Sentiment Analysis For Marathi Language

NLP mini project submitted in partial fulfillment of the
requirements of the degree of

B. E. Computer Engineering

By

Samit Fernandes 03 (212031)

Jaden Franco 04 (212032)

Name of the Guide: Pradnya Sawant
Designation: Assistant Professor

Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

University of Mumbai
2024-2025

CERTIFICATE

This is to certify that the mini project entitled “**Sentiment Analysis For Marathi Language”** is a bonafide work of **Samit Fernandes (03)** and **Jaden Franco (04)** submitted to the University of Mumbai in partial fulfillment of the requirement for the NLP subject in the final year of Computer Engineering.

Ms. Pradnya Sawant
Guide

Declaration

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

Samit Fernandes 03
Jaden Franco 04

Date:

Abstract

This project presents a sentiment analysis system for Marathi text, utilizing Natural Language Processing (NLP) techniques to classify the sentiment as positive, neutral, or negative. The primary focus of the work is on the preprocessing of Marathi text data, which includes tokenization, stop-word removal, and normalization processes tailored to the Marathi language. A Multinomial Naive Bayes model is employed as the predictive mechanism, trained on labeled Marathi text datasets. The model is evaluated based on its ability to accurately classify sentiment across a variety of text samples. Results demonstrate the effectiveness of machine learning methods like Multinomial Naive Bayes in handling sentiment classification tasks in low-resource languages such as Marathi. This study contributes to the growing field of NLP for regional languages and provides a foundational approach for further research and development in sentiment analysis systems.

Contents

List of Figures

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
1	NLP	Natural Language Processing
2	NLTK	Natural Language Toolkit
3	MNB	Multinomial Naive Bayes

Chapter 1

Introduction

1.1 Description

Sentiment analysis, also known as opinion mining, is an essential task in Natural Language Processing (NLP), aimed at determining the sentiment conveyed by a given piece of text. This project focuses on performing sentiment analysis on Marathi text, a widely spoken language in India but relatively underrepresented in the field of NLP research. The main objective is to classify the sentiment of the text as positive, neutral, or negative using machine learning techniques.

Due to the rich morphology and syntactic complexity of Marathi, sentiment analysis poses unique challenges, especially in the preprocessing phase. Marathi text often contains linguistic variations, informal expressions, and a lack of standardized writing, all of which make it difficult to handle using conventional NLP methods. This project employs a Multinomial Naive Bayes model to predict sentiment based on features extracted from preprocessed text data. Preprocessing steps such as tokenization, stop-word removal, and normalization are customized to accommodate the specific characteristics of the Marathi language. The dataset for this project comprises labeled Marathi text samples, which are essential for training and validating the logistic regression model.

The significance of this project lies in its contribution to the growing body of work on sentiment analysis for low-resource languages. While sentiment analysis has been widely explored for languages like English and Hindi, limited research has been conducted for regional languages like Marathi. The implementation of the Multinomial Naive Bayes model demonstrates how machine learning algorithms can be effective when applied to such low-resource languages with appropriate preprocessing techniques. This study provides a foundation for further exploration in NLP for Marathi, opening avenues for future research and more sophisticated language models.

1.2 Problem Formulation

The rise of digital content in regional languages has created a growing demand for sentiment analysis in low-resource languages like Marathi. However, most existing Natural Language Processing (NLP) tools and models are primarily designed for widely spoken languages such as English, leaving regional languages with limited resources and tools for sentiment analysis. Marathi, despite being one of the most widely spoken languages in India, lacks sufficient datasets, linguistic resources, and tailored models for effective sentiment analysis.

This project aims to address the challenge of performing sentiment analysis on Marathi text by leveraging supervised machine learning techniques. The primary problem is to classify the sentiment of a given Marathi text as either positive, neutral, or negative. To achieve this, the project focuses on the following key challenges:

1. Text Preprocessing for Marathi: Marathi text presents unique linguistic characteristics such as complex morphology, informal variations, and the absence of a standardized form of writing. Preprocessing steps such as tokenization, normalization, and stop-word removal need to be carefully adapted to handle these intricacies.

