Automated Detection of Hate Speech and Profanity for Multiple and Mixed Languages

[1]Kalikidhar Reddy, [2]K S Nitish, [3]Vishwanath V Karki,[4]B Deepthi
[1,2,3,4] *PES University*

Abstract **- Detecting profanity, abuse or hate by building efficient Text Content Moderation systems has become an integral process and a practice for various digital and online media platforms. Chat platforms and community discussion forums are essential customer services that are being provided by many online media business platforms which allow the users which to express their opinions. Many users tend to misuse this service to spread profane and abusive content online.**

The aim of this research is to design techniques and create independent models that are able to identify and detect instances of hate, abuse or profane content in English, Hindi and Hinglish. Further discrimination of the inappropriate instances into various classes, namely: (i) HATE (hate speech), (ii) OFFENSIVE (insulting, degrading) and (iii) PROFANE (swear words, cursing) has been performed. This work which aims to solve a prominent problem in the field of Natural Language Processing leverages the Machine Learning and Deep Learning Models to perform the Classification of text.

Index Terms **- Natural Language Processing, Deep Learning, Code-Mixed, Multilingual, Hinglish, Text Classification.**

I.INTRODUCTION

The massive growth in social interactions that take place on online digital platforms has also led to the increase of abusive and hateful content, exploiting those digital services. Offensive and Hate content often tend to target either an individual or a particular group, on the basis of their factors of identity like color, religion, sexual orientation or nationality. Profane content or swear words are abusive in nature which are hurtful to an extent though not as serious or harmful as the content that expresses hatred. However, filtering such content manually is not a scalable solution and is quite challenging due to its tedious nature. Internet users of the Indian Sub-Continent use mainly three forms of language for communicating and posting content online which are – (i) English, (ii) Hindi (Multilingual) and (iii) Hinglish which is a code-mixed mixture of English and Hindi. English is widely adopted all over the world and Hindi is the second most popular language after English that is used by Indian users. However, there has been an increasing gain in popularity for Hinglish Language among the Indian users as it is very convenient to write the text in English compared to any other language and also due to its widespread use among everyday conversations between Indian users. However, it does pose a multitude of challenges as it does not have a fixed vocabulary, grammar or any semantic structure. This often leads to redundancies and spelling mistakes in our vocabulary.

The main contributions described in this paper are as following:

1. Implementing Neural Network Architectures capable of performing Binary as well as Multi Class Classification for English data.
2. Implementing Neural Network Architectures capable of performing Binary as well as Multi Class Classification in a multilingual setting for Hindi data and further improving the performance using the pre-trained multilingual BERT model.
3. Implementing a stacked SVM architecture that is capable of performing Multi Class Classification to detect instances of hate, abuse and profanity in Hindi – English Code-Mixed data.

II. RELATED WORK

In [1], the authors have introduced a novel HOT dataset and also performed the classification of offensive tweets in Hinglish language. They have proposed the MIMCT model that uses multiple embeddings for a CNN-LSTM architecture. They have also compared the performances with the baseline SVMs and Random Forest Classifiers.

In [2], the authors have designed a dictionary-based technique to identify and detect racism in the Dutch social media.

In [3], the authors have created and annotated a Hindi-English Code-Mixed corpus and also built SVM and Random Forest Classifier model architectures to classify the tweets as hate speech or normal speech
In [4], the authors have described in detail the various challenges one might encounter when performing the task of detecting hate speech. They have also built multi-view SVMs in order to detect hate speech.
In [5], the authors have developed a novel technique to create an annotated Hindi-English Code-Mixed corpus that has tweets in both the Roman as well as the Devnagari script. They have also provided us with an extensive discussion of F1 Scores of various trained Machine Learning and Deep Learning Classifiers.
In [6], the authors have proposed a Bayesian Filtering method with approximate string-matching techniques to filter out the profane content.

III. PROPOSED METHODOLOGY

A. Data Description and Preprocessing

Three datasets have been chosen to conduct experiments in three different settings namely – English, Hindi (Multilingual setting) and Hinglish.

