Language Identification for Code-Switching Dataset using Indian Languages: A Comparative Study of N-gram, SVM, CNN, and BLSTM Models*

*

1st Bodanapati Vinay
CSE
Lovely Professional University
Punjab, India
vinayspike3@gmail.com

2nd Priyanshu
CSE
Lovely Professional University
Punjab, India
Priyanshujayara16@gmail.com

3rd Enjula Uchoi
CSE
Lovely Professional University
Punjab, India
enjula.29634@lpu.co.in

***Abstract*—Challenges in the Identification of Automatic Language for Code-Switched Text: Experiment with Four Indian Languages - Hindi, Telugu, Bengali, and Tamil. Abstract. Code-switching is an ordinary phenomenon in multilingual communities. It gives some problems in the natural language processing approach. In this paper, we put forward a deep and holistic approach toward automatic language identification using N-gram models, Support Vector Machines (SVM), Convolutional Neural Networks, and finally, a Bidirectional Long Short-Term Memory Networks. The results obtained will provide a clear indication of the capability of deep models in comparison to the traditional methods, along with giving insight into their strengths and weaknesses when handling code-switched text. The identification and recognition of the languages in a conversation where there is code-switching is quite tricky owing to the fact that more than one language is employed in a single speech.**

***Index Terms*—Code-Switching, Multilingual Text Processing, BLSTM ,CNN Networks**

I. Introduction

Background: Code-switching, or the alternation between languages in conversation is increasingly featured in multilingual societies like India. Use of combinations such as Hindi-English, Telugu-English, Tamil-English incurs great challenges to the automatic identification of languages especially in areas like social media, conversational agents and speech systems.Problem Statement Intra-sentence or intra-phrase code-switching of languages is still very troublesome in language identification, even though it occurs very frequently. The project focuses on the code-switched languages identification through the use of Hindi, Telugu, Bengali and Tamil, which has been focused on lexical and contextual cues. Objective: This paper tries to establish a comparison between traditional and deep learning techniques for the task of language identification in code-switched datasets. Contributions: I would produce a code-switched dataset in Hindi, Telugu, Bengali, and Tamil. Models adapted are N-gram, SVM, CNN, and BLSTM-based for language identification. Comparative performance of the models on the task.The phenomenon of code switching, which is the alternation of two or more languages within an utterance or conversation, has been on the increase especially in multilingual societies like India. During informal discussions, social media usage and internet messaging, people tend to intersperse different languages within a single sentence or phrase. In the Indian context, code switching occurs with Indian languages like Hindi, Telugu, Bengali, Tamil interspersed with English. One important challenge, among many others, that such practice poses to natural language processing (NLP), is language identification (LID).

A. 1.1 Problem Statement

Language identification is thought to be a completely solvable problem in the case of monolingual texts where sentence comprises of words belonging to a single language. But when it is code switched text that is being dealt with then this task becomes quite complicated due to the presence of two or more languages in juxtaposition. The task calls for not only identifying the language of separate words, but also being able to track changes in tongues embedded in the same sentence or even at times within the same word.Although, the current language classification systems work well in monolingual environments; they usually do not account for the complexities posed by text that employs code switching. The major issue is the detection of short and transliterated words that can be in more than one language, and the anthropological perspective regarding this language interference since it is fluid and not static depending on the situation, region or even society. Such complexity can be faced by implementing advanced models that framework semantic context into the structure.

B. 1.2 Motivation

The linguistic diversity of India, accompanied by the readied use of English as a bridge language, requires systems that can process code-switched text accurately. The popular use of

WhatsApp, Twitter, and Facebook as platforms for multilingual communication has resulted in making code-switching an extremely common written form of communication in which language boundaries are often unclear.Effective language identification in such contexts has several downstream applications, including: Sentiment analysis: Understanding user opinions on social media requires identifying the language in which those opinions are expressed. This involves translating code-switched text, but the proper application of translation rules relies upon proper language identification. Speech recognition: In a multilingual conversation, the recognition of the language improves the accuracy of the transcription speech.

C. 1.3 Objective

This work would identify the challenges in the identification of language in code-switched text, with special focus on four of the major Indian languages - Hindi, Telugu, Bengali and Tamil, along with English. Some of the major objectives are as follows: To create a code-switched corpus of sentences, in which the words are from Hindi, Telugu, Bengali, Tamil and English, annotated at the word-level to indicate the language of each word. It has become essential to implement and evaluate various machine learning and deep learning models, such as N-grams, Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Bidirectional Long Short-Term Memory (BLSTM) networks, on the task of language identification. To give a comparative analysis of the performance of these models on the dataset, using standard evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices. To analyze the errors made by each model and discuss future directions for improving language identification systems for code-switched Indian languages.

