embedding gives a better result in the case of WhatsApp for character embedding techniques.

We tend to envision the representations learned by the RNN model by the word embeddings for the selected subset of words from datasets. The above result maps the labels to colors' indicating the defined seven parameters defined in table 4. The color encoding is summarized as follows: 1)Red for label E, 2)Blue for Label H, 3) Black for Label NE, 4) Orange for Label Others, 5) Purple for Label Ambiguous and Mixed, and 7) Yellow for Label Unk (Unrecognized word).

The proposed neural model gives a clearer separation between the different labeling parameters as defined in table 4 along with giving a crystal clear separation between the language Hindi and English used in the code mixed dataset. This result shows that this model can be scaled to detect language in code mixed data without any additional feature engineering.

## 6 Conclusions

The intricacy of language identification in code mixed and code switched data is governed by the following: data source, code switching and code mixing manners, and the relation between the languages involved. We find that the code mixing is more used in social media context as per the evaluation and experiments were undertaken in

this work. Code mixing metrics helps in identifying the code-mixing patterns across language pairs.

By analyzing the code mixing metrics we conclude that Hindi-English words are often mixed in our dataset. It would be a great idea to investigate the emerging trend of code switching and code mixing to bring conclusion about the behavioral patterns in the data of different sources like lyrics of songs, chat data having different language sets, blog data and scripts of plays or movies. We have implemented two different evaluation models: statistical model and neural based learning model and obtained competitive results for the identification of languages. This is probably due to the amount of training and testing data we have.

The results depict that the word embeddings are capable to detect the language separation by identifying the origin of the word and correspondingly mapping to its language label. The BLSTM system performs better for HIN-ENG language pairs. The BLSTM model captures long-distance dependencies in a sequence and this is in line with the observation made above for identifying word level language identification in code mixed data considering the context of the word belonging to labeled languages. Scaling this system to identify other characteristics in linguistics with different language dataset is a potential future direction to explore.

## References

1. **Weiscbedel, R., Carbonell, J., Grosz, B., Lehnert, W., Marcus, M., Perrault, R., & Wilensky, R. (1989).** White paper on natural language processing. *Association for Computational Linguistics,* pp. 481–493.
2. **Kim, Y. (2014)**. *Convolutional neural networks for sentence classification*. arXiv:1408.5882.
3. **Barman, U., Das, A., Wagner, J., & Foster, J. (2014).** *Code mixing: A challenge for Language Identification in the Language of Social Media*. *EMNLP´14*, Vol. 13, pp. 1–23.
4. **King, L., Baucom, E., Gilmanov, T., Kübler, S., Whyatt, D., Maier, W., & Rodrigues, P. (2014)**. The IUCL+ System: Word-Level Language Identification via Extended Markov Models. *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pp. 102–106. DOI:10.3115/v1/W14-3912.
5. **King, B. & Abney, S. (2013).** Labeling the languages of words in mixed-language documents using weakly supervised methods. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1110–1119.
6. **Dong, N. & Doŀruöz A.S. (2013).** Word level language identification in online multilingual communication. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 857–862.
7. **Vyas, Y., Gella, S., Sharma, J., Bali, K., & Choudhury, M. (2014).** Pos tagging of english-hindi code-mixed social media content. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 974–979. DOI:10.3115/v1/D14-1105.
8. **Das, A. & Gamback, B. (2014).** Identifying languages at the word level in code-mixed indian social media text. *ICON´14.*
9. **Sequiera, R., Choudhury, M., Gupta, P., Rosso, P., Kumar, S., Banerjee, S., Naskar, S., Bandyopadhyay, S., Chittaranjan, G., Das, A., & Chakma, K. (2015).** Overview of FIRE´15 Shared Task on Mixed Script Information Retrieval. *Proceedings of FIRE*, Vol. 1587, pp. 19–25.
10. **Jhamtani, H., Bhogi, S. K., & Raychoudhury, V. (2014).** Word-level language identification in bi-lingual code-switched texts. *28th Pacific Asia Conference on Language, Information and Computation*, pp. 348–357.
11. **Ethiraj, R., Shanmugam, S., Srinivasa, G., & Sinha, N. (2015).** NELIS - Named Entity and Language Identification System: Shared Task System Description. *Proceedings of FIRE*, Vol. 1587, pp. 43–46.
12. **Bhargava, R., Sharma, Y., & Sharma, S. (2016).** Sentiment Analysis for Mixed Script Indic Sentences. *International Conference on Advances in Computing, Communications and Informatics, ICACCI´16*, pp. 524–529. DOI:10.1109/ICACCI.2016.7732099.
13. **Sharma, S., Srinivas, P., & Balabantaray, R. (2016).** Emotion Detection using Online Machine Learning Method and TLBO on Mixed Script. *Language Resources and Evaluation Conference*, Vol. 10, No. 5, pp. 47–51.
14. **Barman, U., Wagner, J., & Foster, J. (2016).** Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modeling. *Proceedings of the Second Workshop on*

*Computational Approaches to Code Switchingpp*, pp. 30–39.

