Language Identification in Code-Mixed Data using Multichannel Neural Networks and Context Capture

Soumil Mandal
Department of CSE
SRM University, Chennai, India
soumil.mandal@gmail.com

Anil Kumar Singh
Department of CSE
IIT (BHU), Varanasi, India
aksingh.cse@iitbhu.ac.in

Abstract

An accurate language identification tool is an absolute necessity for building complex NLP systems to be used on code-mixed data. Lot of work has been recently done on the same, but there's still room for improvement. Inspired from the recent advancements in neural network architectures for computer vision tasks, we have implemented multichannel neural networks combining CNN and LSTM for word level language identification of code-mixed data. Combining this with a Bi-LSTM-CRF context capture module, accuracies of 93.28% and 93.32% is achieved on our two testing sets.

1 Introduction

With the rise of social media, the amount of mineable data is rising rapidly. Countries where bilingualism is popular, we see users often switch back and forth between two languages while typing, a phenomenon known as code-mixing or code-switching. For analyzing such data, language tagging acts as a preliminary step and its accuracy and performance can impact the system results to a great extent. Though a lot of work has been done recently targeting this task, the problem of language tagging in code-mixed scenario is still far from being solved. Code-mixing scenarios where one of the languages have been typed in its transliterated from possesses even more challenges, especially due to inconsistent phonetic typing. On such type of data, context capture is extremely hard as well. Proper context capture can help in solving problems like ambiguity, that is word forms which are common to both the languages, but for which, the correct tag can be easily understood by knowing the context. An additional issue is a lack of available code-mixed data. Since most of the tasks require supervised models, the bottleneck of data crisis affects the performance quite a lot, mostly due to the problem of over-fitting.

In this article, we present a novel architecture, which captures information at both word level and context level to output the final tag. For word level, we have used a multichannel neural network (MNN) inspired from the recent works of computer vision. Such networks have also shown promising results in NLP tasks like sentence classification (Kim, 2014). For context capture, we used Bi-LSTM-CRF. The context module was tested more rigorously as in quite a few of the previous work, this information has been sidelined or ignored. We have experimented on Bengali-English (Bn-En) and Hindi-English (Hi-En) code-mixed data. Hindi and Bengali are the two most popular languages in India. Since none of them have Roman as their native script, both are written in their phonetically transliterated from when code-mixed with English.

2 Related Work

In the recent past, a lot of work has been done in the field of code-mixing data, especially language tagging. King and Abney (2013) used weakly semi-supervised methods for building a world level language identifier. Linear chain CRFs with context information limited to bigrams was employed by Nguyen and Doğruöz (2013). Logistic regression along with a module which gives code-switching probability was used by Vyas et al. (2014). Multiple features like word context, dictionary, n-gram, edit distance were used by Das and Gambäck (2014). Jhamtani et al. (2014) combined two classifiers into an ensemble model for Hindi-English code-mixed LID. The first classifier used modified edit distance, word frequency and character n-grams as features. The second classifier used the output of the first classifier for the current word, along with language tag and POS tag of neighboring to give the final tag. Piergallini et al.

Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text, pages 116–120
Brussels, Belgium, Nov 1, 2018.

(2016) made a word level model taking char n-grams and capitalization as feature. Rijhwani et al. (2017) presented a generalized language tagger for arbitrary large set of languages which is fully unsupervised. Choudhury et al. (2017) used a model which concatenated word embeddings and character embeddings to predict the target language tag. Mandal et al. (2018a) used character embeddings along with phonetic embeddings to build an ensemble model for language tagging.

3 Data Sets

We wanted to test our approach on two different language pairs, which were Bengali-English (Bn-En) and Hindi-English (Hi-En). For Bn-En, we used the data prepared in Mandal et al. (2018b) and for Hi-En, we used the data prepared in Patra et al. (2018). The number of instances of each type we selected for our experiments was 6000. The data distribution for each type is shown in Table 1.

	Train	Dev	Test
Bn	3000 27245/6189 22.4	1000 9144/2836 21.4	2000 17967/4624 22.5
Hi	3000 26384/5630 18.8	1000 8675/2485 18.7	2000 16114/4286 18.2

Table 1: Data distribution.