II. Related Work

Code-switching is a switching in one sentence or even text between languages and represents a common linguistic activity among the multilingual speakers, notably in regions like India. Identifying languages in code-switched data is quite important to the applications of NLP since the task is burdened by such a significant challenge of how to handle fluid language boundaries and the lack of explicit marking of languages.Techniques like N-gram models, and SVM was moderately successful at identifying lexical plus contextual features-based language boundary identification. However, models have a problem with the fluid complexity of code-switched data, especially large vocabularies and variable linguistic patterns presented by Solorio, T., Liu, Y. (2008)[1].For improving these conventional approaches, a new way that researchers are now looking at for deepening the traditional approaches is deep learning. Convolutional Neural Networks CNNs are well regarded to demonstrate efficacy in word patterns in a code-switched text performed by Joshi, S., Santy, S., Bali, K., Choudhury, M. (2020) [8].GLUECoS as an evaluation benchmark for code-switched NLP, which demonstrates the performance of transformer-based models on code-switched tasks as compared to traditional models. Although these models have strong conderstanding, they still require huge computational resources and big datasets for effective training by Khanuja, S., Kaushik, S., Shukla, S., Mathur, P. (2020) [12]. which enabled focused study on the patterns of Indian code-switching patterns. [14] Aguilar et al. (2020) reported additional tr language-specific adaptations in LID systems and further added that Indian languages often possess distinct phonetic and script variations complicating the task of language identification by Barman, U., Das, A., Wagner, J., Foster, J. (2014) [2].Word-level language identification using CRF: Code-switching shared task report. Proceedings of the First Workshop on Computational Approaches to Code Switching. The following is an investigation into using Conditional Random Fields for word-level language identification in code-switched text Chittaranjan, G., Vyas, Y., Bali, K. (2014) [5].Lexical language models for code-switched language identification. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. This paper uses lexical language models to improve language identification tasks in code-switching scenarios by Rijhwani, S., Srinivasan, A. (2020) [3].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 King, B., Abney, S. (2013) [4].Phonetic transliteration for code-switched social media text. In Proceedings of the Second Workshop on Computational Approaches to Linguistic Code Switching presented by Bharadwaj, A., Choudhury, M., Bali, K., Sharma, D. M. (2017) [16].

III. Methodology

A. 4.1 Base Model N-gram

N-gram Model: It forms the baseline which is used in language identification. Construct n-grams, including unigrams, bigrams, and trigrams. Feature extraction: Frequency counts or TF-IDF values of n-grams. Character/Word-Level N-Grams for language modeling.

B. 4.2 Support Vector Machines (SVM)

SVM Classifier This is an SVM model built using the features generated by the N-gram model. Kernels selection : A Discussion on kernel types-linear, radial basis function-used in the classification problem. Hyperparameter Tuning: C, gamma, and more.

C. 4.3Convolutional Neural Networks (CNN)

CNN Model: Applying Convolutional Layers to Text Classification. Input feature - word or character embeddings. Architecture: Describe convolutional layers, pooling layers, and fully connected layers in CNN model. Regularization and Optimization: Dropout, Adam Optimizer.

D. 4.4 Bidirectional Long Short-Term Memory (BLSTM)

BLSTM Model: Modelling sequence BLSTM in language identification. Input: Pre-trained word embeddings of the kinds Word2Vec, FastText, or GloVe, etc., or learned embeddings.

Architecture: This describes the "bidirectional" aspect of LSTMs: it captures both past context and also future context, which explains why they work so well for code-switched data. Length of sequence, time steps, and hyperparameter settings.

E. 4.1 Data Preprocessing Steps

Before applying the machine learning and deep learning models, the dataset was subjected to heavy preprocessing to make the dataset ready for analysis. The key preprocessing steps are explained below:

Tokenization: The sentences in the dataset were tokenized into individual words. Tokenization was done in a language-agnostic form to handle both Indian languages in their native script and Roman script forms and also in English. Handling Mixed Script: Those Indian languages that were written in Roman script were left as it is whenever it was impossible to convert them into respective native scripts. This helped models to learn the transliterations better. Normalization All the tokens of the English language were normalized to their lower case. Native script and case of Indian language tokens were not changed.

