SENTIMENTAL ANALYSIS OF TEXT

A MINI PROJECT REPORT

(22IT503 – MINI PROJECT)

Submitted by

PRANAV A 727722EUIT131

LOGESHKUMAR M 727722EUIT096

NITHISH KUMAR P 727722EUIT123

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade

Kuniamuthur, Coimbatore – 641008

Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

NOVEMBER 2024

SUSTAINABLE DEVELOPMENT GOALS

The Sustainable Development Goals are a collection of 17 global goals designed to blue print to achieve a better and more sustainable future for all. The SDGs, set in 2015 by the United Nations General Assembly and intended to be achieved by the year 2030, In 2015, 195 nations agreed as a blue print that they can change the world for the better. The project is based on one of the 17 goals.

Questions	Answer Samples
Which SDGs does the project directly address?	SDG 16: Peace, Justice, and Strong Institutions
What strategies or actions are being implemented to achieve these goals?	SDG 9: Industry, Innovation, and Infrastructure
How is progress measured and reported in relation to the SDGs?	SDG 17: Partnerships for the Goals
How were these goals identified as relevant to the project's objectives?	SDG 11: Sustainable Cities and Communities
Are there any partnerships or collaborations in place to enhance this impact?	SDG 17: Partnerships for the Goals

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2678001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

BONAFIDE CERTIFICATE

Certified that this project report titled **"SENTIMENTAL ANALYSIS OF TEXT"** is the bonafide work of **"Pranav A(727722EUIT31) , Logeshkumar M (727722EUIT096) , Nithish kumar P (727722EUIT123)"** who carried out the project work under my supervision.

SIGNATURE

Dr. M. Arunachalam M.E., Ph.D

HEAD OF THE DEPARTMENT

Information Technology,
Sri Krishna College of Engineering and Technology,
Kuniyamuthur, Coimbatore–641008.

SIGNATURE

Dr.G.Edwin Prem Kumar M.E.,Ph.D

SUPERVISOR

Professor
Information Technology,
Sri Krishna College of Engineering and Technology,
Kuniyamuthur, Coimbatore–641008**.**

Submitted for the Project viva-voce examination held on_____

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDEGEMENT

At this juncture, we take the opportunity to convey our sincere thanks and gratitude to management of the college for providing all the facilities to us. We wish to convey our gratitude to our college principal, **Dr. K. Porkumaran M.E.,Ph.D** for forwarding us to do our project and offering adequate duration to complete our project.

We would like to express our grateful thanks to **Dr. M. Arunachalam Ph.D.**, Head of Department of Information Technology for her encouragement and valuable guidance to this project.

Also, we extend our heartfelt gratitude to our diligent project coordinator **Ms. U.M. Ramya M.E.,** Assistant Professor, Department of Information Technology whose unwavering support and guidance significantly contributed to the success of this project.

We are grateful to our project guide **Dr. G. Edwin Prem Kumar M.E., Ph.D** Assistant Professor, Department of Information Technology for her contributions of time, ideas to our project. We are also grateful for the internal and external members of the project viva voce.

ABSTRACT

Sentimental analysis focuses on developing a sentiment analysis system to classify text as positive, negative, or neutral using the Naive Bayes algorithm. Sentiment analysis is widely used to extract opinions or emotions expressed in text, assisting in understanding customer feedback, reviews, and social media sentiments. Naive Bayes, a probabilistic and efficient algorithm, is chosen for its simplicity and effectiveness in text classification. Additionally, advanced techniques using BERT (Bidirectional Encoder Representations from Transformers) are explored to enhance context comprehension, particularly in complex cases such as sarcasm and ambiguity.The methodology includes preprocessing text through tokenization and stopword removal, extracting features using Bag of Words or TF-IDF, and training a Naive Bayes model on labeled data. To address the limitations of Naive Bayes in understanding deep context, BERT is incorporated for more nuanced predictions, aiming to improve accuracy in scenarios where traditional models may falter. Expected outcomes include a functional Naive Bayes classifier for sentiment prediction and enhanced model accuracy with BERT, which handles complex linguistic structures better.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYMS	ABBREVIATIONS
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
TF-IDF	Term Frequency-Inverse Document Frequency
SVM	Support Vector Machine
P(c/x)	Posterior probability
P(x/c)	Likelihood of features xxx given class
P(x)	Marginal Probability of features xxx
Accuracy	Ratio of correctly predicted Instances

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF SENTIMENT ANALYSIS

Sentiment analysis, or opinion mining, is a technique in natural language processing (NLP) used to analyze and interpret customer emotions, feedback, and opinions by categorizing text data as positive, negative, or neutral. This powerful tool enables businesses to gauge public sentiment and understand customers' perceptions, helping them make informed decisions. In marketing, sentiment analysis can uncover customer preferences, measure brand reputation, and inform targeted campaigns by understanding how customers feel about products or services. In customer service, it helps identify common pain points, allowing companies to address issues proactively and improve customer satisfaction. Social media monitoring further benefits from sentiment analysis by tracking public sentiment in real-time, enabling businesses to respond to crises quickly and manage brand image more effectively. By providing insights into customer emotions and opinions, sentiment analysis is invaluable in refining business strategies, boosting customer satisfaction, and enhancing overall brand perception.

