KL-NF TECHNIQUE FOR FINDING OFFENSIVENESS IN A SENTENCES

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Technology

by

KADIRI MOHAN KUMAR	*411935*
ANSHU KUMAR	*411905*
KASTHALA JAYADEV	*411939*

Supervisor:

Mr. D. Prasad

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH

May - 2023

APPROVAL SHEET

This project work entitled "KL-NF Technique for finding Offensiveness in a sentence" worked out by Kadiri Mohan Kumar (411935), Anshu Kumar (411905), and Kasthala Jayadev (411939) is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor (s)

_____

_____

_____

Chairman

_____

Date: _____

Place: NIT Andhra Pradesh

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Kadiri Mohan Kumar	Anshu Kumar	Kasthala Jayadev
Roll No: 411935	Roll No: 411905	Roll No: 411939
Date: 04-05-2023	Date: 04-05-2023	Date: 04-05-2023

CERTIFICATE

It is certified that the work contained in the thesis titled "**KL-NF Technique for finding offensive in a sentence**," by "Kadiri Mohan Kumar, bearing Roll No: 411935, Anshu Kumar, bearing Roll No: 411905, and Kasthala Jayadev, bearing Roll No: 411939" has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Mr. D. Prasad

Dept. of Computer Science and Engineering

N.I.T. Andhra Pradesh

May,2023

ACKNOWLEDGEMENT

The success and outcome of this project required a lot of guidance and assistance from many people, and we are privileged to have got this all along with the completion of our project. Everything we have done is because of such guidance and help, and we will never hesitate to thank them.

We owe our sincere gratitude to our project guide Mr. D. Prasad, Department of Computer Science, National Institute of Technology, Andhra Pradesh, who took keen interest and guided us all along, till the completion of our project work by providing all the necessary information.

We are grateful and lucky enough to receive consistent motivation, support, and guidance from Head of the Department and all the staff of the Computer Science Department. They have helped us to complete our project work successfully. We would also like to extend our sincere gratitude to all my friends for their timely support.

Kadiri Mohan Kumar	Anshu Kumar	Jayadev Kasthala
Date: 04-05-2023	Date: 04-05-2023	Date: 04-05-2023

ABSTRACT

It is human behaviour to look for others opinion before taking any decision. A lot of documents are available which express opinions on different issues. But the main challenge arises in analysing these documents to produce useful knowledge. Tremendous works in the area of Classification is available for English language. However, there has been little work done for Indian languages. From the last few years, opinion-rich resources are booming in Kannada, Tamil, and Malayalam and hence there is a need to perform classification in Hindi. Classification of movie reviews could help in better rating of movies. This work proposes classification for low-resource languages like Tamil, Kannada, and Malayalam using Neuro-Fuzzy Technique. This report describes the design and implementation of the algorithm. Classification is a field of Natural Language Processing task which deals with classifying subjective information from the e-resources like blogs, reviews, forums etc. in the form text, image or, video to analyze the class or type of that resource.

The purpose is to develop classification model for low-resource language like Tamil, Kannada, Malayalam with use KL-NF and BERT for finding offensiveness in a sentence. We developed language independent model that can work for any language and for small and large datasets. We used datasets of 3 languages to train and test the model. After training the model, we have tested the model and compared the results of both KL-NF and BERT. The comparison among the accuracy shows that BERT model beat KL-NF in multi-lingual classification of offensiveness. The accuracy for BERT model is 82.7% whereas for KL-NF model is 82.5%. This concludes that BERT model can be used for multi-lingual classification to predict more accurately than the KL-NF.

KEYWORDS

Sentiment Analysis, Neuro-fuzzy, KL divergence, Neural Network, Tokenization, Fuzzy Value, Tokens, Attention.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

1. INTRODUCTION

Linguistic resources-based classification is highly dependent on language resources which became a challenge for low-resource languages like Tamil, Malayalam, and Kannada. Low-resource languages suffer from the scarcity of resources; consequently, evolution a method that can be implemented for any language. Classification is a field of Natural Language Processing task which deals with subjective information from the e-resources like blogs, reviews, forums etc. in the form text, image or, video to classify them into classes. The major application areas of classification are in Market research, Healthcare Monitoring, E-Governance, Brand Monitoring [13, 14, 15].

Due to the availability of extensive web-based data in Indian languages such as Kannada, Malayalam, Tamil, and so on. It has become extremely significant to analyze this data and recover valuable and relevant information. Tamil, Kannada, and Malayalam are most spoken languages in southern India, Classification tasks in these languages has turned out to be a critical task particularly for companies and government organizations.

Classification is a metric that conveys the type or class of text or data is. It is performed on textual data to help businesses monitor brand and understand customer needs. It is a time-efficient, cost-friendly solution to analyze huge data. Bad reviews can snowball online, and the longer you leave them the worse the situation will Be, you will be notified about negative brand mentions immediately. E-Governance: It helps is measuring public opinion on government policies. Many developed countries are already using it. Healthcare Monitoring: This may help in monitoring happiness index of the people, their opinions towards some treatment and their feelings.

It helps to determine the reviewer's point of view on a particular topic. It combines the techniques of computational linguistics and Information Retrieval (IR). The increasing user-generated content on the Internet is the motivation behind the classification research Classification is used every day in social media, surveys, feedbacks to identify the needs of the people.

So, there is need of feature extraction technique that is not domain specific and classification technique that can work well even with smaller datasets. Very few research work has been done related to classification in low resource language [9]. Many deep learning algorithms have proved their worth, but that requires big data, which is unavailable in Tamil, Kannada,

and Malayalam. So, need of algorithm which works well with both small datasets and large datasets.

