On-Device Major Indian Language Identification Classifiers to Run on Low Resource Devices

Yashwanth Y S
R.V College of Engineering
Computer Science and Engineering Department
Bengaluru, Karnataka, India

***Abstract*—Language Identification acts a first and necessary step in building intelligent Natural Language Processing (NLP) systems that handle code mixed data. There is a lot of work around this problem, but there is still scope for improvement, especially for local Indian languages. Also, earlier works mostly concentrates on just accuracy of the model and neglects the information like, whether they can be used on low resource devices like mobiles and wearable devices like smart watches with considerable latency. Here, this paper discusses about both binary classification and multiclass classification using character grams as the features. Considering total nine languages in this classification which includes, eight code mixed Indian languages with English (Hindi, Bengali, Kannada, Tamil, Telugu, Gujarati, Marathi, Malayalam) and standard English. Binary classifier discussed in this paper will classify Hinglish (Hindi when written using English script is commonly known as Hinglish) from seven other code-mixed Indian Languages with English and standard English. Multiclass classifier will classify the previously mentioned languages. Binary classifier gave an accuracy of 96% on the test data and the size of the model was 1.4 MB and achieved an accuracy of 87% with multiclass classifier on same test set with model size of 3.6 MB.**

Keywords*—Character grams; code-mixed; deep learning; Indian languages; language identification; NLP; social media text

I. Introduction

In recent years, there has been a boom in the social media usage, especially in India, because of deep penetration of internet connectivity among people. There are close to half a billion active social media users with a growth of 4.2% every year [1]. Due to this rapid increase, people of various demographics and ages have started to use social media and in turn code mixed language has become more popular than ever on social media. Usually, a local regional language is mixed with English. Be it for hate speech detection on social media or to generate auto reply suggestions to incoming text message or any other NLP system that involves code mixed data, requires language tagging as first step and will determine the accuracy of the system as whole to a great extent. There are many difficulties with language tagging. Even though large amount of code-mixed language data is available to us as raw data (tweets, posts, blogs, etc.) on social media, we don't have a readily available tagged data set suitable for supervised learning [2]. Also, there are many dialects of same language, and so there are different spellings and pronunciations of the same word in code mixed context. Another important difficultly is, if the data set is not diverse, then the supervised language tagging model will suffer from over fitting [3]. This paper aims in building code mixed Indian languages classifiers that can run on low resource devices with little latency, so that they can be used in real time applications like auto reply systems.

The reason for choosing the previously mentioned nine languages for classification is, 82% of Indian population speak one of the nine languages as their first language [4]. Since Hindi is spoken by 58% of Indian population [4], this article also considers a binary classifier that distinguishes Hinglish from other languages along with the multi class classifier. Collected close to 120k sentences for all the nine languages to create a separate train set and test set for training and testing model respectively.

In this article presents Stacked Bi-LSTM network that uses trigrams as features to classify Hinglish from other languages and Ensemble CNN - Bi-LSTM with attention network with both trigrams and quad grams as features for multiclass classification [5][6][7].

II. Related Work

In recent years, lot of research on language identification is done, which essentially is the first step in NLP systems, although less work is done where it involves detection of multiple Indian languages. Inumella Chaitanya et al. describe how common word embeddings like Continuous Bag of Words (CBOW) and Skip Grams models can be used to generate embeddings that can be feed to common machine learning models like support vector machine, Logistic Regression and K-Nearest neighbors among other algorithms [8]. Anupam Jamatia et al. in their paper describe about two models i.e., Bi-LSTM classifier and Conditional Random Fields (CRF) classifier and suggest that Bi-LSTM classifier performs better. Ramachandra Joshi and Raviraj Joshi describe about various input representations like character, sub-word and word embeddings, for language identification task in their paper. They also pass these representations as input to CNN and LSTM based models. They indicate that sub-word representation combined with LSTM model gives the best results [9].

Sourya Dipta Das et al. train multiple LSTM models and create an ensemble model using stacking and threshold technique which helping in increasing the accuracy of the model instead of using a single model [10]. Neelakshi Sarma et al. have developed a framework for language identification which is capable of recognizing words that are borrowed from other languages and used in multiple languages and predict the language. This framework also considers the context of the sentence [11].

