

A PROJECT REPORT
ON
**MULTI TASK LEARNING
FOR DRAVIDIAN LANGUAGES**

BY

Anmol Kumar - 19BCS011

Bipul Gautam - 19BCS023

Sejal Rai - 19BCS098

Aman Kumar - 19BCS115

UNDER THE SUPERVISION OF

Dr. Sunil Saumya

Assistant Professor, Computer Science and Engineering



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY




Acknowledgement :

We would like to express our profound gratitude to Dr. Sunil Saumya (Assistant Professor, Computer Science and Engineering department) for his contributions to the completion of our project titled Multi Task Learning for Dravidian Languages. We would also like to express our special thanks to Mr. Shankar Biradar (PhD Scholar, Computer Science and Engineering department) for his time and efforts provided throughout the semester. Their useful advice and suggestions were really helpful to us during the project's completion. In this aspect, we are eternally grateful to them. This project helped us in doing a lot of Research and we came to know about so many new things. We are really thankful to them.

Contents

1. Introduction.....	[04]
2 Problem Statement.....	[05]
3. Motivation.....	[06]
4. Related Works.....	[07]
4.1 Sentiment Analysis	
4.2 MultiTask Learning	
5. DataSet.....	[08]
6. Methodology.....	[10]
7. Data Cleaning.....	[10]
8. Data Preprocessing.....	[11]
8.1 Tokenization	
8.2 iNLTK	
8.3 Removing Stopwords	
9. Embedding.....	[13]
9.1 Using iNLTK	
9.2 Using MBert	
10. Building the model.....	[14]
10.1 BiLSTM model with iNLTK embedding	
10.2 SVM model with iNLTK embedding	
10.3 SVM model with MBERT embedding	



10.4 Random Forest Classifier model with MBERT embedding

11. Problem faced and Solution.....	[17]
12. Constraints.....	[18]
13. Result.....	[19]
14. Conclusion.....	[20]
15. Future Plans.....	[21]
15. References.....	[22]



INTRODUCTION :

Social media has become an important part of our daily lives. A majority of millennials feel empty when they are not connected to social media networks. We acquire all types of information through social media, as well as entertainment that keeps us interested. Twitter, Instagram, Facebook, Whatsapp, YouTube, and other social media platforms have become a way to socialize and keep up with current events and trends.

This affects our mental health in a vulnerable way. Likes, comments, shares, etc become so important sometimes that it decides the state we are in. Reading negative comments can make you feel unwanted and depressed. Once you start taking the comment section seriously every comment affects your personality. Negative comments impact your daily routine and the person tries to find negativity in everything around him. If this continues for a longer period of time , there are chances of people gradually slipping into depression.

This project is an analysis of such comments collected from social media platforms such as youtube in the Dravidian languages (kannada , tamil , malayalam) and analyzing the sentiment of the comments and the type of the sentiment of those comments.



PROBLEM STATEMENT :

The goal of this project is to perform Sentiment Analysis and identify offensiveness in code-mixed social media comments (youtube) for Dravidian languages (Kannada, Malayalam and Tamil).



MOTIVATION :

This project works on code-mixed social-media comments for Kannada language. There are plenty of hate and offensive comments on social media and across the internet in different languages. An appreciable amount of work has been done in mainstream languages like English to curb the threat. Companies like Google, Facebook and Amazon are doing a lot for major languages like English. Not much work has been done in the Dravidian languages, e.g, Kannada, Tamil, Malayalam, etc.

What is amusing is that the population of Karnataka alone is almost equal to the population of the entire France. Yet not much work has been done in the Dravidian languages. This is a matter of concern as the internet has penetrated to distant lands and with affordable rates, everyone is able to get it, so there is a large chunk of people using social media. The offensive and hateful comments are also the reason for violent acts, e.g, riots. So, curbing them is a necessity.



RELATED WORKS :

In order to proceed with the project we have worked and researched on few of the tools techniques algorithms and related work

SENTIMENT ANALYSIS :

Sentiment Analysis is a type of text mining that finds and extracts subjective information from sources, allowing a firm to better understand the social sentiment of its brand, product, or service while monitoring online conversations. Most social media stream analysis, on the other hand, is restricted to rudimentary sentiment analysis and count-based metrics. Sentiment Analysis reviews can be your only source of truth for comprehending what the general public is trying to convey to you, thus it's important to analyze it. Because of its widespread use, there has been a plethora of research in various languages. The Dravidian languages are not in the same boat. As previously stated, there is a significant lack of data in the field to undertake experiments on code-mixed data in Dravidian languages.