In our project, we perform Classification of offensive sentences on multilingual dataset with the help of sentimental Analysis Algorithm. Classification of offensive sentences is a text classification problem in which a text is assigned as offensive, not-offensive, offensive targeted insult single, offensive targeted insult other, offensive targeted insult individual, depending on the offensiveness which it strongly forces. We have used various methods to perform classification. We have developed classification model for low-resource language with use of feature extraction method such as KL-NF and BERT for sentiment classification.

2. LITERATURE REVIEW

An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation by Bijoyan Das, Sarit Chakraborty [4]. In this paper, they have conducted experiments on three datasets, IMDB movie reviews [1], Amazon product reviews [2] and SMS spam detection dataset [3]. After conducting experiment with their proposed model, i.e., using NWN with TF-IDF, they got good accuracy. The accuracy percentages for IMDB review datasets, Amazon product review and SMS spam datasets came as 89.91%, 88.86%, 96.83% respectively. They have concluded that when TF-IDF model is coupled with Next Word Negation then the performance of the sentiment classifier increases by a good percentage.

Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts by Cicero Nogueira dos Santos and Maira Gatti (2015) [5]. In this work they have proposed a new deep convolutional neural network that exploits from character- to sentence-level information to perform sentiment analysis of short texts. they applied the new approach for two corpora of two different domains: the Stanford Sentiment Treebank (SSTb), which contains sentences from movie reviews; and the Stanford Twitter Sentiment corpus (STS), which contains Twitter messages. For the SSTb corpus, their approach achieved state-of-the-art results for single sentence sentiment prediction in both binary positive/negative classification, with 85.7% accuracy, and fine-grained classification, with 48.3% accuracy. For the STS corpus, their approach achieved a sentiment prediction accuracy of 86.4%.

Recurrent Neural Network for Text Classification with Multi-Task Learning by Pengfei Liu, Songfang Huang, and Xipeng Qiu (2016) [6]. This paper presents a recurrent neural network (RNN) model for sentiment classification that uses multi-task learning to leverage related tasks, such as topic classification, to improve sentiment classification performance. The authors show that their model achieves state-of-the-art performance on several benchmark datasets.

Sentimental Analysis Using Product Review Data by Xing_Fang Justin_Zhan [7]. In this paper, Fang and Zhan proposed a sentiment analysis method that utilizes a combination of sentiment lexicons, machine learning algorithms, and feature engineering techniques to classify the sentiment of product reviews. They evaluated their method on three datasets consisting of product reviews from Amazon.com, and report high accuracy and F1 scores for sentiment classification. The authors also conduct experiments to compare their method with other existing sentiment analysis approaches, such as Support Vector Machines (SVMs) and Naive

Bayes (NB) classifiers. Their results show that their proposed method outperforms these baseline methods in terms of accuracy and F1 score.

BERT for Sentiment Analysis of Movie Reviews by Devlin et al. (2018) [8]. This paper proposes a fine-tuning approach for sentiment analysis using BERT on the IMDB movie reviews dataset. The authors show that their BERT-based model outperforms previous state-of-the-art models on this dataset.

3. SYSTEM DESIGN

3.1 EXISTING SYSTEM

There are many classification techniques in the field of computer science which works with large datasets and are not language independent. Most of the researches have been done on English datasets since English language has more resources which can be used for training models. Thus, due to the reason of rich resources for English language, everyone is willing to work with English dataset and building a model which works especially for English and these cannot be used for other language datasets because these models need a plenty of training data which is not found in the other languages [10].

Low resource languages like Tamil, Kannada, and Malayalam cannot used to train a model with normal traditional techniques that are used for English, since those techniques only work when there is a lot of training data. So, due the fact of scarcity in resources, all the existing classification techniques will fail to train a model with small datasets. And, it is impossible to collect large datasets for these languages where few researches are happening. So, existing models have disadvantages when working with small datasets.

3.2 PROPOSED SYSTEM

Classification task in Indian languages with traditional classification technique is very challenging because these languages are a resource scarce language which causes problems in collection and generation of datasets. Also, there are not efficient parsers and taggers for these languages. Indian Languages like Malayalam, Tamil, and Kannada are morphologically rich and a free order language as compared to English language. Limited resources are available for these languages. Thus, we used a technique that can work for any language and for even small datasets.

Our task is finding offensiveness in a sentence for low-resource languages like Tamil, Malayalam, and Kannada. This task has been performed using Neuro Fuzzy Technique by developing neural network with KL-Divergence. We developed language independent model for multilingual dataset using different feature extraction techniques including TF-IDF and BERT. Further, the comparison among accuracy obtained from these models have been done.

3.2.1 SYSTEM ARCHITECTURE

The system architecture of BERT (Bidirectional Encoder Representations from Transformers) [11] consists of two main components: the pre-training phase and the fine-tuning phase.

1. Pre-training phase: During the pre-training phase, BERT is trained on a large corpus of unlabeled text data. The goal of pre-training is to learn general language representations that can be used as a starting point for a variety of downstream natural language processing tasks, including sentiment analysis. BERT uses a multi-layer bidirectional transformer encoder architecture for pre-training. The model is trained on a masked language modeling (MLM) task, where some of the tokens in the input sequence are masked, and the model is trained to predict the masked tokens based on the surrounding context. Additionally, BERT also uses a next sentence prediction (NSP) task, where the model is trained to predict whether two input sentences are contiguous in the original text or not.

