Improving Code-mixed POS Tagging Using Code-mixed Embeddings

Article *in* ACM Transactions on Asian Language Information Processing · March 2020

DOI: 10.1145/3380967

Nagesh Bhattu
National Institute of Technology Andhra Pradesh
21 PUBLICATIONS **77** CITATIONS
SEE PROFILE

N. Satya Krishna
National Institute of Technology, Warangal abd IDRBT
4 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Dvln Somayajulu
National Institute of Technology, Warangal
66 PUBLICATIONS **504** CITATIONS
SEE PROFILE

Improving Code-mixed POS Tagging Using Code-mixed Embeddings

S. NAGESH BHATTU, National Institute of Technology Andhra Pradesh, India
SATYA KRISHNA NUNNA, IDRBT and National Institute of Technology, India
D. V. L. N. SOMAYAJULU, National Institute of Technology and IIITDMKL, India
BINAY PRADHAN, International Institute of Information Technology, India

Social media data has become invaluable component of business analytics. A multitude of nuances of social media text make the job of conventional text analytical tools difficult. Code-mixing of text is a phenomenon prevalent among social media users, wherein words used are borrowed from multiple languages, though written in the commonly understood roman script. All the existing supervised learning methods for tasks such as Parts Of Speech (POS) tagging for code-mixed social media (CMSM) text typically depend on a large amount of training data. Preparation of such large training data is resource-intensive, requiring expertise in multiple languages. Though the preparation of small dataset is possible, the out of vocabulary (OOV) words pose major difficulty, while learning models from CMSM text as the number of different ways of writing non-native words in roman script is huge. POS tagging for code-mixed text is non-trivial, as tagging should deal with syntactic rules of multiple languages. The important research question addressed by this article is whether abundantly available unlabeled data can help in resolving the difficulties posed by code-mixed text for POS tagging. We develop an approach for scraping and building word embeddings for code-mixed text illustrating it for *Bengali-English, Hindi-English,* and *Telugu-English* code-mixing scenarios. We used a hierarchical deep recurrent neural network with linear-chain CRF layer on top of it to improve the performance of POS tagging in CMSM text by capturing contextual word features and character-sequence–based information. We prepared a labeled resource for POS tagging of CMSM text by correcting 19% of labels from an existing resource. A detailed analysis of the performance of our approach with varying levels of code-mixing is provided. The results indicate that the F1-score of our approach with custom embeddings is better than the CRF-based baseline by 5.81%, 5.69%, and 6.3% in *Bengali, Hindi*, and *Telugu* languages, respectively.

CCS Concepts: • **Information systems → Clustering and classification**; • **Computing methodologies → Information extraction**;

Additional Key Words and Phrases: Parts of speech tagging, code-mixed (multi-lingual) text, deep neural networks

Authors' addresses: S. N. Bhattu, National Institute of Technology Andhra Pradesh, Tadepalligudem-534101, Andhra Pradesh, India; email: nageshbhattu@nitandhra.ac.in; S. K. Nunna, IDRBT, Road No 1, Castle Hills, Masab Tank, Hyderabad-500057, Telengana and National Institute of Technology, Warangal-506004, Andhra Pradesh, India; email: satya.krishna.nunna@gmail.com; D. V. L. N. Somayajulu, National Institute of Technology, Warangal, Andhra Pradesh–506004, IIITDMKL, Karnool-518002, Andhra Pradesh, India; email: soma@nitw.ac.in; B. Pradhan, International Institute of Information Technology, Bhubaneswar-751003, Odisha, India; email: pradhan.binay@outlook.com.

2375-4699/2020/03-ART50 $15.00
https://doi.org/10.1145/3380967

ACM Reference format:
S. Nagesh Bhattu, Satya Krishna Nunna, D. V. L. N. Somayajulu, and Binay Pradhan. 2020. Improving Code-mixed POS Tagging Using Code-mixed Embeddings. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 19, 4, Article 50 (March 2020), 31 pages.
https://doi.org/10.1145/3380967

1 INTRODUCTION

The recent past has seen the phenomenal increase in textual sources of data chiefly contributed by social media platforms such as *Facebook*, *Twitter*, *Linkedin*, and *Whatsapp*.[1] The mobiles and hand-held devices further acted as catalysts in popularizing these tools. The success of social media tools is attributed to the virality and reach of messages across the user groups that is otherwise impossible. The short-living nature of these sources of information enables the multitude of users to keep observing, forwarding the messages, liked among their local social groups. These attributes have enticed a variety of businesses to run ad campaigns, cross-sell for popularizing their products. The analysis of text expressed in these forums remains the key to the success of such efforts.

Keeping the objectives of making social messages more viral and compelling, people use code-mixing to crisply convey their intent among the users of their social groups and to express their opinions emphatically. Code-mixing or code-switching occurs when a person uses multiple language words in a single sentence or an utterance, which is commonly seen in multilingual societies in the world [Parshad et al. 2016]. *Code-mixing* is defined as "the embedding of linguistic units such as phrases, words, and morphemes of one language into an utterance of another language" [Gysels 1992; Singh et al. 2018]. Depending on the level of mixing, code-mixing can be broadly classified into intra-sentential and inter-sentential. Example sentences are shown in Table 1 for both cases. Intra-sentential code-mixing involves a number of code-switching points and hence is more difficult to handle [Solorio and Liu 2008a]. We can see that the code-switching happens after every few words in intra-sentential code-mixing. Whereas, in inter-sentential code-mixing, one part of the sentence consists of *Hindi* words and another part is completely *English*, which is easier to handle. The analysis of CMSM text is an important research challenge from the perspectives of both Natural Language Processing (NLP) and Information Retrieval (IR) communities. There have been some research works in this direction, such as automatic word-level language identification for CMSM text [Barman et al. 2014; Solorio et al. 2014; Veena et al. 2018], sentiment analysis in CMSM text [Gupta et al. 2016; Rudra et al. 2016], document ranking in CMSM text [Kumar et al. 2016], parsing pipeline for *Hindi-English* CMSM text [Joshi et al. 2016; Sharma et al. 2016], and POS tagging for CMSM text [Ghosh et al. 2016; Pimpale and Patel 2016; Sarkar 2016; Sequiera et al. 2015; Solorio and Liu 2008b; Vyas et al. 2014]. There have been few works in the recent past for nurturing resources for CMSM text. ICON 2015 and ICON 2016[2] are shared tasks (tool context) on POS tagging for three different CMSM text datasets. The datasets contain text from three code-mixing scenarios (*Bengali-English*, *Hindi-English*, and *Telugu-English*). EMNLP 2014 also conducted a shared task for word-level language identification in CMSM text. The datasets contain the text from four code-mixing contexts (*English-Spanish*, *Mandarin-English*, *Nepali-English*, and *Arabic-Hindi*). Similarly, Rao and Devi [2016] presented Named Entity Recognition (NER) for CMSM text for two code-mixing scenarios (*Hindi-English* and *Tamil-English*). As code-mixing is a niche area of NLP, the datasets are not yet fully standardized and contain numerous tagging errors. This is especially a problem for linguistic tasks such as POS tagging.

[1] www.facebook.com, www.twitter.com, www.linkedin.com, www.whatsapp.com, etc.
[2] http://ltrc.iiit.ac.in/icon2016/, http://amitavadas.com/Code-Mixing.html.

Table 1. Examples for Inter- and Intra-Sentential Code-mixing Sentences with Word-level Language Tags

Intra-sentential:	Hi	bhayya	Bhai	Black	Friday	ki	deal	mein	mila	Kaafi	sastaa
lang-tag:	en	hi	hi	en	en	hi	en	hi	hi	hi	hi
Inter-sentential:	dost	Shadi	karlou	nahi	tou	Bihar	wale	le	jayege	aur	Shadi
lang-tag:	hi	hi	hi	hi	hi	ne	hi	hi	hi	hi	hi
	karwa	denge	....	Free	advice	to	all	Punjab	boys		
	hi	hi	univ	en	en	en	en	en	en		

Table 2. Example for Label Ambiguity in Code-mixed Text

Sentence1:	Rishi	Sharma	AAP	**TO**	DUDE	NIKLE	
Lnag-tag	ne	ne	hi	hi	en	hi	
POS tag:	N_NNP	N_NNP	PR_PRC	RP_RPD	N_NN	V_VM	
Sentence2:	something	big	reveal	aisa	**to**	kuch	nahi
Lnag-tag	en	en	en	hi	hi	hi	hi
POS tag:	RB_ALC	JJ	N_NN	RB_AMN	RP_RPD	RP_INTF	DT

1.1 POS Tagging for Code-mixed Social Media Text

POS tagging is crucial for many upstream NLP tasks. POS tagging is used to improve the performance in Chunking [Aslan et al. 2018; Atserias et al. 2006], NER [Liu and Zhou 2013; Ritter et al. 2011], Question Answering (QA) [Romeo et al. 2017], Dependency parsing [Galitsky 2016; Guo et al. 2010; McDonald et al. 2005; Nivre and Scholz 2004], and many other tasks. POS tagging involves identifying a label (i.e., part of speech) for each word in a sentence based on its definition and its relation with other adjacent words and semantic information of that sentence.

Many supervised and unsupervised POS tagging algorithms were developed for different languages since the work in Rabiner [1989]. Predominantly two types of methods are used in the design of POS taggers—namely, rule-based [Hindle 1989] and machine learning–based approaches [Ratnaparkhi 1996]. Rule-based algorithms are language-dependent and rely on good domain knowledge for effective working. Machine learning algorithms are language-independent, but they require large training data. The success of all state-of-the-art conventional machine learning methods, such as Hidden Markov Model (HMM), Conditional Random Fields (CRF) [Bach et al. 2018; Luo et al. 2015; Passos et al. 2014; Ratinov and Roth 2009], Support Vector Machine (SVM) [Tsochantaridis et al. 2005], and Recurrent Neural Network (RNN) [Athavale et al. 2016], depends on the usage of quality features (such as morphological and lexical language features) extracted from large labeled corpus. The labeled CMSM text for languages such as *Bengali-English*, *Hindi-English,* and *Telugu-English* is scarce. Jamatia et al. [2015] prepared the labeled CMSM text for these languages and then trained the CRF model with n-grams as additional features. It is noteworthy to see that CRF uses a log-linear model based on label-word and label-label features. The core of the algorithm involves dot product between features and parameters. POS tagging for CMSM text faces highly non-linear inference task. Consider the examples shown in Table 2: It is difficult to understand the POS tag of the word "to" (in bold letters), as its language is difficult to infer. The simple extension of such features does not yield good performance for CMSM text. This is partly because of the multitude of OOV words complicating the optimization approaches. Ritter et al. [2011] improved the performance of NER and POS tagging for social media text using *related word usage pattern* features, which are extracted using agglomerative clustering [Brown et al. 1992] on large unlabeled social media corpus. Instead of these high-dimensional discrete features, low-dimensional real-valued word features (word embeddings) offer better representations for the word. Word embeddings consist of contextual, syntactic, and semantic information of a word

[Bojanowski et al. 2017; Mikolov et al. 2013] and have shown their utility in various deep learning models for NLP tasks [Collobert and Weston 2008]. Many trained word embeddings for the monolingual text of different languages are currently available,[3] but not for CMSM text.

In the recent past, deep learning methods [Athavale et al. 2016] have been explored by researchers for sequence labeling tasks such as POS tagging and NER. Deep learning methods, when extended for the CMSM text, face few difficulties. The embeddings learned from the cleaner form of text become not as useful when used for CMSM text [Pratapa et al. 2018b]. The number of variations in which foreign language terms are spelled is too many. As code-mixing phenomena can happen in a large number of combinations, the number of datasets available for fostering research in this area is too few. The POS tagging to non-standardized text, such as CMSM text, is a non-trivial task due to the following reasons:

- Code-mixing causes the syntactic structure analysis to be difficult, as arbitrary portions of text follow different syntactic rules (depending on the language).
- Numerous possible ways of writing foreign words in *English* roman script. For example, “zindagi” is spelled “jindagi,” “zindagy,” and “zyndagy,” and so on.
- Numerous possibilities of OOV words: short-form words (such as using *Tq* instead of *Thank you*, *tx* instead of *thanks*, and *2maro* instead of *tomorrow*) and words to represent the feelings (such as *Oooooo...*, *Hahaaaaaa....*, *Muuu....*).
- Lack of labeled data for CMSM text analysis tasks.

Taking cues from the advancements in NLP research through custom-built embeddings and LSTM-based sequence labelers, the current study addresses the research challenge of translating these gains into POS tagging for CMSM text. The primary impediment to such an approach is the availability of quality labeled data and the sufficient unlabeled social media corpus with different levels of code-mixing. This work looks at collecting CMSM text corpora in three code-mixing scenarios (i.e., *Bengali-English*, *Hindi-English*, and *Telugu-English*) from *Twitter*. Unlabeled social media text is too noisy to be useful for the sequence labeling tasks such as POS tagging. Part of the research challenge being addressed in this work is measuring the level of code-mixing in the unlabeled text corpora such that it contains all possible types of code-mixing (i.e., syntactic mixing, code alterations at various positions, inter- and intra-sentence level). The detailed description of the unlabeled corpora with URL link is provided in Section 4 and for the labeled dataset, Section 4.2. This work also addresses comparing different methods for learning embeddings. As we analyzed various works in the past for sequence labeling of mixed script text, we found that this work is the only effort (to the best of our knowledge) that harnessed the unlabeled corpus of the mixed script to tackle the sequence labeling difficulties in the code-mixed text.

