# IIT (BHU) System for Indo-Aryan Language Identification (ILI) at VarDial 2018

**Divyanshu Gupta, Gourav Dhakad, Jayprakash Gupta and Anil Kumar Singh**
Computer Science and Engineering
Indian Institute of Technology (Banaras Hindu University)
Varanasi-221005, India
{divyanshu.gupta.cse15, gourav.dhakad.cse15, jayprakash.gupta.cse15, aksingh.cse}@iitbhu.ac.in

## Abstract

Text Language Identification is a Natural Language Processing (NLP) task of identifying and recognizing a given language out of many different languages from a piece of text. In the present scenario, this task has become the basis and beginning step of various other NLP tasks, for example, Machine Translation, improving search relevance for a multilingual query, processing code-switched data etc. The biggest limitation of many Language Identification systems is not being able to differentiate between closely related languages. This paper describes our submission to the ILI 2018 shared-task, which includes the identification of 5 closely related Indo-Aryan languages. We used a word-level LSTM (Long Short-Term Memory) model, a specific type of Recurrent Neural Network model, for this task. Given a sentence, our model embeds each word of the sentence and convert into its trainable word embedding, feeds them into our LSTM network and finally predict the language. We obtained an F1 macro score of 0.836, ranking 5th in the task.

## 1 Introduction

In the present scenario, Language Identification (LID) has become an important problem in the field of Natural Language Processing due to its wide range of applications. The Language Identification task has become an important step of various other NLP tasks, for example, Machine Translation, improving search relevance for a multilingual query, named entity recognition in code-switched data etc. The difficulty for Language Identification systems at present is that it is hard for them to differentiate between closely related languages. In this paper, we try out a language identification system that basically focuses on language identification of closely related Indian (Indo-Aryan) languages, i.e., Hindi, Awadhi, Magahi, Bhojpuri, and Braj. Language Identification is of special significance for multilingual countries like India.

The ILI shared task (Zampieri et al., 2018) focuses on identification of 5 closely related languages. The shared task also included an open track that allows additional resources, but we have only participated in the closed track that is, we performed closed training on the ILI dataset provided (Kumar et al., 2018).

The motivation behind our work is to find out how to build a system which can distinguish between closely related language over a domain. In the Indian context, it is a very important problem, since India has a large linguistic diversity, such that many of the languages/dialects/varieties are spoken by a large number of speakers. Many of these are closely related. There are also a large number of text documents which consist of a combination of two or more languages (due to code-switching or code-mixing). So, a system must be developed to serve the purpose. Even though language identification was one of the first NLP problems for which statistical techniques were applied, there is still a lot of scope for improvement in the performance of language identification systems for closely related languages. The shared task provided us a good opportunity to participate and find out the state-of-the-art for this problem. Due to the diversity in its languages, the given task could be an important step in bridging the digital divide between the Indian masses and the world.





*Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 185–190
Santa Fe, New Mexico, USA, August 20, 2018.

For the shared task, we initially used character level n-gram based statistical approaches with various distance measures like mutual information (Zamora et al., 2014), out of place measure (Singh and Goyal, 2014) etc. But the result was not satisfactory for closely related languages in the task, although these approaches worked very well for distant languages. We then moved to an LSTM (Hochreiter and Schmidhuber, 1997) based word-level model which has improved our results. Generally, the traditional statistical approaches, which do not take sequences into account, are useful for distant languages, but in the case of closely related languages, sequence modelling approaches such as LSTM give better results, since these approaches effectively utilize the internal dependencies existing between words of sentences.

## 2 Related Work

Quite often, even human beings are unable to correctly identify similar languages. The accuracy attained by previous works on language identification is over 95 percent for distant languages, but for closely related language, the numbers are still significantly lower. In previous years' reports, there is available a detailed description of the methods, the datasets and their limitations. Here we briefly summarize these.

Most of the reports show that sequence modelling approaches are better than other classical approaches, such as n-gram (with distance metric as out of place measure (Singh and Goyal, 2014)), Support Vector Machines (Noor and Aronowitz, 2006), graph-based n-gram method (Tromp and Pechenizkiy, 2011), Naive Bayes Classifier (Peng et al., 2004) etc. for identifying the language. In some cases, where the task is mainly focused on short sentences, linear SVM and maximum entropy models (Lau et al., 1995) are performing better. It is also noted that Bhojpuri and Magahi are much more similar and too difficult to distinguish through the n-gram approach.

For distant languages, using an n-gram based approach, it was noted that character level n-grams are more appropriate than word-level n-grams. This may be due to the limitation caused by out of vocabulary words present in the given text document. It was also found that using a combination of six, seven and eight character n-grams to train the model gave better accuracy.