2. Fine-tuning phase: During the fine-tuning phase, the pre-trained BERT model is fine-tuned on a specific downstream task, such as sentiment analysis. Fine-tuning involves adding a classification layer on top of the pre-trained BERT model and training the entire model on a labeled dataset specific to the downstream task. The fine-tuning process for BERT involves feeding the input text sequence through the pre-trained BERT model, which generates a contextualized representation for each token in the sequence. These representations are then fed through the classification layer to generate a prediction for the target task.

Overall, BERT's system architecture combines pre-training on a large corpus of text data with fine-tuning on specific downstream tasks to achieve state-of-the-art performance on a variety of natural language processing tasks, including sentiment analysis.

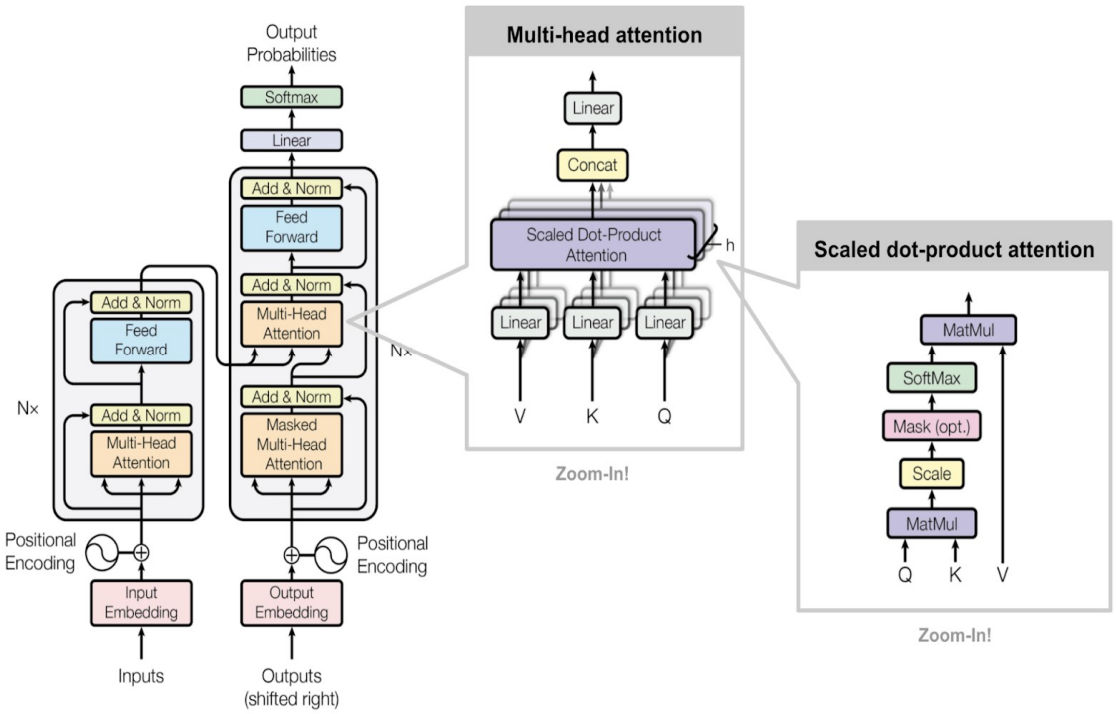

Figure 3.1 BERT Architecture

Neuro-fuzzy logic is a type of machine learning algorithm that combines the strengths of neural networks and fuzzy logic. The system architecture for neuro-fuzzy logic has following layers:

1. Input layer: The input layer is where the system receives input data. The input data can be either crisp (precise) or fuzzy (inexact) and can consist of one or more variables.

2. Fuzzification layer: In the fuzzification layer, the input data is transformed from crisp to fuzzy values. This is done by mapping the input data onto fuzzy sets, which represent a range of possible values for each input variable.

3. Inference layer: In the inference layer, fuzzy logic rules are applied to the fuzzy inputs to determine the output. These rules consist of if-then statements that define the relationship between the input and output variables.

4. Defuzzification layer: In the defuzzification layer, the fuzzy output is transformed back into a crisp value. This is done by calculating the center of gravity of the output fuzzy set, which represents the most likely output value based on the input data and the fuzzy logic rules.

5. Learning layer: In some neuro-fuzzy logic systems, a learning layer is added to the architecture. This layer is used to adapt the fuzzy logic rules based on training data. This allows the system to improve its accuracy over time and adapt to changing input patterns.

Overall, the neuro-fuzzy logic architecture combines the non-linear modeling capability of neural networks with the inexact reasoning of fuzzy logic to create a flexible and adaptable machine learning system. The system can handle both crisp and fuzzy inputs, and can be trained to improve its accuracy over time.