MULTITASK LEARNING :

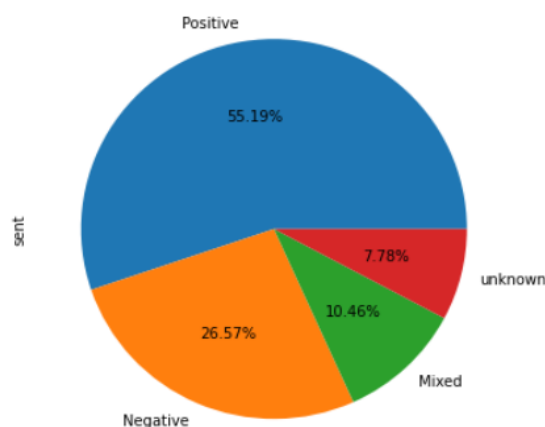
The basic purpose of the MTL model is to improve a model's learning for a single task by adding knowledge from other tasks, whether all or a subset of tasks are connected. MTL is based on the idea that tackling multiple tasks at once results in a shared inductive bias, resulting in a more robust and generalizable system. Machine learning researchers have looked into it extensively. In the NLP sector, it has neural network applications. To the best of our knowledge, no MTL models for Dravidian languages have yet been built.

DATASET :

We make use of the multilingual dataset, consisting of over **60,000** manually annotated YouTube comments. The data set comprises sentences from 3 code-mixed Dravidian languages: **Kannada** , **Malayalam** , and **Tamil** . The code-mixed Kannada dataset consists of **4265** comments, while the corresponding Malayalam and Tamil codemixed datasets consist of **12,711** and **43,349** comments, respectively. There are three columns indicating text , sent and off . The text column consists of the comment in the respective language script .

The sent column consists of four classes :

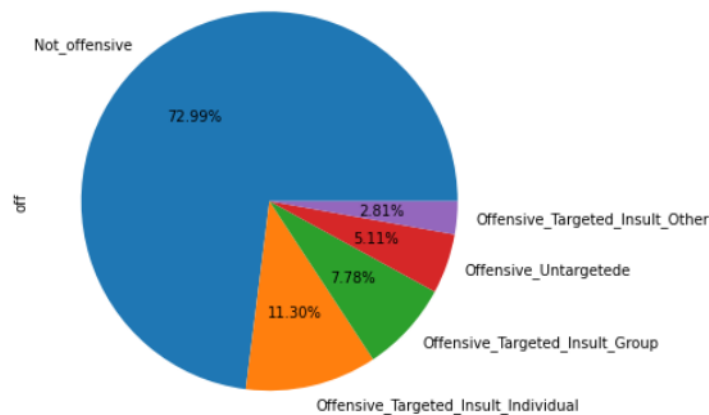
- **Positive state:** comment in the text contains a clue suggesting it is in a positive state. The count is 2354
- **Negative state:** comment in the text contains a clue suggesting it is in a negative state. The count is 1133
- **Mixed :** Comment contains a clue in both positive and negative feeling. The count is 446
- **Unknown:** Comment does not contain any clue such that the emotion or sentiment can be recognized. The count is 332



Above are the pictorial representations of the sent column's data of the kannada dataset