We follow a two-stage BiLSTM recurrent neural network, one at the character level and another at the word level. Character-level embeddings learned by the BiLSTM help in addressing the numerous variations of non-native words. Word-level BiLSTM model leverages the embeddings learned from the corpus of unlabeled CMSM text. We highlight the contributions of this work below:

- We prepare a corpus of unlabeled CMSM text from which embeddings are learned for each of the languages *Bengali, Hindi,* and *Telugu.*
- We propose a hierarchical deep neural network–based approach for addressing the long-short term dependencies at word level and the problem of OOV words using character-level information.

[3]https://fasttext.cc/docs/en/pretrained-vectors.html, https://fasttext.cc/docs/en/crawl-vectors.html, https://github.com/Kyubyong/wordvectors.

Table 3. List of Works Related to CMSM Text Analysis

S.No	Article ref.	Dataset(s)	Task(s)	Model(s)
1	Solorio and Liu [2008b]	EN-SP	POS	SVM
2	Gimpel et al. [2011]	Tweets data	POS	CRF
3	Ritter et al. [2011]	Tweets data	NER and POS	LabeledLDA
4	Owoputi et al. [2013]	Tweets data	POS	CRF
5	Vyas et al. [2014]	HN-EN	POS	SVM
6	Barman et al. [2014]	BN-HN-EN	LI	SVM
7	Solorio et al. [2014]	BN-HN-EN	LI	CRF
8	Bali et al. [2014]	Facebook data(EN-HN)	NER and POS	
9	Jamatia et al. [2015]	EN-HN	POS	CRF
10	Rao and Devi [2016]	HN-EN and TM-EN	NER	CRF
11	Jaech et al. [2016]	TweetLID and Twitter70	LI	CNN-BiLSTM
12	Lample et al. [2016]	CONLL2002 and 2003	NER	BiLSTM-CRF
13	Kann et al. [2018]	Monolingual text	POS	LSTM
14	Murthy et al. [2018]	HN-BN	NER	CNN-BiLSTM

The short forms EN, HN, BN, and TM in third column are denoting *English*, *Hindi*, *Bengali*, and *Tamil* language names, respectively. LI in fourth column denotes the *Language Identification* task.

- We propose a detailed comparison of our approach with the CRF-based baseline and various combinations of character-level embeddings and word-level embeddings.
- A corrected dataset of POS Tagging for CMSM text is prepared and made publicly available (19% of the original dataset is found to be incorrectly tagged).

The rest of this article is organized as follows: Different models related to CMSM text analysis is summarized in Section 2. The details of hierarchical deep neural network approach is summarized in Section 3. The details of corpus preparation, dataset used for experimentation, and summary of results are presented in Section 4. Observations and error analysis of our experiments are given in Section 5.

2 RELATED WORK

This section summarizes various works related to CMSM text. For quick understanding, Table 3 presents the list of previous works related to sequence modeling in CMSM text analysis. Recently, code-mixing has been studied with different applications in the areas of IR [Gupta et al. 2014; Raghavi et al. 2015], NLP [Vyas et al. 2014], and Machine Translation [Li and Fung 2012]. For example, several works addressed different problems in the processing of CMSM text, including the word-level language identification [Chittaranjan et al. 2014; Solorio et al. 2014; Sristy et al. 2017], NER [Ritter et al. 2011] and POS tagging [Jamatia et al. 2015; Solorio and Liu 2008b].

State-of-the-art POS taggers for monolingual text such as Ferro et al. [2017] for *English*, Nakagawa and Uchimoto [2007] for *Japanese*, Zheng et al. [2013] for *Chinese*, Aldarmaki and Diab [2015] for *Arabic*, and so on, use linguistic or word-embedding features. These tools are trained on labeled monolingual text, which is generally collected from news articles. For example, WSJ of the Penn Treebank corpus [Marcus et al. 1993] for *English*, the Vite-Treebank corpus [Nguyen et al. 2009] for *Vietnamese,* and Kyoto corpus [Kawahara et al. 2002] for *Japanese.* Most of these works show that the performance of CRF-based tagging models is better. In code-mixed scenario, Solorio and Liu [2008b], Vyas et al. [2014] used SVM for POS tagging of *English-Spanish*, Barman et al. [2014], Solorio et al. [2014] used SVM and CRF for language identification in *Bengali-Hindi-English*, and Rao and Devi [2016] used CRF for NER in *Hindi-English* and *Tamil-English* code-mixing scenarios. But, these conventional methods perform poorly in CMSM text [Bali et al. 2014]. The works in Ritter et al. [2011] and Bali et al. [2014] elaborately presented the problems introduced by CMSM text. The main disadvantage of these conventional methods is that they require more training data. OOV words due to spelling variations and non-English text written in roman script pose major difficulty for these methods.

2.1 Deep Neural Network–based Tagging Models for CMSM Text

Recent research works improved the sequence modelling to solve the NLP-related tasks using non-linear models, such as Recurrent Neural Network (RNN) [Sutskever et al. 2011], Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997], and Gated Recurrent Unit (GRU) [Chung et al. 2015]. Part of their success is attributed to continuous representations of words such as word2vec [Mikolov et al. 2013], glove [Pennington et al. 2014], and so on. Hu et al. [2016] proposed a neural network architecture with optional Convolutional Neural Network (CNN) and RNN modules for NER and sentiment classification. Chiu and Nichols [2016] proposed a hybrid neural network architecture by combining Bi-directional Long Short Term Memory (BiLSTM) and CNN with a new encoding method for partial lexicon matches. The architecture in Chiu and Nichols [2016] automatically builds the character and word-level features to eliminate the need for feature engineering. Our work partly draws inspiration from such generalizations, though the problem of dealing with mixed script text poses additional difficulties. POS tagging for CMSM text is a niche area of research, more so for Indian languages. To the best of our knowledge, there are very few works for developing POS taggers on bi-lingual CMSM text.

Gimpel et al. [2011], Owoputi et al. [2013] developed a CRF-based POS tagger for *English* text collected from a specific community of *Twitter*. They improved the performance of the model with modified tag set by adding new tags for nuances of social media text such as URLs, hash-tags, re-tweets, emoticons, and so on. However, these works focused on plain social media text instead of CMSM text.

Solorio and Liu [2008b] developed different POS taggers for *English-Spanish* code-mixed text. They prepared the dataset using utterances collected from bilingual speakers. Their approach consists of application of POS taggers selected based on the language identification tags. Such an approach reduces to word-level POS tagging when code-switching points are more. In our work, we follow a holistic approach of tagging the whole code-mixed sentence rather than parts of the sentence.

Vyas et al. [2014] prepared a *Hindi-English* code-mixed dataset collected from *Facebook* and showed the impact of language identification and back-transliteration in the accuracy of POS tagging. We explored such a method using LSTM-based approach (with additional language tag information), which yielded inferior results. Jamatia et al. [2015] explained the procedure for collecting corpus (i.e., from *Twitter* and *Facebook*) and automatic annotation of *English-Hindi* social media text. They presented the experimental results of POS tagging on Coarse-Grained (CR) and Fine-Grained (FI) POS tag-set using various approaches, such as CRF, Naive Bayes, and Random Forests, among which CRF outperformed the remaining approaches. Unlike these approaches, our approach utilizes the advantage of contextual word representations that are learned on large unlabeled code-mixed corpus and the utilization of long-term dependency information captured using BiLSTM deep learning models. In Section 4, we present various combinations of results.

Lample et al. [2016] addressed the NER problem by applying CRF layer on top of BiLSTM. By using character-based word embeddings and contextual word embeddings as input features they achieved better results for *English* monolingual text. Kann et al. [2018] addressed the problem of prediction of POS tags for OOV words in low resource languages using character-based word embedding as input features to a deep neural network. Similar to their work, we use the character-sequence–based information to handle OOV words, with the extension of CRF layer. This layer incorporates the global dependency information among the tags. Kann et al. [2018] used the approach for monolingual POS tagging task, while our approach is meant for handling OOV in CMSM text using character-sequence features and contextual features of a word as input to hierarchical deep neural network.

The work done in Murthy et al. [2018] follows a two-stage approach of CNN and BiLSTM to improve the performance of NER using multi-lingual learning. Multi-lingual learning tries to address the tagging problems with OOV words using shared character representations of labeled corpora of other languages. The datasets in Murthy et al. [2018] represent a cleaner form of normal text as opposed to the CMSM text used in the current work. The current work addresses the problem of OOV words using code-mixed embeddings learned from an independent unlabeled corpus, avoiding the dependence on auxiliary labeled corpora. The current work also compares model performance using different character-based word representations learned using CNN and BiLSTM.

3 OVERALL APPROACH

Training POS taggers for CMSM text has to address the challenge of a limited amount of noisy training data. OOV words pose major difficulty in generalizing for unseen text. The current work partly involves the development of labeled data as well as nurturing unlabeled corpus of CMSM text. In this article, we trained hierarchical deep neural networks with few variations in their architectures for POS tagging. We formalized the sequence learning model for POS tagging task in CMSM text as follows.

3.1 Problem Definition

Consider $\mathcal{D} = (X, Y)$ to be the CMSM dataset. $X = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(N+U)}\}$ is a set of input sentences and $Y = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots \mathbf{y}^{(N)}\}$ is a set of label sequences. Here, N is the number of instances in training dataset $\mathcal{D}$ and $U(>>> N)$ is the number of sentences in unlabeled code-mixed corpus, which is used to learn code-mixed word embeddings. The main objective in CMSM text learning is to find the model parameters $\boldsymbol{\theta}$ by minimizing the following loss function:

$$l(\theta) = \arg\min_{\theta} \frac{-1}{N} \sum_{i=1}^{N} log\left(\mathbf{p}\left(\mathbf{y}^{(i)}|\mathbf{h}^{(i)};\boldsymbol{\theta}\right)\right), \tag{1}$$

where $\mathbf{h}^{(i)}$ is the sequence of hidden state outputs from word-level BiLSTM layer (shown in Figure 1) corresponding to the input sequence $\mathbf{x}^{(i)}$. The conditional probability of label sequence estimation, $\mathbf{p}(\mathbf{y}^{(i)}|\mathbf{h}^{(i)})$, is defined in Equation (9). At time step t the word-level BiLSTM layer gives the output $\mathbf{h}_t^{(i)}$ corresponding to the word $x_t^{(i)}$ in input sequence $\mathbf{x}^{(i)}$, where $\mathbf{h}_t^{(i)}$ is given by

$$\mathbf{h}_t^{(i)} = h\_BiLSTM_{word}\left(\left[h_t^c, V_{we}\left(x_t^{(i)}\right)\right]\right), \tag{2}$$

where "," denotes the concatenation of two vectors. *character -sequence–based word embedding* h_t^c corresponding to input word $x_t^{(i)}$ is defined as

$$h_t^c = h\_BiLSTM_{char}(c_{t1}, c_{t2}, \ldots c_{tl}). \tag{3}$$

Here, c_{tj} denotes the one-hot vector of jth character in tth word of ith sentence. $V_{we}(x_t^{(i)})$ gives the pretrained contextual word embedding corresponding to $x_t^{(i)}$ using *word embedding lookup table*. The contextual word embeddings are learned on a large unlabeled CMSM text $\{\mathbf{x}^{(N+1)}, \mathbf{x}^{(N+2)}, \ldots \mathbf{x}^{(N+U)}\}$ by maximization of the following objective function (4) with the consideration of unlabeled corpus as a large sequence of words. This is achieved using any of the

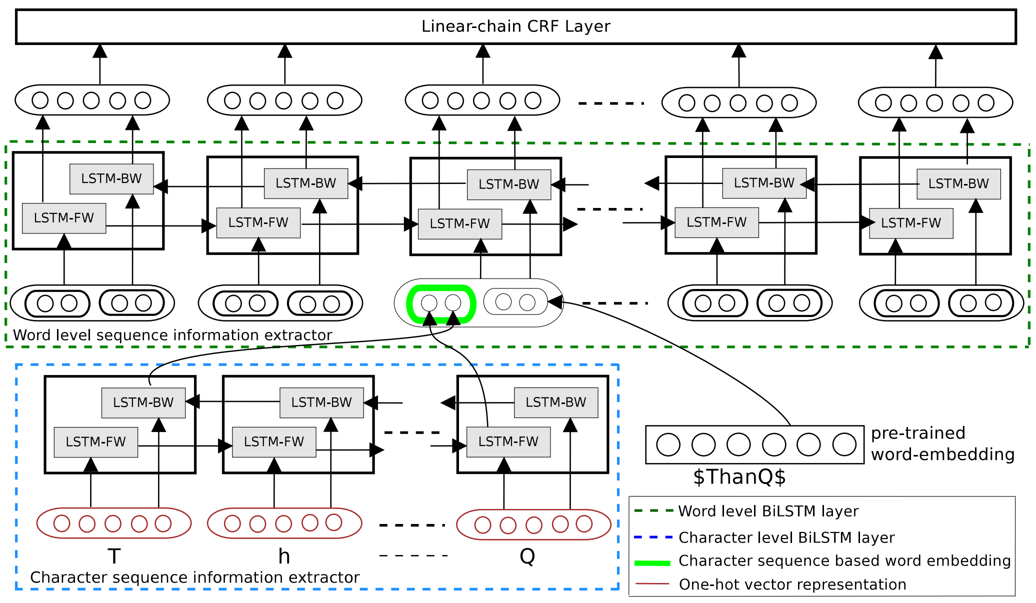
Linear-chain CRF Layer
LSTM-BW
LSTM-FW
Word level sequence information extractor
Character sequence information extractor
T
h
Q
pre-trained word-embedding
$ThanQ$
Word level BiLSTM layer
Character level BiLSTM layer
Character sequence based word embedding
One-hot vector representation

Fig. 1. Hierarchical deep recurrent neural network architecture.

methods discussed in Mikolov et al. [2013], Bojanowski et al. [2017], and Pennington et al. [2014]:

$$O_{we} = \max \sum_{k=1}^{K} \sum_{w_c \in \zeta} \log p\,(w_c | w_k). \tag{4}$$

Here, ζ is the set of context words of kth position word (w_k) and K denotes the length of the word sequence.