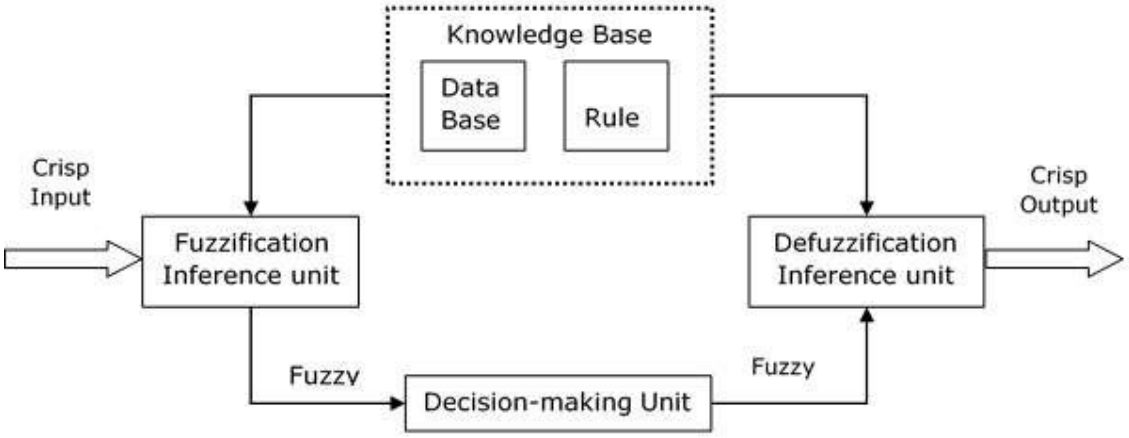

Figure 3.2 Neuro Fuzzy Logic Architecture

3.2.2 MODULES

The following are the modules in the system:

1. Data Importing module: The role of this module is to read the data from the csv files and store into the DataFrame.

2. Preprocessing module: This module cleans the data before passing that data to next module by removing stop words, punctuations, alpha numeric words, emojis, etc.

3. Configuration module: Configuration module provides all the tools to initialize the model and configure with our own architecture.

4. Training module: This model takes care of training the model with the preprocessed data and learn from mistakes by back-propagation.

5. Testing module: This module tests the model with testing data and shows the accuracy, F1 score and other metrics of the model.

4. SYSTEM IMPLEMENTATION

4.1 ALGORITHM

Algorithm1: Algorithm to generate fuzzy values for a word in sentence [12]

Input: Input Document containing sentences. Let |U| is the number of documents in training dataset, |C| is the number of classes of classification and 'S' be the sentence in training document with |S| as the length of the sentence

Output: KL-Fuzzy Value of each word of all sentences in the document.

Step 1: For each word in a sentence:

Compute the probability of a word x_t in a sentence

$p(x) = \frac{x_{it}}{|S|}$ # $x_{it} \in \{0,1,2\ldots\}$ indicates the number of times word x occurred in a sentence

Step 2: For each word in a sentence:

Compute $(x_t) = \sum_{j=1}^{|C|} \frac{N_{jt}}{|U|}$. # $q(x_t)$ is the average number of documents that contain word x

N_{jt} be the number of documents in class c_j that contains word x_t.

$$\widetilde{K_t}(U) = -p(x_t)\log q(x_t)$$

#$\widetilde{K_t}(U)$ is an approximation to average divergence of the word x in individual training document from all training documents

$p(x_t)$ is the probability of the word x in all the training documents (number of occurrences of x_t in all documents divided by all the words in the document)

$$\widetilde{KL}_t(U) = -\sum_{j=1}^{|C|} p(c_j)p(x_t/c_j)\log q(x_t/c_j)$$

$\widetilde{KL}_t(U)$ is an approximation to KL divergence values with assumption that the number of occurrences of word x in all documents is same that contains x

$p(c_j)$ is a prior probability, that evaluates probability of a class

$p(x_t/c_j)$ is a conditional probability for a word x_t to be in class c_j

$q(x_t/c_j)$ has been defined as $q(x_t/c_j) = \frac{N_{jt}}{|c_j|}$

Finally Compute,

$$KL_t(x_t) = \widetilde{K_t}(U) - \widetilde{KL}_t(U)\ ,$$

Step 3: The membership values are obtained

$$\mu_t^j = \frac{KL_t^j(x_t) - \min(KL_t^j(x))}{\max(KL_t^j(x)) - \min(KL_t^j(x))}$$

$\mu_t(x)$ be the membership value of word x_t, and $j \in \{pos, neg\}$:

Figure 4.1 Algorithm to generate fuzzy values

Algorithm2: Algorithm to train and test BERT model to detect offensiveness in a sentence.

Input: Labeled dataset of text samples in Tamil, Kannada, and Malayalam languages
Output: Trained model for multi-lingual classification of offensive in a sentence.

Step 1: Data Preprocessing:

a. Tokenize input text into sub word units using the Word Piece algorithm.

b. Add special tokens (such as [CLS] and [SEP]) to mark the beginning and end of each sequence.

c. Convert text data to numerical representations compatible with BERT model.

Step 2: Model Training:

a. Load a pre-trained BERT model that supports multiple languages.

b. Fine-tune the pre-trained BERT model on the labeled dataset of text samples using a multi-class classification objective.

c. Train the model for a fixed number of epochs, optimizing a chosen loss function using an appropriate optimizer algorithm.

d. Save the trained model for future use.

Step 3: Model Evaluation:

a. Load the trained model and a separate test set of text samples in multiple languages.

b. Preprocess the test set data as in step 1.

c. Evaluate the model on the test set using one or more classification metrics, such as accuracy, precision, recall, and F1-score.

d. Output the evaluation results for analysis and reporting.

Step 4: Deployment:

a. Deploying the model using for classifying the sentences.

4.2 TOOLS/TECHNOLOGIES

Here are the tools and technologies used for implementing BERT and Neuro Fuzzy Logic:

1. PyTorch: PyTorch is popular deep learning frameworks that provide the necessary tools for implementing the BERT model.

2. Hugging Face Transformers: This is a popular library for implementing BERT and other transformer-based models. It provides a range of pre-trained BERT models that can be fine-tuned for specific natural language processing tasks, including sentiment analysis.

3. Tokenizers: Tokenizers are used to convert raw text data into a format that can be processed by the BERT model. The BERT model requires input data to be tokenized into sub word units using the Word Piece algorithm.

