A project report on

MULTILINGUAL SENTIMENT ANALYSIS

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

SHUBHAM OJHA (20BCE1205)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April,2024

MULTILINGUAL SENTIMENT ANALYSIS

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

SHUBHAM OJHA (20BCE1205)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2024

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "MULTILINGUAL SENTIMENT ANALYSIS" submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of bonafide work carried out by me under the supervision of Dr. Rajalakshmi R.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai — SHUBHAM OJHA

Date: — Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"MULTILINGUAL SENTIMENT ANALYSIS"** is prepared and submitted by **Shubham Ojha** (**20BCE1205**) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering programme** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr. Rajalakshmi R

Date:

Signature of the Internal Examiner	Signature of the External Examiner
Name:	Name:
Date:	Date:

Approved by the Head of Department,
B.Tech. CSE

Name: Dr. Nithyanandam P

Date:

(Seal of SCOPE)

ABSTRACT

Multilingual sentiment analysis poses a significant challenge in natural language processing, particularly in languages with limited resources such as Malayalam. This research addresses the problem of sentiment analysis in Malayalam by comparing various machine learning and deep learning models. Leveraging datasets from the CodaLab Fake News Detection in the Dravidian Languages competition, our study investigates the efficacy of Support Vector Machines (SVM), Random Forest (RF), Logistic Regression, Naive Bayes, and deep learning techniques including BERT classifiers. We apply these models to two distinct datasets: one comprising YouTube comments classified as original or fake, and the other categorizing news into five different categories false, mostly false, true, half true, and mostly true. Our experimentation involves three deep learning approaches: utilizing BERT classifiers directly on the Malayalam text, translating Malayalam to English using the Helsinki-NLP pipeline before employing BERT, and employing an Indic BERT tokenizer for enhanced tokenization. Recognizing the issue of fake news and the influence of social media comments on public opinion, our research seeks to address the challenges of identifying misinformation in online content. Our analysis reveals promising results, with the Naive Bayes classifier achieving an accuracy of 78.8% in detecting original YouTube comments, while Random Forest attains 68.6% accuracy in identifying fake news. Apart from this we get 1.59% more accuracy by applying traditional ML models on English translated dataset. Furthermore, we provide comprehensive comparisons using accuracy metrics and confusion matrices, shedding light on the strengths and limitations of each model in the context of Malayalam sentiment analysis. This research contributes to the advancement of sentiment analysis in understudied languages, offering insights into effective modelling techniques and highlighting avenues for future research in multilingual natural language processing, particularly in combating fake news and understanding social media discourse.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Rajalakshmi R., Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of NLP, ML and Deep Learning.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, B.Tech. CSE and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: **Shubham Ojha**

TABLE OF CONTENT

LIST OF FIGURES AND TABLES

Figures-:

Tables-:

Chapter 1

INTRODUCTION

A significant area of research in natural language processing (NLP) is multilingual sentiment analysis, which provides deep insights into the rich and varied fabric of human emotion exhibited in many languages around the world. Sentiment analysis's fundamental goal is to understand and classify the emotional undertone of written material, a process that gets more difficult and subtle when applied to several languages. The interpretation of sentiment can be greatly influenced by the distinctive linguistic characteristics, cultural settings, and colloquial idioms that are peculiar to each language, contributing to the complexity of language use.

Within digital communication, sentiment analysis that is multilingual becomes an essential tool due to the abundance of data in multiple languages and formats. It makes a plethora of unstructured text data from social media platforms, forums, comments, news articles, and blogs accessible to corporations, governments, and researchers. Entities can obtain a more comprehensive understanding of consumer preferences, cultural trends, and public opinion by analysing sentiment across languages. This helps to promote cross-cultural understanding and makes decision-making easier.

This work initially applies conventional machine learning classifiers—Support Vector Machine (SVM), Random Forest (RF), Logistic Regression, and Naive Bayes—to both datasets using a comparative analysis technique. The best model is chosen for testing on the appropriate datasets based on the validation findings from these classifiers. The work then investigates the use of deep learning techniques with a BERT classifier, an Indic-BERT tokenizer to improve tokenization fidelity specifically for Malayalam, and a Helsinki-NLP pipeline for Malayalam-to-English translation followed by BERT classification.

This study attempts to shed light on the difficulties and effectiveness of multilingual sentiment analysis in processing Malayalam literature by contrasting conventional machine learning models with cutting-edge deep learning techniques. The results not only add to the wider conversation on sentiment analysis in a variety of languages, but they also provide useful information about how to best utilise natural language processing (NLP) methods for Malayalam, a language that has great cultural and social significance in the digital era.

Chapter 2

Literature Survey

2.1Deep Learning Model for COVID-19 Sentiment Analysis on Twitter

In the landscape of sentiment analysis research, the work by Salvador Contreras Hernández, María Patricia Tzili Cruz, and José Martín Espínola stands out for its timely focus on public sentiment during the COVID-19 pandemic as expressed in tweets originating from Mexico. The significance of their study lies in the objective to understand the public's reaction to the pandemic, which has had unprecedented global impact. By targeting social media, specifically Twitter, the authors acknowledge the platform's role as a contemporary barometer for public opinion.

The methodology of the study is noteworthy for its application of BERT-based models tailored to the Spanish language, employing a semi-supervised learning framework. This choice is significant as it delves into the linguistic nuances inherent in Spanish tweets, which may be inadequately captured by models trained on data from other languages or contexts. By benchmarking these specialized models against both multilingual BERT variants and traditional machine learning classifiers, the research provides a robust comparative analysis, highlighting the capabilities and limitations of each approach within the scope of sentiment analysis.

The findings of this research are particularly compelling; Spanish language models demonstrated superior precision in sentiment analysis over their multilingual and traditional counterparts. This precision is not just a technical victory but carries profound implications for public health strategies, suggesting that fine-tuned language models can yield insights into public sentiment with high accuracy. These insights have the potential to inform public health policy and decision-making, making sentiment analysis a valuable tool in the ongoing management of the COVID-19 pandemic and in preparing for future public health crises. The study sets a precedent for the importance of language-specific models in sentiment analysis and their practical application in shaping responsive and informed public health interventions.

2.2 Multilingual text categorization and sentiment analysis

In the collaborative work of George Manias, Argyro Mavrogiorgou, and Athanasios Kiourtis, the complexities of classifying Twitter's multilingual content are examined. The study's aim is to advance solutions that transcend specific domains and language barriers, an endeavor crucial in today’s interconnected digital landscape.

The authors' approach involved a detailed comparison of four BERT-based classifiers against a zero-shot classification method, aiming to determine their effectiveness in handling data from multiple languages. This aspect of research stands crucial for enhancing the understanding and development of language models capable of universal application.

The findings of the study offer a dual perspective: multilingual BERT-based classifiers are proficient, but zero-shot classification stands out for its scalable application across languages, despite occasionally lagging in accuracy behind fine-tuned models. This insight provides a valuable direction for future research in multilingual natural language processing, balancing between broad applicability and precision.

2.3Multilingual topic modeling for tracking COVID-19 trends based on Facebook data analysis

In an effort to broaden the scope of computational social science research during the COVID-19 pandemic, Amina Amara, Mohamed Ali Hadj Taieb, and Mohamed Ben Aouicha have directed their study towards the utilization of Facebook data for tracking and analyzing public sentiment and discourse. Unlike Twitter, which has been widely examined, Facebook's rich datasets have not been extensively mined for pandemic-related trends. This oversight in research methodologies presents an opportunity, which the authors address through their multilingual topic modeling investigation.

The researchers employed Latent Dirichlet Allocation (LDA), a sophisticated topic modeling technique, to dissect and understand the multilingual conversations held on Facebook regarding COVID-19. They meticulously analyzed the data spanning five months, from January to May 2020, across seven different languages, reflecting the global impact of the pandemic. The use of graph visualization tools in their

methodology allowed for a dynamic representation of the evolution of public interest and sentiment, providing an illustrative timeline of the pandemic's progression as seen through the eyes of Facebook users worldwide.

The conclusion of their study is particularly insightful, offering a unique window into the ebb and flow of public opinion and the key subjects of discussion during the early and intense months of the COVID-19 crisis. By capturing the chronological development of the pandemic's public discourse in a multilingual context, the study not only enriches the understanding of social media dynamics but also serves as a vital tool for policymakers and health officials. It underscores the importance of leveraging social media analytics to gauge public sentiment, which can guide response strategies and communication plans during global health emergencies.

2.4Sentiment analysis in Portuguese tweets

Daniela Vianna, Fernando Carneiro, and Jonnathan Carvalho have made a significant contribution to the field of sentiment analysis with their investigation into the efficacy of word representations for Brazilian Portuguese tweets. Recognizing the importance of this language in the realm of data analytics, due to the vast number of Portuguese speakers and the corresponding volume of digital communication, their research fills a critical gap in sentiment analysis studies.

The methodology of the study involved a comprehensive assessment of both static and Transformer-based embeddings, utilizing various datasets and classifiers tailored to the Portuguese language. The goal was to meticulously identify which type of word representation could yield the most accurate results in sentiment analysis within the context of Brazilian Portuguese social media interactions.

The results of the research are quite revealing, indicating that the BERTimbau model, which was specifically trained to understand the nuances of the Portuguese language, significantly surpassed other models. This bespoke BERT model's superior performance in sentiment classification underscores the value of language-specific training in natural language processing tasks. The findings serve as a testament to the potential of fine-tuned language models in enhancing the precision of sentiment

analysis, especially for languages other than English, which often lack the depth of research and resources.

2.5Multilingual deep learning framework for fake news detection using capsule neural network

Rami Mohawesh, Sumbal Maqsood, and Qutaibah Althebyan have presented an innovative study focusing on the detection of fake news in a multilingual context, a pressing issue in the digital age where misinformation spreads rapidly across language barriers. Their approach hinges on the use of capsule neural networks, which are particularly adept at managing semantic and contextual nuances that vary from one language to another, an aspect that traditional models often struggle with.

The methodology of the research revolves around constructing a robust multilingual framework. This system integrates the power of capsule neural networks with word embeddings and n-gram feature extraction techniques, enabling the model to be trained on a diverse dataset of fake news spanning multiple languages. This design aims to capture the intricate patterns and signals that denote authenticity or falsehood in news content, regardless of the language.