Table 1:Datasets

LANGUAGE	DATASET	SIZE (Rows x Columns)
English	HASOC 2020	3708 X 5
Hindi	HASOC 2020	2963 X 5
Hinglish	HEOT	3189 X 2

The English and Hindi HASOC 2020 Datasets provide two subtasks – (Sub-Task A: Binary Classification, Sub-Task B: Multiclass Classification).
The task and label description for classification of the English and Hindi HASOC 2020 dataset is as follows

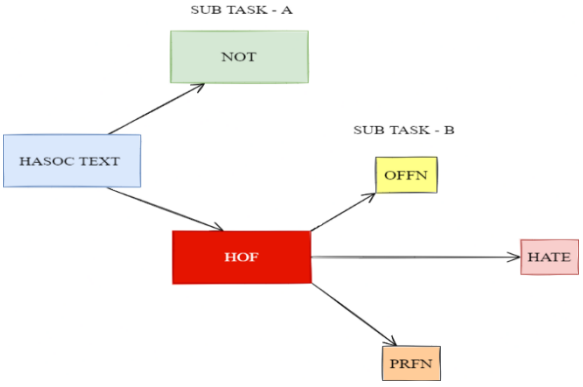

Figure 1:Lables for HASOC

The label description for classification of the HEOT dataset is as follows –

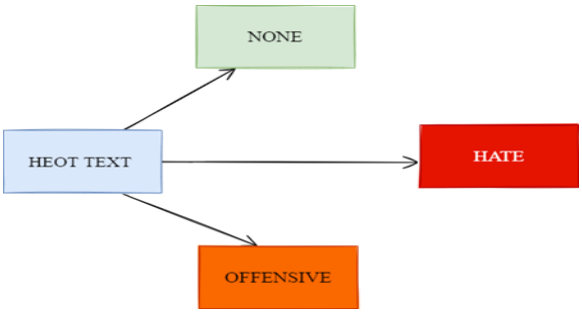

Figure 2:Lables for HEOT

The data obtained is then streamed through a series of steps for preprocessing. Preprocessing is considered to be an important process as the text obtained from the datasets may not be represented in a suitable format which can ultimately hinder the process of training the model.
The steps followed for performing the preprocessing are mentioned as follows:

1. Convert all the characters in the Text to lower case for the sake of uniformity in the data. However, this step does not apply to Hindi data.
2. Remove the stop words that do not contribute to the context or the meaning of text. E.g., (this , and) ,(इस,और) , (iss,aur)
3. Remove the @ mentions if present.
4. Remove the # hashtags if present.
5. Remove the http uniform resource locators or any other web links.
6. Remove digits , dates and special characters.
7. Remove emoticons and Unicode characters.
8. Normalize the Hinglish text by removing unnecessary repetitions in characters which is frequently seen in Hinglish Code-Mixed text. This is a major challenge that leads to the increase in the number of duplicate words representing the same meaning in our corpus. This happens as there is no fixed spelling for a word in the vocabulary. E.g., Name in Hinglish could be written as nam, naam, naaaammm etc.

B. Embeddings Generation and Feature Extraction

The next step after preprocessing is to convert the text into embedding vectors . Higher dimensional vectors for large inputs representing words are translated into a relatively lower dimensional space before passing them as input to the Neural Network.
The maximum length for each text is set to be 120 characters.

1. ENGLISH – GloVe

GloVe embeddings provide an advantage over Word2Vec as they do not just rely on the local context that depicts the information of the words but also considers the global word co-occurrences to derive the word vector representations.

2. HINDI – FastText

For a multilingual setting, fastText embeddings are aligned in a common space of both the languages. In the unsupervised manner to train the embeddings, the mapping between the source space to the target space is learnt.

FastText also provides an advantage of handling the Out-Of-Vocabulary words by taking word parts into account as well.

3. HINGLISH

Word Embeddings are generally considered to be a silver bullet in the field of Natural Language Processing and often outperform BoW and TF-IDF. However, it has been concluded in the experiments conducted in this research that BoW and TF-IDF might be better to use in this scenario due to the following reasons -

a. Since we are building a baseline SVM Model to experiment with.
b. Since our dataset is small, these techniques work better compared to Embeddings.
c. It is difficult to obtain and generate Embeddings for the case of code-mixed data as the context for the data would be specific to the cross-lingual domains.

3.1 BAG OF WORDS

The Bag of Words model is used to represent a text document in a simplified manner by using specific criteria such as the number of times a term is appearing or its frequency. However, it could lead to zeroes in the vector representations resulting in a sparse matrix which has to avoided. If new words are added, the vocabulary size increases resulting in an increase in the vector size too.

3.2 TF-IDF

Term Frequency – Inverse Document Frequency is a method to compute a score for a particular term and signify its importance in the given document and corpus.

As mentioned in [1] , TF-IDF gave higher performance when compared to other techniques.

C. Model Building

1.ENGLISH

1.1 LSTM

Long Short-Term Memory Networks are known to be a more refined architecture of Recurrent Neural Networks as they are able to capture the information regarding the long term dependencies within the text. Every unit in the LSTM Model takes the embedding vector corresponding to the current word as Input and takes the Output from the previous or the preceding unit. It begins to accumulate information about the text in a recursive manner as it uses the Inputs to update its memory cell.