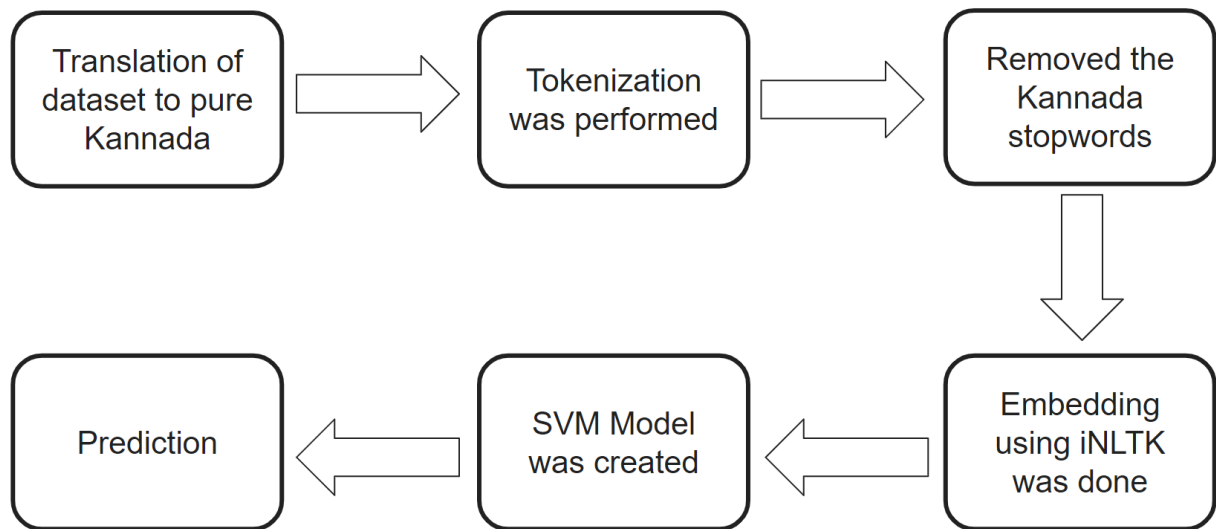
The off column consists of the following labels :

- **Not offensive:** Comment does not contain offense or profanity.
- **Offensive untargeted :** Comment contains offense or profanity without any target.
- **Offensive targeted individual:** Comment contains offense or profanity that targets the individual.
- **Offensive targeted group:** Comment contains offense or profanity that targets the group.
- **Offensive targeted other:** Comment contains offense or profanity that does not belong to any of the previous two categories (e.g., a situation, an issue, an organization, or an event).



Above are the pictorial representations of the off column's data of the kannada dataset

METHODOLOGY :



The workflow we have followed for the sentiment analysis are data collection, data cleaning , data preprocessing which included translation, stemming, tokenization and embedding, setting up the suitable model , training the model and predicting the results .

DATA CLEANING :

If we take the kannada dataset , the comment column had text in kannada script and also a mixture of numbers , english text and other languages such as turkish was also noticed . So at first we performed the cleaning on the dataset by converting the text into pure kannada . we did this by first converting the comments into english using **googleTrans** utility of ms excel and then converted the whole text back to kannada using the same . By this most of the irrelevant words were translated into kannada .


=GOOGLETRANSLATE(cell with text, “source language”, “target language”)

DATA PREPROCESSING:

After cleaning , preprocessing the data is one of the most crucial steps as this is the stage where we transform our data in a format which is acceptable by the model and would be trained smoothly .The steps we have performed are :

1. TOKENIZATION :

Tokenization is breaking the raw text into small chunks. Tokenization breaks the raw text into words, sentences called tokens. These tokens help in understanding the context or developing the model for the NLP. The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words. For example, the text “It is raining” can be tokenized into ‘It’, ‘is’, ‘raining’

For tokenizing the comment column of our dataset we have used the **iNLTK** Library.

iNLTK :

iNLTK is an open-source Deep Learning library built on top of PyTorch aiming to provide out of the box support for Data Augmentation, Textual Similarity, Sentence Embeddings, Word Embeddings, Tokenization and Text Generation in 13 Indic Languages.