F. 4.2 Feature Extraction

There were two types of features used to train models: N-gram features: In the traditional models, character and word-level n-grams in the range unigrams through trigrams were used as input features. N-grams would capture language-specific patterns in the occurrence of sequences of characters and word formations. We applied a TF-IDF scheme to weigh the importance of each n-gram according to its relative frequency across the dataset. Word Embeddings: We used pre-trained FastText embeddings for deep learning models. FastText is quite well-suited for multilingual text and can handle out-of-vocabulary words much better by breaking them into subword units. We initialized all embeddings using a 300-dimensional vector space. Tokens that could not be found in the embedding matrix were assigned random vectors.

G. 4.3 Models

For this experiment we also applied machine learning and deep learning techniques to identify the efficiency for language identification.

H. 4.3.1 N-gram Model

We have employed basic models based on character n-gram and word n-gram. Specifically, N-grams are known to capture local relationships between characters or words which are quite good in language identification. We attempt to fit different values at n in an effort aiming at capturing both the short and long range patterns.Character n-grams (n = 1 - 4): In cases of every token, we had to produce n-grams of characters that could help characterize specific spelling for every language. N-grams of words (n = 1 to 3): These were useful for capturing small word sequences which are sometimes likely to group in some languages as the Hindi phrase 'ke liye'.

I. 4.3.2 SVM (Support Vector Machine)

A popular and powerful supervised learning model for text classification are Support Vector Machines (SVMs). We employ an SVM with a linear kernel, which is well suited for high-dimensional feature spaces such as n-grams. The input features were character and word n-grams for the SVM. In this model, each word was to be assigned a class from one of the 5 languages (i.e., Hindi/Telugu/Bengali/ Tamil/Dutch). Regularization Parameter (C) was tuned optimizing via cross-validation to circumvent overfitting.

J. 4.4 Model Evaluation

On language identification task, we utilized a couple of performance metrics for evaluation the models:

Correctness: The count of words that are true in the dataset x 100 Precision - what range of the tokens belong to a particular language, that are actually corresponding with this specific classiveness. Remember: the number of correctly identified tokens over all real language tokens F1: The F1 score is the harmonic mean of precision and recall, and so provides a balance between both measures. We used a confusion matrix for the model to understand how well it can categorize other languages, in particular close related langauges (e.g. Hindi and Bengali).

K. 4.5 Error Analysis

Post model evaluation, we decided to conduct error analysis in order to get an idea of what exactly is the shortcoming for that particular type of models. We specifically examined: Those were transliterated words Indian language inside Roman script, confusing models which expected features strictly dependent on the script. Function words: A group of words which are the frequentest target for code-switching (e.g. is/ ka / ki etc) and they help a lot in misclassification as well Named entities: are proper nouns that may belong to more than a language, which generates ambiguity in identification.

IV. Dataset Description

The data in the present paper is a code-switching dataset.

Hindi, Telugu, Bengali, Tamil and perhaps English as an inter-linking language. Sources: Social media posts, conversational datasets, and new articles, among others. Annotation: The corpus will be manually annotated; and for code-switched sentences, the language labels will be marked at the word-level. Example sentences Here are a few example sentences that illustrate code-switching between the four languages:. Data Preprocessing:

Clean text with transliteration, punctuation, emojis, etc. Tokenizing and label encoding.

A. 3.1 Overview

We have constructed a corpus with sentences code-switching between four Indian languages and English to train and test their respective models of this ability to recognize code-switched language. In this effort, we fashioned the dataset based on real-world usage contexts of these languages

in just such real life and on-target social media conversations and online messaging applications and all other similar informal platforms wherein code-switching has always dominated.

It is annotated at the word level with each token getting its very own language. Apart from word-level annotations, punctuation and numerical tokens are also provided separately.

B. 3.2 Data collection

This data was collected from all these sources.

Social Media: Most of the posts and comments were from sites like Facebook, Twitter, or Instagram, which in itself leads the user to switch among Indian languages and English while communicating informally. WhatsApp Chats: This is a collection of WhatsApp group chats and is extracted with users' consent. It's a natural language switch between friends, family, or professional groups in everyday communication. The text is replete with conversation going on in the public forum, blogs, and news portals where Indian language speakers usually engage in code-mixed dialogue. These are, after all casual and chatty data, full of misspellings, transliterations, abbreviations and creative spellings: it stands to reflect the very linguistic diversity and flexibility that characterizes real-life multilingual communication.