## 3 Methodology

We formulate the task as a multi-class classification problem where each language is a distinct class. So, for a given sentence, our task is to find the appropriate language class for that sentence. The prediction of language class is carried out with the help of Long Short-Term Memory (LSTM) network architecture (Gonzalez-Dominguez et al., 2014), which is a special kind of Recurrent Neural Networks (RNNs). We particularly, are using Bidirectional LSTMs, a variant of LSTMs, instead of unidirectional LSTMs, since they can see the past and future context of the words present in the sentences and are much better suited for our task. The overall description of our system is given in the following section.

**Input**
The input of the model is a sentence of words of an unknown language to be identified.

**Output**
The output will be a language class (the language name) corresponding to the given sentence. We will get a probability vector of $shape(1, 5)$ that we pass through an $argmax$ layer to extract the index of the most likely language label.

**Steps Involved**

1. The first step is to convert an input sentence into a word vector representation.

2. We train 50-dimensional GloVe (Pennington et al., 2014) word embedding during the training phase of LSTMs.

3. Finally, we feed GloVe representation of each word in a given sentence of unknown language label into the LSTM hidden layers and predict the most appropriate language label for the sentence.

**Mini-batching**
We are using Keras (Chollet and others, 2015) as the framework for implementing LSTM for our task. In our task, we will train LSTMs using mini-batches (Ioffe and Szegedy, 2015). The most basic requirement of a Deep Learning framework is that all sequences in the same mini-batch have the same length. If we had a five words sentence and, say, a seven words sentence, then the computations and calculations needed for them are quite different (one will take five steps of LSTMs, and other takes seven steps), so it is not possible for both to be processed in the same way.

The common solution for this problem is to use padding. We set a maximum limit on the sequence length and pad all the sentences to the same length. The padding could be done in two ways:
1. Forward padding
2. Backward padding

Forward padding is chosen over backward padding since forward padding does not face many problems of vanishing gradient problem as compared to Backward Padding. For example, if we set padding as 40 words, the sentence longer than 40 words will be truncated. In our task, we set the padding length to 40 words for each sentence.

**Embedding Layer**
In Keras, the embedding matrix is displayed as an embedding layer and maps word indices to their word embedding vectors. The main aim of this layer is to convert a matrix of indices of words of input sentences into their GloVe (Pennington et al., 2014) word embedding.

**Building Model Architecture and Hyper-parameter Tuning**
After creating embedding matrix, we need to decide on the architecture of our LSTM model. We choose some number of hidden layers, Dropout value (Srivastava et al., 2014), etc. to create the LSTM model. These details are given in Section 3.1.

After creating the architecture, we compile the model with loss function as **cross-entropy loss** (De Boer et al., 2005), optimizer as **Adam optimizer** and metrics as **accuracy**. The number of epochs and batch size are also tuned for getting improved results.

### 3.1 Training Details

We train the entire LSTM model jointly, including the embedding layers. We used Adam optimization (Kingma and Ba, 2014) with the original parameters that are the default, and the loss function used is cross-entropy. Our implementation is done with the help of Keras framework. The model is run with shuffled mini-batches of sizes 128 and the epochs were ended when the loss in the developmental set stopped improving.

We did hyperparameter tuning by trying out various values, and the best results were observed with following values of hyperparameters: Mini-batch size of 128, Number of hidden layers in LSTM cell to 128, Number of epochs used were 36 and dropout value is 0.45.

## 4 Results

The result of our system on various runs are described in Table 1. The scores of various metrics in the best run is also shown in the Table 2.

| System | F1 (macro) |
|---|---|
| Random Baseline | 0.2024 |
| **01** | **0.8360** |
| 02 | 0.7444 |
| 03 | 0.8269 |

Table 1: Results for the ILI task. Best results out of our runs are in bold.

| Metric | Score |
|---|---|
| Accuracy | 0.8478 |
| F1-micro | 0.8478 |
| F1-macro | 0.8360 |
| F1-weighted | 0.8442 |