The study's conclusion is resoundingly positive, indicating that their proposed model significantly outstrips current leading methods in the field of fake news detection. This advancement is a testament to the capsule neural network’s potential for transforming the landscape of multilingual text analysis, demonstrating a powerful tool for more accurately and efficiently identifying false information across the world's languages. This research not only marks a step forward in combating fake news but also illustrates the growing necessity for advanced computational models that can navigate the complexities of language on a global scale.

2.6A new sentiment analysis method to detect and analyse sentiments of Covid-19 Moroccan tweets using a recommender approach

Youness Madani, Mohammed Erritali, and Belaid Bouikhalene have developed a novel sentiment analysis methodology tailored to evaluate Moroccan tweets concerning COVID-19, integrating a recommender system approach to dissect and categorize public sentiment. Their innovative technique addresses the critical need for understanding public opinion in times of crisis, especially within specific national contexts.

The study's methodology employs collaborative filtering, leveraging four key tweet features alongside sentiment analysis tools—specifically, the SenticNet dictionary and the TextBlob Python library. This approach signifies an adaptation of recommender system strategies, commonly seen in e-commerce, to the domain of sentiment analysis, aiming to identify and track the collective emotional pulse of a nation through its social media output.

Conclusively, the research attained an impressive 86% accuracy in sentiment classification, revealing the sentiment's progression over time and its alignment with the unfolding epidemiological events in Morocco. Notably, the prevalent negative sentiment captured in the tweets mirrors the public's overall response to the pandemic's challenges. These findings are critical for policymakers and health communicators who seek to understand and effectively respond to the population's concerns during health crises, using real-time data to inform strategic decision-making.

2.7HateDetector: Multilingual technique for the analysis and detection of online hate speech in social networks

Anjum and Rahul Katarya have introduced a pioneering approach to combating hate speech online with their tool, HateDetector. In an age where digital communication transcends borders and languages, hate speech on social networks has become a global issue. HateDetector is not just another addition to the arsenal against such online toxicity; it represents a targeted strike against the multilingual nature of hate speech, which adds layers of complexity to its detection and subsequent management.

Hate speech online is not just about the presence of certain keywords or phrases; it's about context, cultural nuance, and language. The authors' approach with HateDetector acknowledges this complexity by incorporating an improved seagull optimization

algorithm to extract features from the data, which is crucial for understanding the subtleties of language that distinguish hate speech from benign communication. Alongside this, a hybrid diagonal-gated recurrent neural network is employed for both detection and sentiment analysis, allowing for a nuanced interpretation of text data. The combination of these two advanced methodologies indicates a deep understanding of both the technical and social dimensions of hate speech.

The study's results are significant: HateDetector has shown considerable improvements in accuracy, precision, recall, and F-measure over existing models. These metrics are vital in the field of natural language processing and sentiment analysis, and improvements in these areas suggest that HateDetector is capable of identifying hate speech more reliably than ever before. In essence, the tool does not just find hateful content; it understands it, and this understanding is crucial for any meaningful moderation strategy on social platforms. It’s a promising development for social media platforms that grapple with the tide of multilingual hate speech, providing a more reliable way to monitor and manage the online dialogue to foster safer digital environments.

2.8Sentiment analysis using deep learning approaches: an overview

Olivier Habimana, Yuhua Li, Ruixuan Li, Xiwu Gu, and Ge Yu have provided a thorough exploration into the application of deep learning techniques within the domain of sentiment analysis, presenting an extensive review that traces the progression and diversification of methodologies in this dynamic field. Their work casts light on how these sophisticated computational approaches have evolved to tackle the multifaceted challenges presented by the subjective nature of sentiment in textual data.

The paper serves as a compendium, delving into a variety of deep learning models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Generative Adversarial Networks (GANs), which have each been adapted to address different dimensions of sentiment analysis. The authors examine the specific utilities of these models, evaluating their performance in intricate tasks that include, but are not limited to, aspect-based sentiment analysis, which focuses on extracting sentiment with

respect to specific aspects of a service or product; multilingual sentiment analysis, which grapples with sentiment expressed across different languages; and multimodal sentiment analysis, which interprets sentiment from combined textual, audio, and visual cues. This multifaceted review provides insights into the strengths and limitations of each model, offering a clear view of the current state of the art.

In conclusion, the authors affirm the potent capabilities of deep learning in sentiment analysis but also recognize the necessity for ongoing innovation. They emphasize the importance of developing more advanced, task-specific deep learning methods to enhance the accuracy and effectiveness of sentiment analysis. By addressing existing challenges and outlining potential avenues for future research, the paper not only celebrates the achievements of deep learning in this area but also serves as a clarion call for the continuation of inventive research efforts to refine these methods further. This is pivotal for the evolution of sentiment analysis, ensuring it remains adept at handling the increasing complexity and nuances of human emotion expressed across diverse and ever-expanding digital platforms.

2.9 Combining sentiment analysis classifiers to explore multilingual news articles covering London 2012 and Rio 2016 Olympics

In their scholarly work, Caio Mello, Gullal S. Cheema, and Gaurish Thakkar delve into the intricate task of sentiment analysis (SA) applied to multilingual news articles that discuss the lasting impacts of the London 2012 and Rio 2016 Olympics. The paper stands out for its utilization of a combination of four distinct sentiment analysis algorithms—SenticNet, SentiStrength, Vader, and BERT. Furthermore, it integrates explainable AI practices to shed light on the limitations of these methods, providing a clearer understanding of how each algorithm performs in the context of a complex, real-world application. The aim is to uncover the narrative frameworks the media uses to portray the legacies of such significant global events across different languages, a challenging area due to the nuanced nature of linguistic expression and cultural context.

To conduct their analysis, the researchers compiled a substantial corpus consisting of 1,271 news articles in both English and Portuguese, reflecting the bilingual nature of the study. They applied the aforementioned SA algorithms to the corpus, treating both

the original texts and their translations to ensure a thorough analysis. Feature visualization was accomplished using SHAP (SHapley Additive exPlanations), enhancing interpretability of the data. They propose an innovative approach that amalgamates three top-performing classifiers—Vader, Amazon BERT, and Sent140 BERT—aiming to leverage the strengths of each to produce a more robust sentiment analysis tool.

The conclusion drawn from this research acknowledges the valuable insights sentiment analysis can offer in media studies, yet it does not shy away from discussing the inherent challenges associated with multilingual and domain-specific datasets. By fusing multiple classifiers and incorporating explainable AI techniques, the authors suggest that the accuracy and reliability of sentiment analysis can be significantly improved. One of the most intriguing outcomes of this study is the identification of a dichotomy in the representation of Olympic legacies, oscillating between utopian and dystopian narratives. This dichotomy not only reflects the multifaceted impact of the Olympics on host cities but also highlights the power of media in shaping public perception and discourse. Through this work, Mello, Cheema, and Thakkar contribute a nuanced understanding of how complex events are represented across cultural and linguistic divides, and how advanced computational techniques can better capture the subtleties of global narratives.

2.10 Deep learning-based sentiment analysis and offensive language identification on multilingual code-mixed datation

Kogilavani Shanmugavadivel, V. E. Sathishkumar, and Sandhiya Raja investigate the nuanced task of sentiment analysis and offensive language identification within code-mixed data, focusing on the linguistic interplay between Tamil and English. Their study is pioneering in its exploration of how machine learning and advanced neural network models, particularly BERT, RoBERTa, and the specialized adapter-BERT, can be adapted for the intricate challenges posed by such low-resource, hybridized datasets.

Leveraging data from the DravidianLangTech@ACL2022 shared task, the research team employed a variety of models including logistic regression, CNN, Bi-LSTM, and the aforementioned transformer models, assessing them against metrics of accuracy, precision, recall, and F1-score. These models were augmented with sophisticated word embedding techniques, like GloVe and BERT, to capture the semantic richness of code-mixed language more effectively. The study presents a comprehensive comparative analysis, culminating in the finding that the adapter-BERT model, tailored for the specific context of code-mixed data, notably outshines others in terms of performance. It achieves 65% accuracy in sentiment analysis and 79% in identifying offensive language, setting a new benchmark for such tasks.

The results underscore the effectiveness of customizing pre-trained models to suit the unique aspects of code-mixed data, a task that has become increasingly relevant as linguistic data becomes more diverse and complex. The researchers advocate for future exploration into multitask learning paradigms and enhancements to dataset quality, aiming to refine the tools for handling the intricacies of mixed-language text. This work not only propels forward the field of sentiment analysis in linguistically diverse settings but also acts as a catalyst for future innovations that could further improve the precision and applicability of natural language processing tools.

2.11 Multilingual hate speech detection sentimental analysis on social media platforms using optimal feature extraction and hybrid diagonal gated recurrent neural network

Purbani Kar and Swapan Debbarma have addressed a pertinent issue in computational linguistics with their recent study: the detection of hate speech and analysis of sentiment in the complex arena of multilingual code-mixed text on social media. They propose a system that not only identifies hate speech but also dissects the sentiment behind it, a particularly challenging task given the nuanced and often ambiguous nature of code-mixed language, which blends elements from multiple languages within a single discourse.

To navigate this linguistic complexity, the authors have developed an improved seagull optimization algorithm specifically for extracting features from these hybrid texts. The

extracted features are then processed through a novel neural network architecture they introduced—a hybrid diagonal gated recurrent neural network (H-DGRNN)—which is tailored for the dual tasks of hate speech detection and sentiment analysis.

The study's conclusive evidence demonstrates notable strides in model performance, showing significant gains in accuracy, precision, recall, and the F-measure. These metrics indicate not only the effectiveness of the proposed method but also its potential to serve as a powerful tool in the current landscape of social media platforms, where multilingual communication and hate speech are pervasive. By enhancing the accuracy of detecting hate speech and analyzing sentiment in code-mixed text, the study by Kar and Debbarma provides a critical contribution to creating safer online environments and advancing the field of natural language processing for multilingual content.

2.12 Sentiment analysis of Hindi language text: a critical review

Simran Sidhu, Surinder S. Khurana, Munish Kumar, and Parvinder Singh have collaborated on a comprehensive review that delves into sentiment analysis specifically within the Hindi language, addressing both the methodological advancements and the evolution of essential tools like Hindi SentiWordNet. This review comes at a crucial time when the Hindi-speaking population is not only substantial but also increasingly active in the digital space, thus necessitating refined techniques for sentiment analysis in this language.