3.2 Hierarchical Deep Neural Network Architecture

In this section, we present a sequence tagging model using a hierarchical deep recurrent neural network with a linear-chain CRF layer as shown in Figure 1. Initially, a sequence of words is given as input to this network. The model employs a *word embedding lookup table* to extract the pre-defined word embedding for each word in the given input sequence. In parallel, the model also employs a network to extract character-level features either using (a) character-level BiLSTM layer or (b) character-level CNN layer. Each character of an input word to this layer is represented in one-hot vector form. A word-level BiLSTM layer is employed, taking as input the concatenation of outputs of character-level word representation layer and word-lookup table. The resultant output is fed as input to linear-chain CRF layer. A variety of architectural variations in the connections of the neural network are tried out in the current work, and extensive experiments are presented in Section 4. The different variations in the architecture are shown in Figure 3. Two variants of the architecture are observed in Figure 3 based on the character-level representation layer being CNN as opposed to BiLSTM. Similarly, two more variations are possible based on the usage of linear-chain CRF module for sequence prediction.

3.3 Model

In this section, we present the details of hierarchical deep neural network–based sequence labeling model. The network architecture is implemented using BiLSTM cells. BiLSTM is a special type

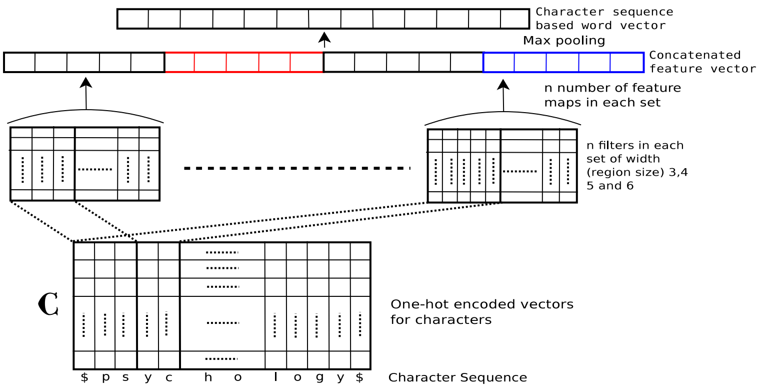
Character sequence based word vector
Max pooling
Concatenated feature vector
n number of feature maps in each set
n filters in each set of width (region size) 3,4 5 and 6
C
One-hot encoded vectors for characters
$ p s y c h o l o g y $
Character Sequence

Fig. 2. Convolutional neural network extracting character-level features from a word *psychology*.

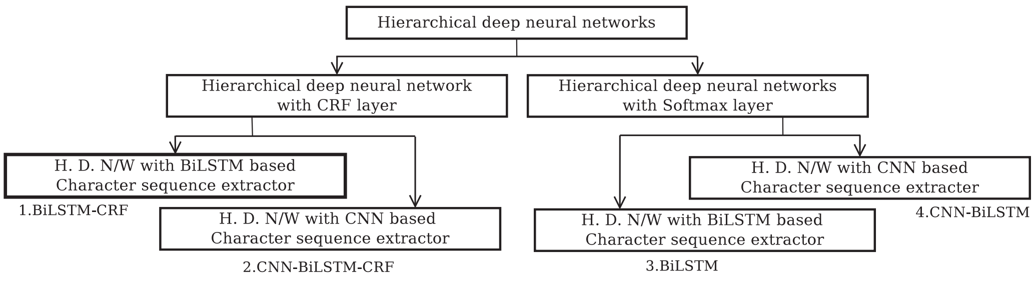
Hierarchical deep neural networks
Hierarchical deep neural network with CRF layer
Hierarchical deep neural networks with Softmax layer
H. D. N/W with BiLSTM based Character sequence extractor
1.BiLSTM-CRF
H. D. N/W with CNN based Character sequence extractor
2.CNN-BiLSTM-CRF
H. D. N/W with BiLSTM based Character sequence extractor
3.BiLSTM
H. D. N/W with CNN based Character sequence extracter
4.CNN-BiLSTM

Fig. 3. Different hierarchical deep neural network architectures employed in this work.

of deep recurrent neural network cell initially introduced [Schuster and Paliwal 1997] for speech processing applications [Schuster and Wiley 1999]. It is implemented using two LSTM [Hochreiter and Schmidhuber 1997] cells to capture the past and future dependency information of an input sequence at a position in sequence prediction tasks. The LSTM model we used in this article is adopted from Hochreiter and Schmidhuber [1997]. The LSTM cell at time step t takes the concatenation of current input (x_t) and previous cell hidden state (h_{t-1}) as input and then updates the current hidden state (h_t) and output gate (o_t). A deep recurrent neural network layer, shown in architecture (Figure 1), is represented as a sequence of BiLSTM cells corresponding to the input sequence ($x_1, x_2, \ldots x_L$). Here, the input sequence is either a sequence of words or sequence of characters. The hidden state of a BiLSTM cell at time step t is the concatenation of hidden states of two LSTM cells—one in forward direction and another in reverse direction, which is represented as $\mathbf{h}_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$.

BiLSTM is used in the current work in two contexts: (1) character level and (2) word level. The model employs character-level BiLSTM Layer to extract the character-sequence information of a word, called as *character-sequence–based word embedding*. This information helps the model to predict the label of OOV words. Let x_t be an input word to the character-level layer at time step t. x_t is represented as a sequence of one-hot encoded characters ($c_{t1}, c_{t2}, \ldots c_{tl}$), where l is the length of the word. Each character c_{ti} (tth word, ith character) has its corresponding BiLSTM cell with forward and backward hidden states denoted as $\overrightarrow{h_{ti}}$ and $\overleftarrow{h_{ti}}$. The concatenation of the last stage hidden state of BiLSTM in both directions—namely, $\overrightarrow{h_{tl}}$ and $\overleftarrow{h_{t1}}$—can be used as an encoding of character sequence of the word. Such embedding is fed as input to the word-level BiLSTM layer.

This concatenated word embedding h_t^c (shown in Figure 1 with green color circle) is denoted as in Equation (5):

$$h_t^c = \left[\overrightarrow{h_{tl}}, \overleftarrow{h_{t1}}\right]. \tag{5}$$

Later, the model employs the concatenation to combine the h_t^c with pre-trained word embedding ($V_{we}(x_t)$) of input word x_t as final input representation to word-level BiLSTM layer as given in Equation (6):

$$\mathbf{h}_t^w = [h_t^c, V_{we}(x_t)]. \tag{6}$$

3.3.1 CNN-based Word Representations. In the previous section, BiLSTM is presented as a tool for extracting character-sequence–based word representations. CNN is also adopted in a similar context [Jaech et al. 2016] chiefly because of the efficiency in training a CNN as opposed to RNN. CNN, when used to extract character-level features of words, uses multiple filter sets, each capturing the extraction of different n-grams from the word. Each character in the input of character-level CNN layer is represented with one-hot encoding. Figure 2 depicts each word as a character word matrix $\mathbb{C}$ of size $d \times l$. Here, d denotes the number of unique characters (one for each dimension of one hot vector) and l denotes the length of input word. CNN layer uses four sets of filters, each with width ranging from 3 to 6. Each filter is a matrix of form $M_{ij} \in \mathrm{R}^{d\times(j+2)}$ indicates ith filter (i ranging from 1 to n) in jth set. Here, n is the number of filters in each set. j value indicates the set number and $j+2$ is the width of each filter. The character to word matrix $\mathbb{C}$ is convoluted with each filter (i) from each set (j) and then added to its corresponding bias b_{ij} that is further fed to a non-linear activation function ReLU. The output is represented as $J_j = ReLU(conv(\mathbb{C}, M_{ij}) + b_{ij})$ of size $n \times l$. The model employs the max-pooling operation to get a character-level word embedding as shown in Equation (7):

$$h_t^c = MaxPool(ReLU(conv(\mathbb{C}, M_{ij}) + b_{ij})). \tag{7}$$

3.3.2 CRF Layer. The model optionally employs the linear-chain CRF [Lafferty et al. 2001] on the sequence of hidden state outputs from word-level BiLSTM layer to capture the global dependency among the tags in a sequence. The goal of CRF layer is to predict the most probable tag sequence $\mathbf{y} = (y_1, y_2, \dots y_L)$ for a given hidden state sequence $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \dots \mathbf{h}_L)$. Let $\mathbf{y} \in \mathcal{Y}(\mathbf{h})$, where $\mathcal{Y}(\mathbf{h})$ is the tag sequence space. The model computes the conditional probability $p(\mathbf{y}|\mathbf{h})$ in CRF layer. A notable difference in CRF presented in Lafferty et al. [2001] compared to the usage here is that the words are represented in continuous word-embedding space as opposed to one-hot encoding. As the CRF layer is used in hierarchy of network connections, the training method of CRF parameters is dependent on the remaining network. Parameter estimation in conventional use of CRF as in Lafferty et al. [2001] could be accomplished using much efficient L-BFGS as opposed to the variants of gradient descent employed here. We find the model parameters by minimizing the negative log-likelihood of conditional probability $p(\mathbf{y}|\mathbf{h})$ as shown in Equation (8):

$$-\sum_{i=1}^{N} log\left(p(\mathbf{y}^i|\mathbf{h}^i, \boldsymbol{\theta})\right). \tag{8}$$

Here, N is the number of instances in training set, $\mathbf{y}^i$ is a tag sequence corresponding to ith input sequence, $\mathbf{h}^i$ is a sequence of hidden state outputs (from word-level BiLSTM layer corresponding to the ith input sequence x^i), and $\boldsymbol{\theta}$ is a parameter vector. In general, the conditional log-linear model is

$$p(\mathbf{y}^i|\mathbf{h}^i, \boldsymbol{\theta}) = \frac{\exp\left\{\mathcal{S}\left(\mathbf{h}^i, \mathbf{y}^i\right)\right\}}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{h}^i)} \exp\left\{\mathcal{S}\left(\mathbf{h}^i, \mathbf{y}'\right)\right\}}, \tag{9}$$

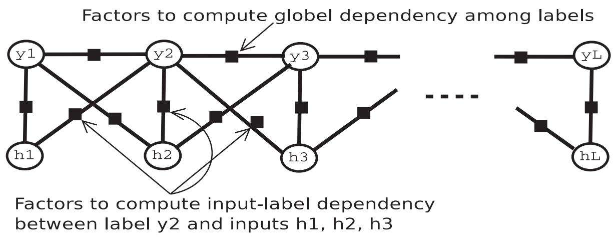
Factors to compute globel dependency among labels
y1
y2
y3
yL
h1
h2
h3
hL
Factors to compute input-label dependency between label y2 and inputs h1, h2, h3

Fig. 4. The computation of dependency among labels and BiLSTM output with context window size 1 in linear-chain CRF layer.

where $\mathcal{S}$ is a score function (feature function) that gives the score for each pair of $\mathbf{h}^i$ and $\mathbf{y}^i$ using following Equation (10):

$$\mathcal{S} = \sum_{t=1}^{L} \boldsymbol{\theta}_{y_t}^T \mathbf{h}_t^i + \sum_{t=1}^{L-1} \mathbf{V}_{y_t, y_{t+1}}. \tag{10}$$

The score $\mathcal{S}$ is the sum of two dependency information quantities, label-label dependency, and input-label dependency. The dependency among input word x_t and the label y_t (as shown in the first term of Equation (10)) is obtained by multiplying the hidden state output ($\mathbf{h}_t^i$) of word-level BiLSTM, corresponding to the input word x_t, with a weight vector $\boldsymbol{\theta}_{y_t}$ corresponding to label y_t. In other words, this is the score to assign the label y_t at position t. Global dependency among the labels (i.e., correlation among labels) is given by the second term in Equation (10). The weight matrix $\mathbf{V}$ contains the dependency information of all possible pairs of labels. The dependency among two labels y_t, y_{t+1} is given by $\mathbf{V}_{y_t, y_{t+1}}$. The score function $\mathcal{S}$ in Equation (10) is defined for context window size zero. If the context window size is 1, then input-label dependency is given by $\sum_{t=1}^{L} \boldsymbol{\theta}_{y_t}^T [\mathbf{h}_{t-1}^i, \mathbf{h}_t^i, \mathbf{h}_{t+1}^i]$, which shown in Figure 4.

3.4 Training

All deep neural network architectures (listed in Figure 3) are trained in end-to-end manner using Adam optimizer [Kingma and Ba 2014] with error back-propagation. The character-sequence–based word embeddings are fine-tuned in the training phase. The model is trained on CMSM labeled data. The optimization parameters employed in these experiments are *learning rate* ($\alpha = 0.001$), both *exponential decay rates* ($\beta_1 = 0.9$ and $\beta_2 = 0.9$), *dropout probability* (dropout = 0.5), *batch size* (20), and *number of epochs* (20). We used *early stopping* if there is no improvement in any three consecutive epochs. It employs the dynamic programming to compute the most probable label sequence in linear-chain CRF layer. All deep learning–based POS taggers are implemented in *python 2.7* using *tensorflow-gpu 1.3.0*. We used NVIDIA GPU driver version 384.130 with 16 GB GPU memory.