15. **Bali, K., Jatin, S., Choudhury, M., & Vyas, Y. (2014).** I am borrowing ya mixing?. An Analysis of English-Hindi Code Mixing in Facebook. *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pp. 116–126.
16. **Vyas, Y., Gella, S., Sharma, J., Bali, K., & Choudhury, M. (2014).** POS tagging of English-Hindi Code-Mixed Social Media Content. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 974– 979.
17. **Rao, P.R.K. & Devi, S. (2016).** CMEE-IL: Code Mix Entity Extraction in Indian Languages from Social Media Text@FIRE´16 - An Overview. *FIRE Workshops*, Vol. 1737, pp. 289–295.
18. **Remmiya-Devi, G., Veena, P.V., Anand-Kumar, M., & Soman, K. P. (2016).** AMRITA-CEN@FIRE 2016: Code-mix Entity Extraction for Hindi-English and Tamil-English tweets. *CEUR Workshop Proceedings*, Vol. 1737, pp. 304–308.
19. **Sapkal, K. & Shrawankar, U. (2016).** Transliteration of Secured SMS to Indian Regional Language. *Procedia Computer Science*, Vol. 78, pp. 748–755. DOI: 10.1016/j.procs.2016.02.048.
20. **Zubiaga, A., Vicente, I.S., Gamallo, P., Pichel, J.R., Alegria, I., Aranberri, N., & Fresno, V. (2015).** TweetLID: A benchmark for tweet language identification. *Language Resources and Evaluation,* Vol. 50, No. 4, pp. 729–766. DOI:10.1007/s10579-015-9317-4.
21. **Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., & Rabinovich, A. (2015).** Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1–9. DOI:10.1109/CVPR.2015.7298594.
22. **LeCun, Y., Haffner, P., Bottou, L., & Bengio. Y. (1999)**. Object recognition with gradient-based learning. *Shape, Contour and Grouping in Computer Vision*, pp. 319–345.
23. **Hochreiter, S. & Schmidhuber, J. (1997).** Long short-term memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780. DOI:10.1162/neco.1997.9.8.1735
24. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013).** *Distributed Representations of Words and Phrases and their Compositionality*. pp. 3111–3119.
25. **Jamatia, A. & Das, A. (2016).** Task Report: Tool Contest on POS Tagging for Code-mixed Indian Social Media (Facebook, Twitter, and Whatsapp) Text@ ICON 2016. *Conference International Conference on Natural Language Processing.*
26. **Banerjee, S., Chakma, K., Naskar, S., Das, A., Rosso, P., Bandyopadhyay, S., & Choudhury, M. (2016).** Overview of the Mixed Script Information Retrieval (MSIR) at FIRE-2016. *CEUR Workshop Proceedings,* Vol. 1737, pp. 94–99. DOI:10.1007/978-3-319-73606-8_3.
27. **Sequiera, R., Choudhury, P., Rosso, P., Kumar, S., Banerjee, S., Naskar, S., Bandyopadhyay, S., Chittaranjan, G., Das, A., & Chakma, K. (2015).** Overview of FIRE´15 Shared Task on Mixed Script Information Retrieval. *Post Proceedings of the Workshops at the 7th Forum for Information Retrieval Evaluation, Gandhinaga*, Vol. 1587, pp. 19–25.
28. **Srinidhi-Skanda, V., Singh, S., Remmiya-Devi, G., Veena, P.V., Anand-Kumar, M., & Soman, K. P. (2016)**. CEN@ Amrita FIRE 2016: Context based Character Embeddings for Entity Extraction in Code-Mixed Text. *CEUR Workshop Proceedings,* Vol. 1737, pp. 321–324.
29. **Alekseev, A. & Nikolenko, S. (2017).** Word embeddings for user profiling in online social networks. *Computación y Sistemas*, Vol. 21, No. 2. DOI:10.13053/cys-21-2-2734.
30. **Shekhar, S., Sharma, D.K., & Beg, M.S. (2018).** Hindi Roman Linguistic Framework for Retrieving Transliteration Variants using Bootstrapping. *Procedia Computer Science*, Vol. 125, pp. 59–67. DOI:10.1016/j.procs.2017.12.010.
31. **Veena, P.V., Anand-Kumar, M., & Soman, K. P. (2018).** Character Embedding for Language Identification in Hindi-English Code-mixed Social Media Text. *Computación y Sistemas*, Vol. 22, No. 1. DOI:10.13053/cys-22-1-2775.

*Article received on 20/02/2019; accepted on 25/07/2020.*
*Corresponding author is Shashi Shekhar.*