The paper explores a variety of methodologies that have been employed for sentiment analysis in Hindi. This includes a focus on semantic analysis and the application of machine learning techniques, which are crucial in accurately interpreting the sentiment of Hindi text. Additionally, the paper scrutinizes the tools integral to the sentiment analysis process, such as lexicons which provide databases of words and their associated sentiments, stemmers which reduce words to their roots, and morphological analyzers which dissect words into their fundamental components. These linguistic tools are vital for processing Hindi text, which often presents challenges due to its complex grammar and rich morphology.

The conclusion drawn from the review is a call to action for further research and development in sentiment analysis methods tailored to the Hindi language. The authors highlight the current gaps in the field, such as the need for more advanced research to develop robust sentiment analysis tools that can keep pace with the growing digital

presence of the Hindi language online. They outline potential future directions for research, underscoring the importance of this work in understanding and harnessing the vast wealth of sentiment expressed by Hindi speakers on digital platforms. This review not only provides a roadmap for future explorations but also serves as a benchmark for the current state of sentiment analysis in Hindi, paving the way for innovations that could significantly impact both academic research and practical applications in the realm of natural language processing.

2.13 An augmented multilingual Twitter dataset for studying the COVID-19 infodemic

Christian E. Lopez and Caleb Gallemore have compiled an extensive multilingual dataset derived from Twitter, specifically designed to probe the social discourse surrounding the COVID-19 pandemic. The purpose of this dataset is to empower research that seeks to decode the complex web of public conversation and sentiment related to one of the most disruptive global events in recent history.

Spanning an impressive volume of over 2.2 billion tweets, the dataset encompasses a range of languages, thereby offering a panoramic view of global sentiment and discussions. This broad linguistic range allows for an inclusive understanding of the pandemic's social impact worldwide. The data is further enriched with sentiment analysis and named entity recognition algorithms, equipping researchers with the tools needed to perform a nuanced analysis of the conversations and to identify key themes, stakeholders, and sentiments within the vast swathes of Twitter data.

The conclusion posited by Lopez and Gallemore is that this curated dataset stands as a critical resource for academics and professionals striving to grasp the evolution of public sentiment throughout the pandemic. Its value lies not just in its size but in its potential to serve as a foundational platform for a variety of social media data analyses, from tracking the spread of information to understanding the public’s emotional and cognitive responses to the pandemic's progression. The dataset offers a unique opportunity to chronicle the pandemic's narrative as told through the eyes of millions on social media, presenting an unfiltered account of humanity's collective experience and reaction to COVID-19.

2.14 Persian Text Sentiment Analysis Based on BERT and Neural Networks

Siroos Rahmani Zardak, Amir Hossein Rasekh, and Mohammad Sadegh Bashkari have conducted a pivotal study in the field of natural language processing, focusing specifically on sentiment analysis within the Persian language. Their work underscores a significant gap in computational linguistics—effective sentiment analysis for Persian text—and seeks to bridge this by adapting the Bidirectional Encoder Representations from Transformers (BERT) algorithm for Persian language sentiment analysis.

BERT, renowned for its context-aware approach to language modeling, is applied to Persian text in this study. The authors compare the BERT model's performance against previous sentiment analysis methods using various datasets. This comparative analysis is crucial because it measures the progress made by employing a state-of-the-art algorithm on a non-English language dataset, which can present unique challenges due to structural and contextual differences.

The conclusion of the study confirms the superiority of the BERT algorithm in analyzing sentiment in Persian texts. The BERT model achieves high accuracy and F1 scores, outstripping earlier methods. This significant improvement not only demonstrates BERT's adaptability across languages but also indicates a promising direction for future research in sentiment analysis for underrepresented languages.

2.15 Multi-class sentiment analysis of Urdu text using multilingual BERT

Lal Khan, Ammar Amjad, Noman Ashraf, and Hsien-Tsung Chang have made a significant contribution to natural language processing by creating a new multi-class Urdu dataset aimed at enhancing sentiment analysis for Urdu, which is classified as a low-resource language in computational linguistics. Their work is particularly important as it addresses the scarcity of robust datasets that are essential for developing effective sentiment analysis models for such languages.

The methods deployed in this research are comprehensive, employing both machine learning and deep learning techniques. These include Support Vector Machines (SVM),

Naive Bayes (NB), AdaBoost, Multi-Layer Perceptrons (MLP), Logistic Regression (LR), Random Forest (RF), one-dimensional Convolutional Neural Networks (CNN-1D), Long Short-Term Memory networks (LSTM), Bidirectional LSTMs (Bi-LSTM), Gated Recurrent Units (GRU), Bidirectional GRUs (Bi-GRU), and a fine-tuned version of the multilingual BERT (mBERT) model.

In their conclusion, the researchers highlight the superior performance of the mBERT model, which utilizes BERT's pre-trained word embeddings. The mBERT model achieved an impressive F1 score of 81.49%, showcasing its proficiency in interpreting and analyzing sentiment in the Urdu language.

Chapter 3

Dataset-Description

This study utilizes two distinct datasets sourced from the CodaLab Fake News Detection in Dravidian Languages competition, specifically designed for the Dravidian-LangTech@EACL 2024. The datasets are instrumental in exploring sentiment analysis and fake news detection within the Malayalam language, representing a significant step towards understanding and analyzing Dravidian languages through advanced natural language processing techniques.

3.1 Malayalam Comments Dataset

The first dataset comprises 3,200 Malayalam comments, derived from various YouTube videos, intended to distinguish between original and fake content. This dataset is split into a 70:30 ratio for training and validation purposes, resulting in 2,240 entries for training and the remainder for validation. An additional set of 1,020 comments is reserved for testing. Each entry consists of two fields:

- **Text:** The comment text in Malayalam, presenting a diverse array of topics and sentiments.
- **Label:** A binary classification indicating whether the comment is deemed 'original' or 'fake'.

3.2 Malayalam News Dataset

The second dataset focuses on Malayalam news articles, comprising 1,700 entries divided similarly into a 70:30 training-validation split, yielding 1,190 entries for training and the rest for validation. The testing set includes 250 news articles. The dataset features two main columns:

- **News:** The full text of the news article in Malayalam.
- **Label:** A categorical classification of the news article into one of five truthfulness categories: 'False', 'Half True', 'Mostly False', 'Partly False', and 'Mostly True'.

2.3 Helsinki-NLP Pipeline and English Translation

To enhance the analysis and applicability of NLP tools traditionally optimized for English, the study incorporates the Helsinki-NLP pipeline for machine translation. This approach involves converting the Malayalam datasets into English, thereby enabling the use of a broader range of NLP models and tools that may not be directly applicable to Malayalam due to language-specific constraints.

Chapter 4

Proposed Methodology

The proposed work focuses on multilingual sentiment analysis in Malayalam, utilizing diverse datasets sourced from the CodaLab Fake News Detection competition. Employing a combination of traditional machine learning algorithms such as SVM, RF, logistic regression, and naive Bayes, alongside deep learning techniques including BERT classifiers and Helsinki-NLP pipeline for translation, the study aims to assess the effectiveness of various approaches in distinguishing between original and fake YouTube comments, as well as categorizing news articles across five truthfulness categories.

4.1 Proposed System Introduction

The capacity to precisely evaluate sentiment and identify bogus news in multilingual environments has become critical in the rapidly changing world of digital communication. There is an urgent need for sophisticated natural language processing (NLP) systems that can handle the complexities of language-specific nuances, cultural contexts, and a wide range of sentiment expressions due to the proliferation of information and the increasing significance of regional languages on digital platforms. This study suggests a thorough approach to deal with these issues in the millions of speakers of the Dravidian language Malayalam, which is rich in linguistic variety.

The suggested method investigates sentiment analysis and false news detection in Malayalam by utilising both conventional machine learning algorithms and state-of-the-art deep learning approaches. This study takes a multipronged strategy to determine the best techniques for identifying and classifying opinions and the accuracy of news reports in Malayalam. In addition to improving the state of NLP research on Dravidian languages, the system is meant to offer useful tools and insights that may be used in a variety of contexts, such as social media monitoring and content moderation, among others.

Our method is based on analysing Malayalam texts for sentiment and authenticity using a variety of machine learning classifiers, such as Support Vector Machine (SVM), Random Forest (RF), Logistic Regression, and Naive Bayes. The system investigates the effectiveness of deep learning models concurrently, concentrating on the BERT classifier and its modifications for the Malayalam language via direct application, translation-based methods, and sophisticated tokenization strategies. Through the use of both classic and new NLP approaches, this dual-path exploration seeks to improve accuracy and provide a deeper understanding of the sentiment and authenticity of Malayalam texts.

The suggested approach would evaluate these models' performance in relation to important indicators using a strict evaluation framework, providing a thorough grasp of their advantages and disadvantages. This methodical approach not only lays the foundation for applying these findings to other languages and circumstances, but it also promises to improve our understanding of sentiment analysis and fake news detection in Malayalam.

The suggested method is proof of the potential of fusing various NLP techniques as we traverse the challenges of multilingual sentiment analysis and fake news identification. By providing a strong framework for evaluating sentiment and authenticity in Malayalam, it seeks to make a substantial contribution to the subject and, in the process, further knowledge of digital communication in multilingual societies.

4.2 Research Objectives

The main objective of this work is to improve the field of natural language processing (NLP) by creating and assessing models that can identify fake news and perform sentiment analysis in Malayalam, a language that is widely spoken in the digital sphere but is not well-represented in NLP studies. The following are the study's particular goals:

4.2.1 To Assess Conventional Models of Machine Learning

Examine how well the classic machine learning classifiers—Random Forest (RF), Logistic Regression, Support Vector Machine (SVM), and Naive Bayes—identify feelings and identify false news in Malayalam text. In order to achieve this goal, these models will be compared in order to determine which method performs best while considering performance indicators like accuracy, precision, recall, and F1 score.

4.2.2 To Explore Deep Learning Techniques

Examine the use of deep learning models for Malayalam sentiment analysis and fake news identification, with a focus on the BERT classifier. This covers the use of translated texts through the Helsinki-NLP pipeline, direct application on Malayalam texts, and enhanced Malayalam text tokenization using the Indic-BERT tokenizer.

4.2.3 To Compare Model Performances

Conduct a comprehensive comparison between traditional machine learning models and deep learning approaches to determine the most effective methodologies for sentiment analysis and fake news detection in Malayalam. This comparison will take into account not only performance metrics but also model efficiency and scalability.