1.2 PURPOSE OF THE PROJECT

The primary goal of this project is to develop an efficient sentiment analysis model by utilizing the Naive Bayes algorithm, while also exploring the role of BERT (Bidirectional Encoder Representations from Transformers) in boosting accuracy and precision. Naive Bayes is a classic probabilistic model that is computationally efficient and performs well with limited training data, making it a popular choice for text classification. However, in pursuit of greater accuracy, the project will also examine how BERT, a powerful language representation model, can capture nuanced language features and context more effectively than traditional models. By combining the simplicity of Naive Bayes with the depth of BERT, the

project aims to advance sentiment analysis techniques, providing robust insights into customer sentiments and opinions. Enhanced models that effectively capture sentiment can help businesses better understand their audience, respond proactively, and make data-driven decisions to improve customer engagement and satisfaction.

1.3 NAIVE BAYES ALGORITHM IN SENTIMENT ANALYSIS

The Naive Bayes algorithm is a probabilistic classifier rooted in Bayes' Theorem, which operates under the assumption that features within a dataset are independent—an assumption that simplifies calculations and makes the model efficient. Despite this "naive" simplification, Naive Bayes is remarkably effective for text classification tasks such as sentiment analysis, as it assigns probabilities to categories based on word frequency. This algorithm is highly favored for its simplicity, ease of implementation, and computational efficiency, especially in handling large datasets, which makes it ideal for applications that require fast, reliable processing. Its effectiveness in categorizing text with minimal preprocessing allows it to accurately capture sentiment, making it a valuable tool for sentiment analysis where both interpretability and speed are crucial.

1.4 MOTIVATION FOR USING BERT

Naive Bayes, while effective in many text classification tasks, has limitations when it comes to handling the complexities of natural language. It struggles with detecting sarcasm, understanding ambiguous phrases, and capturing contextual nuances due to its assumption of feature independence and reliance on word frequencies alone. This can lead to inaccurate sentiment predictions in cases where tone or meaning depends on context, like sarcasm or double meanings.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

Goularte, Fábio Bif, et al. "SentPT: A customized solution for multi-genre sentiment analysis of Portuguese-language texts." Expert Systems with Applications 245 (2024): 123075.

SentPT: A customized solution for multi-genre sentiment analysis of Portuguese-language texts by Goularte, Fábio Bif, et al., addresses the limitations of traditional sentiment analysis models when applied to non-English languages, specifically Portuguese. Most sentiment analysis tools are optimized for English, often failing to capture the linguistic and cultural nuances of other languages. The authors introduce "SentPT," a model tailored for Portuguese-language texts across various genres, such as news articles, social media posts, and reviews.

To create SentPT, the authors collected a dataset from diverse sources and implemented preprocessing techniques to handle unique features of Portuguese, such as tokenization, stemming, and managing idiomatic expressions. The model itself leverages both machine learning and deep learning methods, including support vector machines (SVMs) and neural networks, which they optimized to improve accuracy for Portuguese texts.

Their findings show that SentPT outperforms general-purpose sentiment analysis models when applied to Portuguese, accurately classifying sentiment across different types of content. This highlights the value of language-specific and genre-customized approaches in sentiment analysis. The authors conclude that SentPT represents a significant advancement in multilingual sentiment analysis, setting a foundation for further research into developing customized models for other languages and domains.

Todd, Andrew, James Bowden, and Yashar Moshfeghi. "Text-based sentiment analysis in finance: Synthesising the existing literature and exploring future directions." ***Intelligent Systems in Accounting, Finance and Management*** **31.1 (2024): e1549.**

Text-based sentiment analysis in finance: Synthesising the existing literature and exploring future directions by Todd, Andrew, James Bowden, and Yashar Moshfeghi reviews the current state of sentiment analysis applied to financial texts. Financial sentiment analysis seeks to extract and interpret sentiment from textual data, such as news articles, financial reports, and social media, to understand and predict market behaviors. The authors synthesize existing research, noting how sentiment analysis has been adapted to address the unique challenges in finance, such as interpreting domain-specific language and incorporating real-time data.

The study explores various methods, from traditional machine learning to advanced deep learning techniques, and discusses their effectiveness in analyzing financial sentiment. By examining these methods, the authors identify gaps in current research, such as the need for more robust datasets and models that can handle the nuances of financial language and trends over time.

The paper concludes by suggesting future directions, including the integration of multimodal data sources and the refinement of models to enhance sentiment analysis in finance. This research provides a comprehensive overview of the field and sets a framework for future innovations in sentiment analysis specifically geared toward financial applications.

Bilal, Azhar Ahmed, O. Ayhan Erdem, and Sinan Toklu. "Children's Sentiment Analysis from Texts by Using Weight Updated Tuned with Random Forest Classification." ***IEEE Access*** **(2024).**

Text-based sentiment analysis in finance: Synthesising the existing literature and exploring future directions" by Todd, Andrew, James Bowden, and Yashar Moshfeghi reviews the current state of sentiment analysis applied to financial texts. Financial sentiment analysis seeks to extract and interpret sentiment from textual data, such as news articles, financial reports, and social media, to understand and predict market behaviors.

The authors synthesize existing research, noting how sentiment analysis has been adapted to address the unique challenges in finance, such as interpreting domain-specific language and incorporating real-time data.The study explores various methods, from traditional machine learning to advanced deep learning techniques, and discusses their effectiveness in analyzing financial sentiment. By examining these methods, the authors identify gaps in current research, such as the need for more robust datasets and models that can handle the nuances of financial language and trends over time.

