MULTILINGUAL ABUSIVE COMMENT IDENTIFICATION

BY

SUDHEERA MADDILA (G01324503)
KRISHNA VARDHAN MANDANAPU (G01336669)

ABSTRACT

In the most recent decade, with the increase of an interactive web, particularly on popular online social media like Facebook, Twitter and Instagram, there has been an exponential increment in user-generated content being made accessible over the web. The rise of this phenomenon is mainly due to the anonymity given to social media users and to the lack of effective regulations provided by these platforms.

Negative comments initiate cyber-bullying that targets individuals or a group of people. According to '*Pew Research Center*', one in five internet users (~22%) had been victims of online harassment. This is a serious issue that needs to be addressed without just eliminating the problem itself. In a country like India, with over a hundred of languages being spoken every day these users may be from linguistically different communities which has caused the problem to identify the detection of offensive talk. One of the most effective solutions to cope with this problem is using computational techniques to discriminate offensive content. Hence, this study aims to tackle the Multilingual Offensive Language Detection through our model by using transfer learning models and applying contextual word embeddings and approaches employed in both cross-domain and cross-lingual settings.

The main challenge here is dealing with low-resource linguistic datasets. Our dataset has around 13 Indic languages, and we are developing a model that tries to accurately identify abusive comments in native Indian languages. The dataset is from Kaggle competitions (Indraprastha Institute of Information Technology -Delhi), which contains abusive comments posted on the '*Moj*' app, one of India's largest short video apps in multiple regional languages, that has comments in 10+ languages. Data is massive, multilingual and human annotated. It also has metadata like #likes, #report counts of both posts and comments, etc. The number of positive comments were approximately the same as the negative ones. (52% & 48%). The dataset also has comments in both English (code-mixed language) and native scripts.

1. INTRODUCTION

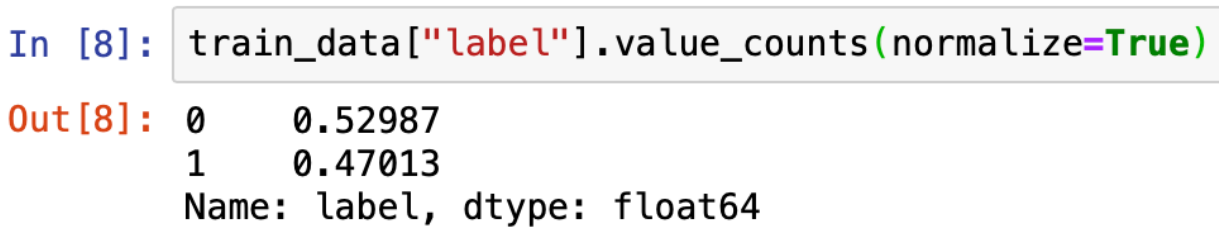
Presently any information online can arrive at billions of web users in mere seconds that has led to not only a positive exchange of ideas but resulted in malicious and offensive content over the web. However, using human moderators to check this offensive content is not anymore, an effective method. This ushers social media administrators to automate the offensive language detection process and supervise the content using Natural Language Processing (NLP) techniques. The Multilingual Offensive Language Identification task is usually modeled as a supervised classification problem where a system is trained on annotated texts containing multilingual abusive or offensive expressions.

Multilingual text classification is defined as the task of classifying simultaneously a set of texts written in different Indic languages and belonging to a set of fixed categories across languages. Majority of the current studies for these negative comment detections only focused on a single language like English, but abusive talk in the social media content is not usually limited to specific languages. This problem is different from cross-language text categorization, when a document written in one language must be classified in a category system learned in another language. The basic idea is to feed various texts with different languages to the same classifier then perform the training on a multilingual dataset.