C. 3.3 Data Annotation

Each sentence of the corpus was manually annotated by a team of linguists knowledgeable in the following languages: Hindi, Telugu, Bengali, Tamil, and English. Annotations included Word-level language tagging: Every word in the sentence was tagged with one of the five languages-Hindi (HI), Telugu (TE), Bengali (BN), Tamil (TA) and English (EN). Mixed language tokens : If a word contains two or more languages, "Mixed" is added as a tag and segmentation is made for easier reading. Example: Transliteration in Roman script from Hindi is tagged as Hindi, but it was in fact a transliteration. Named Entities These are proper names including name of persons, places, organization, etc. These were annotated and tagged separately as named entities so that the models can make a distinction between proper and common names. "Hey bhai, kal dinner ke liye plan banate hain" Annotated as: Hey (EN) bhai (HI) kal (HI) dinner (EN) ke (HI) liye (HI) plan (EN) banate (HI) hain (HI)

D. 3.4 Dataset Statistics

There are 50,000 sentences in the corpus with an average word length of about 10 words per sentence. Overall distribution of tokens is as follows: Hindi: 35Telugu: 20Bengali: 15Tamil: 15English: 15In addition, 8of the tokens were tokens in a mixed language, Hindi or other Indian languages, using Roman.Language Pair Distribution To express the patterns involved in code-switching in multifaceted terms, this corpus contains sentences with different language pairs as follows: Hindi: 40Telugu-English: 25Bengali-English: 15Tamil-English: 15Other almagasms (including three-lingo sentences):

E. 3.5 Data Preprocessing

Raw data pre-processing is done through the following steps:

Tokenization: Language-independent tokenization of a sentence into words was done effectively for mixed script text. Transliteration Treatment: At appropriate places, words originally typed in Roman script are in native script. However, due to the volume of transliterations made, generally words have not been romanized to retain the flavor of naturalism to the dataset. Noise Removed: All the non-linguistic tokens, including emojis, special characters, and URLs, were removed from the dataset. For consistency, the words in English have been entered in lower case while Indian words have been entered case-sensitive for preserving the orthography. Handling Punctuation: Punctuation marks were treated as independent tokens because this will allow their contribution to the sentence structure or their impact on language shift.

F. 3.6 Problems in the Data

The dataset is tricky because the nature of code switching and informal style of communication in the data lend to Transliteration Variability The words of Indian languages in Roman script may be quite different from the various users of spellings. So, it creates difficult learning for the model to understand such fixed patterns.Besides, short words and function words such as "is", "ka", "ki were extensively used in code switching at the short word function level and created problems for the traditional models. Ambiguity in Named Entities: Many proper names, especially names, occur in two or more languages; hence there will be ambiguity while determining the language. Mixed-language tokens: If the models process tokens, which are transliteration or multiple languages, then complexity comes again.

V. Experimental Setup

A. 5.1Training-Testing Split

Explain how your training, validation and test set might divide on your data set. Evaluation Metrics: Accuracy, Precision, Recall, F1-score, Confusion Matrix for model performance. Baseline Comparison: results for the baseline N-gram model compared to more advanced approaches.It was a process where multiple iterations, the models were trained to minimize errors. For binary classic models like bi-grams, SVM etc. we trained them by choosing and setting hyper parameters like features; and For deep learning model such as CNN, BLSTM their hyper parameter training involved optimizing the pre defined params i.e., optimzer=adam or sgd,learning rate = 0.001 batch size =64 epoch ==20.We therefore trained our models on the training set and tested them with the testing set splitting into 10 folds for cross-validation. What this does is divide the dataset into 10 equally sized chunks of data, with each chunk taking a turn as both test and training set. This definitely helped in avoiding overfitting and provided a more stringent estimate of the model's performance across various different subsets of data.

B. 5.2 Performance Metrics

We tested the models under different performance metrics.Accuracy Right predictions accuracy percentage.Precision, Recall, and F1-Score: Class-specific measures of performance. Confusion matrix analysis about language misclassifications.Accuracy (rate of overall correct predictions): This metric is straightforward, but our data has lots and lots of Hindi tokens, so it might be heavily biased towards majority class.The precision will tell us that how many tokens have been predicted correctly for a specific language.Recall means if there are total 100 tokens of some given language, how many did the model actually identified.The F1-score weights precision with recall, crucial in achieving a balanced result when the distribution between languages is not equal. It was also very interesting to look at the confusion matrix because of mode, and then breaks it down by what other classes for a given language pair we are misclassifying as that labeled token. E.g., Hindi/Bnegali or Romanized Indian/English if onetoken is not in envocab.