The paper concludes by suggesting future directions, including the integration of multimodal data sources and the refinement of models to enhance sentiment analysis in finance. This research provides a comprehensive overview of the field and sets a framework for future innovations in sentiment analysis specifically geared toward financial applications.

CHAPTER 3

SYSTEM ANALYSIS

3.1 SYSTEM OVERVIEW

The proposed system architecture for sentiment analysis integrates Naive Bayes and BERT to create a hybrid model that effectively combines the efficiency of traditional machine learning algorithms with the deep contextual understanding offered by modern transformer models, resulting in enhanced accuracy and performance. The system begins with the collection of text data, which can be sourced from diverse platforms such as social media, online reviews, or customer feedback. Once the data is gathered, it undergoes a thorough preprocessing stage, which includes removing noise elements such as special characters, stop words, and irrelevant information that might interfere with accurate sentiment analysis. Additionally, the text is tokenized and converted into a standardized format that is compatible with both Naive Bayes and BERT, ensuring seamless processing for both models.

In the next stage, Naive Bayes is employed for feature extraction. This algorithm analyzes the preprocessed text by converting it into bag-of-words or TF-IDF (Term Frequency-Inverse Document Frequency) features. These features capture the presence and significance of words in the text, and Naive Bayes calculates the probabilities of different sentiment categories—positive, negative, or neutral—based on these features. The Naive Bayes model is computationally efficient and performs well with large datasets, providing a quick and effective sentiment classification. However, Naive Bayes struggles with more complex aspects of language, such as sarcasm, ambiguous phrases, or context-dependent meanings, which leads to the need for a more advanced model.

To address these limitations, BERT is introduced to the system. While Naive Bayes processes the data efficiently, BERT is capable of deeper, more nuanced contextual understanding. Using its bidirectional transformer architecture, BERT evaluates the relationships between words in the context of the entire sentence, allowing it to capture dependencies and semantic meaning that are crucial for understanding sentiment. This includes recognizing sarcasm, understanding tone, and interpreting ambiguous phrases, which Naive Bayes might misclassify. BERT generates contextual embeddings for each word, transforming them into a representation that reflects their meaning within the broader context of the sentence. These embeddings are passed through the model's layers, allowing BERT to predict sentiment based on the overall meaning of the text.

The outputs from both Naive Bayes and BERT are then integrated in a decision-making layer to produce a final sentiment classification. This layer can use a weighted approach to combine the predictions from the two models, with Naive Bayes providing an initial, efficient sentiment classification, and BERT offering a contextual refinement. This hybrid approach enhances the accuracy of sentiment analysis, especially in cases where Naive Bayes might be uncertain or misclassify complex text. For example, in instances where sarcasm or emotional tone plays a significant role in sentiment, BERT's deep contextual analysis is more reliable. The final sentiment classification can be determined using a voting mechanism, where the system checks for consistency between the models, or through a confidence-based approach, where the output with the higher confidence score is selected. In cases where there is a significant discrepancy between Naive Bayes and BERT, BERT's prediction is typically given more weight due to its superior ability to capture complex language nuances.

Once the sentiment classification is made, the result undergoes post-processing. This stage might involve visualizing the sentiment data in a report or interactive dashboard, categorizing the sentiment as positive, negative, or neutral, and assigning an overall sentiment score that reflects the intensity of the sentiment. Post-processing can also include additional features such as sentiment trend analysis over time or summarizing key insights from the text data. The hybrid system, by combining the computational efficiency of Naive Bayes with the contextual depth of BERT, ensures a more robust sentiment analysis process that is capable of handling large datasets quickly while maintaining high levels of accuracy, particularly in understanding the subtleties and complexities of natural language. This approach provides a balanced solution that can be applied to a wide range of text analysis tasks, from customer feedback and social media monitoring to market research and brand sentiment analysis.

3.2 METHODOLOGY

The process of preparing text data for sentiment analysis begins with text preprocessing, which includes several essential steps aimed at cleaning and structuring the data for further analysis. Initially, the text is tokenized, which involves splitting the text into individual words or tokens.

Additionally, normalization is carried out to standardize the text, which includes converting all characters to lowercase, removing punctuation, and stemming orlemmatizing words to their base or root forms. This helps reduce variations of words to a single representation, such as "running" to "run," which improves the model's ability to generalize.

After preprocessing, the next step is feature extraction, where the text data is transformed into a format suitable for model training. Two popular techniques for this are Bag of Words (BoW) and Term Frequency-Inverse Document Frequency

(TF-IDF). In the Bag of Words model, the text is represented as a collection of unique words and their frequencies in the document. Each document is treated as a vector, where the length of the vector is equal to the size of the vocabulary, and the value of each element corresponds to the frequency of a specific word.

While this method is straightforward and computationally efficient, it may lose the order and context of words. TF-IDF, on the other hand, adjusts the word frequencies by considering how frequently words appear across all documents in the dataset. Words that are common across many documents are given lower weight, while those that are unique to specific documents are weighted more heavily. This helps capture the importance of less frequent but more meaningful words, making it particularly useful for tasks like sentiment analysis.

Once the text is preprocessed and the features have been extracted, the next step involves training the Naive Bayes model on labeled data. Naive Bayes is a probabilistic classifier that uses the extracted features to assign probabilities to different sentiment categories (e.g., positive, negative, neutral).

The model learns from labeled examples, calculating the likelihood of each sentiment class based on the features in the text. During training, the model adjusts its parameters to best fit the training data. Afterward, the model can be tested on unseen data to evaluate its performance and accuracy. BERT (Bidirectional Encoder Representations from Transformers), a deep learning- based model, can also be fine-tuned to further enhance sentiment detection.