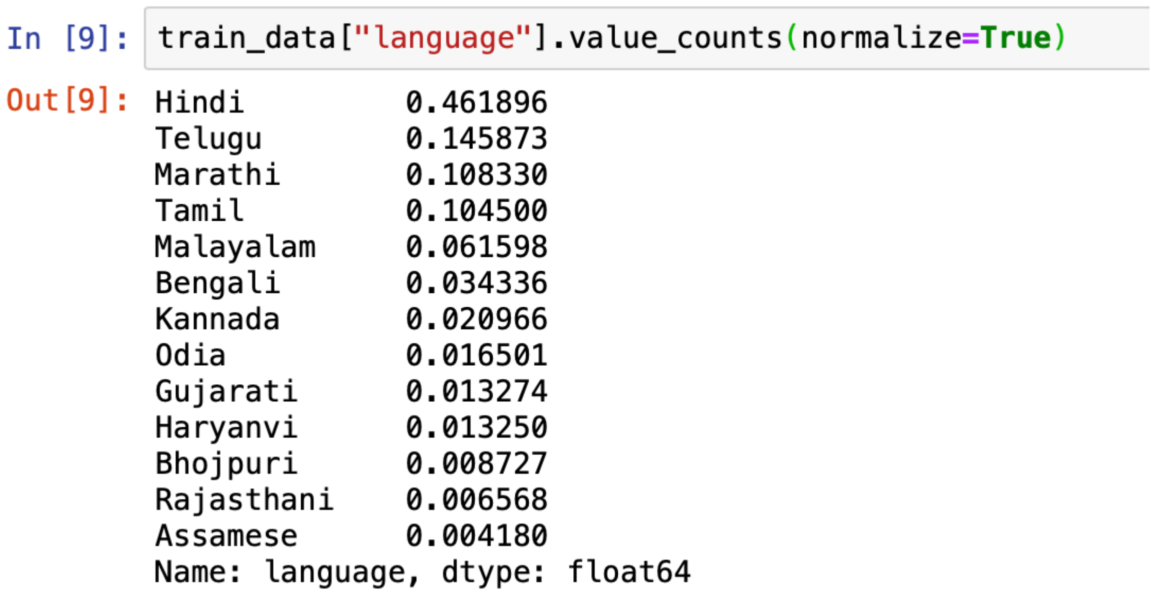
Transformer models have been used successfully for various NLP tasks. Most of the tasks were focused on a single language like English, since most of the pre-trained transformer models were trained on English data. Even though, there were several multilingual models like BERT-m, there were many speculations about its ability to represent all the languages and although BERT-m model showed some cross-lingual characteristics it has not been trained on cross-lingual data. The main idea of the methodology is that we train a classification model on a resource rich language, using a cross-lingual transformer model and initialize the training process for a lower resource language.

2. DATA ANALYSIS

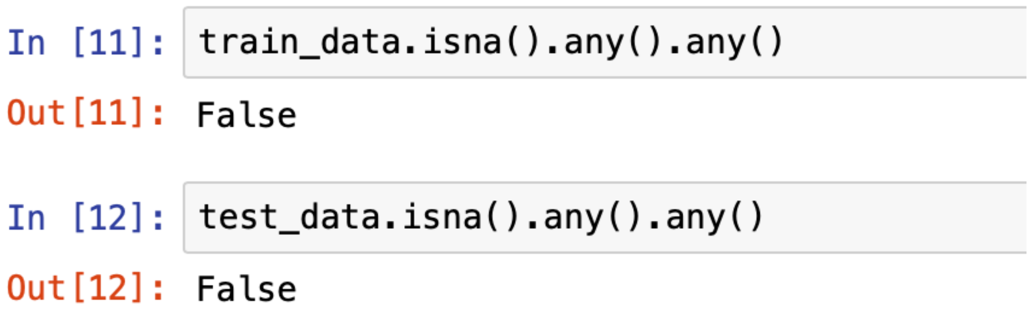
The training dataset contains unique columns like language, post_index, commentText, report_count_comment, report_count_post, like_count_comment, like_count_post, label, val. Dataset has comments in 13 different Indian languages. As part of data analysis, we checked for null values and found that there were no null values. We also looked for whether the dataset is balanced or not.



```
In [8]:  train_data["label"].value_counts(normalize=True)

Out[8]:  0    0.52987
         1    0.47013
         Name: label, dtype: float64
```