3.5 Why Do Character-sequence–based Word Embeddings Improve the POS Tagging for CMSM Text?

Unrestricted and flexible use of words in the social media text causes CMSM text analysis to be difficult. The numerous variations of spelling a word in roman script cause the vocabulary size of the model to explode. Representations of words such as one-hot vector are not good enough to preserve similarity among the numerous variations of the same word. Representations learned based on context using algorithms such as CBOW [Mikolov et al. 2013], skip-gram [Bojanowski et al. 2017], or [Pennington et al. 2014] are also not much help in this regard, because of the sparsity of the context information. We can resolve this problem by correcting all misspelled words using

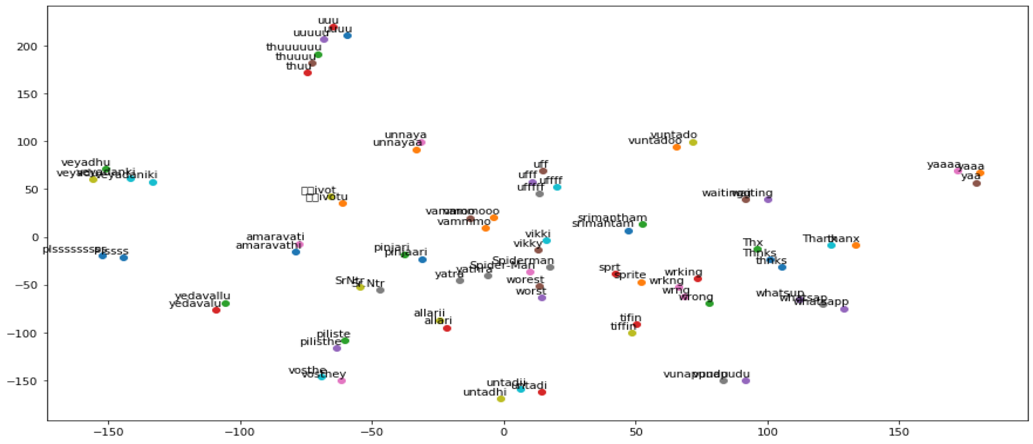

Fig. 5. Similarity between misspelled and correctly spelled words using character-sequence–based word embeddings built using BiLSTM model.

spellcheckers during data pre-processing phase. Even though it will reduce the vocabulary size, this is computationally challenging for spellcheckers to correct the misspelled word if it is not a native language word. To preserve the similarity of numerous variants of single word (in the embedding space), we built the word embeddings using character sequences of the word. Generally, character sequences of variants of a single word share common substrings. For example, the word "psychology" may be spelled as "sychology" or "cichology." These correct and misspelled words share common sub-sequence "chology." Figure 5 depicts tSNE projections of *character -sequence–based word embeddings* of misspelled and correctly spelled words. For example, the sets of words (*Thx, Thnks,* and *thnks*), (*plsssssss* and *plssss*), (*SrNtr* and *Sr.Ntr*), (*whatsup, whatsap,* and *whatsapp*), (*yaaaa, yaaa,* and *yaa*), and so on, are placed closely in vector space.

4 EXPERIMENTAL SETUP

In this section, we discuss the approach for preparation of unlabeled CMSM corpus and details of the CMSM labeled datasets for POS tagging. We also present the experimental details on three Indian language datasets mixed with *English* words.

4.1 Preparation of Unlabeled Corpora

There are many different monolingual word embeddings publicly available for Indian languages. These embeddings are built using text written in respective Indian language script. Such monolingual embeddings cannot be used directly for CMSM text. An explicit transliteration is needed if they have to be used. Such transliteration is noisy, as social media users resort to shorter length words, misspellings, and acronyms. Our experiments corroborate this observation. Instead, we follow a different approach of crawling a large amount of CMSM corpus. Mixed script–based word embeddings are learned using such unlabeled corpus. These embeddings are used as representations of words in an LSTM-based sequence labeling model for POS tagging. We followed the steps below in the preparation of CMSM corpus and used it to build code-mixed word embeddings:

- Scrapping text from social media,
- Word-level language identification,
- Preprocessing.

4.1.1 Scrapping. We crawled text for three Indian languages (*Bengali, Hindi,* and *Telugu*) with *English* code-mixing. To prepare these corpora, we collected an average of 75 candidate keywords in each native language. An initial collection of tweets matching these keywords is obtained using Twitter API.[4] This tweet collection itself cannot be used for learning the word embeddings, as it is collected using specific keywords and is biased. We follow a different approach of expanding the user base of users, possibly using the mixed script for their conversations. From the initial tweet collection, we identified the possible subset of mixed script users usually writing mixed script text. For every such user, we collected the publicly available tweets. We used a binary classifier for weeding out users not representing such native language. To ensure code-mixing in the subsequent tweet collection, we used a word-level language identifier [Sristy et al. 2017]. Once word-level language tagging is done for each of the sentences, we use these tags in the next pre-processing step to remove those sentences having less code-mixing complexity value.

4.1.2 Word-level Language Identification. We applied the approach presented in Sristy et al. [2017] for word-level language identification to find the language tag of each word of a sentence. The approach consists of training a binary classifier for distinguishing the words of *English* and each of the native Indian languages. The binary classifier is trained using 5K words of English and 5K words of native language (collected from Libschitz) represented as n-grams–based feature vector.

4.1.3 Preprocessing. To collect a sufficient number of sentences in different code-switching complexity levels, we pre-processed the above scrapped corpus. In pre-processing phase, we analyzed the level of code-mixing and code-switching in each sentence using its word-level language tags. The sentence-level code-switching is computed using two sources of information—namely, average code-mixing level and an average number of code alteration points per token in a sentence [Gambäck and Das 2016; Jamatia et al. 2015]. The metric CMI (Code-Mixing Index) measures the fraction of minority language tokens over all language-dependent tokens of the sentence. The CMI metric is given as:

$$CMI = \begin{cases} 100 \times \left[1 - \left(\frac{max\{W_{Lang}\}}{T_N - T_u}\right)\right] & : T_N > T_u \\ 0 & : T_N = T_u \end{cases}, \tag{11}$$

where T_N, T_u are the total number of tokens and language-independent tokens in a sentence, respectively. $max\{W_{Lang}\}$ gives the number of words from most frequent language in a sentence. For monolingual sentences, CMI = 0 (since $\max\{W_{Lang}\} = T_N - T_u$). If a sentence only contains language-independent tokens, then its CMI value is zero.

The average code alteration points per token (CSP) is another metric used for addressing the code-switching complexity observed over the whole sequence of the utterance. CSP in an utterance is computed using Equation (12), where P denotes the number of code alternation points in an utterance:

$$CSP = 100 * P/(T_N - T_u). \tag{12}$$

Both CMI and CSP individually do not represent the true complexity of code mixing [Gambäck and Das 2016; Jamatia et al. 2015]. Using CMI and CSP, the level of code-switching complexity in a sentence is defined as:

$$CSL = w_{cm} * CMI + w_{cp} * CSP, \tag{13}$$

where w_{cm} and w_{cp} are weights and $w_{cm} + w_{cp} = 1$. We prepared unlabeled corpora from social media text for three Indian languages using this approach. We computed the code-switching complexity level (CSL) for each sentence in CMSM corpora with $w_{cm} = 0.5$ and $w_{cp} = 0.5$. Tables 4, 5,

[4]https://developer.twitter.com/en/docs/tweets/search/overview.html.

Table 4. The Distribution of Code-mixing Index in Three Corpora

CMI (%)	Bengali	Hindi	Telugu
0–10	132,210	688,059	257,978
11–20	20,509	48,413	26,953
21–30	45,505	51,069	77,443
31–40	58,550	57,412	92,891
40<	44,697	60,416	44,735

Table 5. The Distribution of Average Code-switching Points in Three Corpora

CSP (%)	Bengali	Hindi	Telugu
0–10	123,345	652,528	253,127
11–20	20,267	23,544	15,637
21–30	60,331	60,876	50,046
31–40	74,039	93,418	97,932
40<	23,489	75,003	83,258

Table 6. The Distribution of Code-switching Complexity Level in Three Corpora

CSL (%)	Bengali	Hindi	Telugu
0–10	124,378	656,702	253,916
11–20	21,347	44,162	19,773
21–30	52,909	68,299	62,586
31–40	74,416	77,693	102,816
>40	28,421	58,513	60,909

Table 7. Summary of the Prepared Corpora of Three Languages

Corpus	Instances	Tokens (average tokens)	Lang Tokens (average lang-tokens)
Bengali	301,471	4,155,858 (13.79)	1,817,404 (6.03)
Hindi	905,369	8,135,044 (8.99)	3,088,716 (3.41)
Telugu	500,000	7,237,047 (14.47)	2,534,450 (5.07)

and 6 describe the distribution of sentences in different levels of CMI, CSP, and CSL of three corpora. The values in the first row of these three tables indicate that, in real time, most of the social media users use 0% to 10% of words from other languages in their sentences. The values in these three tables indicate that the corpora being prepared in the current work contain a sufficient number of sentences in different levels of code-mixing. Hence, the word representations learned on this corpus can capture the syntax and semantic information of CMSM text. Table 7 summarizes the number of sentences, tokens, and native language tokens contained in each of the three Indian language CMSM corpora after pre-processing phase. It also summarizes the average number of tokens and language tokens per sentence in parentheses. Table 7 shows that the *Hindi* corpus encodes different CSL levels more evenly compared to *Bengali* and *Telugu* corpora. We have made these corpora publicly available[5] for further utilization.

4.2 Dataset

Our experimental study is based on POS tagging for CMSM text in three Indian languages (*Bengali, Hindi,* and *Telugu* mixed with *English* words) collected from three different sources—namely, *Whatsapp, Facebook,* and *Twitter*. These datasets contain native language sentences in *English* script. Each word of a sentence is labeled with its language tag, POS tags (fine-grained and coarse-grained). POS tags are labeled using Google Universal Tagset (Table 10) [Petrov et al. 2012] with extended tag list related to social media text [Gimpel et al. 2011; Owoputi et al. 2013]. This tag set contains a total of 18 coarse-grained and 38 fine-grained POS tags. Based on the nature of the datasets being considered, there are two different modes of experiments (i) combining datasets from all the three sources (*Whatsapp, Facebook*, and *Twitter*) (ii) treating the datasets separately based on the source as done in Gupta et al. [2017]. The number of sentences, tokens, and native language tokens of each dataset are described in Table 9. The number of sentences based on the source is indicated in parentheses (FB: *Facebook*, TW: *Twitter*, WH: *Whatsapp*). Compared to *Bengali* and

[5]https://drive.google.com/drive/folders/197eZG4A2hF9PYKY9sZ26YsUPucWhNxRa?usp=sharing.

Table 8. Code-switching Complexity Level in Labeled Dataset

CSL (%)	Bengali	Hindi	Telugu
0–10	299	1,328	142
11–20	27	293	88
21–30	97	509	315
31–40	126	361	598
>40	78	140	837

Table 9. Datasets Summary

Language	Instances (FB, TW, WA)	Tokens (FB, TW, WA)	Lang Tokens (FB, TW, WA)
Bengali	625	14,702	6,131
	(148, 173, 304)	(7,461, 3,711, 3,529)	(3,630, 1,805, 696)
Hindi	2,630	41,122	15,183
	(772, 1,095, 763)	(20,615, 17,289, 3,218)	(2,857, 9,772, 2,554)
Telugu	1,979	29,471	8,813
	(743, 743, 493)	(10,037, 12,013, 7,421)	(2,646, 4,052, 2,115)

Telugu datasets *Hindi* dataset contains more number of instances. These datasets are provided by ICON2016 shared task.[6] The number of sentences with different code-switching complexity levels in each dataset is described in Table 8. We can observe that *Bengali* has a relatively lesser number of instances in different CSL levels compared to *Telugu* and *Hindi*.

4.3 POS Tagging Corrections

In this section, we give the details of corrections we made to the tags of ICON 2016 shared task dataset. Being a niche area of NLP, POS tagging for code-mixed text requires skills of multiple languages. We found that many of the POS tags are wrongly labeled in ICON 2016 dataset. A total of 19% of POS tags are corrected. The statistics of corrections we made in each POS tag are described in Table 10. First and third columns describe the number of words that are wrongly tagged in *Hindi* dataset for coarse-grained tags. Similarly, second and fourth columns describe the number of words wrongly tagged for fine-grained tags. More number of corrections are made in **V_VM** (29.19%), **N_NN** (20.26%), **PSP** (24.69%), and **DT** (30.74%). We observed that **V_VM** tag was wrongly labeled 847 (44.7%), 394 (20.8%), and 153 (8.07%) times as **N_NN**, **N_NNP**, and **V_VAUX**, respectively. **N_NN** tag was wrongly labeled 604 (46%), 288 (22.01%), and 140 (10.7%) times as **N_NNP**, **V_VM**, and **JJ**, respectively. **PSP** tag was wrongly labeled 205 (23.51%), 129 (14.8%), and 127 (14.56%) times as **PR_PRP**, **N_NN**, and **PR_PRQ**, respectively. **DT** tag was wrongly labeled 50 (8.17%), 34 (5.56%), and 22 (3.39%) times as **RD_RDF**, **RP_RPD**, and **RP_INJ**, respectively.