Here, the first value represents the number of instances taken, the second line represents the total number of indic tokens / unique number of indic tokens, and the third line represents the mean code-mixing index (Das and Gambäck, 2014).

4 Architecture Overview

Our system is comprised of two supervised modules. The first one is a multichannel neural network trained at word level, while the second one is a simple bidirectional LSTM-CRF trained at instance level. The second module takes the input from the first module along with some other features to output the final tag. Individual modules are described in detail below.

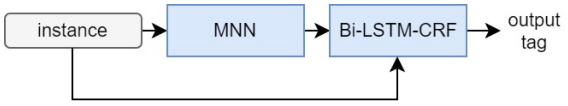

Figure 1: Architecture overview.

5 Word - Multichannel Neural Network

Inspired from the recent deep neural architectures developed for image classification tasks, especially the Inception architecture (Szegedy et al., 2015), we decided to use a very similar concept for learning language at word level. This is because the architecture allows the network to capture representations of different types, which can be really helpful for NLP tasks like these as well. The network we developed has 4 channels, the first three enters into a Convolution 1D (Conv1D) network (LeCun et al., 1999), while the fourth one enters into a Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997). The complete architecture is showed in Fig 2.

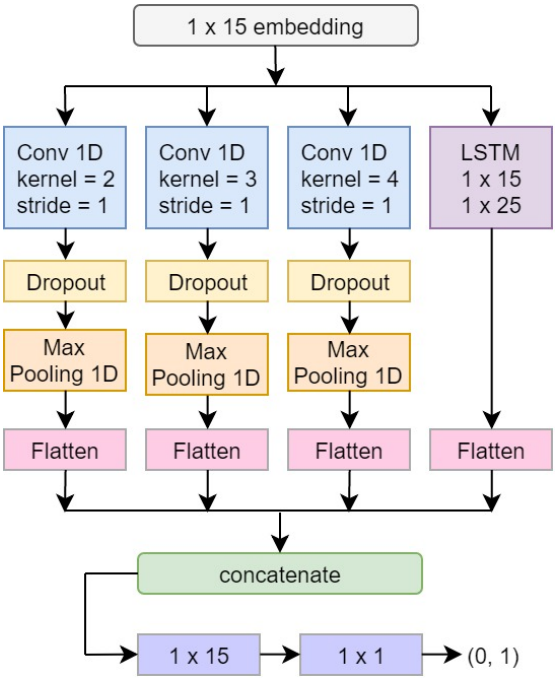

Figure 2: Multichannel architecture for word level tagging.

Character embeddings of length 15 is fed into all the 4 channels. The first 3 Conv 1D cells are used to capture n-gram representations. All the three Conv 1D cells are followed by Dropout (rate 0.2) and Max Pooling cells. Adding these cells help in controlling overfitting and learning invariances, as well as reduce computation cost. Activation function for all the three Conv 1D nets was relu. The fourth channel goes to an LSTM stack with two computational layers of sizes 15, and 25 orderly. For all the four channels, the final outputs are flattened and concatenated. This concatenated vector is then passed on to two dense layers of sizes 15

(activation relu) and 1 (activation sigmoid). For the two models created, Bn-En and Hi-En, target labels were 0 for the Bn/Hi and 1 for En. For implementing the multichannel neural network for word level classification, we used the Keras API (Chollet et al., 2015).

5.1 Training

Word distribution for training is described in Table 1. All indic tagged tokens were used instead of just unique ones of respective languages. The whole model was compiled using loss as binary cross-entropy, optimization function used was adam (Kingma and Ba, 2014) and metric for training was accuracy. The batch size was set to 64, and number of epochs was set to 30. Other parameters were kept at default. The training accuracy and loss graphs for both Bn and Hi are shown below. As the MNN model produces a sigmoid output, to convert the model into a classifier, we decided to use a threshold based technique identical to the one used in Mandal et al. (2018a) for tuning character and phonetic models. For this the development data was used, threshold for Bn was calculated to be $\theta \leq 0.95$, while threshold for Hi was calculated to be $\theta \leq 0.89$. Using these, the accuracies on the development data was 93.6% and 92.87% for Bn and Hi respectively.