3.3 SYSTEM ARCHITECTURE

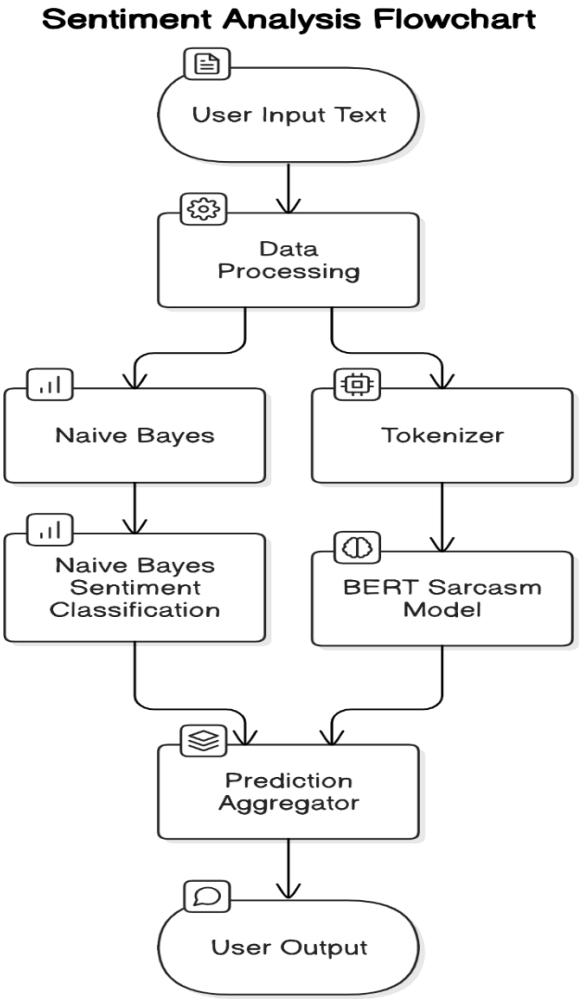

Figure 3.3.1 System architecture diagram

This sentiment analysis and sarcasm detection application is structured with layers for data loading, processing, analysis, and display. The Data Loading Layer uses the Sentiment140 dataset, cleans text, and vectorizes it with TF-IDF for Naive Bayes sentiment classification. The Prediction Combination Module integrates Naive Bayes and BERT outputs, prioritizing sarcasm detection for complex inputs. The User Interface Layer, built with Flask, provides a simple form for user input and displays sentiment and emoji feedback, ensuring a user-friendly experience with nuanced results.

CHAPTER 4

BAYES' THEOREM AND NAIVE BAYES MODEL

4.1 BAYES' THEOREM

Bayes' Theorem is a fundamental concept in probability theory and statistics that allows us to calculate the probability of an event, given some prior knowledge of related events. It provides a way of updating our beliefs about the likelihood of an event occurring, based on new evidence. In simpler terms, Bayes' theorem helps us revise the probability of an event happening based on the occurrence of other related events.

The formula for Bayes' Theorem is: $P(A|B)=P(B)P(B|A)\cdot P(A)/P(B)$

Bayes' Theorem works by combining our prior knowledge (P(A)) with the new evidence (P(B|A)to calculate the updated probability (P(A|B))This updated probability is often referred to as the **posterior probability**.

For example, suppose we want to determine the probability of a person being sick (event A) given that they have a cough (event B). Our prior knowledge of the likelihood of the person being sick, along with the likelihood of having a cough if sick, helps us calculate the probability of sickness after observing the cough. This process is more accurate than merely relying on initial assumptions, as it incorporates the new evidence.

4.2 APPLYING BAYES' THEOREM IN TEXT CLASSIFICATION

Bayes' theorem is widely applied in text classification tasks, especially for sentiment analysis, where the goal is to categorize a piece of text (such as areview or tweet) based on the sentiment expressed (positive, negative, or neutral). In this context, the theorem helps to compute the probability of a particular sentiment given the words or features present in the text. The sentiment categories (such as

"positive" or "negative") are treated as possible outcomes, while the words or terms in the text serve as evidence.

To apply Bayes' theorem for sentiment classification, we calculate the **posterior probability** of each sentiment category (e.g., "positive" or "negative") given the words in the text. This is done by using the following formula:

A key component of applying Bayes' theorem in text classification is the **Naive Bayes assumption**, which simplifies the problem considerably. This assumption posits that all features (i.e., words) are independent of each other given the sentiment class. In other words, the occurrence of each word is assumed to be independent of the others, which is a simplifying approximation. This assumption is "naive" because in reality, words in a text are often correlated, especially when they appear in certain contexts (e.g., "great" and "service" often appear together in positive reviews). However, despite this unrealistic assumption, Naive Bayes models still perform surprisingly well in practice for many text classification tasks.

This independence assumption simplifies the computation of the likelihood, as it allows the product of the individual word probabilities to be computed instead of needing to consider combinations of words. For instance, if we have a text with multiple words, instead of calculating the probability of all words occurring together, we can calculate the probability of each word occurring independently given the sentiment and multiply these probabilities. This drastically reduces the complexity of the model, making Naive Bayes both efficient and effective, particularly in situations with a large number of features (words in this case).