4. Data Loader: Data Loader is a module in PyTorch that is used to efficiently load and preprocess large datasets. It is particularly useful for training BERT models on large corpora of text data.

5. Optimizers: Optimizers are used to update the parameters of the BERT model during the training process. Popular optimizers for BERT include Adam and Stochastic Gradient Descent (SGD).

6. Loss functions: Loss functions are used to calculate the difference between the predicted output of the BERT model and the true labels. The most used loss function for sentiment analysis tasks is binary cross-entropy loss.

7. Train_test_split: This module is part of the scikit-learn library and is used for splitting a dataset into training and testing sets. This is a common technique in machine learning for evaluating the performance of a model on unseen data. The module provides a function that randomly splits a dataset into a training set and a testing set, with the option to specify the size of the split.

8. Pandas: This is a Python library that provides data structures and tools for working with structured data. It is particularly useful for data preprocessing and cleaning, and provides functions for loading and manipulating data in a variety of formats (such as CSV, Excel, and SQL).

9. NumPy: This is a Python library that provides support for large, multi-dimensional arrays and matrices, as well as a range of mathematical functions. It is commonly used in scientific computing and machine learning for data manipulation and numerical operations.

10. Collections: This is a Python library that provides additional data structures beyond the built-in data types, such as lists and dictionaries. Some of the commonly used data structures in collections include Counter (for counting occurrences of elements), defaultdict (for creating dictionaries with default values), and deque (for creating double-ended queues).

11. scheduler: This module is part of the PyTorch library and is used for scheduling the

learning rate during training. The learning rate is a hyperparameter that controls the step size taken during the optimization process, and can have a significant impact on the performance of the model. The scheduler module provides functions for adjusting the learning rate based on various criteria, such as the number of epochs or the validation loss.

12. Google Colab: Google Colab is a cloud-based Jupiter notebook environment that allows users to run Python code and perform machine learning tasks on Google's servers. We used this tool for implementing BERT models because it provides free access to high-performance GPUs and TPUs, which are essential for training large models.

13. GPUs: Graphics Processing Units (GPUs) are specialized hardware components that are optimized for performing parallel computations. We used GPU for training deep learning models, including BERT, because they can perform matrix operations much faster than traditional CPUs. Many cloud-based machine learning platforms, such as Google Cloud Platform, offer GPU instances for training deep learning models.

Overall, these modules provided the necessary tools for implementing and training a BERT model for classification of multi-lingual offensiveness

4.3 DATASET

The dataset has training and testing sentences and labels for 3 languages Tamil, Kannada, and Malayalam [9]. There are 5 types of labels in each set, they are offensive, not-offensive, Offensive Targeted Insult Other, Offensive Targeted Insult Individual, Offensive Targeted Insult Group. Distribution Labels for each language in training and testing sets are plotted in the following sections.

4.3.1 TAMIL OFFENSIVE DATASET

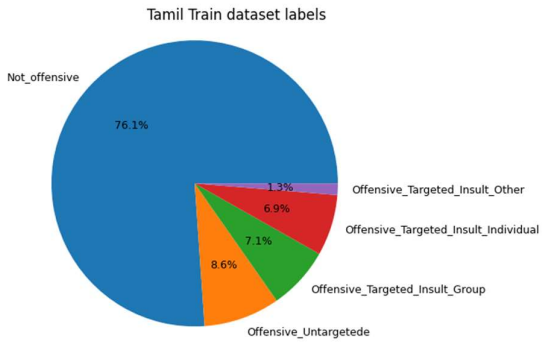

Figure 4.2 Tamil training dataset labels distribution

Label	Count
Not Offensive	23566
Offensive Untargeted	2665
Offensive Targeted Insult Group	2196
Offensive Targeted Insult Individual	2134
Offensive Targeted Insult Other	403

Table 4.1 Tamil training dataset labels count

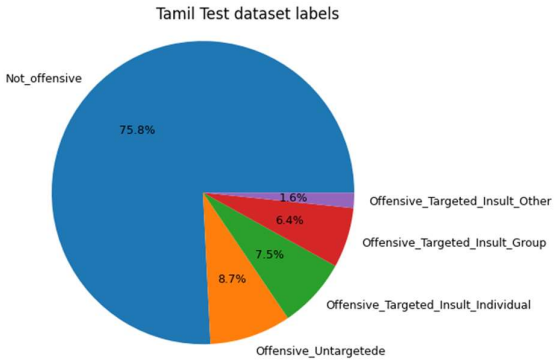

Figure 4.3 Tamil Test Dataset Labels Distribution

Label	Count
Not Offensive	2791
Offensive Untargeted	341
Offensive Targeted Insult Group	251
Offensive Targeted Insult Individual	293
Offensive Targeted Insult Other	64

Table 4.2 Tamil testing dataset labels count

4.3.2 KANNADA OFFENSIVE DATASET

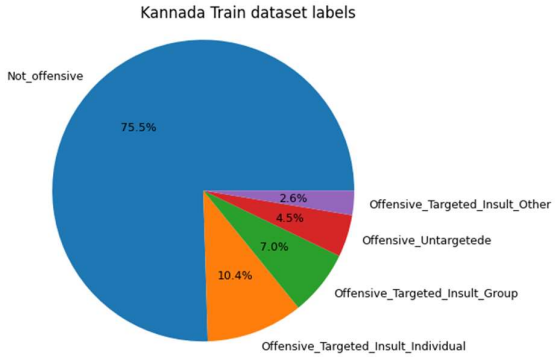

Figure 4.4 Kannada Train Dataset Labels Distribution

Label	Count
Not Offensive	3543
Offensive Untargeted	212
Offensive Targeted Insult Group	329
Offensive Targeted Insult Individual	487
Offensive Targeted Insult Other	123