4.2.4 To Assess the Impact of Language Translation on Model Performance

Evaluate how translating Malayalam to English, using the Helsinki-NLP pipeline, affects the performance of sentiment analysis and fake news detection models. This includes assessing the trade-offs between linguistic authenticity and the accessibility of advanced NLP tools.

4.2.5 To Identify Challenges and Limitations

Identify the main challenges and limitations encountered when applying NLP models to Malayalam text. This objective aims to provide insights into language-specific issues, data scarcity, and the nuances of cultural context in sentiment analysis and fake news detection.

4.2.6 To Offer Insights for Future Research

Based on the findings, provide recommendations for future research directions in the field of NLP, especially concerning underrepresented languages like Malayalam. This includes suggesting improvements to model architectures, data preprocessing techniques, and the exploration of other machine learning and deep learning approaches.

4.2.7 To Demonstrate Real-world Applications

Illustrate the potential applications of the research findings in real-world scenarios, such as enhancing content moderation systems, improving news verification processes, and providing tools for sociolinguistic research in the Malayalam-speaking community.

4.3 Flowchart Diagram

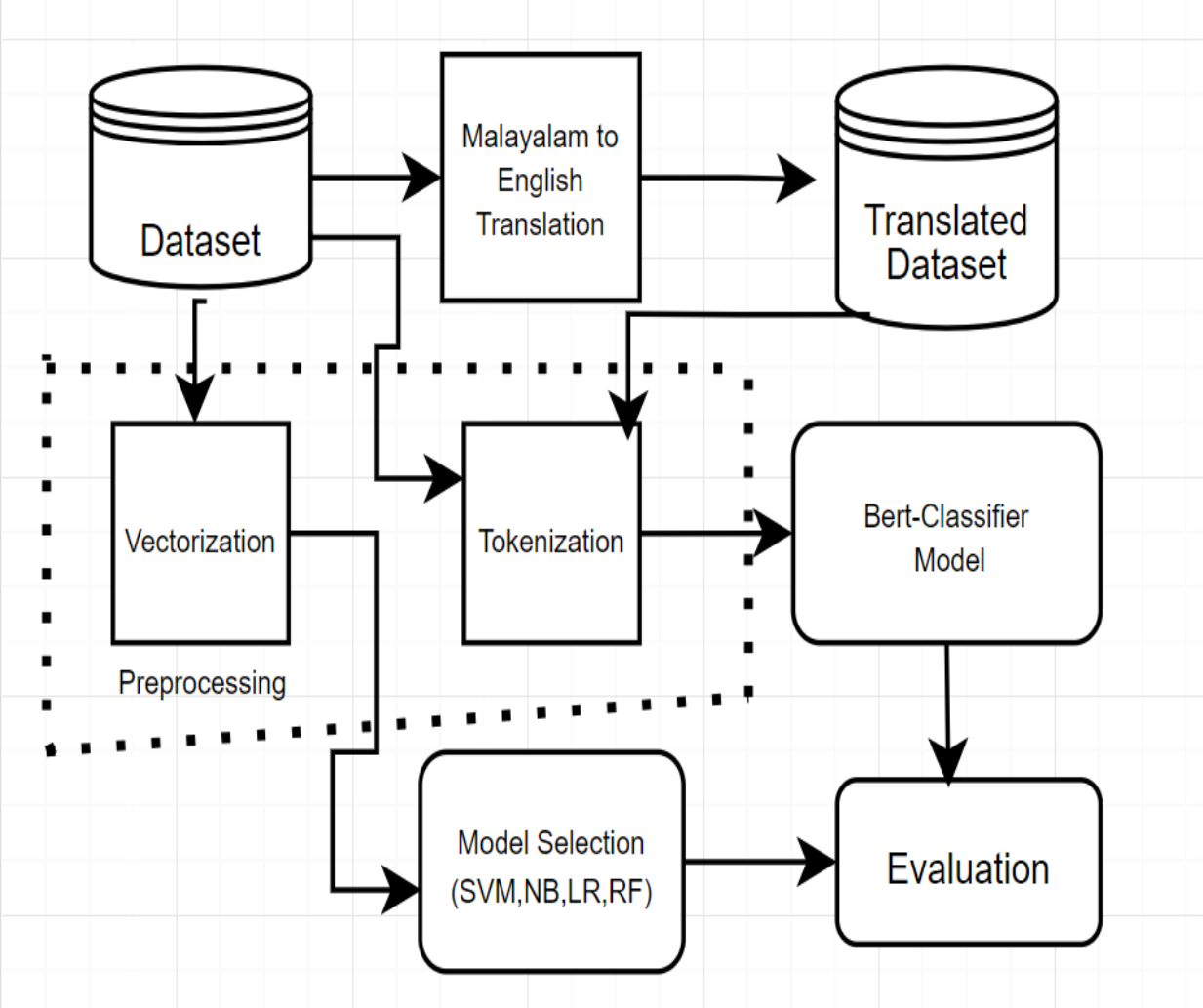

Fig 1

4.4 Flowchart Explanation

4.4.1 Dataset

This is the primary repository of raw data collected for analysis. In this context, it consists of Malayalam comments and news articles that will be the subject of sentiment analysis and fake news categorization. The dataset serves as the foundational input for all subsequent steps in the flowchart.

4.4.2 Malayalam to English Translation

Recognizing the limitations in NLP tools for Malayalam, this stage involves translating the dataset into English. The translation aims to leverage the extensive range of tools and models developed for English, potentially increasing the accuracy of sentiment analysis. This step is a testament to the multilingual aspect of the project and addresses the challenge of language disparity in NLP resources.

4.4.3 Translated Dataset

Post-translation, the dataset exists in a form that is more accessible to NLP tools predominantly designed for the English language. This translated corpus retains the original semantics but is now suitable for processing by models trained on English datasets.

4.4.4 Preprocessing

A critical step in NLP, preprocessing transforms raw text into a structured format that machine learning algorithms can interpret. In the flowchart, this is a compound process consisting of:

- Vectorization: This process converts text to numerical data, creating feature vectors that represent the text's attributes. Vectorization enables the mathematical manipulation of text by models that rely on numerical input.

- Tokenization: This is the practice of dividing text into tokens, which can be words, phrases, symbols, or other meaningful elements. Tokenization is essential for models to analyse and understand the text at a granular level.

4.4.5 Bert-Classifier Model

The BERT (Bidirectional Encoder Representations from Transformers) classifier is a state-of-the-art model for NLP tasks. It’s known for its deep understanding of context and language nuances. The model is fine-tuned for the specific task of sentiment analysis and fake news detection. Given your description, three different applications of BERT classifiers are used:

- One that works directly on the original Malayalam text.

- Another that operates on the English-translated text.

- And a third that employs an Indic-BERT tokenizer, presumably to better handle the Malayalam script's complexities.

4.4.6 Model Selection

This step involves the application and comparison of various traditional machine learning models:

- Support Vector Machine (SVM): A model known for its effectiveness in classification tasks, particularly with high-dimensional data.

- Naive Bayes (NB): A probabilistic classifier that applies Bayes' theorem and is particularly fast and simple.

- Logistic Regression (LR): Despite its name, it's a classification algorithm that predicts the probability of categorical outcomes.

- Random Forest (RF): An ensemble method that operates by constructing multiple decision trees for more reliable and accurate predictions.

4.4.7 Evaluation

Finally, the models are evaluated to assess their performance in classifying sentiments and detecting fake news. Evaluation typically employs metrics such as accuracy (the proportion of correct predictions), precision (the number of true positives over the sum of true positives and false positives), recall (the number of true positives over the sum of true positives and false negatives), and the F1 score (a harmonic mean of precision and recall). The goal of this step is to benchmark the models against each other to determine which yields the most reliable results for the datasets in question.

4.5 Modules Explanation

4.5.1 Vectorization

Vectorization is process of converting script data into numerical form so that machine learning algorithms can understand and process it. There are several steps and methods for vectorizing text data:

i. Text Cleaning: Before vectorization, the text usually undergoes cleaning to remove any noise that could affect the analysis. This includes stripping out HTML tags, correcting typos, and removing special characters and stopwords (common words like 'the', 'is', which don't contribute much meaning).

ii. Normalization: Text normalization includes converting all characters to lowercase, stemming, and lemmatization so that we can easily do our mathematical calculations.

iii. Tokenization: As a pre-step to vectorization, tokenization breaks text into tokens (words or phrases). Each token becomes an individual feature in the final vector.

iv. Vectorization Techniques: There are various techniques for converting tokens into numerical vectors, such as:
    - Count Vector-ization
    - TF-IDF Vectori-zation (where TF stands for Term Frequency and IDF stands for Inverse Document Frequency)
    - Word Embeddings

v. Feature Selection: After vectorization, feature selection may be applied to reduce the dimensionality of the dataset. This step helps to improve model performance by eliminating redundant features and reducing noise.

vi. Output: The output is a feature matrix where each row represents a document, and each column represents a term or token from the text's vocabulary. The values in the matrix depend on the vectorization technique used.

Once the data is vectorized, it is ready for use in machine learning models for various tasks, including classification, clustering, and sentiment analysis.

4.5.2 Tokenization

Tokenization is the process of breaking down text into smaller units called tokens, which can be words, characters, or subwords. It's a fundamental step in preparing raw text for various NLP tasks. Here's an in-depth look at how tokenization is generally performed:

- Selection of Unit for Tokenization: Decide if the tokens should be words, characters, or subwords (like WordPiece used in BERT). The choice depends on the task and the characteristics of the language.

- Implementing Tokenization: Use a tokenization method. For many languages, simple space-based tokenization may be sufficient, but for languages like Malayalam with agglutinative properties, a more sophisticated approach may be required.

- Dealing with Punctuation and Special Characters: Decide whether to keep or remove punctuation and special characters. Sometimes, punctuation can carry significant meaning and should be preserved as separate tokens.

- Handling Compound Words: In languages with compound words, it may be necessary to split these into their constituent parts to better capture their meaning.

- Addressing Case Sensitivity: Convert all tokens to lower case unless there is a specific need to preserve case sensitivity (e.g., 'US' as an abbreviation for the United States versus 'us').

- Applying Language-Specific Rules: Implement rules that handle language-specific characteristics, such as conjuncts in Malayalam script or contractions in English.

- Use of Pre-Trained Tokenizers: Utilize pre-trained tokenizers (such as spaCy, NLTK, or Hugging Face's Transformers library) that have been developed specifically for NLP tasks. These often come with models trained on large corpora and can handle the nuances of tokenization effectively.