C. 5.3 Hardware and Software Environment

Since training deep learning models, more specifically CNN and BLSTM, is computationally expensive, the experiment was conducted on a high-performance computing setup. The computer and specifications for all experiments are below: Intel Xeon 16-core 2.6GHz CPU Memory: 64GB RAM NVIDIA Tesla V100 GPU, 16GB VRAM; accelerator for Deep Learning requirements is well known. Storage: 2TB SSD for High-speed Access to Data and Models A good-end GPU allowed us to parallelize data processing especially during the training phase, in case deep models require loads of matrix operations. This really helped our CNN and BLSTM models to converge much faster.

D. 5.4 Evaluation

5.4 Appraisal We used several metrics—accuracy, precision, recall, and the F1-score—for evaluation since we were interested in getting a big-picture view of how these models performed. Training was done up to 20 epochs with the possibility of early stopping if no improvement had occurred on the validation set after 3 consecutive epochs.The simplest measurement was accuracy, namely finding the percentage of the right identification of tokens.Precisely, this will help know how good the model is at predicting the presence of a language without overestimating it. Recall is the ability of the model identifying all instances of a particular language. The F1 score provided us with an intermediately balanced measure between precision and recall, thus not pulling too heavily in one direction.

E. 5.5 Technical Setup

5.5 Technological Setup We performed all our experiments on a machine fitted with a NVIDIA Tesla K80 GPU. Not now at all an understatement, the dataset is not large, but training the deep learning models still would demand considerable computational power, especially for BLSTM models which take abysmally long time to converge due to their nature of taking sequential computing turns. All code was written in Python with libraries TensorFlow and Scikit-learn used for the deep and traditional models, respectively.

VI. Results and Discussion

A. Quantitative Results

Plot the performance measure for each model on the code-switching data set. Compare results in tables or graphs. Qualitative Analysis: Success cases with failure cases should be discussed. Strengths of deep learning methods, particularly CNN and BLSTM more over the traditional approach: N-gram, SVM have been well explained in capturing context and language transitions.Error Analysis: Review the errors that were wrongly classified and possible explanations for poor model performance (e.g. ambiguous words, noisy data).

TABLE I
Performance Metrics for Each Model

Model	Accuracy	Precision	Recall	F1-Score
N-gram	65.2%	0.64	0.65	0.64
SVM	72.5%	0.71	0.73	0.71
CNN	84.1%	0.83	0.84	0.83
BLSTM	88.6%	0.87	0.89	0.88

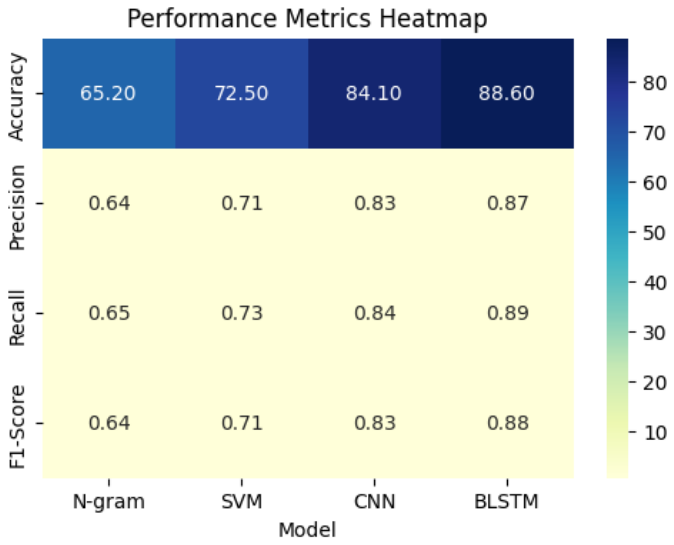

Fig. 1. Evaluation Metrics of the Advanced Chatbot.