Table 4.3 Kannada training dataset labels count

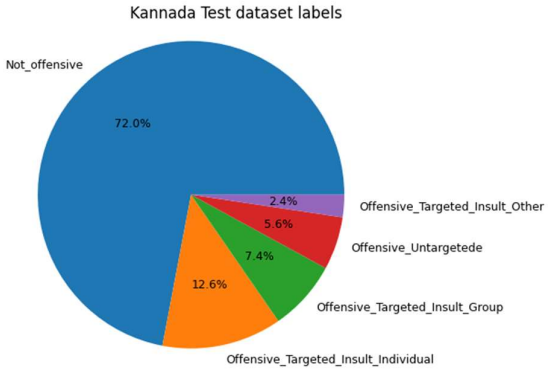

Figure 4.5 Kannada Test Dataset Labels Distribution

Label	Count
Not Offensive	427
Offensive Untargeted	33
Offensive Targeted Insult Group	44
Offensive Targeted Insult Individual	75
Offensive Targeted Insult Other	14

Table 4.4 Kannada testing dataset labels count

4.3.3 MALAYALAM OFFENSIVE DATASET

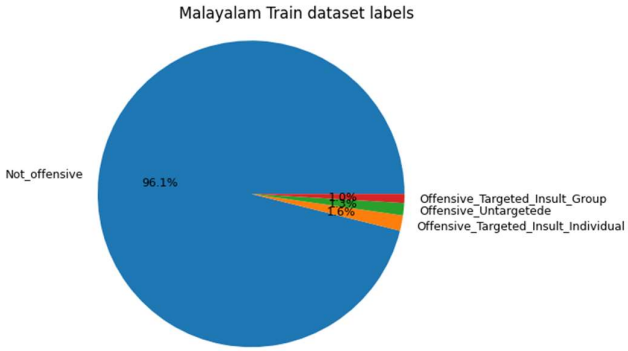

Figure 4.6 Malayalam Train Dataset Labels Distribution

Label	Count
Not Offensive	14153
Offensive Untargeted	191
Offensive Targeted Insult Group	140
Offensive Targeted Insult Individual	239

Table 4.5 Malayalam training dataset labels count

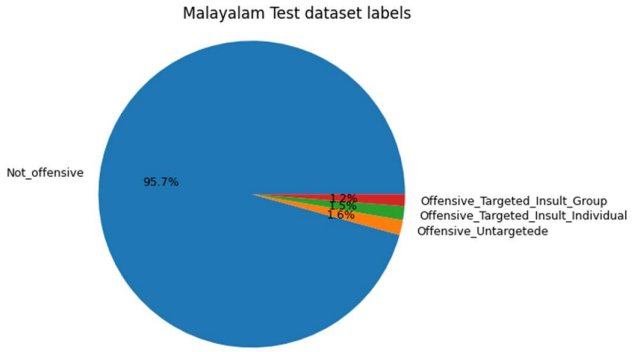

Figure 4.7 Malayalam Test Dataset Labels Distribution

Label	**Count**
Not Offensive	1765
Offensive Untargeted	29
Offensive Targeted Insult Group	23
Offensive Targeted Insult Individual	27

Table 4.6 Malayalam testing dataset labels count

5. EXPERIMENTAL RESULTS

5.1 BERT Model Results

When we went with BERT model, we achieved 82.7% accuracy after training with Tamil, Kannada, and Malayalam Languages Dataset.

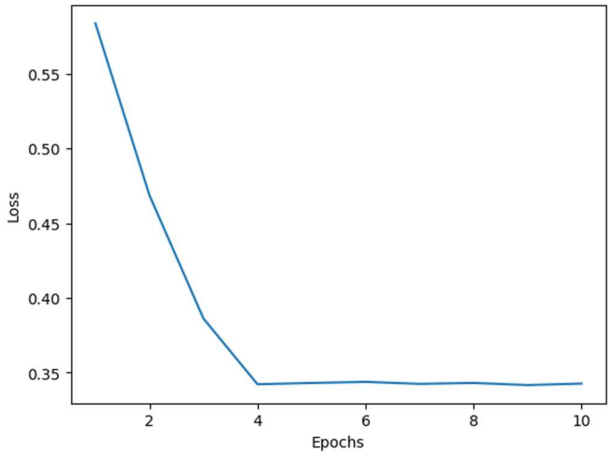

Figure 5.1 Loss Curve (BERT)

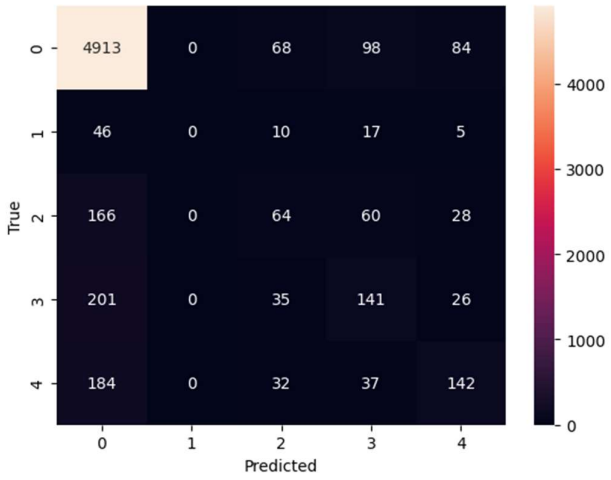

Figure 5.2 Confusion Matrix (BERT)