In Table 11, we presented few example sentences from the dataset with their old label sequences and new label sequences. In sentences 1, 2, and 3, the word tags for *say, marna,* and *karlo* are corrected by replacing their old tags with **V_VM** tag, since these words are representing the action performed by someone or something. The label of the word *it* in sentence-4 is corrected with **PR_PRP** tag, since this word is used to substitute a noun. The word *ghussa* is indicating the feature of the person in sentence-5, hence it is corrected with **JJ** tag. Explanation for the corrections we made is provided for each token.[7]

4.4 Results

This section describes the experimental results on three different code-mixed Indian language datasets. We compare the results of deep neural network model (with four variations in neural architecture) with baseline CRF model. For easy understanding, we used different names for these architectures, as shown in Figure 3. The different neural network variants tried out are based on differences in character-level network being (i) BiLSTM and (ii) CNN and an optional CRF layer at the end. A total of 4 such combinations are given below. (1) BiLSTM-CRF: It uses BiLSTM in

[6] http://www.amitavadas.com/Code-Mixing.html.
[7] https://drive.google.com/drive/folders/197eZG4A2hF9PYKY9sZ26YsUPucWhNxRa?usp=sharing.

Table 10. Description of Corrections Made in Each POS Tag in *Hindi* Dataset

CR Tag (# wrong / # total)	**FI Tag** (# wrong / # total)	**CR Tag** (# wrong / # total)	**FI Tag** (# wrong / # total)
G_N (Noun) (1,186/9,101)	(Common Noun) N_NN (1,308/6,454)	(Quantifier) G_SYM (576/642)	(General) QT_QTF (229/374)
	(Verbal Noun) N_NNV (158/327)		(Cardinal) QT_QTC (148/168)
	(Spatio-temporal) N_NST (48/80)		(Ordinal) QT_QTO (66/100)
	(Proper Noun) N_NNP (608/2,240)	(Particles) G_PRT (1,338/2,050)	(Default) RP_RPD (539/938)
G_PRP (Pronoun) (1,122/3,529)	(Personal) PR_PRP (552/2,389)		(Negation) RP_NEG (86/191)
	(Relative) PR_PRL (224/397)		(Intensifier) RP_INTF (193/211)
	(Reflexive) PR_PRF (135/148)		(Interjection) RP_INJ (365/710)
	(Reciprocal) PR_PRC (14/27)	(Residual) G_X (2,490/7,217)	(Foreign Word) RD_RDF (20/140)
	(Wh-Word) PR_PRQ (126/249)		(Symbol) RD_SYM (3/249)
G_V (Verb) (2,121/7,872)	(Main verb) V_VM (1,895/6,490)		(Punctuation) RD_PUNC (33/4,175)
	(Auxiliary verb) V_VAUX (246/1,382)		(Unknown) RD_UNK (2/12)
G_J (Adjective) (514/2,045)	(Adjective) JJ (514/2,045)		(Echo Word) RD_ECH (0/0)
G_R (Adverb) (563/1,754)	(Locative Adverb) RB_ALC (292/446)	Conjunction / Preposition	(Conjunction) CC (137/1,032)
	(Adverb of Manner) RB_AMN (224/1,308)		(Pre-/Postposition) PSP (872/3,531)
G_PRP (Demonstrative) (1,122/3,529)	(Absolute) DM_DMD (76/181)	G_X (Twitter related tags proposed by Gimple)	(At-mention) @ (135/1,128)
	(Indefinite) DM_DMI (58/82)		(Emoticon) E (244/557)
	(Wh-word) DM_DMQ (28/46)		(URL or email) U (1/214)
	(Relative) DM_DMR (10/10)		(Hashtag) # (13/561)
Determiner (612/1,991)	(Determiner) DT (612/1,991)		(Re-Tweet / discourse) ~ (148/181)

both character- and word-level layers. CRF layer on top of the word-level BiLSTM layer. (2) CNN-BiLSTM-CRF: It uses CNN in character-level layer and BiLSTM in word-level layer. It includes CRF layer on top of the word-level BiLSTM layer. (3) BiLSTM: Similar to the BiLSTM-CRF architecture with no additional CRF layer. (4) CNN-BiLSTM: This is similar to the CNN-BiLSTM-CRF architecture with no additional CRF layer. We performed experiments by varying the number of

Table 11. Few Examples from Corrected Sentences

Sentence1:	they	used	to	say	hello	baby	kis	school	mein	ho
Old-tag-seq:	PR_PRP	V_VM	RP_RPD	**PSP**	RP_INJ	N_NN	PR_PRQ	N_NNP	PSP	V_VM
New-tag-seq:	PR_PRP	V_VM	RP_RPD	**V_VM**	RP_INJ	N_NN	PR_PRQ	N_NNP	PSP	V_VM
Sentence2:	abe	....	joke	marna	hai	to	aur	kahi	maar	
Old-tag-seq:	N_NN	RD_PUNC	N_NN	**N_NN**	RP_INJ	PSP	CC	N_NN	N_NN	
New-tag-seq:	RP_INJ	RD_PUNC	N_NN	**V_VM**	V_VAUX	PSP	CC	N_NN	V_VM	
Sentence3:	....	kuch	dino	ka	intezaar	karlo	....			
Old-tag-seq:	RD_PUNC	RP_INJ	N_NN	PSP	**N_NNP**	**N_NNP**	RD_PUNC			
New-tag-seq:	RD_PUNC	RP_INJ	N_NN	PSP	**N_NNV**	**V_VM**	RD_PUNC			
Sentence4:	it	is	too	long	....					
Old-tag-seq:	**N_NNP**	V_AUXV	RB_AMN	JJ	RD_PUNC					
New-tag-seq:	**PR_PRP**	V_AUXV	RB_AMN	JJ	RD_PUNC					
Sentence5:	thora	kam	ghussa	kary	us	py				
Old-tag-seq:	RB_AMN	QT_QTF	**V_VM**	V_VM	PR_PRP	PSP				
New-tag-seq:	RB_AMN	QT_QTF	**JJ**	V_VM	PR_PRP	PSP				

dimensions in character and word embeddings. The baseline CRF model is implemented using CRF++0.58[8] package. We consider the input features as specified in Jamatia et al. [2015]. The input features to CRF are given using a template of [The current word, previous word, next word, prefix and suffix of the current word, language tag of the current word, previous word label, previous to the previous word, next to next word].

4.4.1 CRF vs. Deep Neural Networks with Code-mixed Embeddings. Tables 12, 13, and 14 present the average F1-scores and accuracies of four approaches (CRF baseline, BiLSTM, BiLSTM-CRF, and CNN-BiLSTM-CRF) on both fine-grained and coarse-grained CMSM datasets. These experiments are performed in 5-fold cross-validation by considering 80% training data and 20% test data. The overall average F1-score improvement of BiLSTM-CRF compared with CRF is 10.18%, 4.47%, and 4.13% in *Bengali, Hindi*, and *Telugu* languages, respectively, in fine-grained datasets. Similarly, improvements of 4.08%, 4.30%, and 2.14% are observed for respective language-specific coarse-grained datasets. BiLSTM model also moderately improved the average F1-score compared to the baseline CRF model. The results in these three tables describe that the CNN-BiLSTM-CRF model is inferior compared to the BiLSTM-CRF. We compared the performance of BiLSTM-CRF and CNN-BiLSTM-CRF on *Bengali, Hindi*, and *Telugu* datasets using the paired t-test. The p-values for three datasets are well below the accepted significance level of 0.05, which indicates that the performance improvement of BiLSTM-CRF over CNN-BiLSTM-CRF is statistically significant. The overall F1-score improvement of BiLSTM-CRF compared to CNN-BiLSTM-CRF is 3.58%, 2.81%, and 1.64% on *Bengali, Hindi*, and *Telugu* fine-grained datasets. A possible reason for this observation could be that the BiLSTM used in character-level layer in BiLSTM-CRF captures the longer-range dependencies better than that of CNN-based character-level layer.

4.4.2 BiLSTM-CRF with Mono-lingual Embeddings. The neural network configurations presented in tables 12, 13, and 14 demonstrate the effectiveness of word representations learned from external code-mixed corpora. Taking cues from different methods of learning word representations [Murthy et al. 2018; Pratapa et al. 2018a] and additional language tag information given to baseline CRF model, we designed one more variant of neural network using mono-lingual word representations. Monolingual word embeddings are trained on the respective language text alone and capture the co-occurrence of words in that language. We use pre-trained *fasttext* monolin-

[8]https://taku910.github.io/crfpp/.

Table 12. F1-score and Accuracy for *Bengali* and *English* Code-mixed Dataset

Model				F1-Score		Accuracy	
				FI	CR	FI	CR
CRF				0.576	0.7019	0.7016	0.7561
	Word embedding method	Char Dim	Word Dim				
BiLSTM	Word2Vec	300	300	0.567	0.6822	0.6785	0.7309
		300	500	0.5973	0.7038	0.6968	0.7437
	FastText	300	300	0.6054	0.7119	0.7033	0.7569
		300	500	0.6443	0.7294	0.7116	0.7616
	Glove	300	300	0.5935	0.7057	0.6965	0.7526
		300	500	0.6212	0.7175	0.7029	0.7567
BiLSTM-CRF	Word2Vec	300	300	0.6016	0.7235	0.6921	0.758
		300	500	0.6137	0.7277	0.7008	0.7595
	FastText	300	300	0.6354	0.7313	0.7112	0.7638
		300	500	**0.6778**	**0.7427**	**0.7233**	**0.7715**
	Glove	300	300	0.6136	0.7355	0.6981	0.7623
		300	500	0.6471	0.7405	0.7063	0.7639
CNN-BiLSTM-CRF	Word2Vec	300	300	0.5582	0.6714	0.6674	0.7288
		300	500	0.5887	0.6941	0.6759	0.7356
	FastText	300	300	0.674	0.7203	0.703	0.7527
		300	500	0.6420	0.7420	0.7091	0.7691
	Glove	300	300	0.5836	0.6860	0.6638	0.7314
		300	500	0.6030	0.6936	0.6700	0.7339

gual embeddings[9] for *Bengali, Hindi*, and *Telugu*. The corpora used for these languages consist of text written in unicode native script as opposed to the roman script being used in code-mixed text. We use transliteration[10] to match words written in roman script to native uni-code script for retrieving the corresponding mono-lingual word embedding. As a single word in code-mixed text can be either of the native languages (*Bengali, Hindi*, and *Telugu*) or *English* or others, we use a word representation of size double that of respective mono-lingual embeddings. The first half of this representation captures *English* embeddings and latter half captures native language mono-lingual embedding.

There are four different combinations in which such concatenation of individual embeddings is possible. If an *English* word is encountered, we use the corresponding word's embedding from *English* embeddings in the first half of the representation. We fill the latter half of such word's representation with that of UNK, as it is not a native word. Similarly, if a native word is encountered, we fill the latter half of the representation with that of native word-embedding and use UNK's representation in the first half. If a word is in both English and native language dictionaries (universal), we use concatenation of both representations. If the word is in neither of these dictionaries, we use UNK for both halves of the representation. The results of this approach for three languages are presented in the first row (*approach* A1) of Table 15. The overall average F1-score improvement of BiLSTM-CRF using code-mixed word embeddings compared with BiLSTM-CRF

[9] monolingual embeddings collected from: https://s3-us-west-1.amazonaws.com/fasttext-vectors/.
[10] Code for transliteration: https://github.com/libindic/indic-trans.

Table 13. F1-score and Accuracy for *Hindi* and *English* Code-mixed Dataset

Model				F1-Score		Accuracy	
				FI	CR	FI	CR
CRF				0.6109	0.7556	0.7542	0.8203
	Word embedding method	Char Dim	Word Dim				
BiLSTM	Word2Vec	300	300	0.6179	0.7617	0.7538	0.8263
		300	500	0.6221	0.7630	0.7545	0.8280
	FastText	300	300	0.6404	0.7891	0.7791	0.8456
		300	500	0.6409	0.7907	0.7792	0.8469
	Glove	300	300	0.6356	0.7800	0.7708	0.8375
		300	500	0.6400	0.7857	0.7725	0.8414
BiLSTM-CRF	Word2Vec	300	300	0.6276	0.7792	0.7636	0.833
		300	500	0.628	0.7817	0.7636	0.8358
	FastText	300	300	0.6504	0.7952	0.785	0.8504
		300	500	**0.6556**	**0.7986**	**0.787**	**0.8526**
	Glove	300	300	0.6424	0.7888	0.7736	0.8425
		300	500	0.6467	0.7919	0.7786	0.8459
CNN-BiLSTM-CRF	Word2Vec	300	300	0.6003	0.7414	0.7336	0.8068
		300	500	0.6058	0.7427	0.7421	0.8070
	FastText	300	300	0.6219	0.7655	0.7576	0.8255
		300	500	0.6275	0.7748	0.7664	0.8325
	Glove	300	300	0.6050	0.7472	0.7357	0.8100
		300	500	0.6075	0.7550	0.7462	0.8166

using mono-lingual embeddings is 15.75%, 4.25%, and 9.97% in *Bengali, Hindi*, and *Telugu* languages, respectively. There are two apparent reasons for the inferior performance of the approach based on mono-lingual embeddings: (1) The mono-lingual text from which mono-lingual embeddings are learned does not reflect the numerous variations used in social media text. (2) The mono-lingual embeddings do not capture the co-occurrence of words in multiple languages.

4.4.3 Ablation Study. An ablation study for understanding the role of different embeddings used in the current work is summarized in the last three rows in Table 15. The ablation study considers the performance of POS tagger with and without character-based word embeddings. A further comparison of differences in performances of the POS tagger with and without custom code-mixed embeddings is also part of this study.