1.2 CNN

Convolutional Neural Networks are popularly used for Text Classification. After three convolution operations where each convolution uses a filter size of 128, they can identify the patterns and similarities in text sequences by capturing the context irrespective of the position of the word tokens. Pooling layers in the network convert each text sequence into a vector of fixed length and thus are able to extract the information from the entire sequence. Max-Pooling has been used as this layer is responsible for capturing the crucial latent semantic factors from the text. Finally, a SoftMax layer is added that can calculate the probability distributions of all the classes for each input text. The number of units int this layer needs to be configured according to the type of classification e.g., Binary and Multi Class Classification.

2.HINDI

2.1 CNN

It is imperative to prepare the pre-trained multilingual embeddings provided by fast Text before proceeding with building the CNN architecture for classifying Hindi text. The input sequences representing the word embeddings are passed through three convolutions that are 1-Dimensional. A filter size of 128 is used has been used for each convolution operation. Finally, a dense layer is added along with softmax where the size of the layer or the number of units is equal to the number of Output classes specified by the classification task.

2.2 M-BERT

BERT multilingual base model (cased) is a pre-trained model which was pre-trained on top of 104 languages including Hindi using the largest Wikipedia corpus with the Masked Language Modelling or MLM objective. It was pre-trained in a self-supervised fashion without any human intervention for labelling the texts. The raw model provided by huggingface

could be used for either Next Sentence Prediction or Masked Language Modelling. However, since this research, the intention is to perform a downstream task such as Sentence Classification, the BERT model had to be fine-tuned accordingly.

3.HINGLISH

STACKED SVM MODELLING

The Support Vector Machines or SVMs are used for classifying instances by determining the best decision boundary that divides the instance space into two subspaces in the case of a Binary Classification.

Stacking or Stacked generalization is a concept which utilizes a hierarchical architecture in which we have various learning strategies that are ultimately grouped or combined by a combiner. We gather the outputs of each base learner (SVM +TF-IDF) , (SVM+BoW) in our hierarchy for every instance and combine them with the actual output value to create a meta instance. The learner at a higher level in the hierarchy is then trained on these meta instances which then outputs a final result.

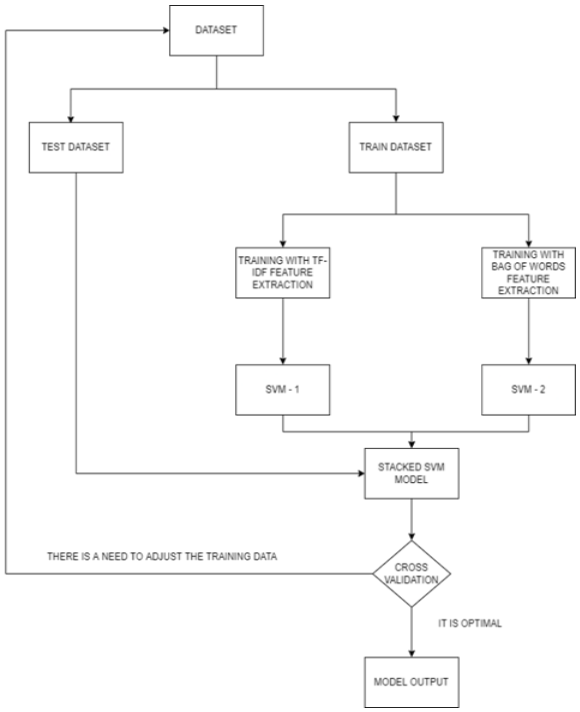

Figure 3:SVM Model

IV. EVALUATION

A. Real Time Monitoring

Twitter is a popular social media platform and has been chosen for performing evaluation on real time data for this research.

Twitter API has been utilized to stream tweets in real time and a user defined limit of the number of tweets to be streamed is set.

The tweets then undergo a series of Preprocessing steps as outlined earlier to represent them in a clean and suitable format to pass as input to the trained model. For the sake of convenience and testing purposes , the LSTM model that has been trained on English data was deployed in the backend using the Python based Flask framework ,since majority of the tweets we encounter were in English as it is the universal and the most convenient language.

The Classification results were then represented in a tabular format displaying the information about the tweet streamed along with its corresponding label that has been predicted by the model.

B. Training Details and Fine-Tuning SVM

The SVM learners need to be finetuned in order to find the best and optimal hyperparameter values that the classifier model should be fitted with. The technique Grid Search Cross Validation has been employed in order to do so by keeping the number of cross validation splits of the training data to be 10. The ‘C’ hyperparameter plays a crucial role in deciding the penalty parameter for the error term .The Grid Search algorithm is iterated for multiple pre-defined values of the ‘C’ hyperparameter and the most optimal value is then chosen.