2. REMOVING STOPWORDS :

Stopwords are words that contribute little meaning to a statement in any language. They can be safely ignored without compromising the sentence's meaning. Stop words should be removed from text classification or sentiment analysis tasks since they contribute no information to our model, i.e. keeping undesired words out of our corpus; however, stopwords are useful in language translation tasks because they must be translated with other words. We collected the list of Kannada stopwords from the internet and then we removed the matching stopwords from our translated dataset using INLTK.

```
stopwords={'ಮತ್ತು', 'ಈ', 'ಒಂದು', 'ರಲ್ಲಿ', 'ಹಾಗೂ', 'ಎಂದು', 'ಅಥವಾ', 'ಇದು', 'ರ', 'ಅವರು', 'ಎಂಬ', 'ಮೇಲೆ', 'ಅವರ', 'ತನ್ನ', 'ಆದರೆ', 'ತಮ್ಮ', 'ನಂತರ', 'ಮೂಲಕ', 'ಹೆಚ್ಚು', 'ನ', 'ಆ', 'ಕೆಲವು', 'ಅನೇಕ', 'ಎರಡು', 'ಹಾಗು', 'ಪ್ರಮುಖ', 'ಇದನ್ನು', 'ಇದರ', 'ಸುಮಾರು', 'ಅದರ', 'ಅದು', 'ಮೊದಲ', 'ಬಗ್ಗೆ', 'ನಲ್ಲಿ', 'ರಂದು', 'ಇತರ', 'ಅತ್ಯಂತ', 'ಹೆಚ್ಚಿನ', 'ಸಹ', 'ಸಾಮಾನ್ಯವಾಗಿ', 'ನೇ', 'ಹಲವಾರು', 'ಹೊಸ', 'ದಿ', 'ಕಡಿಮೆ', 'ಯಾವುದೇ', 'ಹೊಂದಿದೆ', 'ದೊಡ್ಡ', 'ಅನ್ನು', 'ಇವರು', 'ಪ್ರಕಾರ', 'ಇದೆ', 'ಮಾತ್ರ', 'ಕೂಡ', 'ಇಲ್ಲಿ', 'ಎಲ್ಲಾ', 'ವಿವಿಧ', 'ಅದನ್ನು', 'ಹಲವು', 'ರಿಂದ', 'ದ', 'ದಕ್ಷಿಣ', 'ಗೆ', 'ಅವನ', 'ಅತಿ', 'ನೆಯ', 'ಬಹಳ', 'ಕೆಲಸ', 'ಎಲ್ಲ', 'ಪ್ರತಿ', 'ಇತ್ಯಾದಿ', 'ಇವು', 'ಬೇರೆ', 'ಹೀಗೆ', 'ನಡುವೆ', 'ಇದಕ್ಕೆ', 'ಎಸ್', 'ಇವರ', 'ಮೊದಲು', 'ಶ್ರೀ', 'ಮಾಡುವ', 'ಇದರಲ್ಲಿ', 'ರೀತಿಯ', 'ಮಾಡಿದ', 'ಕಾಲ', 'ಅಲ್ಲಿ', 'ಮಾಡಲು', 'ಅದೇ', 'ಈಗ', 'ಅವು', 'ಗಳು', 'ಎ', 'ಎಂಬುದು', 'ಅವನು', 'ಅಂದರೆ', 'ಅವರಿಗೆ', 'ಇರುವ', 'ವಿಶೇಷ', 'ಮುಂದೆ', 'ಅವುಗಳ', 'ಮುಂತಾದ', 'ಮೂಲ', 'ಬಿ', 'ಮೀ', 'ಒಂದೇ', 'ಇನ್ನೂ', 'ಹೆಚ್ಚಾಗಿ', 'ಮಾಡಿ', 'ಅವರನ್ನು', 'ಇದೇ', 'ಯ', 'ರೀತಿಯಲ್ಲಿ', 'ಜೊತೆ', 'ಅದರಲ್ಲಿ', 'ಮಾಡಿದರು', 'ನಡೆದ', 'ಆಗ', 'ಮತ್ತೆ', 'ಪೂರ್ವ', 'ಆತ', 'ಬಂದ', 'ಯಾವ', 'ಒಟ್ಟು', 'ಇತರೆ', 'ಹಿಂದೆ', 'ಪ್ರಮಾಣದ', 'ಗಳನ್ನು', 'ಕುರಿತು', 'ಯು', 'ಆದ್ದರಿಂದ', 'ಅಲ್ಲದೆ', 'ನಗರದ', 'ಮೇಲಿನ', 'ಏಕೆಂದರೆ', 'ರಷ್ಟು', 'ಎಂಬುದನ್ನು', 'ಬಾರಿ', 'ಎಂದರೆ', 'ಹಿಂದಿನ', 'ಆದರೂ', 'ಆದ', 'ಸಂಬಂಧಿಸಿದ', 'ಮತ್ತೊಂದು', 'ಸಿ', 'ಆತನ', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', 'a'}
```

3 .EMBEDDING :

This is the last stage in our data preprocessing. The words are converted into vectors during embedding. We need scalar values or matrices of scalar

values as inputs for every machine learning or deep learning technique. The words are then converted into vectors. Each word is given a distinct value, so that each vector represents the word's context, meaning, and semantics. Words with similar meanings and context are grouped together in this fashion.