Table 2: Table showing result of various metric in best run

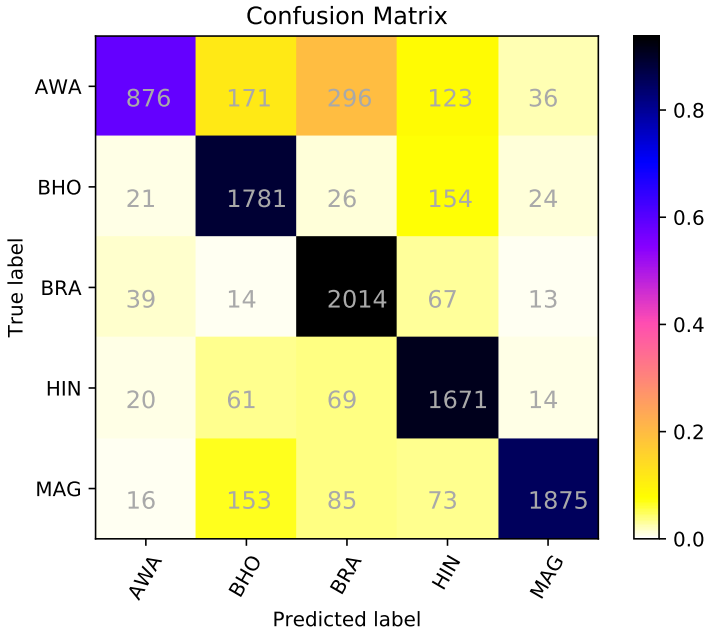


Figure 1: ILI task: Confusion Matrix of our best run

The confusion matrix for our best run is shown in Figure 1.

The results of different approaches used for implementing the task reveal that the best result is observed when we used sequence modelling approaches, i.e., by using the LSTM architecture, with which we achieved F1-macro score of 0.8360. In our second run, we used statistical n-gram approach with mutual information (Zamora et al., 2014) as the distance measure, in which we achieved F1-macro score of 0.7444. The reason for above result, could be the inability of n-gram approach to distinguish closely related languages, particularly, Bhojpuri and Magahi. The above method was unable to model the internal dependencies between the words of the sentence. So, we used sequence modelling approaches using LSTMs. The reason is that LSTMs can remember long-range dependencies among the words of the sentences.

## 5 Conclusion

The major conclusions which we can draw from our work in this shared task are:

1. In n-gram model, the result were improved when we increased the n-value. The combination of 6, 7 and 8-gram model yield the best result.

2. While tuning our n-gram model, we concluded that results of character-level n-gram model were much better than that of word-level n-gram model.

3. N-gram approach was not able to distinguish effectively between Bhojpuri and Magahi due to the high degree of similarity between them. So, the sequence modelling approach was implemented to

remember internal dependencies existing between words in the sentences.

4. The hyperparameter tuning was performed, and the best results were observed for LSTM model when mini-batch size was 128, the number of Hidden layers in LSTM was 128, with the dropout of 0.45 and number of epochs used were 36.

5. For smoothing the gradient descent, we used Adam optimization algorithm and the loss function used was the cross-entropy loss.

## 6 Future work

In future, we would like to apply our method to other natural language processing tasks such as multilingual search query, dialect identification, etc.

## References


François Chollet et al. 2015. Keras.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67.

Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Haşim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno. 2014. Automatic language identification using long short-term memory recurrent neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association*.

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

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

Ritesh Kumar, Bornini Lahiri, Deepak Alok, Atul Kr. Ojha, Mayank Jain, Abdul Basit, and Yogesh Dawar. 2018. Automatic Identification of Closely-related Indian Languages: Resources and Experiments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*.

Raymond Lau, Ronald Rosenfeld, and Salim Roukos. 1995. Building scalable n-gram language models using maximum likelihood maximum entropy n-gram models, November 14. US Patent 5,467,425.

Elad Noor and Hagai Aronowitz. 2006. Efficient language identification using anchor models and support vector machines. In *Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The*, pages 1–6. IEEE.

Fuchun Peng, Dale Schuurmans, and Shaojun Wang. 2004. Augmenting naive bayes classifiers with statistical language models. *Information Retrieval*, 7(3-4):317–345.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Anil Kumar Singh and Pratya Goyal. 2014. A language identification method applied to twitter data. In *TweetLID@ SEPLN*, pages 26–29. Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Erik Tromp and Mykola Pechenizkiy. 2011. Graph-based n-gram language identification on short texts. In *Proc. 20th Machine Learning conference of Belgium and The Netherlands*, pages 27–34.

Juglar Díaz Zamora, Adrian Fonseca Bruzón, and Reynier Ortega Bueno. 2014. Tweets language identification using feature weighting. In *TweetLID@ SEPLN*, pages 30–34. Citeseer.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Ahmed Ali, Suwon Shuon, James Glass, Yves Scherrer, Tanja Samardžić, Nikola Ljubešić, Jörg Tiedemann, Chris van der Lee, Stefan Grondelaers, Nelleke Oostdijk, Antal van den Bosch, Ritesh Kumar, Bornini Lahiri, and Mayank Jain. 2018. Language Identification and Morphosyntactic Tagging: The Second VarDial Evaluation Campaign. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, USA.