2. Feature Extraction and Representation: Converting raw text into a structured format suitable for machine learning involves selecting appropriate features, such as word frequencies or TF-IDF scores, that can capture the sentiment-related patterns in Marathi text.

3. Sentiment Classification: The task is to develop a model that can accurately predict whether a piece of text conveys a positive, neutral, or negative sentiment. Multinomial Naive Bayes is chosen as the classification algorithm due to its efficiency, scalability, and strong performance in text classification tasks, particularly when dealing with word frequency features in multiclass problems.

4. Model Evaluation and Optimization: It is necessary to assess the model's performance using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.

Additionally, techniques like hyperparameter tuning may be required to optimize the model's performance.

The goal of this project is to create a sentiment analysis system that effectively handles the challenges of working with Marathi text, offering reliable sentiment classification for low-resource languages.

1.3 Motivation

The explosion of user-generated content on social media, blogs, and forums in regional languages has created an increasing demand for sentiment analysis in languages beyond English and other globally dominant languages. Marathi, spoken by over 83 million people, is one such regional language that has seen significant growth in online content. However, despite its widespread use, Marathi remains a low-resource language in terms of Natural Language Processing (NLP) tools and datasets, making it difficult to apply advanced sentiment analysis techniques. This project is motivated by the need to bridge that gap.

The growing influence of sentiment analysis in various fields such as market research, public opinion monitoring, and customer feedback analysis further highlights the importance of extending these capabilities to regional languages. In multilingual societies like India, analyzing sentiment in native languages such as Marathi can provide deeper insights into public sentiment that are otherwise inaccessible through analysis in English or Hindi. Developing NLP tools for Marathi can enhance communication and engagement across sectors, including government, businesses, and social platforms, by providing a better understanding of user opinions and emotions.

While deep learning methods have gained popularity in recent years, traditional algorithms, when paired with language-specific preprocessing techniques, can still deliver reliable and interpretable results, especially in low-resource settings. This project seeks to demonstrate that by focusing on effective preprocessing tailored to Marathi text, a simple yet powerful machine learning model can achieve satisfactory sentiment classification outcomes.

1.4 Proposed Solution

1. Dataset: L3CubeMahaSent

The dataset used for this project is L3CubeMahaSent, which is the largest publicly available Marathi sentiment analysis dataset to date. This dataset consists of 18,378 Marathi tweets, manually labeled into three sentiment categories: positive (1), neutral (0), and negative (-1). The dataset is representative of real-world data as the tweets are provided in their original, unprocessed form.

Dataset Statistics:

Training set: 12,114 tweets (4,038 per class)

Test set: 2,250 tweets (750 per class)

Validation set: 1,500 tweets (500 per class)

The dataset is carefully split to maintain balance across sentiment classes, avoiding class imbalance. An additional set of 2,514 tweets is provided in a separate file but was not used during the baseline experiments. This dataset is ideal for training the sentiment analysis model due to its size, diversity, and well-structured labeling.

2. Text Preprocessing

Due to the nature of raw text data, especially in regional languages like Marathi, preprocessing is a critical step. The following NLP techniques are employed to clean and prepare the text for the classification model:

- Tokenization: The process of breaking down the input text into individual words or tokens is crucial for analyzing text. For this, the NLTK library is used to tokenize Marathi sentences into words.
- Stop-word Removal: Stop-words (common words that do not contribute significantly to the meaning of the text, such as “आहे”, “होतो” in Marathi) are removed to reduce noise and improve the focus on sentiment-bearing words.
- Normalization: The text is cleaned by removing special characters, punctuation, and numbers, and by standardizing words that have various forms due to Marathi’s rich morphology.