We have performed the embedding using two methods :

1 . USING INLTK:

- Using iNLTK, we have generated the embedding vectors and we have stored it in an np_array of size 4265 X 400.
- Here, for each comment using iNLTK, we have got vectors of 400 size.
- Then, we have stored embeddings in a .npy file for further use.

```
for j in range(4265):
    np. set_printoptions(threshold=np. inf)
    s=data['translated'][j]
    print(s)
    s.strip()
    res = len(s.split())
    v=get_embedding_vectors(s,'kn')
    sum_vector=[]
    sum_vector = [0 for i in range(400)]

    for i in range(0,res):
        sum_vector=sum_vector+v[i]
    for i in range(0,400):
        sum_vector[i]=sum_vector[i]/res
    emb[j]=sum_vector
```

2. USING MBERT :

- Using mBert and providing padding, we have generated the embedding vectors and we have stored it in an np_array of size 4265 X 768.
- Here, for each comment using mBert, we have got vectors of 768 size.
- Then, we have stored embeddings in a .npy file for further use.

```
from transformers import BertTokenizer, TFBertModel
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
model = TFBertModel.from_pretrained("bert-base-multilingual-cased")
tokenized = data['translated'].apply((lambda x: tokenizer.encode(x, add_special_tokens=True,max_length=40,truncation=True)))
```

```
max_len = 0
for i in tokenized.values:
    #print(len(i))
    if len(i) > max_len:
        max_len = len(i)
padded = np.array([i + [0]*(max_len-len(i)) for i in tokenized.values])
```

```
output = model(padded)
```

BUILDING THE MODEL :

After the data is preprocessed and embeddings are ready , we passed these embedding vectors into different models like BiLSTM , SVM , Random forest . The description of all the models is given below one by one

1. BiLSTM Model with iNLTK embedding :

Bidirectional LSTM (BiLSTM) model maintains two separate states for forward and backward inputs that are generated by two different LSTMs. The first LSTM is a regular sequence that starts from the beginning of the sentence, while in the second LSTM, the input sequence is fed in the opposite order. The idea behind bi-directional networks is to capture information of surrounding inputs. It usually learns faster than a one-directional approach although it depends on the task.

We passed on the embeddings to this BiLSTM Model which has two Bidirectional layers and 2 dense layers . and we received an accuracy of **51%**

```
model = Sequential()
model.add(Bidirectional(LSTM(100, input_shape=(2985,400), return_sequences=True)))
model.add(Dropout(0.3))
model.add(Bidirectional(LSTM(20)))
model.add(Dropout(0.2))
model.add(Dense(20, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

2. SVM MODEL WITH INLTK EMBEDDING :

We have created a SVM model using **OneVsRestClassifier**.

One-vs-rest is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.

Here after passing the embedding to this model we got an accuracy of **48%**

Matrix for 'sent' label:

Accuracy: 0.478125				
	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.74	0.70	0.72	752
2	0.31	0.50	0.38	197
3	0.00	0.00	0.00	0
micro avg	0.49	0.65	0.56	949
macro avg	0.26	0.30	0.28	949
weighted avg	0.65	0.65	0.65	949
samples avg	0.49	0.48	0.48	949

Matrix for 'off' label:

Accuracy: 0.7140625				
	precision	recall	f1-score	support
0	0.92	0.79	0.85	1101
1	0.33	0.77	0.46	60
2	0.00	0.00	0.00	0
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
micro avg	0.72	0.79	0.75	1161
macro avg	0.25	0.31	0.26	1161
weighted avg	0.89	0.79	0.83	1161
samples avg	0.72	0.72	0.72	1161

3. SVM Model with MBERT Embeddings :

For the above model we have passed the embedding received using INLTK , In this model we have passed the Mbert Embeddings and received an

accuracy of -

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn import svm
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svc = svm.SVC()
clf = OneVsRestClassifier(GridSearchCV(svc, parameters,cv=10))
clf.fit(X_features,X_labels)
```