4.3 NAIVE BAYES MODEL IN SENTIMENT ANALYSIS

Implementing Naive Bayes for sentiment analysis involves several steps, starting with data collection and preprocessing. First, a dataset containing labeled text samples, such as product reviews with sentiment labels (positive, negative, neutral),

is gathered. The next step is to preprocess the text, which includes tokenizing the text into words, removing stopwords (common but meaningless words like "and", "the"), and converting the text to a standardized format (e.g., lowercase). Afterward, a feature extraction technique, like **Bag of Words** or **TF-IDF**, is used to convert the text data into numerical features that can be used by the model. These features represent the frequency or importance of words in the text.

Once the data is prepared, the Naive Bayes algorithm can be trained. This is done by calculating the prior probability for each sentiment category (how frequently each sentiment appears in the training set), and the likelihood of each word given the sentiment (how frequently each word appears in texts labeled with that sentiment). The **Laplace smoothing** parameter is often applied to avoid zero probabilities for words not seen during training. The model is then trained by using these probabilities to classify new, unseen text data.

During the training process, several parameters and settings are crucial for effective model performance. Key parameters include the **smoothing parameter** (usually α\alphaα) to handle unseen words, which ensures the model doesn't assign a zero probability to any unseen word. In addition, the model might require tuning based on the specific text corpus, such as adjusting the feature extraction method (e.g., choosing between **Bag of Words** and **TF-IDF**) or selecting the right threshold for classification. After training, the Naive Bayes model uses the learned probabilities to classify new texts by comparing the posterior probabilities of each sentiment category based on the words in the text. The sentiment with the highest posterior probability is selected as the predicted class.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DATA COLLECTION AND PREPROCESSING

For sentiment analysis, data can be sourced from various platforms where people express their opinions, such as Twitter, product review sites (e.g., Amazon, Yelp), or forums. Twitter is particularly popular for sentiment analysis due to its vast volume of real-time text data, which captures public sentiment on trending topics. Product reviews from e-commerce sites like Amazon provide rich data that is already labeled by star ratings, making it easy to infer sentiment categories (e.g., reviews with five stars as positive and one star as negative). Similarly, datasets from movie or restaurant reviews also offer sentiment-rich content suitable for training sentiment classifiers. The dataset chosen for analysis typically contains thousands of text entries, each labeled with a sentiment category (e.g., positive, negative, or neutral), and can vary in length from short sentences (as on Twitter) to detailed multi-sentence reviews (as on Amazon).

Preprocessing is an essential step before feeding the text data into a Naive Bayes model. It starts with **lowercasing** all text, which removes any inconsistencies caused by capitalized words (e.g., "Good" and "good" are treated as the same word). **Removing punctuation** follows, as symbols like periods and commas generally do not contribute to sentiment and can clutter the data. **Tokenization** is used to break down sentences into individual words, making it easier to analyze word-level sentiment. **Stopwords** (common words like "is", "the", "and") are removed because they do not carry specific sentiment and can dilute the model's focus on impactful words.

Further preprocessing steps include **stemming** and **lemmatization**. Stemming reduces words to their root forms (e.g., "running" to "run"), while

lemmatization reduces words to their base or dictionary form (e.g., "better" to "good"). This helps standardize words with the same meaning but different forms, allowing the model to capture sentiment more effectively. Finally, **vectorization** techniques like Bag of Words or TF-IDF convert the cleaned, processed text into numerical features that the Naive Bayes model can interpret. These steps ensure that only meaningful, sentiment-bearing words are used, improving the accuracy and efficiency of the model in classifying text.

5.2 MODEL BUILDING

To implement a Naive Bayes model for sentiment analysis, we can leverage the **scikit-learn** library, which provides robust tools for both the Naive Bayes classifier and feature extraction methods like **TF-IDF**. The process begins with setting up the Naive Bayes classifier, typically the Multinomial Naive Bayes variant, as it is well-suited for text classification. First, we prepare our dataset by splitting it into training and test sets, ensuring that our model is trained on a portion of the data and evaluated on unseen samples for accuracy. Using scikit-learn's **TfidfVectorizer**, we convert the text data into a **TF-IDF matrix**. This matrix represents each word's importance within individual documents, balancing word frequency with the word's overall rarity across the dataset. Once the TF-IDF features are ready, we initialize the Naive Bayes classifier, fit it to the training data, and then evaluate its performance on the test set, allowing us to assess its accuracy in predicting sentiment categories.

In addition to Naive Bayes, more advanced models like **BERT (Bidirectional Encoder Representations from Transformers)** can be used for sentiment analysis, especially for applications where context and word order are crucial. BERT is a pre-trained, transformer-based model that captures deep semantic and syntactic information from text, making it a

powerful choice for complex language tasks. To fine-tune BERT for sentiment analysis, we begin by loading a **pre-trained BERT model** (e.g., from the Hugging Face Transformers library) and adapt it to our specific dataset. We convert the text into embeddings using BERT's tokenizer, which transforms text into numerical representations suitable for model input. Fine-tuning involves training the model on labeled sentiment data, adjusting BERT's weights slightly to optimize it for our task without entirely retraining from scratch. This process typically includes adding a classification layer on top of BERT that outputs probabilities for each sentiment category. By leveraging pre-trained embeddings, BERT can accurately capture contextual nuances in sentiment, even when words or phrases are highly context-dependent, resulting in a powerful sentiment classifier.