These preprocessing steps ensure that the text is in a clean and standardized format, suitable for feature extraction and model training.

3. Feature Extraction

Once the text has been preprocessed, it is transformed into a numerical format suitable for machine learning. This is done using the CountVectorizer technique from Scikit-learn:

CountVectorizer: This technique converts the cleaned text into a bag-of-words representation, where each unique word in the corpus is treated as a feature. The vectorizer counts the occurrences of each word in the text and transforms it into a numerical matrix that can be used for classification. This method captures the frequency of words across documents, which is crucial for the model to understand word importance and patterns.

4. Multinomial Naive Bayes Algorithm

For sentiment classification, Multinomial Naive Bayes (MNB) is chosen as the machine learning model. This algorithm is well-suited for text classification tasks where the features are word counts or frequencies.

Why Multinomial Naive Bayes?

Multinomial Naive Bayes is particularly effective in dealing with text data because it models the probability of a word belonging to a particular class (sentiment) based on its frequency in the text. It assumes that each word contributes independently to the sentiment of the text, making it computationally efficient and suitable for large datasets. The algorithm works by estimating the conditional probability of a class given the features (words), and it is known for its strong performance in document classification tasks. Additionally, it scales well with larger datasets and provides reliable results.

5. Model Training and Evaluation

The Multinomial Naive Bayes model is trained on the preprocessed and vectorized Marathi text from the training dataset. The model learns to associate word frequency patterns with sentiment categories (positive, neutral, and negative).

The model’s performance is evaluated using the validation and test datasets. Classification reports are generated to measure key performance metrics, such as accuracy, precision, recall, and F1-score, for each sentiment class. Additionally, confusion matrices are plotted to visualize the model’s performance in correctly classifying the sentiment categories.

6. Sentiment Prediction

Once trained, the system allows users to input Marathi text and receive real-time sentiment predictions. The user's input is preprocessed, vectorized, and classified using the trained Multinomial Naive Bayes model. This functionality demonstrates the practical applicability of the system in predicting sentiment from unseen Marathi text data.

1.5 Scope of The Project

The scope of this project centers on the development and implementation of a sentiment analysis system for Marathi text using traditional machine learning techniques, specifically logistic regression. The project is designed to provide a practical and scalable solution for classifying sentiment as positive, neutral, or negative. The main aspects covered within the scope include:

1. Marathi Text Processing:

   This project focuses on handling text data in the Marathi language, which includes adapting standard NLP preprocessing techniques like tokenization, stop-word removal, and normalization for Marathi-specific text characteristics.

2. Sentiment Classification:

   The sentiment analysis is restricted to classifying text into three categories—positive, neutral, and negative. The Multinomial Naive Bayes model is trained to work on Marathi text and is evaluated based on its ability to perform this multiclass classification.

3. Dataset and Domain:

   The project uses a labeled dataset of Marathi text, collected from publicly available sources such as social media, reviews, or blogs. The dataset is balanced to include examples of positive, neutral, and negative sentiment. However, the dataset is limited to the domain of general opinions and may not cover specialized fields.

4. Machine Learning Model:

   While the project focuses on using Multinomial Naive Bayes, the framework developed in this study can be expanded to accommodate other machine learning or deep learning models in the future. The primary scope, however, is to demonstrate the effectiveness of Multinomial Naive Bayes when coupled with proper preprocessing in a low-resource language environment.

5. Evaluation and Performance Metrics:

   The project aims to evaluate the performance of theMultinomial Naive Bayes model using common evaluation metrics such as accuracy, precision, recall, and F1-score. The performance is tested against the dataset prepared, but its scope does not include real-time deployment or handling of unseen, highly specialized forms of Marathi.

6. Limitations:

   The scope of this project is limited to the Marathi language, and the methods developed may not generalize directly to other low-resource languages without modifications. Additionally, while this project focuses on Multinomial Naive Bayes, it does not explore more advanced deep learning models such as neural networks or transformers, which could potentially yield higher accuracy in sentiment analysis tasks.