- Subtokenization: For models like BERT, subtokenization is often used where words are broken down into smaller pieces that allow the model to handle a wider variety of words it hasn't seen before.

- Output: The output is a list or sequence of tokens. This output serves as input for further processing steps like vectorization or direct use in models like BERT, which require tokens as input.

Steps to Execute Tokenization:

i. Import a tokenization library or use a custom tokenizer if language-specific adjustments are required.
ii. Prepare the text data, ensuring it's in a suitable format for tokenization.
iii. Apply the tokenization process to the dataset.
iv. Review and potentially refine the list of tokens to ensure quality.
v. Proceed to use these tokens for further NLP tasks, such as feeding them into a vectorization process or a machine learning model for training and inference.

Tokenization is an essential preprocessing step that directly affects the performance of subsequent NLP tasks. It must be tailored to the specific linguistic features and requirements of the language and the objectives of the NLP application.

4.5.3 Translation (Malayalam to English)

Translation in the context of NLP involves converting text from one language to another, in this case, from Malayalam to English. It's a complex task that requires understanding and preserving the meaning of the source text. Here's how the translation process unfolds in detail:

- Choosing a Translation Model: Select an appropriate machine translation model or service. For Malayalam to English translation, one might use state-of-the-art models like those from the Helsinki-NLP group, which offer pre-trained models for a variety of language pairs.

- Preparing the Dataset: Before translation, ensure the dataset is cleaned and formatted properly. This may involve removing any non-textual information, correcting encoding issues, and segmenting the text into translatable units.

- Handling Ambiguity and Context: Determine how to deal with words or phrases that have multiple meanings and require context to translate correctly. Some translation models can handle this inherently, while others may need additional context as input.

- Translation Execution: Run the translation process on the prepared dataset. This involves feeding the text into the translation model, which then outputs the translated text.

- Post-Translation Review: After the initial translation, it's crucial to review the output to check for accuracy and fluency. Post-editing may be required to correct any errors or awkward phrasings.

- Quality Assurance: Implement quality assurance measures, such as back-translation (translating the text back to the original language) or human review, especially for critical applications.

- Use in Subsequent Tasks: The translated text is now ready to be used in further NLP tasks, like sentiment analysis or classification, often with tools that are optimized for or exclusively available in English.

Steps to Execute Translation:

i. Choose and configure a machine translation service or library.
ii. Preprocess the Malayalam text data to ensure it's clean and compatible with the translation tool.
iii. Input the Malayalam text into the translation system in a batch or stream, as supported by the tool.
iv. Collect the translated English text as output from the system.
v. Conduct post-translation checks to ensure the quality of the translated text.
vi. Utilize the translated dataset for downstream NLP tasks or models that are designed for or perform better with English language input.

4.5.4 Support Vector machine (SVM)

Often used to solve classification problems, the Support Vector Machine (SVM) is a strong and adaptable supervised machine learning technique that may be applied to regression as well as classification applications. It is especially helpful in situations when precise decision boundaries are required because of its capacity to identify the hyperplane that most effectively partitions a dataset into classes. Because of its versatility in handling different kinds of data and its efficiency in high-dimensional environments, this approach is particularly well-liked.

Using Support Vector Machines (SVM) to analyse datasets for plagiarism detection can be quite helpful in determining the similarities and differences between texts or documents. Word frequency, n-grams, and semantic similarity are just a few of the aspects that may be analysed using SVM to determine if a document is authentic or plagiarised. The SVM-generated decision planes facilitate the differentiation of authentic content from instances of plagiarism, which in turn allows for the detection and mitigation of content reuse or unauthorised copying.

Furthermore, SVM's capacity to manage intricate decision-making in datasets with a multitude of features is helpful in tasks involving plagiarism detection,

where the ability to discern minute differences in text is crucial. Support vector machines (SVM) are able to identify patterns and similarities across texts by using them as a representation of the data points that define the hyperplane.

Steps typically involved in calculating the solution for an SVM:

i. Feature Mapping: Map input data to a high-dimensional feature space.
ii. Hyperplane Definition: Identify the optimal hyperplane that separates the classes in the feature space.
iii. Support Vectors Identification: Find the data points closest to the hyperplane; these are the support vectors.
iv. Margin Maximization: Optimize the hyperplane by maximizing the margin between the support vectors and the hyperplane.
v. Solve Optimization Problem: Use an optimization algorithm (like Sequential Minimal Optimization or another quadratic programming method) to solve for the hyperplane.
vi. Kernel Trick (if needed): Apply a kernel function to enable separation in the high-dimensional space without explicitly mapping data to it.
vii. Decision Function: Establish the decision function based on the support vectors and their coefficients to classify new data.

Steps to Perform SVM:

i. Data Preparation: Ensure your data is in the right format, cleaned, and normalized. If necessary, convert categorical data into a numerical format through encoding.
ii. Choose a Kernel: Decide on the kernel function to use (linear, polynomial, radial basis function, etc.), which will allow the algorithm to fit the training data as accurately as possible.

iii. Model Training: Train the SVM model on your dataset. This involves feeding the vectorized and preprocessed data into the SVM algorithm to build the model.

iv. Hyperparameter Tuning: Adjust the model parameters, such as the regularization parameter (C), the kernel coefficient (gamma), and others as needed to find the best performing model. This is often done using techniques like grid search or cross-validation.

v. Model Evaluation: Evaluate the performance of the SVM model using appropriate metrics, such as accuracy, precision, recall, and F1-score, on a validation set or with cross-validation.

vi. Prediction: Use the trained SVM model to make predictions on new, unseen data, by feeding it the vectorized form of this new data.

vii. Model Fine-Tuning: Depending on the evaluation results, fine-tune the model by possibly retraining with different parameters or kernel functions until you achieve a satisfactory level of accuracy.

4.5.5 Naïve Bayes

Naive Bayes is mathematical and probabilistic classifiers based on Maths important topic Bayes' Theorem with the high assumption of non-dependence between their features. They are particularly known for their simplicity, efficiency, and their applicability to high-dimensional datasets. Naive Bayes is best suited for categorical input variables compared to numerical variables. Despite the simplicity of the assumption, Naive Bayes classifiers can outperform more sophisticated algorithms and are especially useful for text classification tasks such as spam detection or sentiment analysis.

Steps typically involved in calculating the solution for an NB:

i. Bayes' Theorem: Apply Bayes' Theorem to compute probabilities.

ii. Class Conditional Independence: Assume feature independence for each class.
iii. Likelihood Calculation: Calculate the likelihood of the data given each class.
iv. Posterior Probability: Determine the posterior probability for each class.
v. Classification: Assign each data point to the class with the highest posterior probability.

Steps to Perform Naive Bayes:

i. Preprocess Data: Clean the dataset and handle missing values. Convert text data into a numerical format through vectorization.
ii. Feature Selection: Choose or extract features to be used as input variables for the classifier.
iii. Model Training: Train the Naive Bayes classifier on the dataset. This involves calculating the prior probability of each class and the likelihood of each feature given each class.
iv. Hyperparameter Tuning: Adjust the smoothing parameter if necessary to account for features not present in the learning samples.
v. Model Evaluation: Assess the classifier using metrics like accuracy, precision, recall, and F1-score on the validation set.
vi. Prediction: Use the model to predict the class of new instances based on their feature measurements.

4.5.6 Logistic Regression

Logistic Regression is also a mathematical and stats related method for dividing a more than one dataset independent variables that determine an result. The result event is measured with a dich-otomous variable. It is used extensively in fields like medicine and social sciences, as well as machine learning, where it serves as a baseline algorithm for binary classification problems.

Steps typically involved in calculating the solution for an LR:

i. Sigmoid Function: Use the sigmoid function to map predicted values to probabilities.
ii. Log-Odds: Compute the log-odds of the probabilities, which is the linear relationship of the features.
iii. Maximum Likelihood Estimation: Optimize the model parameters to maximize the likelihood of the observed data.
iv. Gradient Descent: Use gradient descent (or other optimization algorithms) to find the weights that minimize the cost function.
v. Probability Threshold: Classify predictions based on a probability threshold (usually 0.5).

Steps to Perform Logistic Regression:

i. Data Preparation: Preprocess the data by normalizing or standardizing numerical variables and encoding categorical variables.
ii. Feature Selection: Identify significant variables that have a more substantial influence on the outcome.
iii. Model Fitting: Fit the logistic regression model to the dataset, estimating the probability of the dependent variable given the independent variables.
iv. Model Validation: Validate the model's performance with metrics such as ROC curve, AUC, or confusion matrix.
v. Optimization: Use techniques like regularization to prevent overfitting and to enhance the model's generalization capabilities.
vi. Prediction: Employ the model to estimate the probability of the target class for new data points.

4.5.7 Random Forest

For classification, regression, and other problems, Random Forest is an ensemble learning technique that builds a large number of decision trees during training. The class that the majority of the trees choose is the random forest's output for classification problems. Its excellent accuracy, resilience, and user-friendliness are well-known, and it handles numerical and categorical features as well as data without the need for scaling.

Steps typically involved in calculating the solution for an RF:

i. Bootstrap Aggregating: Create multiple bootstrap samples from the original data.
ii. Decision Tree Construction: Build a decision tree for each bootstrap sample. At each node:
iii. Randomly select a subset of features.
iv. Choose the best split based on an impurity measure (like Gini impurity or entropy).
v. Random Feature Selection: For each tree, use a random subset of features at each split.
vi. Ensemble Prediction: Make predictions by averaging the output of all the individual trees (regression) or by majority voting (classification).
vii. Error Reduction: Achieve error reduction through the aggregation of less correlated trees.

Steps to Perform Random Forest:

i. Data Preprocessing: Clean the data, handle missing values, and if necessary, encode categorical features.
ii. Feature Engineering: Though RF can handle a large number of features, it's often beneficial to create new features or select the most informative ones.
iii. Model Training: Train the random forest by building a multitude of decision trees. This includes deciding on the number of trees to build (n_estimators) and the depth of each tree.
iv. Hyperparameter Tuning: Tune parameters like the number of trees, the maximum depth of trees, minimum samples split, and minimum samples leaf.
v. Cross-Validation: Perform cross-validation to ensure that the model does not overfit the training data.
vi. Model Evaluation: Evaluate the random forest model using appropriate metrics to determine its accuracy and generalization to unseen data.
vii. Prediction: Apply the trained model to predict outcomes on new data.