5.3 MODEL TRAINING AND TESTING

To ensure reliable model performance, the data is first split into **training** and **testing sets**, typically using an 80-20 or 70-30 ratio. The training set is used to teach the model the relationship between text features and sentiment labels, while the test set provides a way to evaluate how well the model generalizes to unseen data. This split is crucial to avoid overfitting, where the model performs well on training data but poorly on new examples.

The **training process** for the Naive Bayes model and BERT varies due to their structural differences. For Naive Bayes, once the training data is vectorized (e.g., using TF-IDF), the model is trained by estimating probabilities for each word given a sentiment category. This step involves calculating prior probabilities for each class and the likelihood of each word within each class. Training Naive Bayes is relatively fast and straightforward since it relies on simple probabilistic relationships. In contrast, **training BERT** for sentiment analysis typically involves loading a pre-trained BERT model and fine-tuning it on the labeled dataset. BERT training requires text to be tokenized into embeddings, capturing word meanings in context.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 SUMMARY OF FINDINGS

In summary, the project aimed to explore and compare the effectiveness of two sentiment analysis models—**Naive Bayes** and **BERT**—in classifying text data across a range of sentiment complexities. The primary goal was to understand how each model performs on both straightforward and nuanced sentiment tasks, highlighting strengths and trade-offs. Through a combination of training, evaluation, and metric analysis, we found that Naive Bayes excels in **basic sentiment classification** due to its simplicity and speed. It is especially well-suited for datasets where sentiment is conveyed through clear, unambiguous language, making it ideal for applications with minimal computational resources and time constraints. However, Naive Bayes' independence assumption and lack of contextual understanding limit its effectiveness in scenarios where sentiment depends on context or tone, such as in sarcastic or ambivalent expressions.

BERT, on the other hand, demonstrated a strong capacity for handling **complex, context-dependent sentiment**. Its transformer-based architecture, which reads text bidirectionally, allows BERT to capture the nuances of language, making it ideal for tasks where accurate sentiment detection relies on understanding the relationship between words within context. This makes BERT especially effective in social media sentiment analysis, product reviews with mixed feedback, and any scenario involving subtle emotional cues. Although BERT provides superior accuracy and sensitivity to language subtleties, this comes at a cost; BERT requires significant computational resources, longer processing times, and specialized hardware for optimal performance.

In conclusion, the choice between Naive Bayes and BERT ultimately depends on the specific needs and resources of the application. **Naive Bayes** is highly suitable for **basic sentiment classification tasks**, offering quick, reasonably accurate results with minimal computational demands. **BERT** shines in **complex sentiment analysis**, where understanding context and depth of language is critical, though at the expense of higher computational cost. Together, these findings highlight the importance of aligning model choice with project goals, ensuring a balance between **efficiency** and **accuracy** based on the complexity of the sentiment analysis task at hand.

INPUT	OUTPUT
He is happy	Positive
He is dead	Negative
Hey hello	Neutral

Table 6.1.1 – Result table

6.2 FUTURE IMPROVEMENTS

To further enhance the accuracy and effectiveness of sentiment analysis, several strategies could be explored. One key area for improvement is the **detection of sarcasm**, which remains a significant challenge in traditional sentiment analysis models. Sarcasm often relies on context, tone, or ironic statements, which models like Naive Bayes and even BERT can struggle with. To address this, **dedicated sarcasm detection models** could be integrated into the sentiment analysis pipeline. These models, trained specifically to recognize patterns indicative of sarcasm (such as contradictory word usage or exaggerated statements), can improve sentiment classification, especially in social media posts and informal text, where sarcasm is prevalent.

Another enhancement lies in leveraging **domain-specific datasets** to tailor sentiment analysis to particular industries or subject areas. For instance, sentiment analysis in the context of healthcare, finance, or customer service can benefit from models trained on specialized terminology and phrases. By integrating **domain-specific knowledge**, sentiment models can become more accurate in interpreting specialized language, improving the precision ofsentiment predictions in those areas. Fine-tuning general models like BERT on industry-specific data can help capture nuances that would otherwise be missed by a more generic model.

Additionally, there is potential in exploring **hybrid models or ensembles**, which combine the strengths of different techniques to improve sentiment analysis accuracy. One promising approach is the combination of Naive Bayes with more advanced deep learning models, such as BERT or LSTMs (Long Short-Term Memory networks). A hybrid model could leverage Naive Bayes for quick, computationally efficient sentiment classification on straightforward cases, while using a more sophisticated deep learning model like BERT for analyzing complex, ambiguous, or context-sensitive text. Ensemble methods, where the predictions of multiple models are combined, could further improve accuracy by reducing the impact of errors from individual models, particularly in cases with mixed sentiments or subtle emotional tones.

Incorporating these suggestions—sarcasm detection, domain-specific datasets, and hybrid or ensemble models—could lead to significant improvements in sentiment analysis, making it more robust, context-aware, and accurate across diverse types of text and sentiment complexity. These advancements would be especially beneficial in real-world applications like social media monitoring, customer feedback analysis, and content moderation, where the variety of language use and emotional expression can pose challenges to traditional models.

6.3 FINAL THOUGHTS

The findings from this project have significant **implications for real-world applications** in sentiment analysis, particularly in industries where understanding customer sentiment, public opinion, or social media trends is crucial. For example, **Naive Bayes** could be highly effective in environments where quick and resource-efficient sentiment classification is needed, such as in **automated customer service**, **feedback analysis**, and **spam detection**. Its speed and efficiency make it suitable for processing large volumes of simple text data in real-time, offering businesses a way to quickly gauge customer reactions or flag negative feedback for immediate response. On the other hand, **BERT's** advanced capabilities in understanding nuanced language make it invaluable in settings where sentiment is more complex, such as in **social media monitoring**, **brand reputation management**, and **market research**. By accurately detecting sarcasm, mixed emotions, or subtle changes in sentiment, BERT can help companies gain deeper insights into consumer feelings, even in the face of ambiguous or intricate language.