For *Bengali* language, BiLSTM-CRF performs better in approach A2 compared with approach A3, which tells us that the code-mixed embeddings are more impactful than the character-based embeddings in this case. For *Hindi* language, performance of approach A3 is better than approach A2, indicating that the character-based embeddings play key role in the POS tagging task. The performance is identical in approach A2 and A3 for *Telugu* language.

The last row of Table 15 for approach A4 indicates that the combination of the code-mixed embeddings and character-based embeddings is much better than that of either one of them alone. We can also observe that the impact of such combination is much higher for *Telugu,* as each of these embeddings covers different aspects of POS tagging.

4.4.4 Domain-wise Performance Comparison. The earlier experiments in 4.4.1, 4.4.3, and 4.4.2, all use the dataset by combining the three genres of data—namely, *Facebook*, *Twitter,* and

Table 14. F1-score and Accuracy for *Telugu* and *English* Code-mixed Dataset

Model				F1-Score		Accuracy	
				FI	CR	FI	CR
CRF				0.6559	0.7141	0.67	0.7128
	Word embedding method	Char Dim	Word Dim				
BiLSTM	Word2Vec	300	300	0.67	0.7223	0.6931	0.736
		300	500	0.6806	0.7301	0.6981	0.736
	FastText	300	300	0.6621	0.7221	0.6881	0.7333
		300	500	0.6769	0.7288	0.6945	0.7351
	Glove	300	300	0.6646	0.7179	0.6868	0.7341
		300	500	0.6698	0.7309	0.6941	0.7351
BiLSTM-CRF	Word2Vec	300	300	0.6517	0.6883	0.6906	0.7394
		300	500	0.6816	0.719	**0.6996**	**0.7488**
	FastText	300	300	0.6584	0.7304	0.6877	0.7348
		300	500	0.6831	0.7343	0.6957	0.7387
	Glove	300	300	0.667	0.7207	0.6867	0.7321
		300	500	**0.6972**	**0.7355**	0.6989	0.7364
CNN-BiLSTM-CRF	Word2Vec	300	300	0.6634	0.7123	0.6803	0.7254
		300	500	0.6672	0.7127	0.6812	0.7253
	FastText	300	300	0.6547	0.7104	0.6777	0.7299
		300	500	0.6592	0.7191	0.6846	0.7332
	Glove	300	300	0.6486	0.705	0.6708	0.7248
		300	500	0.6636	0.7162	0.6774	0.7269

Table 15. Ablation Study of BiLSTM-CRF

Approach	Bengali-English				Hindi-English				Telugu-English			
	FI		CR		FI		CR		FI		CR	
	F1-Scr	Acc	F1-Scr	Acc	F1-Scr	Acc	F1-Scr	Acc	F1-Scr	Acc	F1-Scr	Acc
A1	45.64	59.70	56.58	67.00	61.01	75.95	76.45	82.89	56.87	66.88	62.99	71.61
A2	60.58	66.82	68.97	73.47	50.48	68.84	66.83	76.17	49.36	64.19	61.02	70.24
A3	41.99	51.97	52.43	62.89	63.00	74.13	76.82	82.40	50.26	64.16	62.19	69.81
A4	63.54	73.13	71.12	76.38	65.04	79.52	78.50	85.04	65.84	73.04	68.77	73.48

A1 Embeddings: with character embeddings and plain text word embeddings. **A2:** without character embeddings and with pre-trained CM-word embeddings. **A3:** with character embeddings and without pre-trained CM-word embeddings. **A4:** with character embeddings and pre-trained CM-word embeddings.

Whatsapp—into a single dataset and report the model performance in 5-fold cross-validation setting. In this section, we perform the experiments on these three genres separately. Two baselines based on CRF are used in this section. The first baseline uses the implementation and features of the model in Jamatia et al. [2015], referred to as CRF-Jamtia. The second baseline uses the implementation in Gupta et al. [2017], referred to as CRF-Gupta. CRF-Gupta [Gupta et al. 2017] is the best open source implementation available for this dataset (as per our study). In addition to these baselines, we analyzed the performance of BiLSTM-CRF model with and without character-sequence–based word embeddings. Table 16 (for fine-grained datasets) and Table 17 (for coarse-grained datasets) present the results on three code-mixed language datasets from each

Table 16. Evaluation of BiLSTM-CRF Model Using the Evaluation Setup Given by ICON2016 Shared Task on FI Dataset

Features	Model	Metric	Bengali-English Dataset			Hindi-English Dataset			Telugu-English Dataset		
			FB	TWT	WA	FB	TWT	WA	FB	TWT	WA
Rich language features	CRF-Jamatia	P	55.67	51.71	57.22	58.14	64.11	58.86	42.25	38.20	42.55
		R	79.95	67.59	74.80	70.56	80.99	70.62	59.78	48.77	61.12
		F	65.63	58.58	64.80	63.74	71.56	64.16	49.48	42.83	50.12
		A	72.72	67.39	68.51	75.31	75.87	66.34	63.13	59.82	58.43
	CRF-Gupta	P	77.25	62.00	67.50	75.25	68.75	65.00	61.50	55.75	53.50
		R	75.75	62.00	66.50	75.25	68.50	64.25	60.75	53.75	51.00
		F	74.75	60.00	65.00	74.50	67.50	63.75	59.75	53.00	50.50
		A	75.99	62.05	66.60	75.18	68.46	64.34	60.64	53.66	50.85
BERT Embeddings	BiLTM-CRF-A	P	47.81	34.04	44.39	42.82	49.76	43.56	31.75	31.27	39.64
		R	68.94	51.95	63.64	52.89	53.39	55.41	51.77	45.68	57.11
		F	56.46	41.13	52.31	47.32	51.51	48.77	39.36	37.13	46.79
		A	68.55	60.28	60.19	68.92	66.41	60.51	57.78	55.32	57.95
	BiLTM-CRF-B	P	60.26	57.73	59.91	58.62	63.66	58.23	56.15	52.58	66.56
		R	73.11	70.78	72.21	72.93	65.95	69.98	77.10	59.41	79.50
		F	66.07	63.60	65.49	65.01	64.78	63.56	64.98	55.79	72.46
		A	76.12	72.23	68.33	78.23	81.82	75.02	74.64	72.76	75.25
Fasttext Embeddings	BiLTM-CRF-A	P	67.46	41.54	56.61	60.19	59.60	45.94	49.27	49.62	61.58
		R	83.48	58.35	77.21	73.01	64.27	57.21	62.32	60.64	71.36
		F	74.62	48.53	65.32	65.98	61.85	50.96	55.03	54.58	66.11
		A	79.69	65.65	66.77	81.42	75.32	62.59	77.52	72.76	76.24
	BiLTM-CRF-B	P	76.79	57.68	76.18	76.77	76.25	71.76	76.63	83.19	85.99
		R	85.02	72.85	84.83	88.93	80.43	77.68	93.55	86.76	90.17
		F	**80.70**	**64.38**	**80.27**	**82.43**	**78.28**	**74.60**	**84.25**	**84.93**	**88.03**
		A	**85.21**	**71.56**	**82.00**	**88.63**	**90.81**	**85.33**	**90.82**	**85.70**	**89.75**

domain. The sub-field names FB, TWT, and WA in the second row refer to *Facebook*, *Twitter*, and *Whatsapp*, respectively. A total of 18 combinations of experiments are used in this analysis (3 languages, 3 domains, 2 types of labels (coarse- and fine-grained)). The names BiLSTM-CRF-A and BiLSTM-CRF-B in the second column refer to the BiLSTM-CRF model without character embeddings and with character embeddings, respectively. P, R, F, and A in the third column refer to the metrics *precision, recall, F1-score*, and *accuracy*, respectively.

Several recent works in NLP, such as question answering [Yang et al. 2019; Zhu et al. 2018], document ranking [MacAvaney et al. 2019], NER [Straková et al. 2019], and sentiment analysis [Xu et al. 2019] have shown good performance using BERT (Bidirectional Encoder Representations from Transformers)-based embeddings. These works use pre-trained embeddings—namely, *BERT_base* or *BERT_large* [11] trained on large plain text corpus. Instead of these pre-trained models, we build BERT model by training the Masked Language Modeling (MLM) objective of Devlin et al. [2018] on unlabeled code-mixed text corpus. Further word-embedding features are extracted using this model. While training, we mask each token 80% of the time, 10% of the time replace with another random word, and 10% of the time, we consider itself only. We conducted experiments on POS tagging on two more baselines where the embeddings are based on BERT model with

[11]The BERT pre-trained language models are available at https://github.com/google-research/bert#pre-trained-models.

Table 17. Evaluation of BiLSTM-CRF Model Using the Evaluation Setup Given by ICON2016 Shared Task on CR Dataset

Features	Model	Metric	Bengali-English Dataset			Hindi-English Dataset			Telugu-English Dataset		
			FB	TWT	WA	FB	TWT	WA	FB	TWT	WA
Rich language features	[Jamatia et al. 2015]	P	65.13	55.73	57.16	75.18	67.27	62.67	43.50	48.30	43.97
		R	84.00	77.78	80.09	81.94	83.83	76.72	67.01	69.10	70.27
		F	73.33	64.57	66.70	78.41	74.64	68.92	52.73	56.84	54.07
		A	79.06	72.93	74.18	80.69	79.85	74.33	70.08	64.73	64.80
	[Gupta et al. 2017]	P	82.25	70.50	75.75	80.50	76.50	73.75	69.00	61.00	62.25
		R	80.75	69.50	74.50	80.75	76.50	73.00	68.75	59.25	60.25
		F	80.00	67.50	72.75	80.00	75.50	71.50	67.25	58.50	58.75
		A	81.10	69.48	74.44	80.56	76.51	72.83	69.06	59.28	60.33
BERT Embeddings	BiLTM-CRF-A	P	53.35	40.44	51.27	62.89	55.47	52.06	39.70	41.51	45.97
		R	78.05	55.78	61.6	73.00	65.37	66.89	60.29	66.62	75.01
		F	63.38	46.89	55.96	67.57	60.02	58.55	47.88	51.15	57.01
		A	73.97	65.03	66.88	76.60	75.26	73.85	67.04	60.40	65.16
	BiLTM-CRF-B	P	76.91	73.24	66.74	85.58	76.34	70.46	71.45	71.45	76.79
		R	86.94	83.27	73.91	88.31	81.24	86.36	83.28	81.02	87.26
		F	81.62	77.93	70.14	86.93	78.71	77.60	76.91	75.94	81.69
		A	81.81	77.39	73.65	86.63	87.26	85.52	80.36	75.35	78.86
Fasttext Embeddings	BiLTM-CRF-A	P	63.53	61.54	67.76	77.34	69.02	51.35	68.10	68.51	59.25
		R	73.52	72.30	81.02	84.45	85.25	70.59	83.17	85.88	72.59
		F	68.16	66.49	73.80	80.74	76.28	59.45	74.88	76.21	65.25
		A	75.12	79.95	76.71	83.99	82.39	75.50	82.23	76.84	73.35
	BiLTM-CRF-B	P	84.77	71.92	77.40	96.85	92.79	85.13	88.47	83.93	87.53
		R	89.65	85.11	85.97	97.38	96.57	94.52	95.03	91.49	91.51
		F	**87.14**	**77.96**	**81.46**	**97.11**	**94.58**	**89.58**	**91.63**	**87.55**	**89.48**
		A	**87.15**	**81.68**	**83.87**	**97.45**	**96.56**	**93.23**	**91.30**	**86.31**	**90.42**

sequence labeling based on BiLSTM-CRF-A and BiLSTM-CRF-B. These two models are referred to as BERT-BiLSTM-CRF-A and BERT-BiLSTM-CRF-B, respectively. Finally, the performance of all the baselines is compared with the model based on *fasttext* code-mixed embeddings combined with BiLSTM-CRF-A and BiLSTM-CRF-B, referred to as FastText-BiLSTM-CRF-A and FastText-BiLSTM-CRF-B, respectively.

As shown in Tables 16 and 17, the FastText-BiLSTM-CRF-B model gives more accuracy and F1-score (in bold font) on all datasets compared with two baselines CRF-Jamatia and CRF-Gupta and FastText-BiLSTM-CRF-A model. FastText-BiLSTM-CRF-B also performs better compared to base-lines BERT-BiLSTM-CRF-A and BERT-BiLSTM-CRF-B.

Though the first two baseline approaches CRF-Jamatia and CRF-Gupta are using the CRF-based sequence labeling model, CRF-Gupta performs better than CRF-Jamatia due to consideration of few more task-specific features (such as word probability, word normalization, stemming and phonetic normalization, word position, number of upper-case characters).

As shown in Table 16, CRF-Gupta has shown approximately 10% of F1-score improvement compared to CRF-Jamatia for *Facebook* data of all three languages. In *Twitter* and *Whatsapp* domains of *Bengali* and *Telugu* languages, CRF-Gupta performs on par with CRF-Jamatia.