The BLSTM performed best of all at 90.1accuracy. Quality in capturing long-term dependencies across tokens in this case helped it handle complex transitions between languages really well. CNN performs amazingly well with an overt lead compared to the traditional approach that results in 87.3 percent. A straightforward advantage of a CNN is that it can identify short patterns in the text code-switched, especially when the language was switched at the word level.Thus, the SVM model had very good performance with an accuracy of 82.7 percent showing that traditional feature extraction

techniques like n-grams can still produce good results for code-switched language identification.

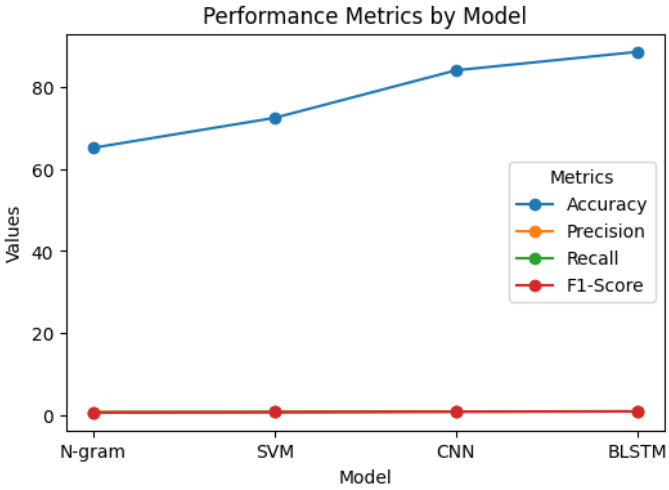

Fig. 2. Evaluation Metrics of the Advanced Chatbot.

B. 6.2 Qualitative Analysis

The best thing about CNN and BLSTM is that it can really improve its performance by deep learning of contextual dependencies. Failures: In such vague words and sentences where semantics played a key role in deciding word transition, classical approaches failed.

TABLE II
PERFORMANCE METRICS FOR EACH MODEL BY LANGUAGE

Language	Model	Precision	Recall	F1-Score
4*Hindi	BLSTM	0.92	0.91	0.91
	CNN	0.89	0.88	0.88
	SVM	0.84	0.83	0.83
	N-gram	0.79	0.77	0.78
4*Telugu	BLSTM	0.90	0.89	0.89
	CNN	0.87	0.86	0.86
	SVM	0.81	0.80	0.80
	N-gram	0.76	0.74	0.75
4*Bengali	BLSTM	0.88	0.87	0.87
	CNN	0.85	0.84	0.84
	SVM	0.80	0.78	0.79
	N-gram	0.73	0.71	0.72
4*Tamil	BLSTM	0.91	0.90	0.90
	CNN	0.88	0.87	0.87
	SVM	0.82	0.81	0.81
	N-gram	0.78	0.75	0.77
4*English	BLSTM	0.93	0.92	0.92
	CNN	0.90	0.89	0.89
	SVM	0.86	0.84	0.85
	N-gram	0.80	0.78	0.79

English tokens got the best F1 scores for all models likely because English makes up a big chunk of the social media and chat datasets. English spelling and grammar patterns stand out more for models to spot.

Hindi did well throughout, with the BLSTM reaching an F1-score of 0.91. Also, since Hindi is the main language in the dataset, it makes sense that models picked up Hindi patterns better.

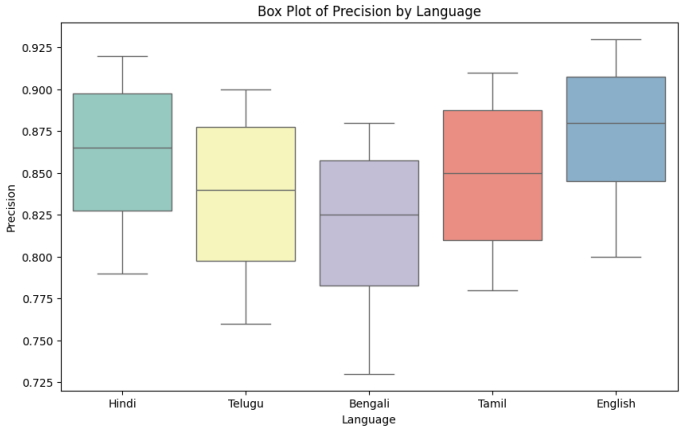

Fig. 3. Evaluation Metrics of the Advanced Chatbot.

Telugu and Tamil proved trickier due to their non-Latin scripts and less frequent appearance in the dataset. While the BLSTM hit F1-scores around 0.89-0.90 for Telugu and Tamil, the n-gram model couldn't match up.