Matrix for 'sent' label:

Accuracy: 0.5296875

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.85	0.63	0.73	946
2	0.29	0.58	0.38	154
3	0.00	0.00	0.00	0
micro avg	0.54	0.63	0.58	1100
macro avg	0.28	0.30	0.28	1100
weighted avg	0.77	0.63	0.68	1100
samples avg	0.54	0.53	0.54	1100

Matrix for 'off' label:

Accuracy: 0.7265625

	precision	recall	f1-score	support
0	0.95	0.79	0.86	1135
1	0.27	0.84	0.41	45
2	0.00	0.00	0.00	0
3	0.00	0.00	0.00	0
4	0.00	0.00	0.00	0
micro avg	0.73	0.79	0.76	1180
macro avg	0.24	0.33	0.25	1180
weighted avg	0.92	0.79	0.84	1180
samples avg	0.73	0.73	0.73	1180

4. Random Forest Classifier using MBERT Embeddings :

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. It is one of the famous bagging techniques of ensemble learning .As the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.

We have passed on the MBERT Embedding to this classifier and got the accuracy of 45% when we took n_estimators as 50 and the accuracy remained the same when the value was changed to 5000.

```
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 42)
classifier.fit(x_train, Y_train)
```

Matrix for 'sent' label :

```

Accuracy: 0.45390625
      precision    recall  f1-score   support

     0       0.00      0.00      0.00         1
     1       0.77      0.64      0.70        854
     2       0.11      0.60      0.19         60
     3       0.01      0.33      0.01          3

 micro avg       0.45      0.63      0.53        918
 macro avg       0.22      0.39      0.23        918
weighted avg       0.72      0.63      0.66        918
 samples avg       0.45      0.45      0.45        918

```

Matrix for 'off' label:

```

Accuracy: 0.72578125
      precision    recall  f1-score   support

     0       0.96      0.76      0.85       1187
     1       0.14      1.00      0.25         20
     2       0.01      0.50      0.02          2
     3       0.00      0.00      0.00          0
     4       0.00      0.00      0.00          1

 micro avg       0.73      0.77      0.75       1210
 macro avg       0.22      0.45      0.23       1210
weighted avg       0.95      0.77      0.84       1210
 samples avg       0.73      0.73      0.73       1210

```

PROBLEMS FACED AND SOLUTIONS:

P. Our dataset was not primarily only in Kannada but mixed.

S. So we translated the comments to Kannada.



P. While using the Google Trans API to translate the comments into Kannada, there was a limit to how many translations the API endpoint could make.

S. So the entire dataset was not getting translated. We used MS Excel to overcome this problem using the formula.

P. While getting the embeddings using iNLTK, the RAM was getting exhausted.

S. So we divided the dataset into batches of 250 data points and ran it for 20 times to get the embeddings. Simultaneously, we stored the embeddings in the dataset

P. While extracting the embedding we were unable to do it because embeddings were saved in string format.

S. We saved it in a numpy array and stored it in .npy file.

P. We tried to implement BiLSTM model, due to inconsistency of labels in the dataset, the model was classifying only one class in both the labels, i.e, sent and off.