By focusing on these core areas, the project lays the groundwork for sentiment analysis in Marathi and provides a model that can be extended or improved upon in future studies. The project does not extend to developing complex production-level systems or addressing highly domain-specific sentiments, keeping its scope limited to general sentiment classification tasks in Marathi.

Chapter 2

Review of Literature

Paper [1] explored the effectiveness of the Multinomial Naive Bayes (MNB) algorithm in sentiment analysis. Their work focused on the automatic classification of text, specifically movie reviews, into positive and negative sentiments using machine learning techniques. The authors highlight the importance of efficient document sorting due to the overwhelming growth of online textual data. The MNB model was applied with a "bag of words" approach, which takes word frequency into account rather than simple word occurrence, thus improving classification accuracy compared to simpler Naive Bayes models like the Bernoulli Naive Bayes (BNB). To enhance performance, they incorporated Term Frequency-Inverse Document Frequency (TF-IDF) for weighting words based on their importance, resulting in an improved model performance.

The experiments were conducted on a dataset of movie reviews, and the results showed an initial accuracy of 91%. While the model proved to be computationally efficient and effective on large datasets, one of the key limitations identified was overfitting on smaller datasets. Furthermore, the authors pointed out that while MNB provides a robust method for sentiment analysis, future work could involve integrating advanced techniques like deep learning to further enhance accuracy, particularly in more complex text scenarios. Despite these limitations, the research demonstrates the strong potential of MNB for real-time sentiment classification tasks.

Chapter 3

System Analysis

3.1 Functional Requirements

The functional requirements of the system are the key operations and features that the sentiment analysis system for Marathi text is expected to perform. These requirements are essential for the successful implementation and functionality of the sentiment classification model. The following are the major functional requirements of the system:

1. **Input Text Handling:**

    - The system must accept raw Marathi text as input from users.
    - It should be able to handle a variety of text formats, such as plain text input, social media posts, or online reviews, provided in Marathi.
    - The system must support the processing of multiple sentences or paragraphs at a time.

2. **Text Preprocessing:**

    - Tokenization: The system should break the input Marathi text into tokens or individual words, ensuring that punctuation and special characters are appropriately handled.
    - Stop-word Removal: It should remove Marathi stop-words (e.g., “आहे”, “होतो”).
    - Normalization: The system must normalize the text by converting inflected or non-standardized words into a standard form.
    - Special Character and Noise Removal: The system should remove unnecessary symbols, numbers, and irrelevant characters that do not contribute to sentiment classification.

3. **Feature Extraction:**

- The system should convert the preprocessed text into numerical features using techniques like Term Frequency-Inverse Document Frequency (TF-IDF) or bag-of-words representation, making the text data usable for machine learning.
- It should accurately weigh words according to their frequency and importance within the context of sentiment.

4. **Sentiment Classification:**

- The system must classify the sentiment of the input Marathi text as either positive, neutral, or negative.
- The logistic regression model should be trained to output the probability of each sentiment class and predict the most likely sentiment based on the input features.
- The system should provide the sentiment label (positive, neutral, or negative) as output for each text sample processed.

5. **Model Training and Learning:**

- The system must support the training of the logistic regression model using a labeled dataset of Marathi text.
- The system must also support hyperparameter tuning, allowing for optimization of the model's performance based on evaluation metrics such as accuracy, precision, recall, and F1-score.

6. **Evaluation and Feedback:**

- The system must provide performance metrics such as accuracy, precision, recall, and F1-score for the sentiment classification model.
- It should display the evaluation results after model training, enabling users to assess how well the model is performing.
- The system should provide feedback on the classification of test samples to verify its accuracy and reliability.