6 Context - Bi-LSTM-CRF

The purpose of this module is to learn representation at instance level, i.e. capture context. For this, we decided to use bidirectional LSTM network with CRF layer (Bi-LSTM-CRF) (Huang et al., 2015) as it has given state of the art results for sequence tagging in the recent past. For converting the instances into embeddings, two features were used namely, sigmoid output from MNN (*fe1*), character embedding (*fe2*) of size 30. The final feature vector is created by concatenating these two, *fe* = (*fe1*, *fe2*). The model essentially learns code-switching chances or probability taking into consideration character embeddings and sigmoid scores of language tag. We used the open sourced neural sequence labeling toolkit, NCRF++ (Yang and Zhang, 2018) for building the model.

6.1 Training

Instance distribution for training is described in Table 1. The targets here were the actual language labels (0 for the Bn/Hi and 1 for En). The hyper-parameters which we set mostly follow Yang et al. (2018) and Ma and Hovy (2016). Both the models (Bn-En & Hi-En) had identical parameters. L_2 regularization λ was set at 1e-8. Learning rate η was set to 0.015. Batch size was kept at 16 and number of epochs was set to 210. Mini-batch stochastic gradient descent (SGD) with decaying learning rate (0.05) was used to update the parameters. All the other parameters were kept at default values. This setting was finalized post multiple experiments on the development data. Final accuracy scores on the development data was 93.91% and 93.11% for Bn and Hi respectively.

7 Evaluation

For comparison purposes, we decided to use the character encoding architecture described in Mandal et al. (2018a) (stacked LSTMs of sizes 15, 35, 25, 1) with identical hyper-parameters for both Bn and Hi. Training data distribution while creating the baseline models were in accordance with Table 1. The thresholds for the baseline models calculated on the development data was found to be $\theta \leq 0.91$ and $\theta \leq 0.90$ for Bn and Hi respectively. The results (in %) for each of the language pairs are shown below.

	Acc	**Prec**	**Rec**	**F1**
baseline	88.32	89.64	87.72	88.67
word model	92.87	94.33	91.84	93.06
context model	**93.28**	94.33	92.68	93.49

Table 2: Evaluation on Bn.

From Table 2 we can see that the jump in accuracy from baseline to the word model is quite significant (4.55%). From word to context model, though not much, but still an improvement is seen (0.41%).

	Acc	**Prec**	**Rec**	**F1**
baseline	88.28	88.57	88.01	88.28
word model	92.65	93.54	91.77	92.64
context model	**93.32**	93.62	93.03	93.32

Table 3: Evaluation on Hi.

In Table 3, again a similar pattern can be seen, i.e. a significant improvement (4.37%) from baseline to word model. Using the context model, accuracy increases by 0.67%. In both the Tables, we see that precision has been much higher than recall.

8 Analysis & Discussion

The confusion matrices of the language tagging models are shown in Table 4 and Table 5 for Bn and Hi respectively. Predicted class is denoted by Italics, while Roman shows the True classes.

Confusion Matrices					
1	Bn	En	2	Bn	En
Bn	16502	1465	*Bn*	16652	1315
En	991	15521	*En*	1000	15512

Table 4: Confusion matrices for Bn.

From Table 4 (1 - word model, 2 - context model), we can see that the correctly predicted En tokens has not varied much, but in case of Bn, the change is quite substantial, and the accuracy improvement from word to context model is contributed by this. Upon analyzing the tokens which were correctly classified by context model, but misclassified by word model, we see that most of them are rarely used Bn words, e.g. *shaaotali* (tribal), *lutpat* (looted), *golap* (rose), etc, or words with close phonetic similarity with an En word(s) or with long substrings which belong to the En vocabulary, e.g. *chata* (umbrella), *botol* (bottle), *gramin* (rural), etc. For some instances, we do see that ambiguous words have been correctly tagged by the context model unlike the word model, where the same language tag is given.

E.g 1. Amar\bn shob\bn rokom\bn er\bn e\bn fruit\en like\en aam\bn, jam\bn, kathal\bn bhalo\bn lage\bn. (Trans. I like all kinds of fruits like aam, jam, kathal.)

E.g 2. Sath\bn shokale\bn eto\bn jam\en eriye\bn office\en jawa\bn is\en a\en big\en headache\en amar\bn boyeshe\bn. (Trans. Early morning commuting through traffic for office is a big headache at my age.)