These findings underscore the **value of combining traditional and modern NLP approaches** in sentiment detection. **Naive Bayes** offers simplicity, efficiency, and effectiveness in situations where quick sentiment categorization is necessary, while **BERT** brings the power of contextual language understanding for tasks where precision and the ability to handle complex language are paramount. By integrating both methods, companies and researchers can **optimize sentiment analysis systems**, using Naive Bayes for straightforward classification tasks and deploying BERT for more complex, context-sensitive analyses. This hybrid approach ensures a balance between **computational efficiency** and **deep contextual understanding**, enabling organizations to tailor sentiment analysis systems to their specific needs and resource constraints. Ultimately, this combination enhances the ability to capture a more comprehensive and accurate picture of public sentiment, unlocking valuable insights for decision-making and strategy in various domains.

APPENDICES

APPENDIX 1

SOURCE CODE

app.py

```
from flask import Flask, render_template, request

from data_processing import load_and_preprocess_data

from model_training import train_naive_bayes_model

from sentiment_prediction import combine_predictions,
get_sarcasm_pipeline


# Initialize Flask app

app = Flask(__name__)


# Load data and models

file_path = './sentiment140.csv'

train_vectors, train_labels, vectorizer =
load_and_preprocess_data(file_path)

nb_classifier = train_naive_bayes_model(train_vectors, train_labels)

sarcasm_pipeline = get_sarcasm_pipeline()


@app.route('/')

def index():

    return render_template('index.html')
```

```
@app.route('/analyze', methods=['POST'])
def analyze_sentiment():
    text = request.form['text']  # Capture the user input
    combined_result = combine_predictions(text, vectorizer, nb_classifier, sarcasm_pipeline)

    # Determine sentiment label and emoji based on the prediction
    if combined_result == 1:
        emoji = '😊'  # Positive
        sentiment_label = "Positive"
    elif combined_result == 0:
        emoji = '😞' # Negative
        sentiment_label = "Negative"
    else:
        emoji = '😐'  # Neutral
        sentiment_label = "Neutral"

    # Pass the input text, sentiment label, and emoji to the template
    return render_template('index.html', input_text=text, sentiment_label=sentiment_label, emoji=emoji)

if __name__ == '__main__':
    app.run(debug=True)
```

data_processing.py

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB


def load_and_preprocess_data(file_path):
    # Load Sentiment140 Dataset
    sentiment140 = pd.read_csv(file_path, encoding='latin-1', header=None, names=['polarity', 'id', 'date', 'query', 'user', 'text'])
    # Map polarity: 0 (negative), 2 (neutral), 4 (positive)
    sentiment140['label'] = sentiment140['polarity'].apply(lambda x: 2 if x == 2 else (1 if x == 4 else 0))
    # Use only text and label columns
    df_sentiment140 = sentiment140[['text', 'label']]
    # Vectorize text data
    vectorizer = TfidfVectorizer()
    train_vectors = vectorizer.fit_transform(df_sentiment140['text'])
    train_labels = df_sentiment140['label']
    return train_vectors, train_labels, vectorizer
```

model_training.py

```
from sklearn.naive_bayes import MultinomialNB
def train_naive_bayes_model(train_vectors, train_labels):
```

```
    nb_classifier = MultinomialNB()
    nb_classifier.fit(train_vectors, train_labels)
    return nb_classifier
```

sentiment_analysis.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from transformers import BertTokenizer, BertForSequenceClassification, pipeline
import torch

# Load Sentiment140 Dataset
file_path = './sentiment140.csv'  # Replace with your file path
sentiment140 = pd.read_csv(file_path, encoding='latin-1', header=None, names=['polarity', 'id', 'date', 'query', 'user', 'text'])

# Map polarity: 0 (negative), 2 (neutral), 4 (positive)
sentiment140['label'] = sentiment140['polarity'].apply(lambda x: 2 if x == 2 else (1 if x == 4 else 0))

# Use only text and label columns
df_sentiment140 = sentiment140[['text', 'label']]
```

```
    # Preprocess and Vectorize
    vectorizer = TfidfVectorizer()
    train_vectors = vectorizer.fit_transform(df_sentiment140['text'])
    train_labels = df_sentiment140['label']

    # Train Naive Bayes Model
    nb_classifier = MultinomialNB()
    nb_classifier.fit(train_vectors, train_labels)

    # Load BERT for sarcasm detection
    tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
    sarcasm_model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=2)
    sarcasm_pipeline = pipeline("text-classification", model=sarcasm_model, tokenizer=tokenizer)

    # Define prediction functions
    def get_sentiment_nb(text, threshold=0.7):
        text_vector = vectorizer.transform([text])
        prediction_proba = nb_classifier.predict_proba(text_vector)
        max_proba = max(prediction_proba[0])
        if max_proba >= threshold:
            return nb_classifier.predict(text_vector)[0]  # 0 (negative), 1 (positive), 2 (neutral)
        else:
```

```
            return -1  # Indicate low confidence for neutral


    def get_sarcasm_bert(text):
        prediction = sarcasm_pipeline(text)[0]
        return 1 if prediction['label'] == 'LABEL_1' else 0


    def combine_predictions(text):
        nb_prediction = get_sentiment_nb(text)
        if len(text.split()) <= 3:
            return nb_prediction if nb_prediction != -1 else 2 # Assume neutral
if uncertain


        sarcasm_prediction = get_sarcasm_bert(text)
        if nb_prediction == -1:
            return sarcasm_prediction
        return nb_prediction if nb_prediction == sarcasm_prediction else
sarcasm_prediction


    # User Interface
    while True:
        print("\nEnter a sentence for analysis (type 'exit' to quit):")
        user_input = input()
        if user_input.lower() == 'exit':
            print("Exiting...")
```

```
        break


    combined_result = combine_predictions(user_input)

    sentiment_label = "Positive" if combined_result == 1 else "Negative"
if combined_result == 0 else "Neutral"

    print(f"Combined Sentiment Prediction: {sentiment_label}")
```