S. Therefore, we tried the SVM model to classify.

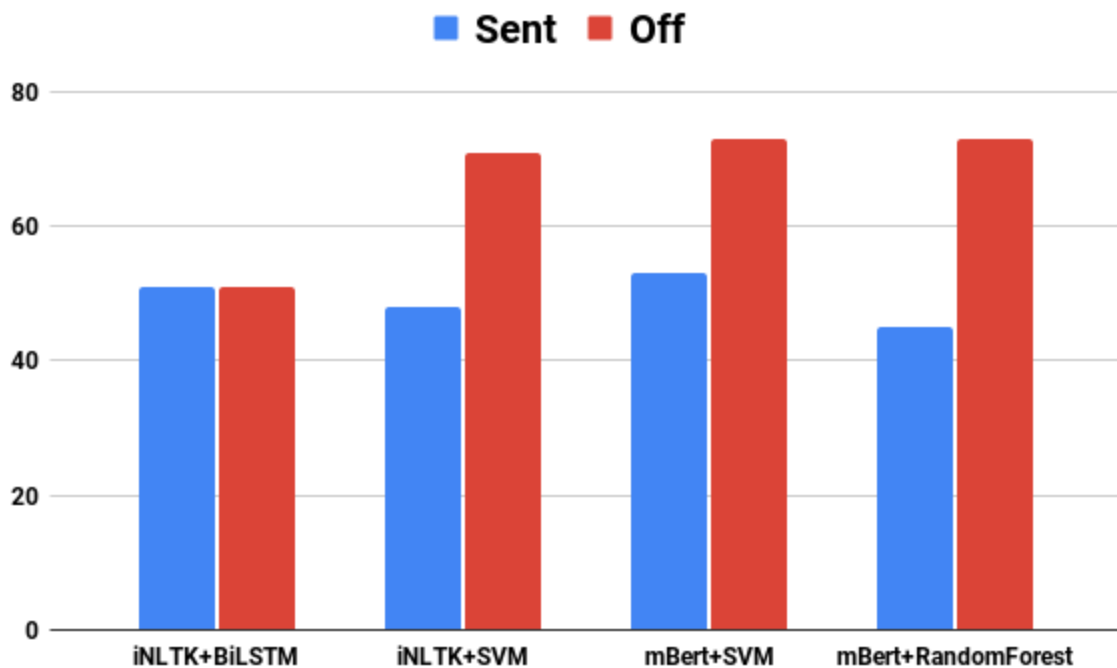


CONSTRAINTS :

Due to unavailability of computing resources (GPU and RAM), we could only perform it for Kannada language. Tamil dataset had around 30000 data points whereas Malayalam had around 9000 data points. So CPU and RAM were getting exhausted while generating the embeddings.

RESULT :

We have successfully implemented all the four models and predicted the outputs for both sent column and off column and we have received a nearby accuracy count for all the models .Below is the pictorial and tabular representation of the accuracy of the models.



Accuracy

	INLTK+BILSTM	INLTK+SVM	MBERT+SVM	MBERT+RANDOM FOREST
SENT	51%	47.8%	52.9%	45.4%
OFF	51%	71.4%	72.7%	72.6%

CONCLUSION:

Numerous advanced methodologies and approaches for language processing tasks are examined in depth in this work. Extensive reviews of language processing are offered, with a focus on Offensive language detection. The building elements for numerous natural language processing tasks, such as tokenization, transliteration, embedding and so on are covered. Two types of famous embedding techniques are also covered namely embedding using INLTK and m-BERT which is the crucial part of this project .Training of these embedded vectors are performed on few of the best models in terms of accuracy like the random forest which is well known to prevent the overfitting of the data by reducing the variance , SVM which is also a famous classification algorithm and provides best accuracy an takes less time for training. SVM works well with a clear margin of separation and with high dimensional space. The other one is BiLSTM which also provides best predictions due to two LSTM models for training frontwards and backwards .They provide us with a large range of parameters such as learning rates, and input and output bias. We finally succeeded in building the models and predicting the outputs where in all the models gave nearby accuracy .



FUTURE PLANS :

- We will improve the consistency of the data set to improve the model performance using Augmentation.
- We will be replicating the model for Tamil and Malayalam languages.
- We will integrate the model into a web application.
- We will also make it available as an API so that others are able to use it.



REFERENCES :

1. iNLTK Documentation:
[-https://inltk.readthedocs.io/en/latest/api_docs.html](https://inltk.readthedocs.io/en/latest/api_docs.html)
2. m-Bert:
[-https://huggingface.co/bert-base-multilingual-cased](https://huggingface.co/bert-base-multilingual-cased)
3. BiLSTM:
[-https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/](https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/)
4. SVM:
[-https://scikit-learn.org/stable/modules/svm.html](https://scikit-learn.org/stable/modules/svm.html)
5. Random Forest Classifier:
[-https://www.codementor.io/@agarrahu101/multiclass-classification-using-random-forest-on-scikit-learn-library-hkk4lwawu](https://www.codementor.io/@agarrahu101/multiclass-classification-using-random-forest-on-scikit-learn-library-hkk4lwawu)