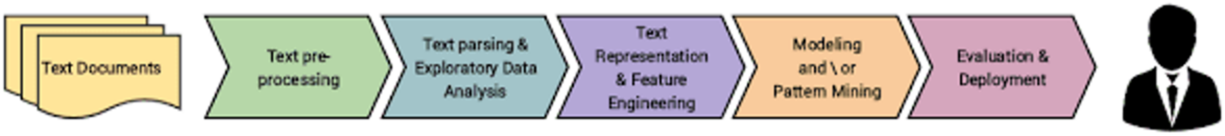
```
In [9]:  train_data["language"].value_counts(normalize=True)

Out[9]:  Hindi         0.461896
         Telugu        0.145873
         Marathi       0.108330
         Tamil         0.104500
         Malayalam     0.061598
         Bengali       0.034336
         Kannada       0.020966
         Odia          0.016501
         Gujarati      0.013274
         Haryanvi      0.013250
         Bhojpuri      0.008727
         Rajasthani    0.006568
         Assamese      0.004180
         Name: language, dtype: float64
```



```
In [11]:  train_data.isna().any().any()

Out[11]:  False

In [12]:  test_data.isna().any().any()

Out[12]:  False
```

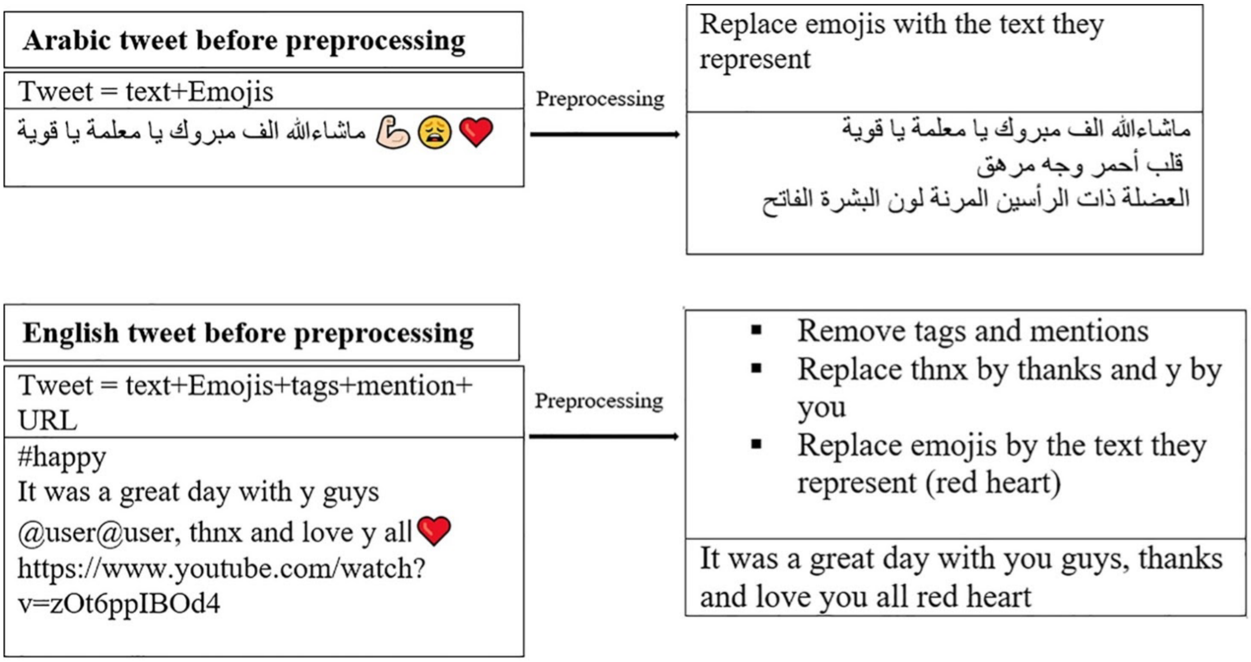
3. DATA PRE-PROCESSING

To preprocess our text means to simply bring the text into a form that is ***predictable*** and ***analyzable*** for our task. A task here is a combination of approach and domain. We used Natural Language Processing to achieve our goal. NLP is a way for computers to analyze, understand and derive meaning from human language in a smart and useful way. They are used to provide automatic summarization of the main points in each text or document. NLP is also used to classify text according to predefined categories or classes and is used to organize information. Our multilingual offensive language detection system comprises several modules, including tweet preprocessing, BERT tokenization, text representation, and tweet classification. The pre-processing phase consists of several steps in order to keep only pertinent information.