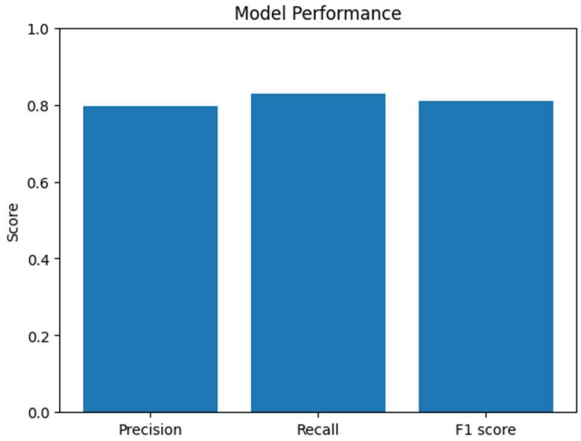

Figure 5.3 Model Performance (BERT)

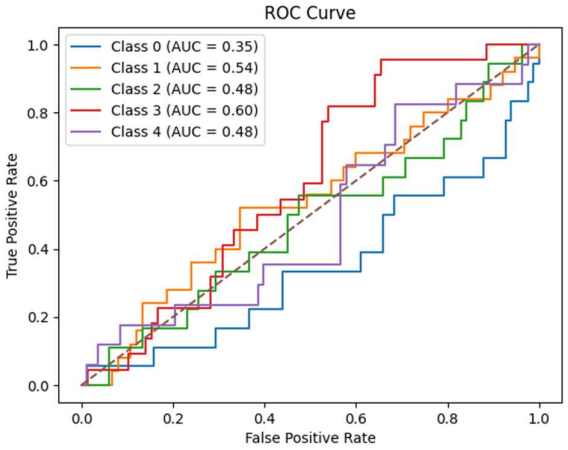

Figure 5.4 ROC Curve (BERT)

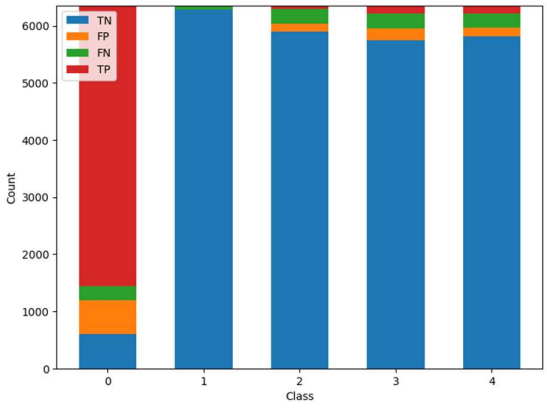

Figure 5.5 TP, TN, FP, FN for each label (BERT)

5.2 KL-NF Technique Results

When we used KL Divergence, Neuro Fuzzy Logic and TF-IDF for the same task of multi-lingual Classification, we achieved approximately 82.5% accuracy after training with same Tamil, Kannada, and Malayalam Datasets

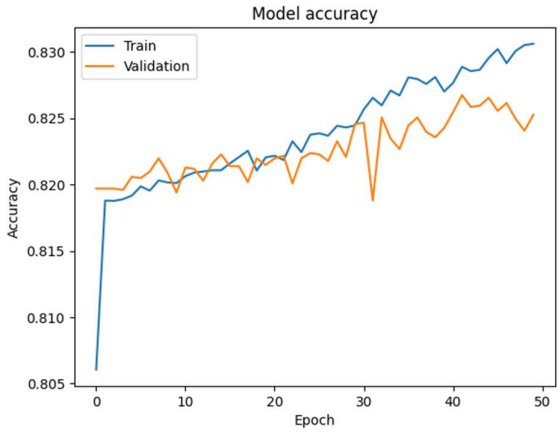

Figure 5.6 Model Accuracy Curve (KL-NF)

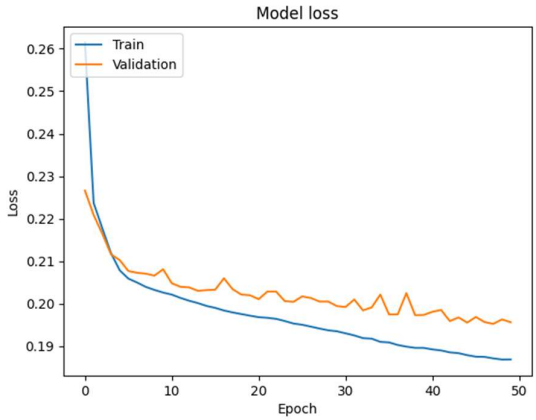

Figure 5.7 Loss Curve (KL-NF)

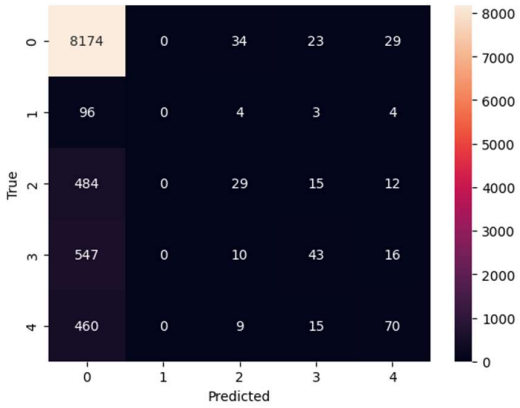

Figure 5.8 Confusion Matrix (KL-NF)