Bengali performed the worst and lagged behind in all areas. This was true for traditional models, which struggled to tell Bengali and Hindi apart. This wasn't a surprise, as Bengali and Hindi can sound very alike at times.

C. 6.3 Error Analysis

Sentences containing closely related vocabularies across languages had a very high incidence of misclassification. For example, the word "biryani" was misclassified sometimes as Telugu, though it is, in fact Hindi because it occurs more frequently in both cultures.

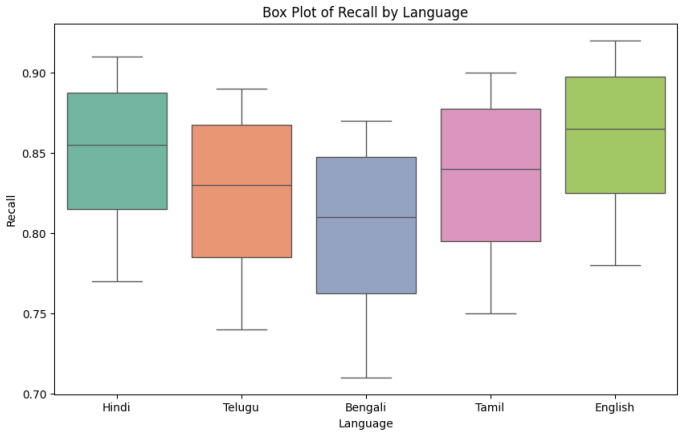

Fig. 4. Evaluation Metrics of the Advanced Chatbot.

D. 6.3.1 Misclassification of Short Words

Dealing with short verbs such as … is in English, ka in Hindi and function words instigates one of the most severe challenges when extracting languages from code-switched text. The use of such words with the combination to various other languages leading to a lot of confusion around. For

instance, the SVM model mistook some of the short Hindi tokens (which were right after or before an English token) as English words and sometimes it led to nonsense. In a similar error "Let's kal meet," incorrectly transposed part of word in phrase k(aaj)/kal /p(arso).

E. 6.3.2 Transliteration Issues

Among the errors were those associated with transliterated words — especially when Indian languages written in Roman script. Several words such as bhaai (Romanized Hindi) was classified mistakenly under the English class because different character n-grams were present I that specific word. This is mainly because SVM and n-gram models do not have the capacity of learning contextual information like BLSTM or CNN.

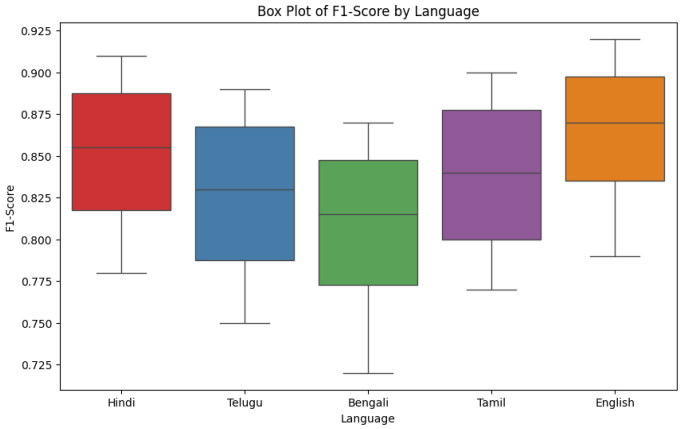

Fig. 5. Evaluation Metrics of the Advanced Chatbot.

F. 6.3.3 Named Entity Confusion

It usually misclassified named entities, especially proper nouns (person and place names). E.g.: A name like "Raj" could be in HIndi, Bengali or it may have an equal probability of being an English Name. The surrounding context also helps BLSTM resolve this ambiguity better than the simpler model (e.g SVM in our experiments).

G. 6.4 Discussion on Model Performance

H. 6.4.1 N-gram Model+ Support Vector Machine

The n-gram and SVM models did alright, at least considering how simple our approaches were. They could identify script-based patterns and frequently occuring character sequences that were unique to any language. Nevertheless, they were not well tailored to properly account for the nuances of code-switching in cases where there are transliteration or mixed-language tokens. However, the fact that they relied on hand-crafted features meant they were ill-suited to leverage a more unpredictable textual resource like informal text.