Text Documents
Text pre-processing
Text parsing & Exploratory Data Analysis
Text Representation & Feature Engineering
Modeling and \ or Pattern Mining
Evaluation & Deployment

We follow the upcoming steps:

- We proceed by removing all the hashtags, HTML tags, mentions, and URLs.
- For English text, we further replace contractions with their full forms, fix misspelled words, and convert text to lowercase.
- We replace emoji’s if they exist with the text they represent since emoji’s or emoticons play an essential role in defining a tweet.


Arabic tweet before preprocessing
Tweet = text+Emojis
ماشاءالله الف مبروك يا معلمة يا قوية 💪😩❤️
Preprocessing
Replace emojis with the text they represent
ماشاءالله الف مبروك يا معلمة يا قوية
قلب أحمر وجه مرهق
العضلة ذات الرأسين المرنة لون البشرة الفاتح
English tweet before preprocessing
Tweet = text+Emojis+tags+mention+ URL
#happy
It was a great day with y guys
@user@user, thnx and love y all ❤️
https://www.youtube.com/watch?v=zOt6ppIBOd4
Preprocessing
▪ Remove tags and mentions
▪ Replace thnx by thanks and y by you
▪ Replace emojis by the text they represent (red heart)
It was a great day with you guys, thanks and love you all red heart

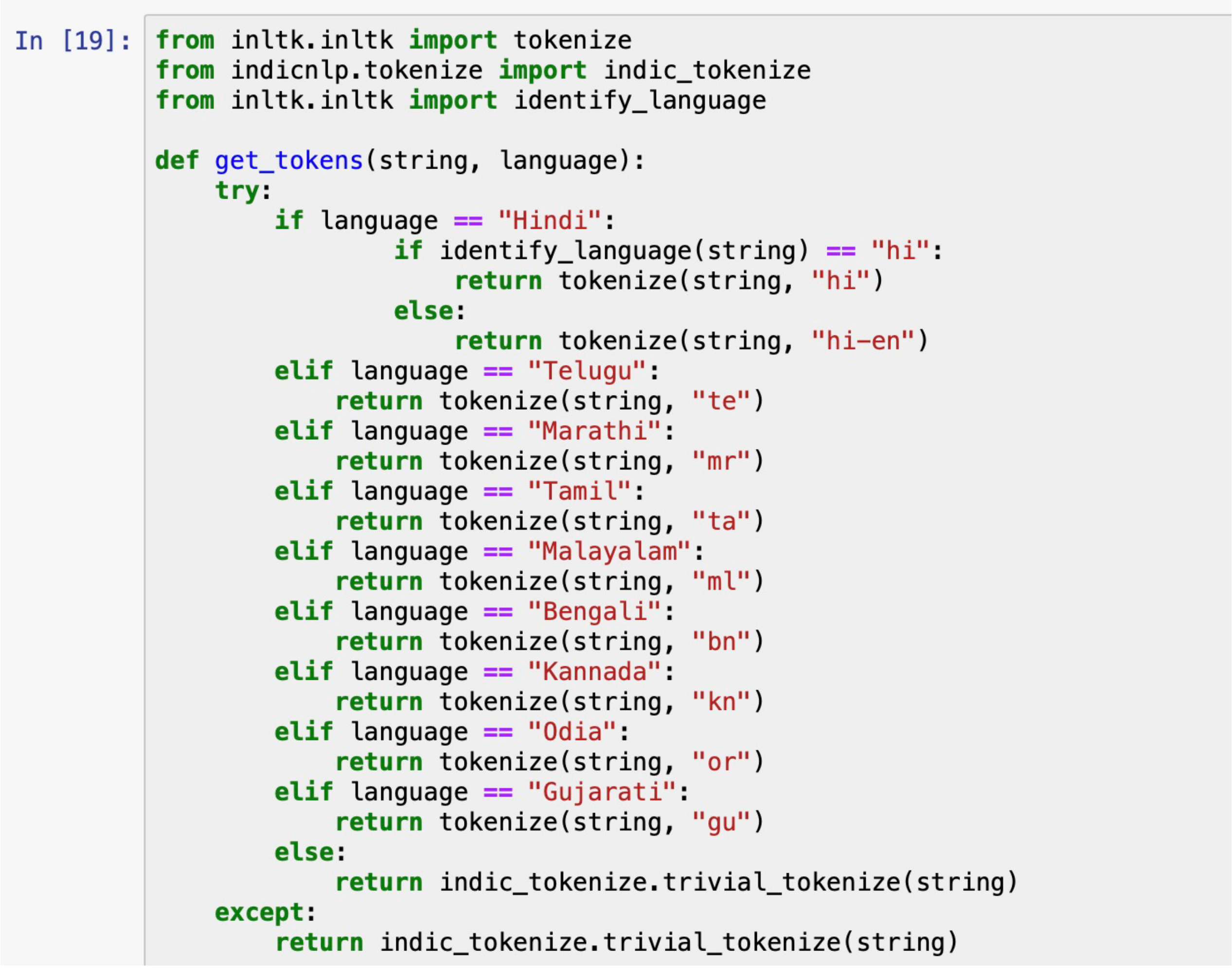
Solving an NLP problem is a multi-stage process. As part of this text processing, we perform the next following steps:

NORMALIZATION: Normalization is the process of transforming a text into a canonical (standard) form that allows the processing of the data effectively. Text normalization is important for noisy texts such as social media comments, text messages and comments to blog posts where abbreviations, misspellings and use of out-of-vocabulary words are prevalent. It has been effective for analyzing highly unstructured clinical texts in non-standard ways.

*TOKENIZATION***:** Here we used ‘inltk’ and ‘indicnlp’ for tokenization. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens. Before processing a natural language, we need to identify the *words* that constitute a string of characters. That’s why tokenization is the most basic step to proceed with NLP (text data). This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.

STEMMING: Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

LEMMATIZATION: Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma*. Here we used stanza for lemmatization. Tokenization & lemmatization were implemented based on the language of comment.



```
In [19]: from inltk.inltk import tokenize
         from indicnlp.tokenize import indic_tokenize
         from inltk.inltk import identify_language

         def get_tokens(string, language):
             try:
                 if language == "Hindi":
                         if identify_language(string) == "hi":
                             return tokenize(string, "hi")
                         else:
                             return tokenize(string, "hi-en")
                 elif language == "Telugu":
                     return tokenize(string, "te")
                 elif language == "Marathi":
                     return tokenize(string, "mr")
                 elif language == "Tamil":
                     return tokenize(string, "ta")
                 elif language == "Malayalam":
                     return tokenize(string, "ml")
                 elif language == "Bengali":
                     return tokenize(string, "bn")
                 elif language == "Kannada":
                     return tokenize(string, "kn")
                 elif language == "Odia":
                     return tokenize(string, "or")
                 elif language == "Gujarati":
                     return tokenize(string, "gu")
                 else:
                     return indic_tokenize.trivial_tokenize(string)
             except:
                 return indic_tokenize.trivial_tokenize(string)
```

3.1 VECTOR CONVERSION

Once we had the processed data, it was converted to a vector format using the Term Frequency-Inverse Document Frequency.

Term Frequency (TF)

It is a measure of the frequency of a word (w) in a document (d). TF is defined as the ratio of a word’s occurrence in a document to the total number of words in a document. The denominator term in the formula is to normalize since all the corpus documents are of different length

$$TF(w,d) = \frac{\textit{occurences of w in document d}}{\textit{total number of words in document d}}$$

Inverse Document Frequency (IDF)

It is the measure of the importance of a word. Term frequency (TF) does not consider the importance of words. Some words such as’ of’, ‘and’, etc. can be most frequently present but are of little significance. IDF provides weightage to each word based on its frequency in the corpus D. IDF of a word (w) is defined as the following-

$$IDF(w,D) = \ln(\frac{\textit{Total number of documents } (N) \textit{ in corpus D}}{\textit{number of documents containing w}})$$

4. CROSS-VALIDATION

Once we have the data that has been processed, we perform cross-validation on our dataset. Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. A model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called **overfitting**. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set ‘X_test’, ‘Y_test’.

4.1 CLASSIFICATION MODELS

Classification is a data-mining technique that assigns categories to a collection of data to aid in more accurate predictions and analysis. Classification is one of several methods intended to make the analysis of very large datasets effective. The different classifiers that we implemented in our project were-

- *LOGISTIC REGRESSION*: Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. This model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

- *RANDOM FOREST:* Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time.