4.5.8 Bi-directional Encoder Representation from Transformer (BERT)

BERT is a revolutionary model in the field of Natural Language Processing (NLP) developed by Google. Unlike previous models that process text in one direction, either from left to right or vice versa, BERT reads text bidirectionally, which allows it to understand context more effectively.

Here is a detailed explanation of BERT, including steps to execute it:

i. Pre-trained Models: BERT is first pre-trained on a large corpus of text. This pre-training involves learning the relationships between words in a sentence through two primary tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

ii. Fine-tuning for Specific Tasks: After pre-training, BERT can be fine-tuned with additional output layers to perform a wide variety of NLP tasks. This fine-tuning involves training BERT on a labeled dataset specific to the task at hand, such as sentiment analysis, question answering, or fake news detection.

Steps to Execute BERT for a Task Like Sentiment Analysis:

i. Load Pre-trained BERT Model
   Choose a pre-trained BERT model appropriate for your dataset. For English, bert-base-uncased is commonly used, while for Malayalam, you would use a model pre-trained on Malayalam texts or a multilingual version.

ii. Preprocess the Text Data

Preprocess your text data as per BERT's requirements. This involves tokenizing the text into tokens that BERT was trained on, adding special tokens (like [CLS] at the beginning of each sentence and [SEP] between sentences), and padding or truncating sentences to a fixed length.

iii. Tokenization

Use BERT's tokenizer to convert each token into its corresponding ID in BERT's vocabulary.

iv. Input Representation

BERT requires three kinds of input data: input IDs (token IDs from the tokenizer), attention masks (which tokens should be attended to), and token type IDs (used to distinguish different sentences).

v. Fine-tuning BERT

Pass your input data through BERT and add a classification layer on top. Train the model on your labeled dataset. During training, backpropagation is used to fine-tune the pre-trained BERT parameters and learn the weights of the added output layer.

vi. Evaluation

After fine-tuning, evaluate the model on a validation set to check its performance. Use relevant metrics like accuracy, precision, recall, F1-score, etc.

vii. Inference

Use the fine-tuned BERT model to make predictions on new data. The [CLS] token's output embedding can be used as the aggregate sequence representation for classification tasks.

4.6 Advantages and Disadvantages of models used

4.6.1 Naïve Bayes (NB)

Advantages:

- Speed: It's known for its speed in training and prediction, making it highly efficient for large datasets.
- Simplicity: Its algorithmic simplicity and the assumption of feature independence make it straightforward to implement and understand.
- Performance: Despite its simplicity, Naive Bayes can perform remarkably well in text classification tasks such as spam detection and document categorization.

Disadvantages:

- Feature Independence: The assumption that all features are independent is rarely met in real-world data, which can limit its effectiveness in certain applications.
- Complex Relationships: Struggles to capture complex relationships between features, limiting its use in more sophisticated classification tasks.

4.6.2 Random Forest (RF)

Advantages:

- Versatility: Excellent for handling both numerical and categorical data and performs well across a wide range of tasks.
- Feature Importance: Provides useful insights into which features are most influential in predicting the outcome.
- Overfitting: Tends to reduce overfitting through its ensemble approach, creating multiple decision trees and averaging their predictions.

Disadvantages:

- Complexity and Size: Models can become large and unwieldy, requiring significant memory storage.
- Computation Time: Training time can be long, especially as the number of trees increases.

4.6.3 Logistic Regression (LR)

Advantages:

- Efficiency: Offers fast training and prediction times, making it suitable for problems with a linear decision boundary.
- Interpretability: Coefficients can be easily interpreted, providing insights into the importance of each feature.
- Probabilistic Results: Outputs have a probabilistic interpretation, allowing for decision making with a threshold level.

Disadvantages:

- Non-Linear Problems: Performs poorly on problems where the relationship between variables is non-linear.
- Complex Relationships: Not the best choice for capturing complex interactions between features.

4.6.4 Support Vector Machine (SVM)

Advantages:

- Effectiveness in High Dimensions: Particularly powerful in spaces with a high number of dimensions.
- Flexibility: The use of different kernel functions allows SVM to adapt to various data types and structures.

Disadvantages:

- Scalability: Training time can grow quickly with the size of the data, making it less suitable for large datasets.
- Parameter Tuning: Requires careful tuning of parameters, including the choice of kernel and regularization parameters, which can be complex and time-consuming.

4.6.5 BERT (Bidirectional Encoder Representations from m Transformers)

Advantages:

- Contextual Understanding: BERT's deep learning approach allows it to capture the context and semantics of words in text, leading to state-of-the-art performance on a wide range of NLP tasks.
- Pre-training and Fine-tuning: The model can be pre-trained on a large corpus of text and then fine-tuned for specific tasks, making it versatile and powerful.

Disadvantages:

- Computational Resources: Requires substantial computational power and memory, limiting accessibility for some users.
- Training Time: Due to its complexity, training from scratch or fine-tuning on large datasets can take a significant amount of time.
- Risk of Overfitting: Its capacity to model complex relationships can lead to overfitting, especially with smaller training sets.

Chapter 5

Result and Discussion

5.1 Evaluation Metrices

When evaluating the performance of classification models, several evaluation metrics and visualizations can be used to understand how well a model is performing. Three commonly used methods are accuracy, the confusion matrix, and bar graphs. Each provides different insights into the model's performance.

i. Accuracy

One of the easiest metrics to employ when assessing categorization models is accuracy. It calculates the percentage of all predictions that the model accurately predicts, including true positives and true negatives:

The ratio of accurate predictions to total predictions determines accuracy.

Even though accuracy can give a fast overview of a model's performance, it can not always provide a full picture, particularly when the dataset is imbalanced (that is, when there are dramatically varying numbers of examples in different classes). For instance, a model that consistently predicts the majority class in a dataset where 95% of the examples belong to that class will have a high accuracy of 95% even if it is unable to accurately identify any instances of the other class.

ii. Confusion Matrix

The confusion matrix is a more detailed tool for evaluating classification model performance. It is a matic that is used to describe performance of a classification model on a set of data for which the true values are known. It shows the actual labels

against the model's predictions, providing insights into the types of errors made by the model.

The matrix is especially useful for binary classification tasks and can be extended to multiclass classification problems.

A confusion matrix for binary classification has four parts:

True Positives (TP): The number of positive instances correctly identified by the model.

False Positives (FP): The number of negative instances incorrectly identified as positive by the model.

True Negatives (TN): The number of negative instances correctly identified by the model.

False Negatives (FN): The number of positive instances incorrectly identified as negative by the model.

From the confusion matrix, other metrics like precision, recall, and F1-score can be calculated to provide further insights into model performance.

iii. Bar Graph

Bar graphs (or bar charts) are a visual form of displaying data that uses bars to compare different categories or groups. In the context of model evaluation, bar graphs can be very useful for visually comparing the performance of multiple models across different metrics (such as accuracy, precision, recall, etc.).

By using bar graphs, you can quickly identify which models are performing well and which ones are lagging in certain areas. For instance, you might plot the accuracy of several models on a bar graph to see which model has the highest accuracy. You can also use bar graphs to visualize the performance of a single model across different metrics, making it easier to assess the model's strengths and weaknesses comprehensively.

5.2 Observation and Explanation

Below are the tables summarizing the performance (Accuracy and Training Time) of each model for the YouTube Comment Originality dataset and the Fake News Detection dataset.

Youtube Comment Originality Detection

Model	***Accuracy(%)***	***Training Time (HH : MM : SS)***
Naïve Bayes	78.8	00:00:51
SVM	50.1	00:00:27
Random Forest	75.5	00:12:31
Logistic Regression	78.6	00:01:57
BERT	75.2	01:30:22
IndicBERT	75.1	01:29:27

Table 1- YouTube Comment Originality Detection Accuracy

Model	***Accuracy(%)***	***Training Time (HH : MM : SS)***
Naïve Bayes	76.8	00:00:47
SVM	58.3	00:00:25
Random Forest	78.5	00:10:41
Logistic Regression	71.6	00:01:37
BERT	74.0	01:16:20

Table 2 -YouTube Comment Originality Detection Accuracy on English translated Dataset

Table 1 presents the accuracy and training times of various models applied to detect the originality of YouTube comments in Malayalam. While Naïve Bayes and logistic regression exhibit relatively high accuracies of 78.8% and 78.6% respectively, SVM and BERT show lower performance. Random Forest achieves a competitive accuracy of 75.5%. In Table 2, the performance on an English-translated dataset is provided, where the models generally demonstrate lower accuracy compared to the Malayalam dataset, with Naïve Bayes and Random Forest performing relatively well. These results

suggest the impact of language translation on the effectiveness of the models for originality detection in YouTube comments.

Fake News Detection

Model	***Accuracy(%)***	***Training Time (HH : MM : SS)***
Naïve Bayes	63.5	00:00:59
SVM	65.8	00:00:37
Random Forest	68.6	00:20:76
Logistic Regression	61.7	00:12:07
BERT	60.52	01:56:12
IndicBERT	62.31	01:50:18

Table 3 - Fake News Detection Accuracy

Model	***Accuracy(%)***	***Training Time (HH : MM : SS)***
Naïve Bayes	60.5	00:00:56
SVM	67.7	00:00:28
Random Forest	72.2	00:14:76
Logistic Regression	64.7	00:11:72
BERT	65.7	01:33:12

Table 4 - Fake News Detection Accuracy on English translated Dataset

Table 3 presents the accuracy and training times of various models applied to detect fake news in Malayalam. Among these models, Random Forest achieves the highest accuracy of 68.6%, followed closely by SVM with 65.8%. Naïve Bayes and IndicBERT exhibit moderate accuracies, while BERT demonstrates the lowest performance. Table 4 shows the performance on an English-translated dataset, where Random Forest again performs the best with an accuracy of 72.2%. However, the overall performance of the models on the translated dataset is slightly lower compared to the Malayalam dataset, indicating potential challenges in cross-lingual fake news detection.