APPENDIX 2

SCREENSHOTS

Figure A.2.1 – Positive sentence input

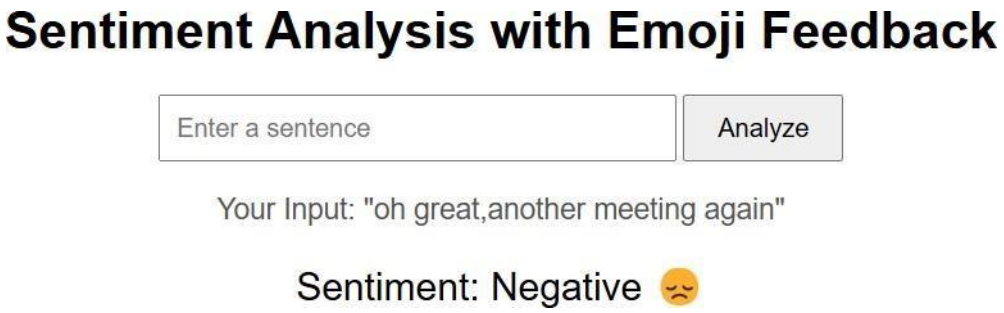

Figure A.2.2 – Negative sentence input

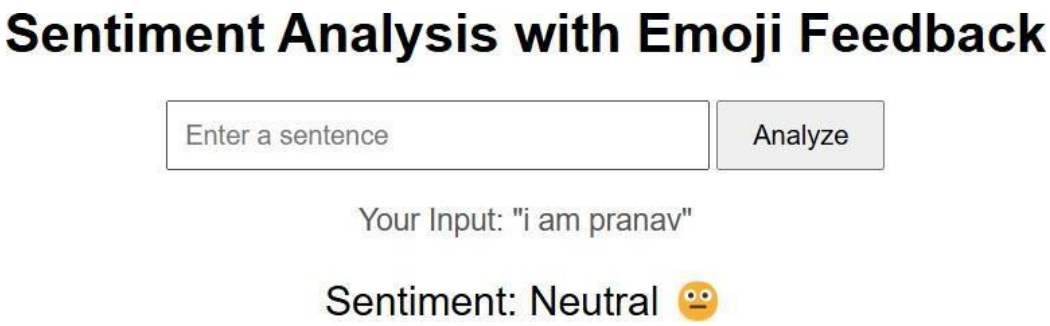

Figure A.2.3 – Neutral sentence input

REFERENCES

[1] Alnasrawi, Ali Mohamed, Asia Mahdi Naser Alzubaidi, and Ahmed Abdulhadi Al-Moadhen. "Improving sentiment analysis using text network features within different machine learning algorithms." *Bulletin of Electrical Engineering and Informatics* 13.1 (2024): 405-412.

[2] Ba Alawi, Abdulfattah, and Ferhat Bozkurt. "Performance Analysis of Embedding Methods for Deep Learning-Based Turkish Sentiment Analysis Models." *Arabian Journal for Science and Engineering* (2024): 1-23.

[3] Das, Ringki, and Thoudam Doren Singh. "Multimodal sentiment analysis: a survey of methods, trends, and challenges." *ACM Computing Surveys* 55.13s (2023): 1-38.

[4] Goularte, Fábio Bif, et al. "SentPT: A customized solution for multi-genre sentiment analysis of Portuguese-language texts." *Expert Systems with Applications* 245 (2024): 123075.

[5] Loginova, Ekaterina, et al. "Forecasting directional bitcoin price returns using aspect-based sentiment analysis on online text data." *Machine Learning* 113.7 (2024): 4761-4784.

[6] Nath, Deena, and Sanjay K. Dwivedi. "Aspect-based sentiment analysis: approaches, applications, challenges and trends." *Knowledge and Information Systems* 66.12 (2024): 7261-7303.

[7] Taherdoost, Hamed, and Mitra Madanchian. "Artificial intelligence and sentiment analysis: A review in competitive research." *Computers* 12.2 (2023): 37.

[8] Tan, Kian Long, Chin Poo Lee, and Kian Ming Lim. "A survey of sentiment analysis: Approaches, datasets, and future research." *Applied Sciences* 13.7 (2023): 4550.

[9] Todd, Andrew, James Bowden, and Yashar Moshfeghi. "Text-based sentiment analysis in finance: Synthesising the existing literature and exploring future directions." *Intelligent Systems in Accounting, Finance and Management* 31.1 (2024): e1549.