As shown in Table 16, CRF-Gupta, has shown a minimum 12% improvement in F1-score than the BERT-BiLSTM-CRF-A across all domains and across all languages (except *Telugu Whatsapp*

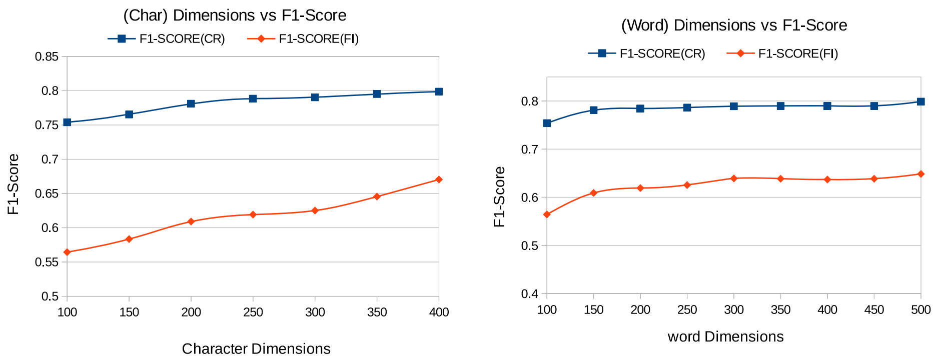

Fig. 6. Performance of deep learning–based POS tagger for different dimensions of character-sequence–based embeddings (left) and word embeddings (right).

fine-grained data). CRF-Gupta has also shown the improvement in F1-score by a minimum of 4.72% compared to that of FastText-BiLSTM-CRF-A for all combinations of fine-grained data (except *Telugu Twitter* and *Whatsapp* data and *Hindi Facebook* data). Finally, as shown in Table 16, the improvement in performance of F1-score of FastText-BiLSTM-CRF-B varies between 5.8% to 42.1% compared to that of CRF-Jamatia, between 4.38% to 37.53% compared to that of CRF-Gupta, and between 0.78% to 29.14% compared to BERT-BiLSTM-CRF-B over all combinations of fine-grained datasets.

Similarly, as shown in Table 17, the performance improvement in F1-score of FastText-BiLSTM-CRF-B varies between 10.43% and 38.9% compared to CRF-Jamatia, between 6.79% and 30.73% compared to CRF-Gupta, and between 7.08% and 19.6% compared to BERT-BiLSTM-CRF-B for all combinations of coarse-grained labeling. Further analysis of the improvements in the performance of FastText-BiLSTM-CRF-B over CRF-Jamatia and CRF-Gupta reveals that the lower level of improvement in performance is observed in *Bengali* and *Hindi*, while the largest improvement in performance is observed in *Telugu* language. From the level of code-mixing indicated in Table 8, we can observe that *Telugu* has the highest fraction of sentences (837 out of 1,979) with CSL greater than 40 (42%), while *Hindi* and *Bengali* have 5% to 10% of total sentences with CSL above 40. As CSL level in the code-mixed corpus increases, the F1-score improvement margin of FastText-BiLSTM-CRF-B over CRF-Jamatia and CRF-Gupta is also observed to increase.

4.4.5 *The Effect of Contextual Word Embeddings and Character-sequence–based Word Embeddings:.*

Figure 6(a) depicts the F1-score improvement in BiLSTM-CRF model while increasing the dimensions of character-sequence–based embeddings from 100 to 400 by fixing word-embedding dimensions to 100. Increasing the character-embedding dimensions up to 400 has resulted in F1-score improvement of up to 79.86% and 67.03% for coarse-grained and fine-grained *Hindi* datasets, respectively. Figure 6(b) depicts the F1-score improvement in BiLSTM-CRF model while increasing the word-embedding dimensions from 100 to 500 by fixing character-embedding dimensions to 100. Increasing the word-embedding dimensions up to 500 the F1-score improved up to 79.86% and 65.0% for coarse-grained (CR) and fine-grained (FI) *Hindi* datasets, respectively.

5 DISCUSSION

In this section, we analyze the performance of CRF and BiLSTM-CRF with various test-set examples of CMSM text. Code-mixing poses significant challenges for sequence labeling tasks such as POS tagging, NER, and so on. There are many reasons for the inadequacy of CRF-based POS tagger.

Table 18. Example Code-mixed Sentence from Test Dataset

			Ambiguity						
Sentence :	Mastttt	hoti	h	but	abhi	main	tujhse	naraj	hu
BiLSTM-CRF:	JJ	V_VM	V_VAUX	CC	RB_ALC	PR_PRP	PR_PRP	N_NN	V_VM
CRF:	N_NN	V_VM	**V_VM**	CC	RB_ALC	**PSP**	PR_PRP	**V_VM**	V_VM

Table 19. An Example for Intra-code-mixed Sentence from Test Dataset

sentence:	Hi	bhayya	Bhai	Black	Friday	ki	deal	mein	mila	Kaafi	sastaa
lang-tag:	en	hi	hi	en	en	hi	en	hi	hi	hi	hi
BiLSTM-CRF:	RP_INJ	N_NN	N_NN	JJ	N_NNP	PSP	N_NN	PSP	V_VM	QT_QTF	N_NN
CRF:	RP_INJ	N_NN	N_NN	JJ	**N_NN**	PSP	N_NN	PSP	**N_NN**	**V_VM**	**V_VM**

Table 20. Inter Code-mixing Examples-1

Sentence:	dost	Shadi	karlou	nahi	tou	Bihar	wale	le	jayege	aur	Shadi	karwa	denge	....	Free	advice	to	all	Punjab	boys
lang-tag:	hi	hi	hi	hi	hi	ne	hi	hi	hi	hi	hi	hi	hi	univ	en	en	en	en	en	en
CRF :	N_NN	N_NN	V_VM	DT	**V_VM**	N_NNP	RP_RPD	**PSP**	V_VM	CC	N_NN	V_VM	V_VM	RD_PUNC	**V_VM**	N_NN	PSP	DT	**QT_QTF**	N_NN
BiLSTM-CRF:	N_NN	N_NN	V_VM	DT	RP_RPD	N_NNP	RP_RPD	V_VM	V_VM	CC	N_NN	V_VM	V_VM	RD_PUNC	JJ	N_NN	PSP	DT	N_NNP	N_NN

5.1 Resolving Ambiguity Problems in Label Prediction

CRF-based POS tagger can capture only the structural information (general sequence pattern) in training, whereas deep learning–based POS taggers use contextual word representations that contain both structural and semantic information. This additional semantic information helps the model to resolve the ambiguity of labels in sequence labeling.

Let us consider the word *"h"* (it means *hai* in *Hindi*) in the following example sentence in Table 18. This table shows the test instance and its label sequences predicted by BiLSTM-CRF and CRF models. Usually the word *"h"* is associated with "V_VM" or "V_VAUX," depending on the context. In this test sentence, the correct tag is "V_VAUX." The mis-classification of CRF can be attributed to number of "V_VM" tags associated with the word *"h"* in identical neighboring words and tags. BiLSTM-CRF uses embeddings as representations of words that helped it to predict the correct label based on the usage of "h" and its other variants such as "hai," "hi," and so on.

5.2 Predicting Labels for Varying Code-mixing Levels

Let us consider an intra code-mixed sentence, shown in Table 19, in which the *Hindi* and *English* words are mixed together in a single sentence. Though CRF uses language-tag information, it has failed to predict the labels of *Friday, mila, Kaafi*, and *sastaa* words. The BiLSTM-CRF model is superior in these cases, as its layered structure helps model the non-linearity in the prediction task. Let us consider an example for the inter-sentential code-mixing in Table 20. In this case, we observed that most of the time CRF model agrees with BiLSTM model, because inter-code-mixed sentences have only one code-switching point in the input sequence. Except at this code-switching point, they have a linear relation between the word sequence and its label sequence. Let us consider another inter code-mixed sentence in Table 21, in which the code-mixed structure (underlined) is uniform and both CRF and BiLSTM predicted the fine-grained sequence correctly.

5.3 Predicting Labels for OOV Words

As discussed in the Section 1, predicting labels of OOV words in CMSM data is a challenging issue. If the input word is an OOV word or its context has OOV words, then CRF model may predict

Table 21. Inter-code-mixing Examples-2

	univ	eng	eng	eng	univ	hi	hi	hi	hi	hi	hi	univ
Sentence:	@ranote_aman	it	was	good	.	Wapis	aane	ka	Mann	nahi	tha	.
BiLSTM-CRF:	@	PR_PRP	V_VM	JJ	RD_PUNC	N_NN	V_VM	PSP	N_NN	DT	V_VM	RD_PUNC
CRF:	@	PR_PRP	V_VM	JJ	RD_PUNC	N_NN	V_VM	PSP	N_NN	DT	V_VM	RD_PUNC

Table 22. Example Sentence with OOV Words

			rare		*rare*				
Sentence :	It'ss	all	abt	wo	H10	wali	LADKI	in	AUS
BiLSTM-CRF :	PR_PRP	DT	CC	PR_PRP	N_NNP	RP_RPD	N_NN	PSP	N_NNP
CRF :	PR_PRP	DT	**N_NN**	PSP	**$**	RP_RPD	N_NN	PSP	N_NNP

Table 23. Analysis Report on Hindi-English Test Results

CSL Level (%)	Total Sentences	Total Tokens (Avg sentence length)	No. of tokens correctly predicted (% of tokens correctly predicted) CRF	BiLSTM-CRF
0–10	253	4,423 (17.48)	3,405 (76.98)	3,578 (80.90)
11–20	46	898 (19.52)	609 (67.82)	685 (76.28)
21–30	131	2,284 (17.44)	1,689 (73.95)	1,784 (78.11)
31–40	72	1,239 (17.21)	901 (72.72)	963 (78.13)
41<	24	221 (9.20)	172 (77.81)	174 (78.74)

Table 24. Analysis Report on Bengali-English Test Results

CSL Level (%)	Total Sentences	Total Tokens (Avg sentence length)	No. of tokens correctly predicted (% of tokens correctly predicted) CRF	BiLSTM-CRF
0–10	59	736 (12.47)	516 (70.11)	558 (75.84)
11–20	7	232 (33.14)	163 (70.26)	174 (75.00)
21–30	21	799 (38.05)	561 (70.21)	608 (76.09)
31–40	23	631 (27.43)	450 (71.32)	510 (80.82)
41<	15	290 (19.33)	212 (73.10)	235 (81.03)

a wrong label, because CRF fails in the utilization of character-sequence information. BiLSTM-CRF–based POS tagger captures the character-sequence information of a word in character-level BiLSTM layer. Such information enables the tagger to deal with OOV words gracefully. Consider an example sentence from test set, as shown in Table 22, which contains two OOV words. The OOV word "abt" and rare word "H10" are not handled by CRF, whereas BiLSTM predicted the correct labels.

5.4 Performance at Various Code-switching Complexity (CSL) Levels

In this section, we analyze the tagging performance at various code-switching levels. Tables 23, 24, and 25 summarize the token-level performance of CRF and BiLSTM-CRF in different CSL levels. We can observe that, as length of the sentences increases along with CSL, the performance gap

Table 25. Analysis Report on Telugu-English Test Results

CSL Level (%)	Total Sentences	Total Tokens (Avg sentence length)	No. of tokens correctly predicted (% of tokens correctly predicted) CRF	BiLSTM-CRF
0–10	27	187 (6.93)	140 (74.86)	163 (87.17)
11–20	16	265 (16.56)	183 (69.06)	201 (75.85)
21–30	62	1,050 (16.94)	725 (69.04)	799 (76.09)
31–40	124	2,112 (17.03)	1,329 (62.93)	1,491 (70.59)
41<	167	2,409 (14.43)	1,671 (69.36)	1,703 (70.69)

between BiLSTM-CRF and CRF is further widening. For CSL of below 10%, the difference between BiLSTM-CRF and CRF is lowest. As either the average length of sentences or CSL level increases, the difference between the two models widens.

5.5 Error Analysis

In the discussion section, we explained how the BiLSTM-CRF model could perform better compared to the CRF model by presenting many practical examples. Though we handle the misspelled and OOV words using contextual information and character-sequence information of a word, the BiLSTM-CRF accuracies are 77.15%, 85.26%, and 74.88% for *Bengali*, *Hindi*, and *Telugu* languages, respectively. This indicates that the accuracies are not in line with accuracies observed on uni-language corpora. In this section, we do the error analysis for the errors seen in the output of our model.

We analyzed the output of BiLSTM-CRF model on the *Hindi* dataset for each POS tag. If we neglect the unnecessary misprediction cases such as predicting RD_SYM (Symbol) instead of RD_PUNC (Punctuation symbol), predicting E (emoticon) instead of RD_UNK (unknown word), predicting RD_UNK instead of RD_RDF (foreign word), and so on, nearly 62% of mispredictions are lying in the prediction of V_VM(main verb), V_VAUX (auxiliary verb), JJ (adjective), N_NNP (proper noun), and N_NN (common noun) labels. We summarize our observations in error analysis below:

- 17.21% of the errors are due to misprediction of common nouns (N_NN) as main verbs or proper nouns.
- 12% of the errors are due to misprediction of adjectives (JJ) as proper nouns N_NNP or main verb V_VM.
- 9.8% of the errors are due to misprediction of auxiliary verbs of which 90% are tagged as main verbs (V_VM). Such mispredictions are mainly due to the confusion in the order of auxiliary and main verb word usage between the two languages of the mixed script. *English* text usually contains auxiliary verb followed by the main verb, while *Hindi* has the main verb followed by the auxiliary verb.
- 14.26% of the errors are due to misprediction of main verbs as an auxiliary verb (V_VAUX) or common noun (N_NN).

6 CONCLUSION

The conventional approach to NLP tasks involving POS tagging uses CRF as a baseline technique. Recently, Recurrent Neural Networks—specifically, BiLSTM as a deep learning technique—has shown appreciable improvement also for cases with unstructured long sentences in monolingual texts. We extended the deep learning approach for multilingual code-mixed texts in our proposed model, BiLSTM-CRF.

In this article, we proposed a model that uses BiLSTM and CRF consecutively for POS tagging in code-mixed text. The implemented process first uses BiLSTM for generating a sequence of tags and then uses CRF to find the best score on the generated sequence. Although BiLSTM captures the long-term dependencies and labels a word based on neighboring tags, in code-mixed scenarios, the grammatical structure changes drastically, thereby there is need to re-score and re-model the sequence. Therefore, we build CRF on top of BiLSTM to take tagging decision globally for this NLP task.