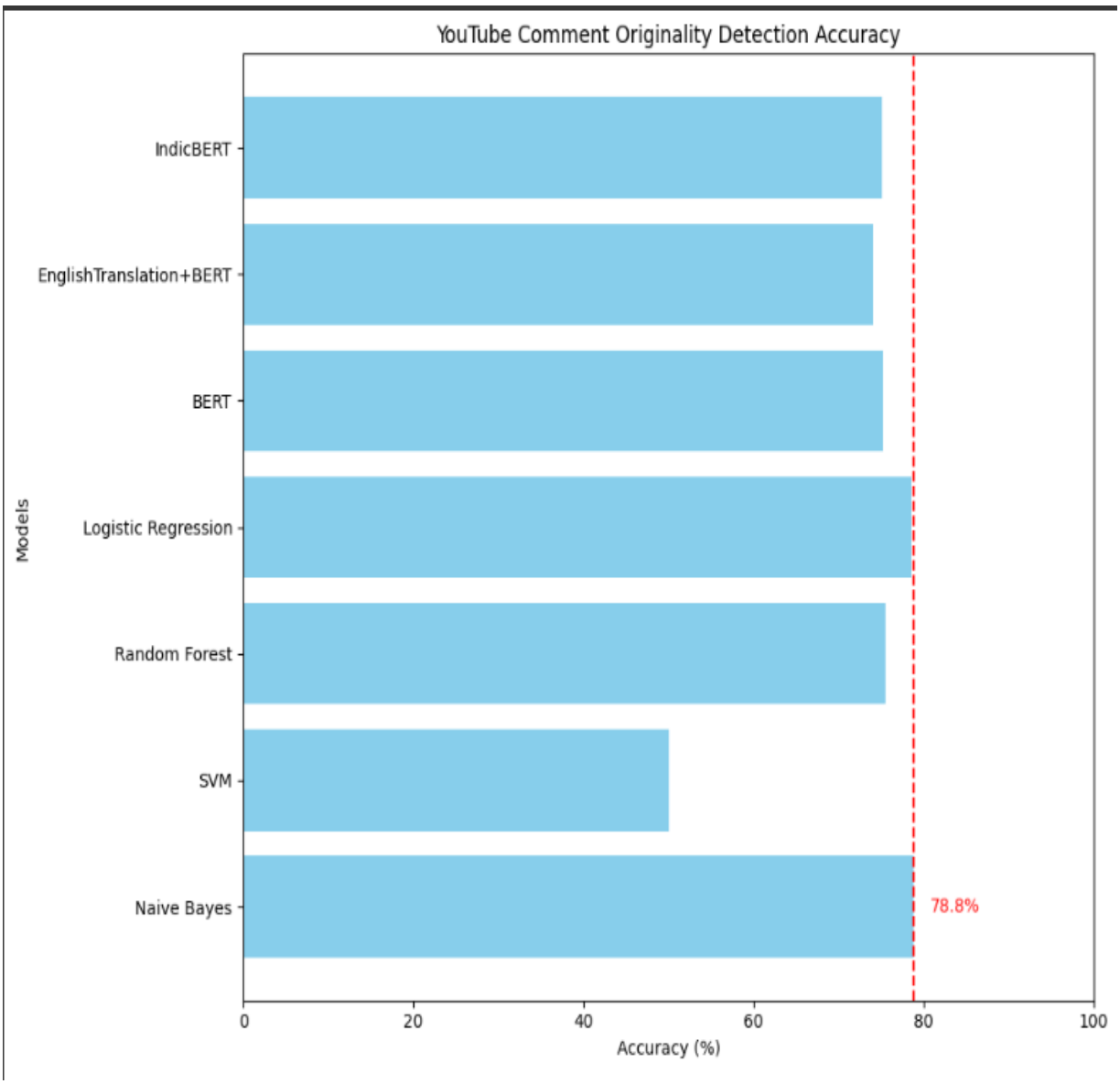

Fig 2- YouTube Comment Originality Detection Accuracy

Figure 2 and Figure 3 depict bar graphs illustrating the accuracy of various models for YouTube comment originality detection and fake news detection respectively. Each bar represents the accuracy achieved by a specific model, allowing for easy comparison of their performance. In both figures, Random Forest consistently emerges as one of the top-performing models, while Naïve Bayes and Logistic Regression exhibit mixed performance across the tasks. These visual representations provide clear insights into the relative effectiveness of different models for the respective tasks of originality detection and fake news detection.

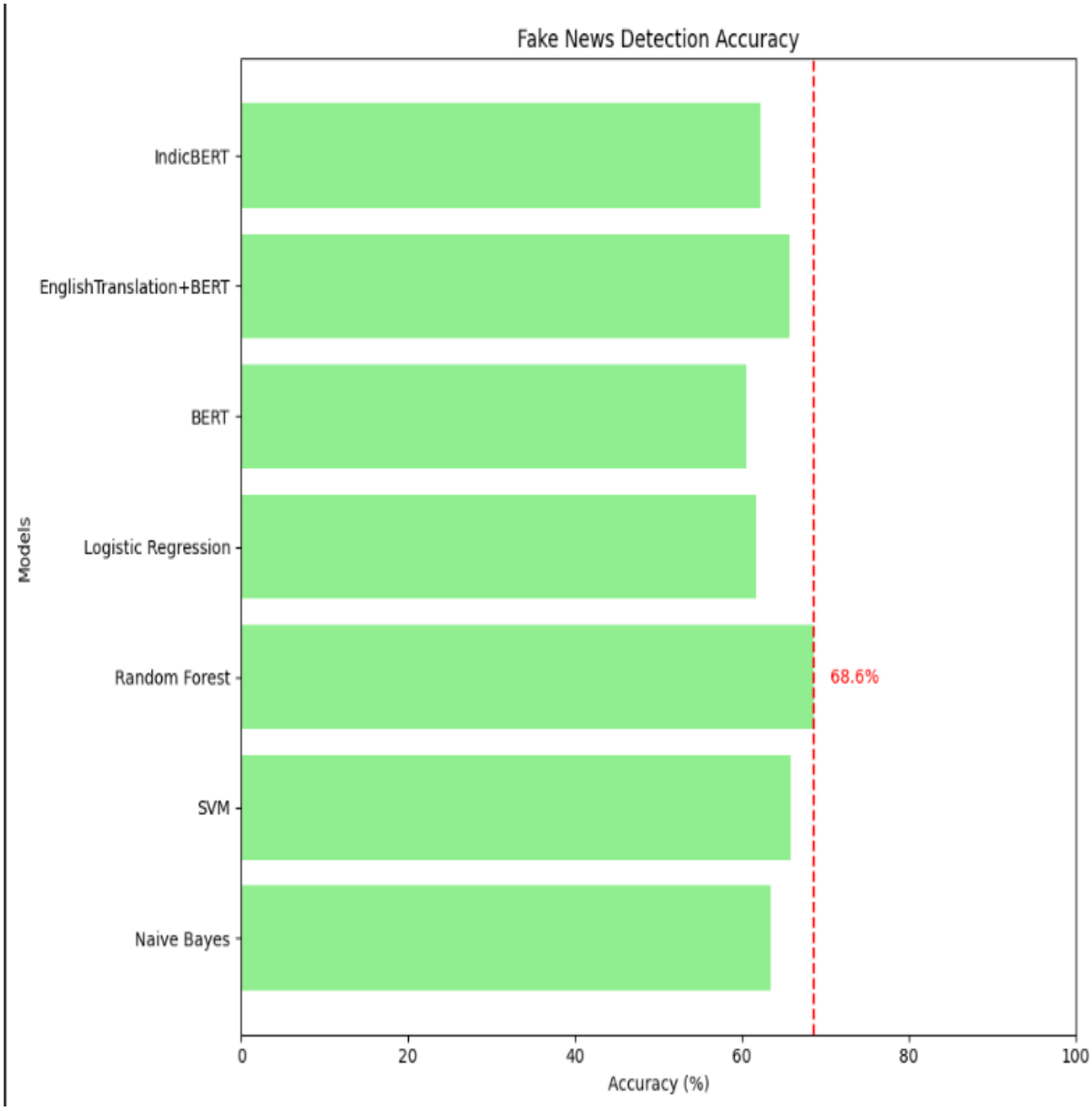

Fig 3- Fake News Detection Accuracy

Youtube Comment Dataset Models Confusion Matrices Result-:

Model	***Precision***	***Recall***	***F1 Score***
SVM	0.4137	0.3156	0.3651
Random Forest	0.4324	0.2573	0.3204
Naïve Bayes	0.4209	0.2823	0.3388
Logistic Regression	0.3828	0.3027	0.3049
BERT	0.4277	0.2663	0.3291
IndicBERT	0.4194	0.2762	0.3344

Table 5 – F1 Score for Youtube Comment Originality Detection on Malayalam Language Dataset

Model	***Precision***	***Recall***	***F1 Score***
SVM	0.4025	0.3331	0.3713
Random Forest	0.4280	0.2485	0.3088
Naïve Bayes	0.4030	0.2548	0.3119
Logistic Regression	0.3670	0.3338	0.3543
BERT	0.4272	0.2612	0.3237

Table 6 – F1 Score for Yobute Comment Originality Detection on English Translated Dataset

Tables 5 and 6 show the F1 scores of different models for detecting the originality of YouTube comments in Malayalam and English translated datasets, respectively. Random Forest performs best in both cases, followed closely by Naïve Bayes. The results suggest that while certain models maintain consistent performance across languages, translation may slightly affect their effectiveness.

Fake News Dataset Models Confusion Matrices Result-:

Model	***Precision***	***Recall***	***F1 Score***
SVM	0.5095	0.4442	0.4750
Random Forest	0.5280	0.3411	0.4199
Naïve Bayes	0.4700	0.3562	0.4036
Logistic Regression	0.4929	0.4092	0.4440
BERT	0.5682	0.3616	0.4461
IndicBERT	0.5542	0.2989	0.3832

Table 7 – F1 Score for Fake News Detection on Malayalma Language Dataset

Model	***Precision***	***Recall***	***F1 Score***
SVM	0.5588	0.4975	0.5222
Random Forest	0.5245	0.3008	0.3872
Naïve Bayes	0.4942	0.3556	0.4146
Logistic Regression	0.5202	0.4145	0.4606
BERT	0.5693	0.3465	0.4294

Table 8 – F1 Score for Fake News Detection on English Translated Dataset

Tables 7 and 8 display the F1 scores of various models for detecting fake news in Malayalam and English translated datasets, respectively. Across both datasets, SVM and Logistic Regression exhibit strong performance, with BERT also showing competitive results. However, there are variations in model effectiveness between the original and translated datasets, suggesting potential impacts of language translation on model performance.