Monojit Choudhury et al. used both word embeddings and character embeddings, then concatenated both of them to predict the language [12]. Harsh Jhamtani et al describes a method which combines word embeddings, character n-grams and POS tagging to predict the language [13]. Shruti Rijhwani et al presents a unsupervised model for language detection that does not require any annotated data to train the model. It is also capable of detecting large number of languages [14].

III. Proposed Solution

This section discusses about the dataset creation, pre-processing steps, embeddings generation and finally about the classifiers i.e., binary and multiclass classifiers. In summary, first a data set is created by collecting data from various sources. Vocabulary or Embeddings are generated from the data set, which will be used to encode the input before giving it the model. All the steps are explained in detail, in following sections and summary of the steps in very high level is shown in Fig. 1.

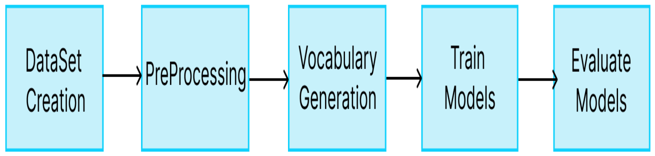


Fig. 1. Summary of Steps of Proposed Solution.

A. Dataset

There was no readily available dataset for the previously mentioned nine languages. Had to create one by collecting data from multiple sources like reddit, Facebook posts, twitter tweets, WhatsApp chats and blogs. English language sentences are easy to collect but it is difficult to find and collect other Indian local languages code mixed with English. Data for Indian languages code mixed with English, was collected by scrapping from dedicated subreddit topics and twitter tweets. Also, used Google translate API, where we provide a English sentence as input to translate function of the API and set the destination language to any one of the Indian languages [15]. The translate function returns an object that has a pronunciation attribute. This pronunciation is a close approximation of Indian language sentences written in English script. In addition to this, after collecting the data, some irregularities to the spellings were introduced in the data set, so that the model does not suffer from overfitting problem. In total, the dataset consists of little above 100k sentences and the test set consists of about 13k sentences. The test data consists mostly of real world data and the approximation data generated from google translate API is not included. The distribution of train set and test set is shown in Table I and Table II, respectively.

B. Pre-Processing

These common preprocessing steps are applied to all the sentences in dataset and also to the input sentence given to the trained model to predict. Removed the components of the sentences that don't help us in identifying the language. Digits, punctuation marks, extra whitespaces, HTML tags and emojis are removed.

TABLE I. Train Set

Languages	Number of Sentences
Hinglish	26804
Bengali	11570
Kannada	11148
Tamil	11115
English	11069
Gujarati	11032
Telugu	9231
Marthi	8166
Malayalam	6836
Total	106971

Apart from these common preprocessing steps, irregularities to spellings of Indian languages is introduced into only train set. This is because, Indian language words when written in English script can have different spellings. For instance, the word "why" in Telugu language is spelled as "enduku" or "yenduku", when written in English script. Similarly, "where" in Kannada language is spelled as "ellige" or "yellige". So, introduced these extra letter in some cases into train set so that the model generalizes well and does not suffer from overfitting problem. The summary of preprocessing steps are shown in Fig. 2.

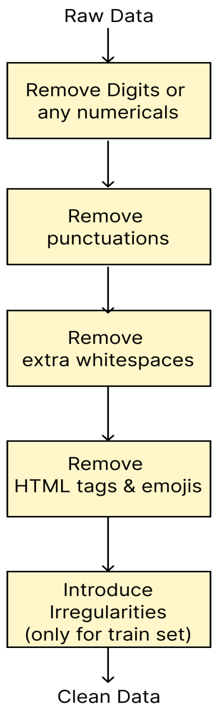

Raw Data
Remove Digits or any numericals
Remove punctuations
Remove extra whitespaces
Remove HTML tags & emojis
Introduce Irregularities (only for train set)
Clean Data

Fig. 2. Summary of Pre-Processing Steps.