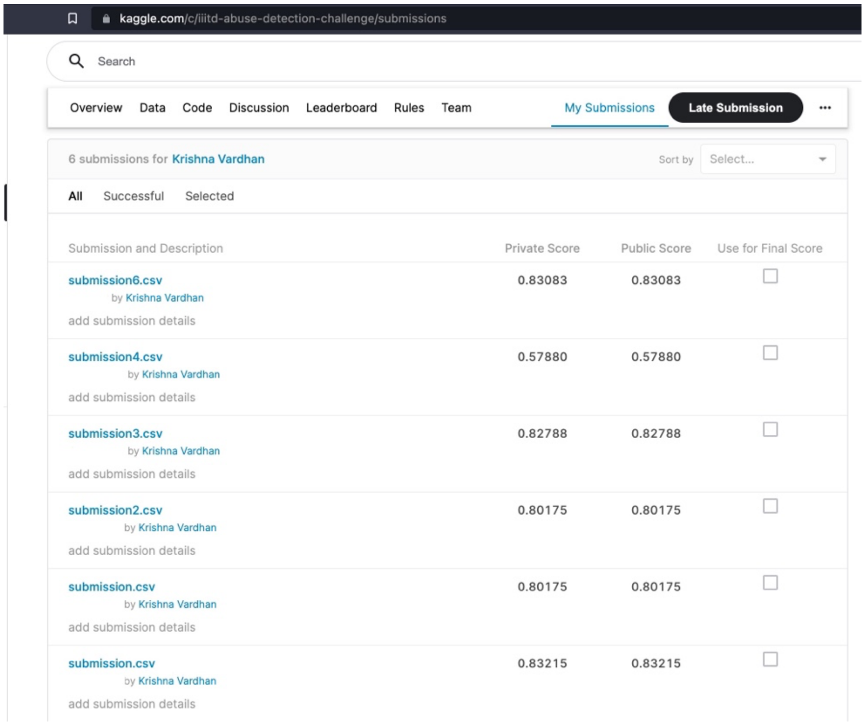
- *K-NEAREST NEIGHBOR ALGORITHM:* The k-nearest neighbors (KNN) algorithm is a data classification method for estimating the likelihood that a data point will become a member of one group, or another based on what group the data points nearest to it belongs to.

5. ACCURACY METRIC

Here we have used the ‘Accuracy’ as the performance metric for our problem solution. We observed that Logistic Regression with ‘lbfgs’ solver gave the best results with an accuracy of 83% in ’Kaggle’ competitions.

$$\boldsymbol{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

RESULTS submitted to KAGGLE:



6 submissions for Krishna Vardhan

Submission and Description	Private Score	Public Score	Use for Final Score
submission6.csv by Krishna Vardhan add submission details	0.83083	0.83083	□
submission4.csv by Krishna Vardhan add submission details	0.57880	0.57880	□
submission3.csv by Krishna Vardhan add submission details	0.82788	0.82788	□
submission2.csv by Krishna Vardhan add submission details	0.80175	0.80175	□
submission.csv by Krishna Vardhan add submission details	0.80175	0.80175	□
submission.csv by Krishna Vardhan add submission details	0.83215	0.83215	□

6. CONCLUSION

The goal of our project is to develop a paradigm for predicting offensive comments in various Indian languages, given the regional data along with contextual user data. This challenge drew our attention mainly because it addresses the problems in Indic languages and solving this would help in creating a safe online space for social media users. This task has given us a better understanding of the detection of speech on social media. It not only has given us an understanding of the importance of the detection of offensive speech but also improved our ability to solve such problems. We believe that tackling hate and offensive content on public platforms is a serious challenge and hope that our model will be useful, specifically in the Indian context.

7. REFERENCES

https://arxiv.org/pdf/2103.10730.pdf
https://www.mdpi.com/2078-2489/12/8/306/pdf
https://cdn.iiit.ac.in/cdn/precog.iiit.ac.in/pubs/AAAI-19_paper_211.pdf
http://ceur-ws.org/Vol-2517/T3-9.pdf