3.2 Non Functional Requirements:

3.2.1 Performance Requirements

1. **Accuracy of Sentiment Classification:**
   The system must achieve high accuracy in classifying Marathi text as positive, neutral, or negative. The performance should be evaluated using metrics such as precision, recall, and F1-score. The goal is to maintain at least 80% accuracy on a balanced test set to ensure the system is reliable in real-world applications.

2. **Processing Speed:**
   The system must be able to process and classify text efficiently. Given that Marathi text datasets may vary in size, the sentiment analysis model should perform with a reasonable processing time for both training and prediction tasks. The system should:

   - Process input text and provide sentiment classification within a few seconds for individual sentences or small text samples.
   - Handle larger datasets (e.g., batch classification) within a reasonable time frame, ensuring that the classification task completes without significant delays.

3. **Scalability:**
   The system should be scalable, allowing it to handle increasing amounts of Marathi text data without significant degradation in performance. The model must be capable of processing larger datasets or retraining with additional data as needed.

4. **Memory and Resource Usage:**
   The system should be optimized for memory and resource consumption, particularly in environments with limited computational power. It should efficiently manage the system's memory, especially during preprocessing and model training, to prevent excessive resource usage. The feature extraction and training process should be lightweight enough to run on standard machines.

3.2.2 Software Quality Attributes

1. **Maintainability:**
   The system must be designed for easy maintainability, allowing for updates, such as improvements to the preprocessing pipeline or model enhancements. The sentiment analysis code should be modular, ensuring that each component (e.g., preprocessing, feature extraction, classification) can be modified independently.

2. **Extensibility:**
   While the current project focuses on using Multinomial Naive Bayes, the system should be designed with extensibility in mind, allowing for the integration of other machine learning models (e.g., decision trees, support vector machines, or deep learning models) in the future.

3. **Reusability:**
   The preprocessing and feature extraction components should be reusable across other NLP tasks involving Marathi text. The system should be designed in such a way that these components can be easily adapted for other machine learning applications, such as text classification or topic modeling, involving Marathi language data.

4. **Reliability:**
   The system must reliably produce sentiment classifications across various text inputs. It should consistently handle different forms of Marathi text, including informal language, spelling variations, and different sentence structures, without breaking or producing erroneous results.

5. **Portability:**
   The system should be portable and able to run on different platforms or environments without extensive reconfiguration. This includes compatibility with standard data science environments (e.g., Python, Jupyter notebooks, and libraries such as scikit-learn and NLTK) so that the system can be used in various academic and industrial settings.

6. **Accuracy and Robustness:**

   The system should not only perform with high accuracy but also be robust to variations in Marathi text. It should handle errors or noise in the text (such as typos or slang) without a significant drop in performance. This robustness is critical for dealing with real-world data, which may be noisy or unstructured.

7. **Documentation and Clarity:**

   Proper documentation must be provided to ensure that the system's functionality is clear and understandable to future users or developers. This includes documenting the steps involved in text preprocessing, model training, and evaluation,.

3.3 Specific Requirements:

Hardware :
Minimum Requirements

CPU: Dual-core processor (Intel i5 or AMD Ryzen 5)

RAM: 8 GB

Storage: 256 GB SSD

Network: Stable internet connection (for downloading datasets and libraries)

Recommended Requirements

CPU: Quad-core processor (Intel i7 or AMD Ryzen 7)

RAM: 16 GB

Storage: 512 GB SSD

Network: Stable internet connection (for collaborative work and data fetching)

Software :
Operating System

Windows: Windows 10 or later

Programming Language

Python: Version 3.7 or later

Libraries and Packages

Data Handling

pandas: For data manipulation and analysis

numpy: For numerical operations

Natural Language Processing (NLP)

nltk: For tokenization and natural language processing tasks

re: Built-in module for regular expressions

Machine Learning

scikit-learn: For implementing the Multinomial Naive Bayes algorithm

Data Visualization

matplotlib: For plotting graphs and visualizing results

seaborn: For enhanced data visualization

Other Utilities

joblib: For saving and loading models

IDE or Text Editor

Google Colab Notebook: Recommended for interactive data analysis and visualization