TABLE II. Test Set

Languages	Number of Sentences
Hinglish	2496
Bengali	689
Kannada	1338
Tamil	1554
English	3226
Gujarati	262
Telugu	1599
Marthi	1942
Malayalam	442
Total	13548

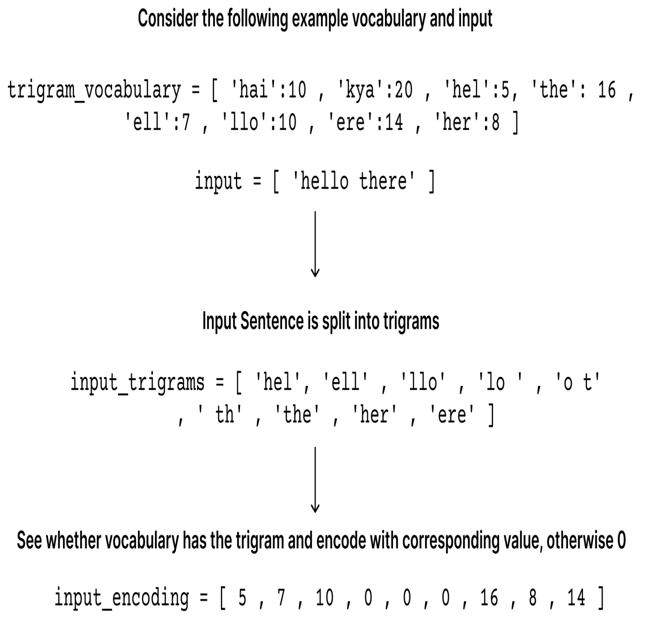
C. Vocabulary/Embeddings Generation

Generated trigrams and quadgrams using all the sentences from the dataset and stored these character grams in form of a dictionary, where the character gram string is the key and a unique integer starting from 1 as the value for the key. So, each character gram is unique identified by an integer. Zero (0) is used to indicate out of vocabulary (OOV) character grams. Only top 20k (in terms of number times it appears in dataset) trigrams and quadgrams are selected. We are limiting the size of the embeddings to limit the size of the model, keeping in mind that we need models that can run on-device with as little resource utilization as possible. Separate dictionaries are created for trigrams and quadgrams. Trigrams are used as embeddings in binary classifier and both trigrams and quadgrams are used in case of multiclass classifier. These dictionaries are stored as csv files.

The process of encoding the input that must be given to the model, using the vocabulary dictionaries generated is described next. For instance, lets consider the following trigram vocabulary dictionary and input sentence in English as shown in Fig. 3. The input sentence is then split into trigrams and each of these trigrams are searched in the vocabulary dictionary. If there is a match, then value associated with the trigram in the dictionary is used for encoding. If the trigram match is not found, then it is a case of out of vocabulary (OOV) and 0 is used to encode such trigrams. Same is done when quadgrams are used as features.

D. Binary Classifier

After following the preprocessing steps, the sentences in the train set are encoded using the earlier generated embeddings. Each sentence encoded is padded to have a size of 50. The first layer of the model is Embedding layer with vocab size of 20k, as previously mentioned and with both the input and output dimensions to be 50. The activation function for this layer was relu. The embedding layer is followed by a dropout layer (rate = 0.5). This followed by stacked Bi-LSTM layers with output dimensionality of 32 and 16. Stacking the Bi-LSTM layers, helped in boosting the accuracy of the model. This is followed by a dense layer with output dimensionality of 8 followed by drop out layer (rate = 0.5). Last is the output layer with sigmoid as the activation function. The summary of the model is shown in Fig. 4. The entire model was compiled with adam as optimization function and binary cross-entropy



Consider the following example vocabulary and input

```
trigram_vocabulary = [ 'hai':10 , 'kya':20 , 'hel':5, 'the': 16 ,
          'ell':7 , 'llo':10 , 'ere':14 , 'her':8 ]

input = [ 'hello there' ]
```

Input Sentence is split into trigrams

```
input_trigrams = [ 'hel', 'ell' , 'llo' , 'lo ' , 'o t'
          , ' th' , 'the' , 'her' , 'ere' ]
```

See whether vocabulary has the trigram and encode with corresponding value, otherwise 0

```
input_encoding = [ 5 , 7 , 10 , 0 , 0 , 0 , 16 , 8 , 14 ]
```

Fig. 3. Steps to Encode Input.

as loss. The number of epochs was set to 3 and batch size was set to 64.

The trained model is then saved as a TensorFlow Lite (tflite) model in order to compress the size and decrease the latency of the model [16], without losing much on accuracy and tflite models are easier to use on low resource devices like mobiles or even embedded devices. The tflite model size came to be 1.4 MB. The coming sections discuss about the accuracy and performance of the model in detail.