We evaluated the performance of various machine learning models on two distinct datasets: one composed of Malayalam YouTube comments aimed at distinguishing between original and fake content, and the other focusing on Malayalam news articles for fake news detection. Initially, we observed that across both datasets, models such as Naive Bayes, Random Forest, Logistic Regression, BERT, and IndicBERT exhibited competitive accuracies. However, there was a notable discrepancy when comparing the performance on the original Malayalam dataset versus the English translated dataset.

Interestingly, upon applying the models to the English translated dataset, we found a notable increase in accuracy for certain models, notably Naive Bayes and Random Forest, in both the YouTube comment originality and fake news detection tasks. Additionally, the training time for the translated dataset was generally reduced compared to the original Malayalam dataset. This observation suggests that while the translation process introduces linguistic variations, it may also enhance the model's ability to discern patterns in the data. Moreover, the reduced training time for the translated dataset implies that the models may require less computational resources to achieve comparable performance, potentially enhancing their scalability and efficiency in real-world applications. These findings underscore the importance of considering language translation techniques in multilingual text analysis tasks, offering insights into optimizing model performance and resource utilization.

Chapter 6

Conclusion and Future Work

In conclusion, the empirical exploration of sentiment analysis within Malayalam datasets reveals critical insights into the performance of both traditional and deep learning models. The study's comparative approach highlights Naive Bayes as the frontrunner in accuracy for YouTube comment classification and Random Forest for fake news detection. Despite longer training times, BERT-based models provided in-depth linguistic analyses, though their accuracies did not surpass those of the traditional models.

A pivotal factor in the observed performance discrepancies is the size of the datasets. Deep learning models, like BERT, are inherently data-hungry and designed to excel with vast quantities of information. With 3,200 comments for YouTube and 1,700 articles for news, the datasets may be considered undersized for the deep learning models to fully capture and generalize the nuances of the Malayalam language. Meanwhile, the traditional models showed notable efficiency, possibly due to their suitability for smaller datasets and less complex feature spaces.

The constrained dataset size, therefore, emerges as a significant variable, possibly curbing the potential of the deep learning algorithms. It is plausible that with an expanded dataset, these models could outperform their traditional counterparts, given their capacity to understand and process complex patterns in language.

This research thus not only sheds light on the capabilities of current sentiment analysis models for Malayalam texts but also underscores the critical need for larger datasets to fully exploit the prowess of deep learning techniques. Moving forward, expanding the datasets or employing data augmentation strategies might be necessary to elevate the performance of deep learning models and to establish a more definitive benchmarking within the field of multilingual sentiment analysis. The study serves as a stepping stone for future research, emphasizing the balance between model choice, dataset size, and computational efficiency in the pursuit of accurate sentiment analysis.

Appendices

Appendix 1

Code

Code Snippets for all the steps involved and models used -:

Vectorization

```
# Vectorize the text using TF-IDF
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(fake_train['text'])
y_train = fake_train['label']
```

```
X_dev = vectorizer.transform(fake_dev['text'])
y_dev = fake_dev['label']
```

Naïve Bayes

```
# Initialize a Naive Bayes classifier
classifier_nb = MultinomialNB()
classifier_nb.fit(X_train, y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
# Predict on the development set
predictions_nb = classifier_nb.predict(X_dev)
```

SVM

```
    classifier_svm = SVC(kernel='rbf')
    classifier_svm.fit(X_train, y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

```
    # Predict on the development set
    predictions_svm = classifier_svm.predict(X_dev)
```

Random Forest

```
param_grid = {'n_estimators': [50, 100, 200],
              'max_depth': [None, 10, 20, 30],
              'min_samples_split': [2, 5, 10],
              'min_samples_leaf': [1, 2, 4]}
```

+ Code + Markdown

```
classifier = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(classifier, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score='raise',
       estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=10, n_jobs=1, oob_score=False, random_state=42,
            verbose=0, warm_start=False),
       fit_params={}, iid=True, n_jobs=-1,
       param_grid={'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='accuracy', verbose=0)
```

```
# Predict on the development set
best_classifier_rf = grid_search.best_estimator_
predictions_rf = best_classifier_rf.predict(X_dev)
```

Logistic Regression

```
param_grid_lr = {'C': [0.001, 0.01, 0.1, 1, 10, 100],
                 'solver': ['liblinear', 'lbfgs', 'newton-cg', 'sag']}
```

```
classifier_lr = LogisticRegression(max_iter=1000, random_state=42)
grid_search_lr = GridSearchCV(classifier_lr, param_grid_lr, cv=5, scoring='accuracy', n_jobs=-1)
grid_search_lr.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score='raise',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=1000, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False),
       fit_params={}, iid=True, n_jobs=-1,
       param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100], 'solver': ['liblinear', 'lbfgs', 'newton-cg', 'sag']},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='accuracy', verbose=0)
```

```
# Predict on the development set
best_classifier_lr = grid_search_lr.best_estimator_
predictions_lr = best_classifier_lr.predict(X_dev)
```

BERT Tokenizer

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize the text
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=512, return_tensors="tf")
val_encodings = tokenizer(val_texts, truncation=True, padding=True, max_length=512, return_tensors="tf")
```

INDIC BERT Tokenizer

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained('ai4bharat/indic-bert')


# Tokenize the text
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=512, return_tensors="tf")
val_encodings = tokenizer(val_texts, truncation=True, padding=True, max_length=512, return_tensors="tf")
```

BERT MODEL

```
from transformers import TFBertForSequenceClassification

model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(label_mapping))

# Prepare the model for training
optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
```

Malayalam to English Translation Coede

```
from transformers import pipeline
import pandas as pd
translator = pipeline("translation_ml_to_en", model="Helsinki-NLP/opus-mt-ml-en")
```

```
from transformers import pipeline
from tqdm.auto import tqdm

# Initialize the translation pipeline
translator = pipeline("translation", model="Helsinki-NLP/opus-mt-ml-en")

def translate_texts(texts):
    translations = []
    for text in tqdm(texts, desc="Translating"):
        # Skip translation if the text is an empty string
        if text.strip():  # Checks if text is not just whitespace
            translation = translator(text, max_length=512)[0]['translation_text']
            translations.append(translation)
        else:
            # Append an empty string or a placeholder for empty inputs
            translations.append('')
    return translations

# Apply translation to the cleaned 'text' column
df['translated_text'] = translate_texts(df['News'].tolist())
```

```
Translating: 100% 1669/1669 [1:19:53<00:00, 1.76s/it]
```

REFERNCES

[1] Contreras Hernández, S., Tzili Cruz, M. P., Espínola Sánchez, J. M., & Pérez Tzili, A. (2023). Deep learning model for covid-19 sentiment analysis on twitter. New Generation Computing, 41(2), 189-212.

[2] Manias, G., Mavrogiorgou, A., Kiourtis, A., Symvoulidis, C., & Kyriazis, D. (2023). Multilingual text categorization and sentiment analysis: a comparative analysis of the utilization of multilingual approaches for classifying twitter data. Neural Computing and Applications, 35(29), 21415-21431.

[3] Amara, A., Hadj Taieb, M. A., & Ben Aouicha, M. (2021). Multilingual topic modeling for tracking COVID-19 trends based on Facebook data analysis. Applied Intelligence, 51, 3052-3073.

[4] Vianna, D., Carneiro, F., Carvalho, J., Plastino, A., & Paes, A. (2023). Sentiment analysis in Portuguese tweets: an evaluation of diverse word representation models. Language Resources and Evaluation, 1-50.

[5] Mohawesh, R., Maqsood, S., & Althebyan, Q. (2023). Multilingual deep learning framework for fake news detection using capsule neural network. Journal of Intelligent Information Systems, 60(3), 655-671.

[6] Madani, Y., Erritali, M., & Bouikhalene, B. (2023). A new sentiment analysis method to detect and Analyse sentiments of Covid-19 moroccan tweets using a recommender approach. Multimedia Tools and Applications, 82(18), 27819-27838.

[7] Anjum, & Katarya, R. (2023). HateDetector: Multilingual technique for the analysis and detection of online hate speech in social networks. Multimedia Tools and Applications, 1-28.

[8] Habimana, O., Li, Y., Li, R., Gu, X., & Yu, G. (2020). Sentiment analysis using deep learning approaches: an overview. Science China Information Sciences, 63, 1-36.

[9] Mello, C., Cheema, G. S., & Thakkar, G. (2023). Combining sentiment analysis classifiers to explore multilingual news articles covering London 2012 and Rio 2016 Olympics. International Journal of Digital Humanities, 5(2), 131-157.

[10] Shanmugavadivel, K., Sathishkumar, V. E., Raja, S., Lingaiah, T. B., Neelakandan, S., & Subramanian, M. (2022). Deep learning based sentiment analysis and offensive language identification on multilingual code-mixed data. Scientific Reports, 12(1), 21557.

[11] Kar, P., & Debbarma, S. (2023). Multilingual hate speech detection sentimental analysis on social media platforms using optimal feature extraction and hybrid diagonal gated recurrent neural network. The Journal of Supercomputing, 79(17), 19515-19546.

[12] Sidhu, S., Khurana, S. S., Kumar, M., Singh, P., & Bamber, S. S. (2023). Sentiment analysis of Hindi language text: a critical review. Multimedia Tools and Applications, 1-30.

[13] Lopez, C. E., & Gallemore, C. (2021). An augmented multilingual Twitter dataset for studying the COVID-19 infodemic. Social Network Analysis and Mining, 11(1), 102.

[14] Zardak, S. R., Rasekh, A. H., & Bashkari, M. S. (2023). Persian Text Sentiment Analysis Based on BERT and Neural Networks. Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 47(4), 1623-1634.

[15] Khan, L., Amjad, A., Ashraf, N., & Chang, H. T. (2022). Multi-class sentiment analysis of urdu text using multilingual BERT. Scientific Reports, 12(1), 5436.

[16] Subramanian, M., Chakravarthi, B. R., Shanmugavadivel, K., Pandiyan, S., Kumaresan, P. K., Palani, B., Singh, M., Raja, S., Vanaja, & S, Mithunajha. (2023). Overview of the Shared Task on Fake News Detection from Social Media Text. In Proceedings of the Third Workshop on Speech and Language Technologies for Dravidian Languages. Varna, Bulgaria: Recent Advances in Natural Language Processing.

[17] Kakwani, D., Kunchukuttan, A., Golla, S., Gokul, N. C., Bhattacharyya, A., Khapra, M. M., & Kumar, P. (2020, November). IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In Findings of the Association for Computational Linguistics: EMNLP 2020 (pp. 4948-4961).