PyCharm: A powerful IDE for Python development

3.4 Use-Case Diagrams and description

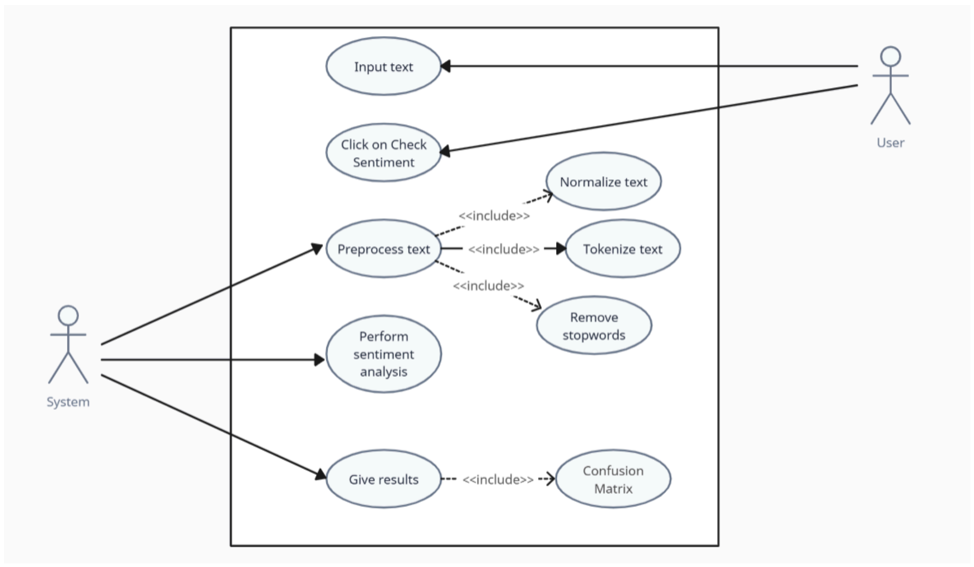


Figure 1: Use Case Diagram for sentiment analysis

There are two actors: the User and the System. The User interacts with the system by providing input through Input Text and Click on Check Sentiment. The system then begins with Preprocess Text, which involves steps such as Normalize Text, Tokenize Text, and Remove Stopwords to prepare the input for analysis.Then it executes the Perform Sentiment Analysis. The system completes the process with the Give Results use case, displaying the sentiment outcome along with a Confusion Matrix to evaluate the accuracy of the analysis.