E. Multiclass Classifier

The preprocessing and encoding steps are same as for binary classifier, except both trigrams and quadgrams are used. Siamese neutral networks are used, whose output is concatenated and given to a dense network to form a ensemble model [17][18]. Built two models, with trigrams as features for one and quadgrams as features for the other, then concatenated the outputs and feed it to a deep network as shown in Fig. 5. After the embedding layer we use Conv1D layers with varying kernel sizes (here 3, 4 and 5) with relu as activation function and followed spatial dropout (rate = 0.2) and max pooling cells. Using filters of various sizes, dropout and max pooling cells helped reduce overfitting and variance largely. This is followed by Bi-LSTM stack with attention with 32 and 16 sizes orderly. The output from two identical networks using different features was combined by a concatenate layer followed by two dense layers of 16 and 9 orderly as shown. The entire model was compiled with adam as optimization function and categorical cross-entropy as loss. The number of epochs was set to 3 and batch size was set to 64. Even the multiclass classifier is stored as a tflite model and the size comes to 3.6 MB. The accuracy and performance of the model is discussed in the coming sections.

TABLE III. Models Performance Summary

Models	Accuracy	F1-score	Model Size	Latency	Peak RAM Usage
Binary Classifier	96%	0.96	1.4MB	8-10ms	4.7MB
Multiclass Classifier	87%	0.88	3.6MB	30-35ms	8.3MB

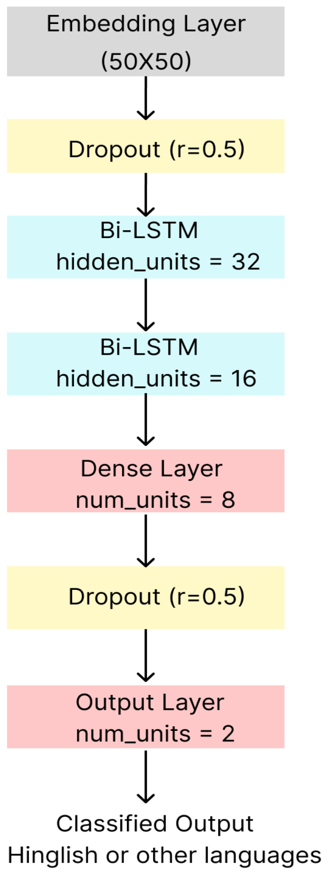


Fig. 4. Binary Classifier Model Summary.

TABLE IV. Binary Classifier Performance for Each Class

Languages	Precision	Recall	F1-score
Hinglish	0.95	0.96	0.95
Other Languages	0.98	0.97	0.97

TABLE V. Multiclass Classifier Performance for Each Class

Languages	Precision	Recall	F1-score
Hinglish	0.92	0.96	0.94
Bengali	0.84	0.98	0.90
Kannada	0.83	0.86	0.85
Tamil	0.87	0.82	0.84
English	0.90	0.90	0.90
Gujarati	0.75	0.62	0.69
Telugu	0.86	0.88	0.87
Marthi	0.98	0.98	0.98
Malayalam	0.67	0.48	0.59

The peak RAM usage was estimated using massif tool, which is heap profiler [21]. Massif comes as part of valgrind, which is collection of tools for memory profiling. The peak usage for both the models is recorded in Table III.

The same vocabulary dictionaries saved as csv files earlier are used to encode the test input sentences given to the trained models to predict the output. The predicted labels given by the trained model is stored and compared with the actual labels, then the classification report is generated based on it. The classification report for both binary classifier and multiclass classifier generated using classification_report part of scikit-learn package [22] and is reported in Table IV and Table V, respectively.

IV. Experimental Results

The parameters on which the models was judged are accuracy, f1-score, size, latency and the peak RAM usage by the model. The test set described earlier is used to measure the performance of the model. A c++ program was written to perform the earlier discussed preprocessing steps on the input which is to be given to trained model and to load the saved tflite model and predict the output. The reason for writing c++ code to estimate the performance of the models rather than python was, c++ code runs faster than python [19].