In the first example, the word ”jam” is a Bengali word meaning rose apple (a type of tropical fruit), while in the second example, the word ”jam” is an English word referring to traffic jam i.e. traffic congestion. Thus, we can see that despite having same spellings, the word has been classified to different languages, and that too correctly. This case was observed in 47 instances, while for 1 instance, it tagged the ambiguous word incorrectly. Thus we see that when carefully trained on standard well annotated data, the positive impact is much higher than negative.

In Table 5 (3 - word model, 4 - context model) we can see substantial improvement in prediction of En tokens as well by the context model, though primary reason for accuracy improvement is due to better prediction of Hi tokens.

Confusion Matrices					
3	Hi	En	4	Hi	En
Hi	14788	1326	*Hi*	14992	1122
En	1034	14968	*En*	1021	14981

Table 5: Confusion matrices for Hi.

Here again, on analyzing the correct predictions by the context model but misclassified by the word model, we see a similar pattern of rarely used Hi words, e.g. *pasina* (sweat), *gubare* (balloon), or Hi words which have phonetic similarities with En words, e.g. *tabla* (a musical instrument), *jangal* (jungle), *pajama* (pyjama), etc. In the last two cases, we can see that the words are actually borrowed words. Some ambiguous words were correctly tagged here as well.

E.g 3. First\en let\en me\en check\en fir\hi age\hi tu\hi deklena\hi. (Trans. First let me check then later you takeover.)

E.g 4. Anjan\hi woman\en se\hi age\en puchna\hi is\en wrong\en. (Trans. Asking age from an unknown woman is wrong.)

In the first example, ”age” is a Hindi word meaning ahead, while in the next instance, ”age” is an English word meaning time that a person has lived. Here, correct prediction for ambiguous words was seen in 39 instances while there was no wrong predictions.

9 Conclusion & Future Work

In this article, we have presented a novel architecture for language tagging of code-mixed data which captures context information. Our system achieved an accuracy of 93.28% on Bn data and 93.32% on Hi data. The multichannel neural network alone got quite impressive scores of 92.87% and 92.65% on Bn and Hi data respectively. In future, we would like to incorporate borrowed (Hoffer (2002), Haspelmath (2009)) tag and collect even more code-mixed data for building better models. We would also like to experiment with variants of the architecture shown in Fig 1 on other NLP tasks like text classification, named entity recognition, etc.

References

François Chollet et al. 2015. Keras. https://keras.io.

Monojit Choudhury, Kalika Bali, Sunayana Sitaram, and Ashutosh Baheti. 2017. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 65–74, Kolkata, India. NLP Association of India.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text.

Martin Haspelmath. 2009. Lexical borrowing: Concepts and issues. *Loanwords in the world's languages: A comparative handbook*, pages 35–54.

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

Bates L Hoffer. 2002. Language borrowing and language diffusion: An overview. *Intercultural communication studies*, 11(4):1–37.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Harsh Jhamtani, Suleep Kumar Bhogi, and Vaskar Raychoudhury. 2014. Word-level language identification in bi-lingual code-switched texts. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1110–1119.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Soumil Mandal, Sourya Dipta Das, and Dipankar Das. 2018a. Language identification of bengali-english code-mixed data using character & phonetic based lstm models. *arXiv preprint arXiv:1803.03859*.

Soumil Mandal, Sainik Kumar Mahata, and Dipankar Das. 2018b. Preparing bengali-english code-mixed corpus for sentiment analysis of indian languages. *arXiv preprint arXiv:1803.04000*.

Dong Nguyen and A Seza Doğruöz. 2013. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 857–862.

Braja Gopal Patra, Dipankar Das, and Amitava Das. 2018. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task@ icon-2017. *arXiv preprint arXiv:1803.06745*.

Mario Piergallini, Rouzbeh Shirvani, Gauri S Gautam, and Mohamed Chouikha. 2016. Word-level language identification and predicting codeswitching points in swahili-english language data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 21–29.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1971–1982.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. 2015. Going deeper with convolutions. Cvpr.

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 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.

Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. *arXiv preprint arXiv:1806.04470*.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.