We tested our model with the baseline approach CRF for POS tagging in code-mixing sentences and saw an improved performance in handling many problems, such as non-linearity, ambiguity, OOV words, rare word handling, and so on. In our work, we are also providing our datasets and tools for further research into the domain. Most importantly, we have incorporated a different approach for creating contextual word vector (with character-sequence embedding and word embedding) and have provided the repository with different dimensions to be used by others.

REFERENCES

Hanan Aldarmaki and Mona Diab. 2015. Robust part-of-speech tagging of Arabic text. In *Proceedings of the 2nd Workshop on Arabic Natural Language Processing*. Association for Computational Linguistics, 173–182. DOI: https://doi.org/10.18653/v1/W15-3222

Ozkan Aslan, Serkan Gunal, and Bekir Taner Dincer. November 2018. On constituent chunking for Turkish. *Inf. Proc. Manag.* 54, 6 (Nov. 2018), 1262–1276.

Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. 2016. Towards deep learning in Hindi NER: An approach to tackle the labelled data sparsity. In *Proceedings of the 13th International Conference on Natural Language Processing*. NLP Association of India, 154–160. Retrieved from http://www.aclweb.org/anthology/W16-6320.

J. Atserias, B. Casas, E. Comelles, M. González, L. Padró, and M. Padró. 2006. FreeLing 1.3: Syntactic and semantic services in an open-source NLP library. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA). Retrieved from http://www.aclweb.org/anthology/L06-1108.

Ngo Xuan Bach, Nguyen Dieu Linh, and Tu Minh Phuong. 2018. An empirical study on POS tagging for Vietnamese social media text. *Comput. Speech Lang.* 50 (2018), 1–15.

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. "I am borrowing ya mixing?" An analysis of English-Hindi code mixing in Facebook. In *Proceedings of the 1st Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics, 116–126. DOI: https://doi.org/10.3115/v1/W14-3914

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the 1st Workshop on Computational Approaches to Code Switching*. 13–23.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Trans. Assoc. Computat. Ling.* 5, 1 (2017), 135–146.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computat. Ling.* 18, 4 (1992), 467–479.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identification using CRF: Code-switching shared task report of MSR india system. In *Proceedings of the 1st Workshop on Computational Approaches to Code Switching*. 73–79.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Trans. Assoc. Computat. Ling.* 4, 1 (2016), 357–370.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*. 2067–2075.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. ACM, New York, NY, 160–167. DOI: https://doi.org/10.1145/1390156.1390177

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

Manuel Vilares Ferro, Víctor Manuel Darriba Bilbao, and Francisco José Ribadas Pena. 2017. Modeling of learning curves with applications to POS tagging. *Comput. Speech Lang.* 41 (2017), 1–28.

Boris A. Galitsky. 2016. Generalization of parse trees for iterative taxonomy learning. *Inf. Sci.* 329 (2016), 125–143.

Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), 23–28.

Souvick Ghosh, Satanu Ghosh, and Dipankar Das. 2016. Part-of-speech tagging of code-mixed social media text. In *Proceedings of the 2nd Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics, 90–97. DOI: https://doi.org/10.18653/v1/W16-5811.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proceedings of the 49th Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2 (HLT'11)*. Association for Computational Linguistics, 42–47. Retrieved from http://dl.acm.org/citation.cfm?id=2002736.2002747.

Yi Guo, Zhiqing Shao, and Nan Hua. 2010. A cognitive interactionist sentence parser with simple recurrent networks. *Inf. Sci.* 180, 23 (2010), 4695–4705.

Deepak Gupta, Ankit Lamba, Asif Ekbal, and Pushpak Bhattacharyya. 2016. Opinion mining in a code-mixed environment: A case study with government portals. In *Proceedings of the 13th International Conference on Natural Language Processing*. NLP Association of India, 249–258. Retrieved from http://www.aclweb.org/anthology/W/W16/W16-6331.

Deepak Gupta, Shubham Tripathi, Asif Ekbal, and Pushpak Bhattacharyya. 2017. SMPOST: Parts of speech tagger for code-mixed Indic social media text. *arXiv preprint arXiv:1702.00167* (2017).

Parth Gupta, Kalika Bali, Rafael E. Banchs, Monojit Choudhury, and Paolo Rosso. 2014. Query expansion for mixed-script information retrieval. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'14)*. ACM, New York, NY, 677–686. DOI: https://doi.org/10.1145/2600428.2609622.

Marjolein Gysels. 1992. French in urban Lubumbashi Swahili: Codeswitching, borrowing, or both? *J. Multiling. Multicult. Dev.* 13, 1–2 (1992), 41–55.

Donald Hindle. 1989. Acquiring disambiguation rules from text. In *Proceedings of the 27th Meeting on Association for Computational Linguistics (ACL'89)*. Association for Computational Linguistics, 118–125. DOI: https://doi.org/10.3115/981623.981638

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computat.* 9, 8 (1997), 1735–1780.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2410–2420. DOI: https://doi.org/10.18653/v1/P16-1228

Aaron Jaech, George Mulcaire, Shobhit Hathi, Mari Ostendorf, and Noah A. Smith. 2016. Hierarchical character-word models for language identification. In *Proceedings of the 4th International Workshop on Natural Language Processing for Social Media (SocialNLP@EMNLP'16)*. 84–93. DOI: https://doi.org/10.18653/v1/W16-6212

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed English-Hindi Twitter and Facebook chat messages. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. INCOMA Ltd., 239–248. Retrieved from http://www.aclweb.org/anthology/R15-1033.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of Hindi-English code mixed text. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, 2482–2491. Retrieved from http://aclweb.org/anthology/C16-1234.

Katharina Kann, Johannes Bjerva, Isabelle Augenstein, Barbara Plank, and Anders Søgaard. 2018. Character-level supervision for low-resource POS tagging. In *Proceedings of the Workshop on Deep Learning Approaches for Low-resource NLP*. Association for Computational Linguistics, 1–11. Retrieved from http://aclweb.org/anthology/W18-3401.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'02)*. European Language Resources Association (ELRA). Retrieved from http://www.aclweb.org/anthology/L02-1302.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980 (2014).

Subham Kumar, Anwesh Sinha Ray, Sabyasachi Kamila, Asif Ekbal, Sriparna Saha, and Pushpak Bhattacharyya. 2016. Improving document ranking using query expansion and classification techniques for mixed script information retrieval. In *Proceedings of the 13th International Conference on Natural Language Processing*. NLP Association of India, 81–89. Retrieved from http://www.aclweb.org/anthology/W/W16/W16-6311.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*. Morgan Kaufmann Publishers Inc., 282–289. Retrieved from http://dl.acm.org/citation.cfm?id=645530.655813.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 260–270. DOI: https://doi.org/10.18653/v1/N16-1030

Ying Li and Pascale Fung. 2012. Code-switch language model with inversion constraints for mixed language speech recognition. In *Proceedings of the International Conference on Computational Linguistics (COLING'12)*. The COLING 2012 Organizing Committee, 1671–1680. Retrieved from http://aclweb.org/anthology/C12-1102.

Xiaohua Liu and Ming Zhou. 2013. Two-stage NER for tweets with clustering. *Inf. Proc. Manag.* 49 (2013), 264–273. DOI: https://doi.org/10.1016/j.ipm.2012.05.006

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 879–888. Retrieved from https://aclweb.org/anthology/D/D15/D15-1104.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. *arXiv preprint arXiv:1904.07094* (2019).

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguist.* 19, 2 (June 1993), 313–330. Retrieved from http://dl.acm.org/citation.cfm?id=972470.972475.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Meeting on Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, 91–98. DOI: https://doi.org/10.3115/1219840.1219852

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 3111–3119.

Rudra Murthy, Mitesh M. Khapra, and Pushpak Bhattacharyya. 2018. Improving NER tagging performance in low-resource languages via multilingual learning. *ACM Trans. Asian Low-Resour. Lang. Inf. Proc.* 18, 2, Article 9 (Dec. 2018), 20 pages. DOI: https://doi.org/10.1145/3238797

Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and POS tagging. In *Proceedings of the 45th Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, 217–220.

Phuong-Thai Nguyen, Xuan-Luong Vu, Thi-Minh-Huyen Nguyen, Van-Hiep Nguyen, and Hong-Phuong Le. 2009. Building a large syntactically annotated corpus of vietnamese. In *Proceedings of the 3rd Linguistic Annotation Workshop (ACL-IJCNLP'09)*. Association for Computational Linguistics, 182–185. Retrieved from http://dl.acm.org/citation.cfm?id=1698381.1698416.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*. Association for Computational Linguistics, Article 64. DOI: https://doi.org/10.3115/1220355.1220365

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 380–390.

Rana D. Parshad, Suman Bhowmick, Vineeta Chand, Nitu Kumari, and Neha Sinha. 2016. What is India speaking? Exploring the "Hinglish" invasion. *Phys. A: Statist. Mech. Applic.* 449 (2016), 375–389.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the 18th Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 78–86. Retrieved from http://www.aclweb.org/anthology/W/W14/W14-1609.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1532–1543.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA).

Prakash B. Pimpale and Raj Nath Patel. 2016. Experiments with POS tagging code-mixed Indian social media text. *arXiv preprint arXiv:1610.09799* (2016).

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018a. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1543–1553.

Adithya Pratapa, Monojit Choudhury, and Sunayana Sitaram. 2018b. Word embeddings for code-mixed language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 3067–3072. Retrieved from https://www.aclweb.org/anthology/D18-1344.

L. R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (Feb. 1989), 257–286. DOI: https://doi.org/10.1109/5.18626

Khyathi Chandu Raghavi, Manoj Kumar Chinnakotla, and Manish Shrivastava. 2015. "Answer Ka Type Kya He?": Learning to classify questions in code-mixed language. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15 Companion)*. ACM, New York, NY, 853–858. DOI: https://doi.org/10.1145/2740908.2743006

Pattabhi R. K. Rao and Sobha Lalitha Devi. 2016. CMEE-IL: Code mix entity extraction in Indian languages from social media Text@ FIRE 2016-An overview. In *Proceedings of the Forum for Information Retrieval Evaluation (FIRE'16) (Working Notes)*. 289–295.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL'09)*. Association for Computational Linguistics, 147–155. Retrieved from http://dl.acm.org/citation.cfm?id=1596374.1596399.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Retrieved from http://www.aclweb.org/anthology/W96-0213.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1524–1534.

Salvatore Romeo, Giovanni Da San Martino, Yonatan Belinkov, Alberto Barrón-Cedeño, Mohamed Eldesouki, Kareem Darwish, Hamdy Mubarak, James Glass, and Alessandro Moschitti. 2019. Language processing and learning models for community question answering in Arabic. *Inf. Proc. Manag.* 56, 2 (2019), 274–290.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi-English speakers do on Twitter? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1131–1141.

Kamal Sarkar. 2016. Part-of-speech tagging for code-mixed Indian social media text at ICON 2015. *arXiv preprint arXiv:1601.01195* (2016).

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Sig. Proc.* 45, 11 (1997), 2673–2681.

Mike Schuster and John Wiley. 1999. Neural networks for speech processing. *Encyclopedia of Electrical and Electronic Engineering*. John Wiley and Sons.

Royal Sequiera, Monojit Choudhury, and Kalika Bali. 2015. POS tagging of Hindi-English code mixed text from social media: Some machine learning experiments. In *Proceedings of the 12th International Conference on Natural Language Processing*. NLP Association of India, 237–246. Retrieved from http://www.aclweb.org/anthology/W/W15/W15-5936.

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Shrivastava, Radhika Mamidi, and Dipti M. Sharma. 2016. Shallow parsing pipeline—Hindi-English code-mixed social media text. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 1340–1345. Retrieved from http://www.aclweb.org/anthology/N16-1159.

Rajat Singh, Nurendra Choudhary, and Manish Shrivastava. 2018. Automatic normalization of word variations in code-mixed social media text. *arXiv preprint arXiv:1804.00804* (2018).

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad Al Ghamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the 1st Workshop on Computational Approaches to Code Switching*. 62–72.

Thamar Solorio and Yang Liu. 2008a. Learning to predict code-switching points. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*. Association for Computational Linguistics, 973–981. Retrieved from http://dl.acm.org/citation.cfm?id=1613715.1613841.

Thamar Solorio and Yang Liu. 2008b. Part-of-speech tagging for English-Spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*. Association for Computational Linguistics, 1051–1060. Retrieved from http://dl.acm.org/citation.cfm?id=1613715.1613852.

Nagesh Bhattu Sristy, N. Satya Krishna, B. Shiva Krishna, and Vadlamani Ravi. 2017. Language identification in mixed script. In *Proceedings of the 9th Meeting of the Forum for Information Retrieval Evaluation*. ACM, 14–20.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Meeting of the Association for Computational Linguistics*. 5326–5331.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 1017–1024.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.* 6, Sept. (2005), 1453–1484.

P. V. Veena, M. Anand Kumar, and K. P. Soman. 2018. Character embedding for language identification in Hindi-English code-mixed social media text. *Comput. Sistemas* 22, 1 (2018), 65–74.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. POS tagging of English-Hindi code-mixed social media content. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 974–979.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232* (2019).

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 72–77.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 647–657. Retrieved from http://www.aclweb.org/anthology/D13-1061.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2018. SDNet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593* (2018).

Received April 2019; revised October 2019; accepted January 2020