The latency of the model was estimated by recording the time (say t1) when the input is given for preprecessing and again recording the time (say t2) when model predicts the output. The latency is calculated as difference of t2 and t1. So, this latency also includes the time taken for preprocessing steps as well. The now() function of high_resolution_clock class as part of chrono c++ header was used to record time t1 and t2 [20]. The range of latency of both the models is recorded in Table III.

V. Analysis

The binary classifier performs well on most kinds of data. The only drawback observed was, when the input sentence is very short and mostly filled with English, then it misclassifies Hinglish as English. For instance, “Relatives aya” (which means “Relatives came” in English) is classified as English sentence instead Hinglish. This is because the sentence is very short and has only a three letter Hindi word as part of it and the model finds it difficult to predict the correct label. Similarly, “Ek spoon” (which means one spoon) is also misclassified as English.

The multiclass classifier does well on most languages, except Malayalam and Gujarati. The F1-scores for Malayalam and Gujarati classes are 0.59 and 0.69 respectively, which is way less compared to other classes as seen from Table V. These classes also pull down the overall accuracy of the multiclass classifier. The main reason for this is because Malayalam language is very closely related to Tamil Language. Out of 442 Malayalam sentences in the test set, 190 of are missclassified

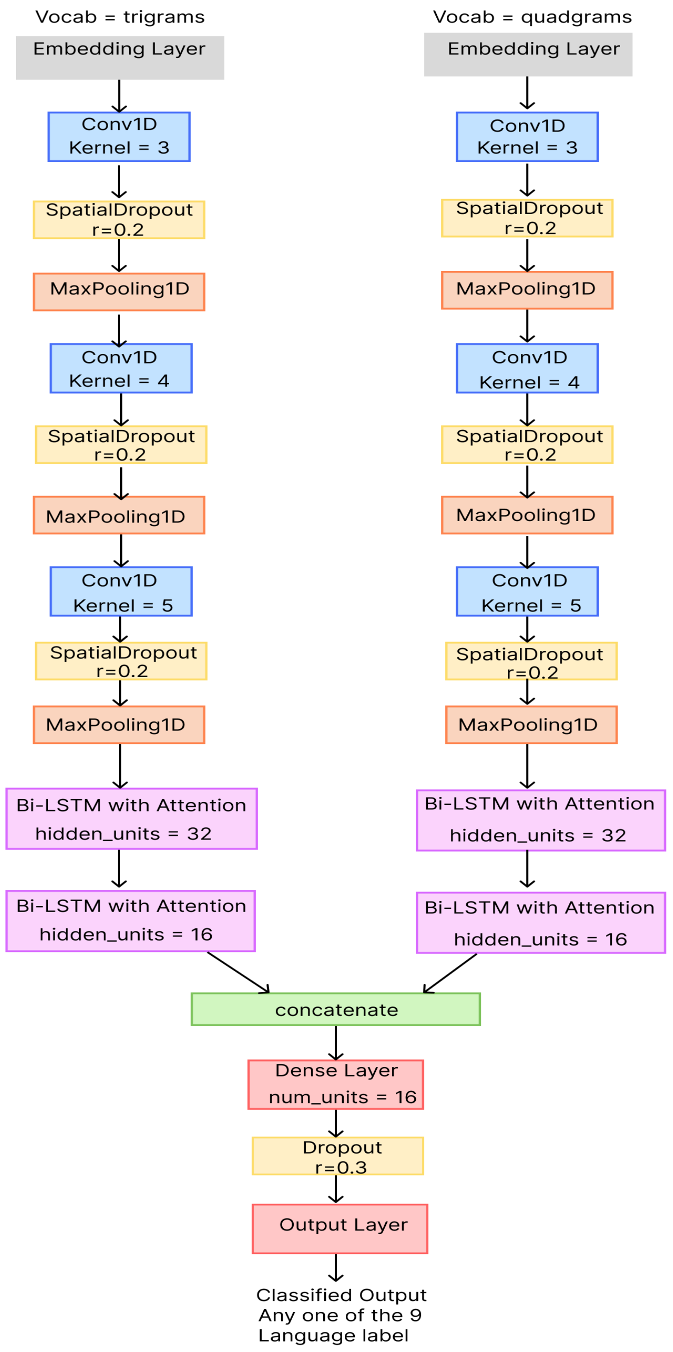


Fig. 5. Multiclass Classifier Model Summary.

as Tamil. These two languages are spoken by people of adjacent states of South India, and the root words of both languages are very similar. For instance, “come home” in Malayalam is “Vittir varu” and in Tamil is “Vittirku va”. The encoded sequences of these two sentences using trigrams or quadgrams will be very close. This is the reason the model confuses for Malayalam sentences and classifies them as Tamil sentences. The same reason goes for Gujarati language class, where Gujarati language is very closely related to Hindi and encoded sequences are also close and the model misclassifes. Out of 262 Gujarati sentences in the test set, 70 are misclassified as Hinglish.