Chapter 4

Analysis Modeling

4.1 Class Diagram

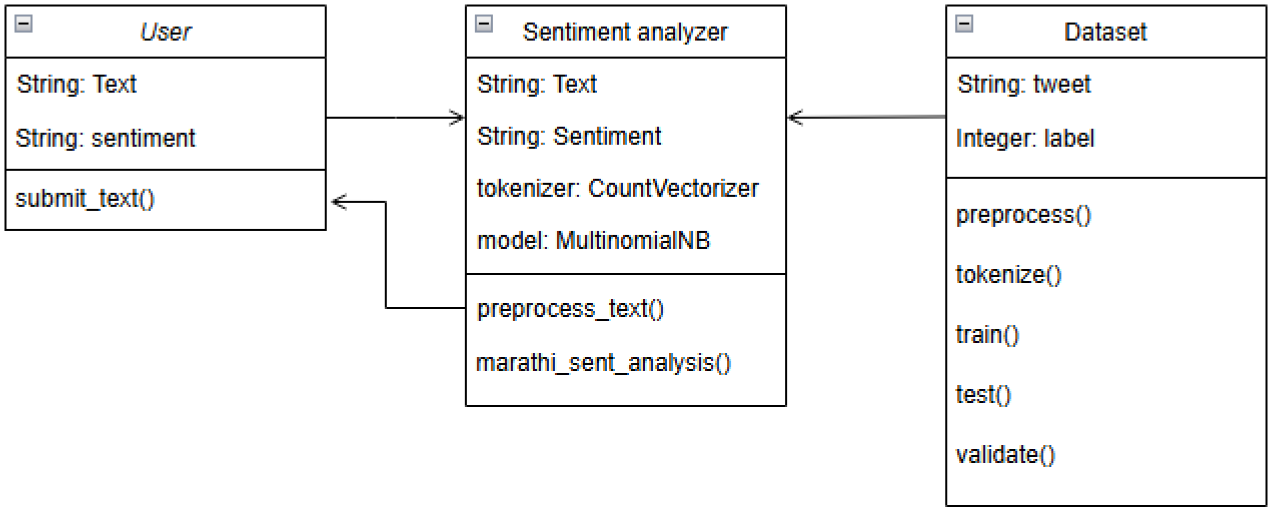


Figure 2: Class Diagram for sentiment analysis

In this class diagram, there are three main classes: User, Sentiment Analyser, and Dataset. The User class has two attributes: text and sentiment, both as strings, and it uses the function submit_text() to provide input. The Sentiment Analyser class also has the attributes text and sentiment as strings, along with a tokenizer implemented using CountVectorizer, and a model using MultinomialNB. It includes the functions preprocess_text() to clean and prepare the input and marathi_sent_analysis() to perform sentiment analysis. The Dataset class contains two attributes: tweet (a string) and label (an integer representing sentiment), with functions preprocess(), tokenize(), train(), test(), and validate() for handling and preparing data for sentiment analysis.