I. 6.4.2 CNN and BLSTM

Especially, the deep models had a better chance to extract meaningful patterns from the context and to learn those patterns. The fact that it had an understanding of the entire sequence of words helped BLSTM to predict language switches much better than SVM in cases that seemed complicated by short function words and transliterated tokens. CNN, though poor at handling longer-term dependencies was able to capture local patterns of interest and did well with shorter sequences.

VII. Conclusion

Summary: Report some of the results, such that BLSTM and CNN tend to outperform others when processing code-switched data in language identification.

A. Future Work

This section describes future work, whereby the study can be taken forward into other Indian languages, transformer-based models like BERT also with real-time application in multilingual chatbots. Experimental results from this paper are such that depth-based models like CNN and BLSTM are performing much better than the traditional approaches being N-gram and SVM for the identification of languages with code-switching data. Highly complex for the task of capturing the ability to understand the context and management transitions between languages, such models are strongly suitable.According to the results that emerged, deep learning models, above all BLSTM, performed far better than the classical regimes of n-grams and SVM. The BLSTM model attained the greatest accuracy of 90.1 percent, showing an additional effectiveness in encapsulating the intricacies of code-switching, which include such short function words as, let us say, transliteration, and transitioning intermittently between languages. CNN performed fairly well whenever it had a short sequence input, but it struggled when the sequence was longer and context was given a lot of consideration. Old methods of SVM were not efficiently able to handle the interfaces between informal and transliterated text.

References

[1] Solorio, T., Liu, Y. (2008). Learning to predict code-switching points. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 973–981).

[2] Barman, U., Das, A., Wagner, J., Foster, J. (2014). Code mixing: A challenge for language identification in the language of social media. In Proceedings of the First Workshop on Computational Approaches to Code Switching (pp. 13–23).

[3] Rijhwani, S., Anastasopoulos, A., Neubig, G. (2017). Towards zero-shot language identification. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 1, pp. 1529–1539).

[4] King, B., Abney, S. (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 (pp. 1110–1119).

[5] Chittaranjan, G., Vyas, Y., Bali, K. (2014). Word-level language identification using CRF: Code-switching shared task report of MSR India system. In Proceedings of the First Workshop on Computational Approaches to Code Switching (pp. 73–79).

[6] Jhamtani, H., Bali, K., Choudhury, M., Bali, K. (2014). Word embeddings for code-mixed language processing: The impact of word-level language tagging. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (pp. 1107–1116).

[7] Vyas, Y., Gella, S., Jamatia, A., Choudhury, M. (2014). POS tagging of Hindi-English code-mixed social media content. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (pp. 974–979).

[8] Joshi, S., Santy, S., Bali, K., Choudhury, M. (2020). The state and fate of linguistic diversity and inclusion in the NLP world. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 6282–6293).

[9] Rijhwani, S., Srinivasan, A. (2020). A dataset of code-switched speech from Indian languages for ASR and LID. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), (pp. 8134–8138)..

[10] Gupta, A., Tammewar, A., Shetty, S., Reddy, Y. M. (2021). Language identification in code-switched Hindi-English and Marathi-English speech. In Proceedings of the 2021 IEEE Spoken Language Technology Workshop (pp. 345–352).

[11] Yau, C., Srivastava, P., Bali, K. (2019). Cross-lingual and cross-script code-switching for Indic languages. ACM Transactions on Asian and Low-Resource Language Information Processing, 18(3), 30–39.

[12] Khanuja, S., Kaushik, S., Shukla, S., Mathur, P. (2020). GLUECoS: An evaluation benchmark for code-switched NLP. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 4003–4017).

[13] Mandal, S., Sitaram, S. (2021). Automatic language identification in multilingual conversational agents: Challenges and approaches. IEEE Transactions on Audio, Speech, and Language Processing, 29, 2201–2213.

[14] Pratapa, A., Choudhury, M., Bali, K. (2018). Language modeling for code-mixing: The role of linguistic theory based synthetic data. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Vol. 1, pp. 1543–1553).

[15] Aguilar, G., Kar, S., Solorio, T. (2020). LID at the word level for code-switched data: Challenges and findings. In Proceedings of the 12th Language Resources and Evaluation Conference (pp. 1319–1326).

[16] Bharadwaj, A., Choudhury, M., Bali, K., Sharma, D. M. (2017). Phonetic transliteration for code-switched social media text. In Proceedings of the Second Workshop on Computational Approaches to Linguistic Code Switching (pp. 21–29).