VI. Conclusion and Future Work

The binary classifier performs really on diverse data and generalizes well, expect for really small sentences. The multi-class classifier performs well on seven out of nine languages. With model sizes of 1.4 MB and 3.6 MB for binary classifier and multiclass classifier, these models can used on any low resource devices like smart watches, mobiles or any embedded system where memory and RAM consumption are a constraint.

The future work would be to collect more real world data to train, in place of approximate data generated using Google translate API. Also, the vocabulary size was restricted to limit the resource usage on the device. So, if the models are run on powerful devices, then the vocabulary dictionaries can be further extended further and models can be retrained with the extended vocabulary and check if there is any effect on accuracy of the models. Also, the embedding layer size and the number of hidden layers of the neural network can be increased to increase the accuracy the model.

REFERENCES

[1] https://www.theglobalstatistics.com/india-social-media-statistics/

[2] Osisanwo F.Y., Akinsola J.E.T., Awodele O., Hinmikaiye J. O., Olakanmi O., Akinjobi J. "Supervised Machine Learning Algorithms: Classification and Comparison". International Journal of Computer Trends and Technology (IJCTT) V48(3):128-138, June 2017. ISSN:2231-2803

[3] https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

[4] https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers_in_India

[5] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory" .Neural Computation. Volume 9, Issue 8. November 15, 1997. pp 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[6] Dong, X., Yu, Z., Cao, W. et al. A survey on ensemble learning. Front. Comput. Sci. 14, 241–258 (2020). https://doi.org/10.1007/s11704-019-8208-z

[7] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[8] Inumella Chaitanya, Indeevar Madapakula, Subham Kumar Gupta and S thara. 2018. "Word Level Language Identification in Code-Mixed Data using Word Embedding Methods for Indian Languages". 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI). ISBN: 978-1-5386-5214-2

[9] Joshi, R., Joshi, R. (2022). "Evaluating Input Representation for Language Identification in Hindi-English Code Mixed Text". In: Kumar, A., Senatore, S., Gunjan, V.K. (eds) ICDSMLA 2020. Lecture Notes in Electrical Engineering, vol 783. Springer, Singapore. https://doi.org/10.1007/978-981-16-3690-5_73

[10] Sourya Dipta Das, Soumil Mandal, Dipankar Das. "Language Identification of Bengali-English Code-Mixed Data using Character & Phonetic based LSTM Models". FIRE '19: Proceedings of the 11th Forum for Information Retrieval Evaluation. December 2019. Pages 60–64. https://doi.org/10.1145/3368567.3368578

[11] Neelakshi Sarma, Ranbir Sanasam Singh, Diganta Goswami. "Switch-Net: Learning to switch for word-level language identification in code-mixed social media text". Natural Language Engineering. Volume 28 Issue 3. DOI: 10.1017/s1351324921000115

[12] 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.

[13] Harsh Jhamtani, Bhogi Suleep Kumar, Vaskar Raychoudhury. "Word-level Language Identification in Bi-lingual Code-switched Texts".Pacific Asia Conference on Language, Information and Computing. December 2014.

[14] 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.

[15] https://stackabuse.com/text-translation-with-google-translate-api-in-python/

[16] https://towardsdatascience.com/model-compression-a-look-into-reducing-model-size-8251683c338e

[17] https://towardsdatascience.com/what-are-siamese-neural-networks-in-deep-learning-bb092f749dcb

[18] https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/

[19] https://towardsdatascience.com/how-fast-is-c-compared-to-python-978f18f474c7

[20] https://en.cppreference.com/w/cpp/chrono/high_resolution_clock

[21] https://valgrind.org/docs/manual/ms-manual.html

[22] https://scikit-learn.org/stable/