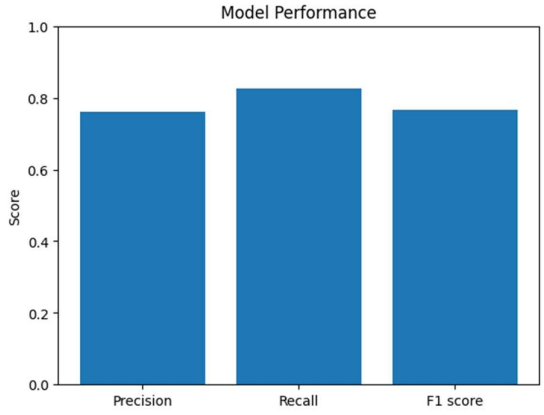

Figure 5.9 Model Performance (KL-NF)

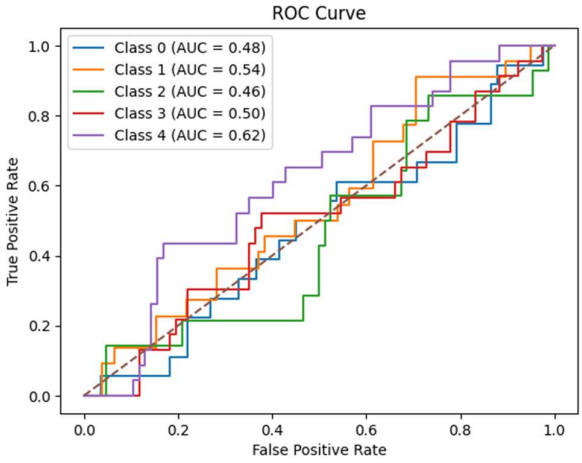

Figure 5.10 ROC Curve (KL-NF)

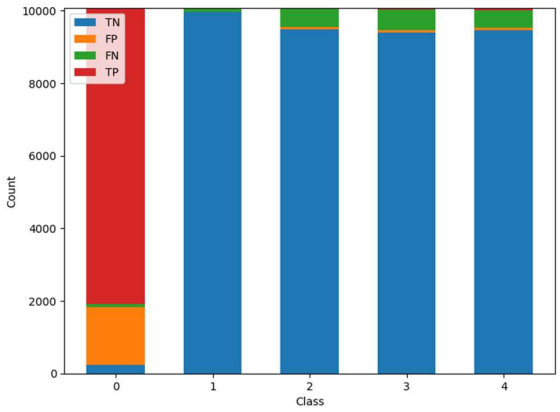

Figure 5.11 FP, FN, TP, TN counts for each label (KL-NF)

From Comparison of Both models Accuracy after training and testing with same Dataset (Tamil, Kannada, and Malayalam), we observed that both models are equally likely classifying the sentence correctly. If we consider with more precision BERT is slightly better in accuracy than the KL-NF.

6. CONCLUSION AND FUTURE SCOPE

The research shows how to use low resource languages for multi-lingual classification with small and large datasets. The approach used TF-IDF and Neuro Fuzzy technique for producing the fuzzy values. Because of the unique approach to represent input sentence with fuzzy values, the algorithm can be used for any language. The results shows that BERT can slightly learn better than the KL-NF because of its ability to generate attention masks and considering those attention masks while learning.

If we need to further improve the model, we can pretrained model for initializing the BERT model weights at starting to reduce the training time and to improve the accuracy to better than the present. Also, we can use this approach for any classification task with any number of labels for any language and for any size of datasets. Overall, the research presents a promise approach for multi-lingual classification task for low resource languages with both KL-NF and BERT.

REFERENCES

[1]. Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). “Learning word vectors for sentiment”

[2]. Kaggle Open Data Source, Adam Mathias Bittlingmayer, “Amazon Reviews for Sentiment Analysis,”

[3]. UCI Machine Learning, “SMS Spam Collection Dataset,” “www.kaggle.com/uciml/sms-spamcollection-dataset”.

[4]. Das, Bijoyan and Chakraborty, Sarit (2018). An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation.

[5]. Cicero Nogueira dos Santos and Maira Gatti (2015). “Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts”

[6]. Pengfei Liu, Songfang Huang, and Xipeng Qiu (2016). "Recurrent Neural Network for Text Classification with Multi-Task Learning"

[7]. Xing Fang Justin Zhan. “Sentimental Analysis Using Product Review Data.”

[8]. Devlin et al. (2018). "BERT for Sentiment Analysis of Movie Reviews"

[9]. Adaikkan Kalaivani, Durairaj Thenmozhi. “Multilingual Sentiment Analysis in Tamil, Malayalam, and Kannada code-mixed social media posts” using MBERT.

[10] Alhanoof Althnian, Duaa AlSaeed, Heyam Al-Baity, Amani Samha , Alanoud Bin Dris, Najla Alzakari, Afnan Abou Elwafa and Heba Kurdi, (2021). “Impact of Dataset Size on Classification Performance: An Empirical Evaluation in the Medical Domain”

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, (2019). Google AI Language. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.”

[12] Kanika Garg, D.K. Lobiyal (2021). KL-NF technique for sentiment classification.

[13] Bakliwal A, Arora P, Varma V (2012) Hindi subjective lexicon: A lexical resource for hindi polarity classification. Eighth Int Conf Lang Resour Eval:1189–1196

[14] Balamurali AR, Joshi A, Bhattacharyya P (2011) Robust sense-based sentiment classification. In: Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA), pp 132–138.

[15] Wilson T (2005) Recognizing contextual polarity in phrase-level sentiment analysis in HLT-EMNLP, pp 347–354.