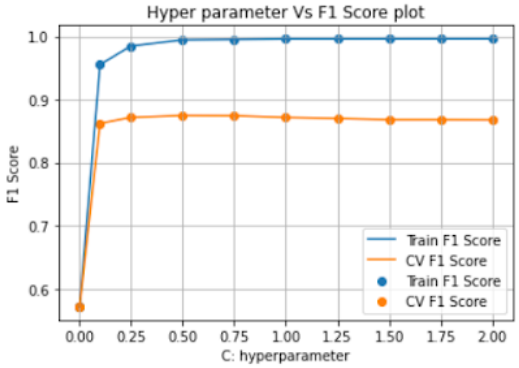

Figure 4:Hyper parameter Vs F1 Score plot

C. Discussion of Results

The results of all the experiments have been tabulated with the help of F1 Scores that were obtained using the Testing data of each task respectively. Both the LSTM + GloVe and the CNN + GloVe architectures gave similar results when tested for English data. The pre-trained multilingual BERT model showed an

improvement for Hindi data when compared to the CNN + FastText model. The SVM architecture gave optimal results for Hinglish data which was in accordance to the results obtained in [1] .The work conducted in this research significantly outperforms many other approaches that deal with code-mixed data. In [2] , the most optimal F1 score obtained to detect racism using a dictionary-based approach was only 50% and in [3] , Support Vector Machine and Random Forest Classifiers are used to detect hate speech and the maximum accuracies obtained are 69% and 71% respectively.

The results of the experiments are summarized as follows –

Table 2:Summary of Results

LANGUAGE	CLASSIFICATION	MODEL	TEST F1
ENGLISH	BINARY	LSTM+ GloVe	86%
ENGLISH	BINARY	CNN+ GloVe	86%
ENGLISH	MULTI	LSTM+ GloVe	79%
ENGLISH	MULTI	CNN+ GloVe	80%
HINDI	BINARY	CNN+ FastText	75%
HINDI	BINARY	M-BERT	78%
HINDI	MULTI	CNN+ FastText	72%
HINDI	MULTI	M-BERT	75%
HINGLISH	MULTI	SVM	88%

V. CONCLUSION AND FUTURE WORK

In conclusion, we have experimented with various model architectures in order to train our models either for a binary class labelling or a multiclass labelling. Among these models, Multilingual BERT and the SVM architecture have performed considerably well than the others.

Leveraging the State of the Art pre-trained models gave better results.

The use of the stacking concept along with the finetuning of the SVMs has ensured that the model is trained using the best hyperparameters without being prone to any overfitting and also would give the most optimal results.

For further analysis and experimentation, Deep Neural Networks along with cross-lingual embeddings for the Hinglish Code-Mixed data could be built and tested as well and a comparison of its performance with the existing baseline SVM could be recorded.

It can be inferred from [1] that Random Forest Classifiers are not always the most suitable when compared with SVMs, especially for scenarios where the size of the dataset is not too huge. This is another aspect that can be considered for further experimentation. Other popular Neural Network architectures like the GRUs or Bidirectional LSTMs have not been considered in this research . Our research does not take into account , the abundance of spelling mistakes and an improper vocabulary that could result when dealing with Hinglish data . However , this major problem could be tackled by generating Embeddings that capture information at the Character level rather than the word level. N-grams could also be used as an alternative to BoW for extracting the language features. This might however result in an increase in the computation and lead to a bottleneck.

REFERENCES

[1] Mathur, Puneet, et al. "Did you offend me? classification of offensive tweets in hinglish language." *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. 2018.

[2] Tulkens, Stéphan, et al. "A dictionary-based approach to racism detection in dutch social media." *arXiv preprint arXiv:1608.08738* (2016).

[3] Bohra, Aditya, et al. "A dataset of hindi-english code-mixed social media text for hate speech detection." *Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media*. 2018.

[4] MacAvaney, Sean, et al. "Hate speech detection: Challenges and solutions." *PloS one* 14.8 (2019): e0221152.

[5] Rani, Priya, et al. "A comparative study of different state-of-the-art hate speech detection methods in Hindi-English code-mixed data." *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*. 2020.

[6] Ghauth, Khairil Imran, and Muhammad Shurazi Sukhur. "Text censoring system for filtering malicious content using approximate string matching and Bayesian filtering." *Computational Intelligence in Information Systems*. Springer, Cham, 2015. 149-158.