4.2 Functional Modeling

Data Flow Diagram

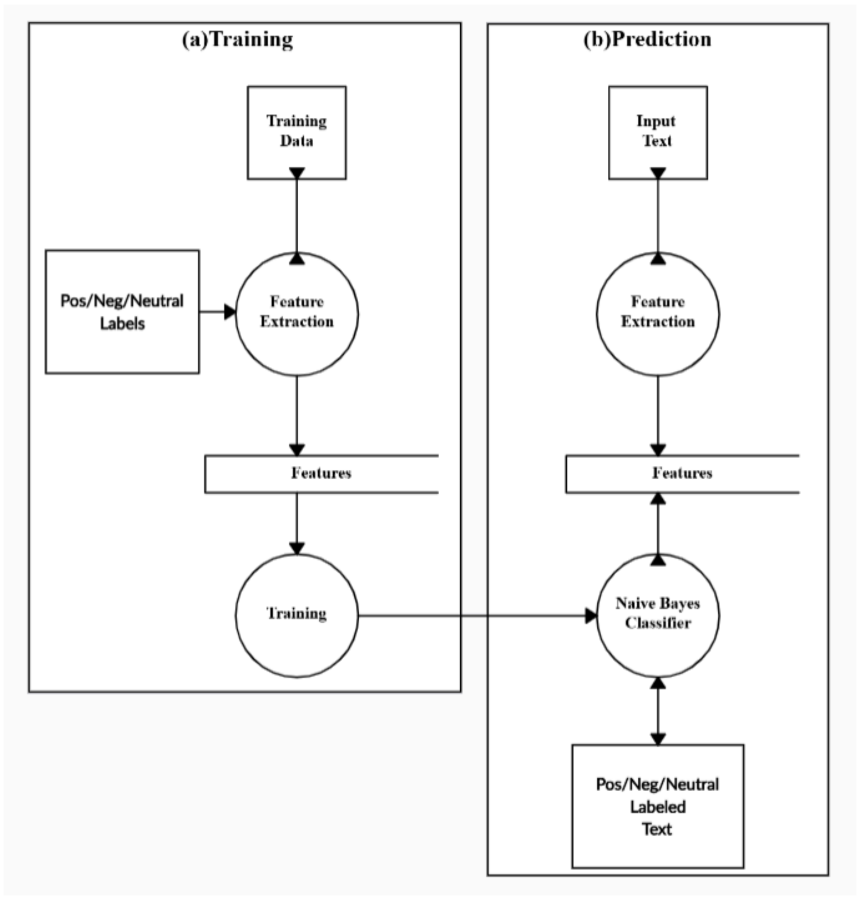


Figure 3: Data Flow Diagram for sentiment analysis

In this data flow diagram, there are two main parts: Training and Prediction. In the Training phase, the process starts with Training Data, which undergoes Feature Extraction to convert the raw text into numerical features. These features are labeled with Pos/Neg/Neutral sentiment labels, and the Training process uses these labeled features to train the sentiment analysis model. In the Prediction phase, the system receives an Input Text, performs Feature Extraction to generate features, and then passes them through the Multinomial Naive Bayes Classifier. The classifier outputs the predicted sentiment as Pos/Neg/Neutral text based on the analysis.

Chapter 5

Design

5.1 Architectural Design

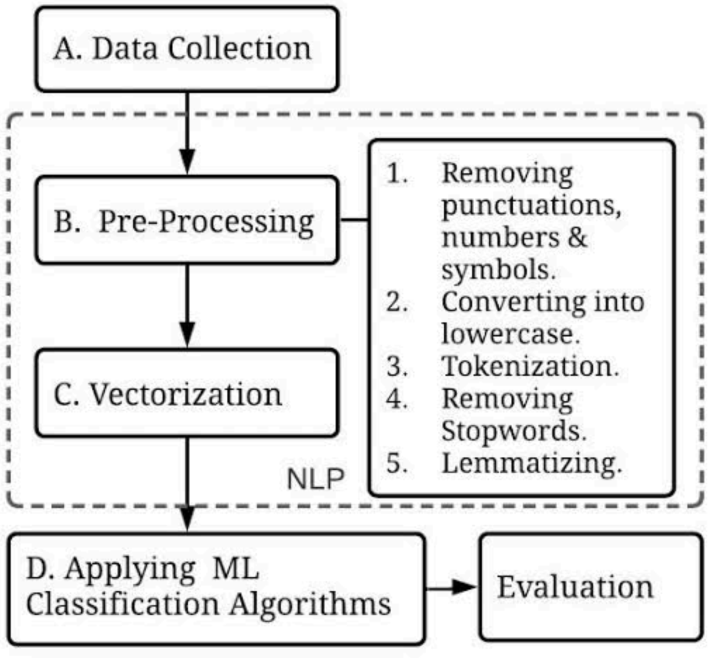


Figure 4: Architectural Diagram for sentiment analysis

In this architectural diagram, the process begins with Data Collection, where the raw data is gathered. This data moves to the Preprocessing stage, where it undergoes several steps: removing punctuation, numbers, and symbols, normalization, tokenization, and stopword removal to clean the text. After preprocessing, the data is passed to the Vectorization step, where the CountVectorizer is used to convert the text into numerical feature vectors. Then, the system proceeds to Applying ML Classification Algorithm, where MultinomialNB is applied to classify the text into categories such as positive, negative, or neutral. Finally, the model's performance is assessed during the Evaluation phase.

5.2 User Interface Design

The user interface is implemented using Streamlit, a Python framework that allows for the rapid development of interactive web applications. The primary goal of the user interface is to provide a simple, intuitive way for users to input Marathi text and obtain sentiment predictions (positive, neutral, or negative). The design of the interface ensures ease of use while maintaining responsiveness and clarity. The following components make up the design:

1. **Title and Introduction**

The interface starts with a clear title and a brief description of the system's purpose:

Title: "Marathi Sentiment Analysis"

Description: The system prompts the user to enter a Marathi sentence to determine its sentiment (positive, neutral, or negative).

This ensures that users immediately understand the function of the application.

```
st.title('Marathi Sentiment Analysis')
st.write('Enter a Marathi sentence and get its sentiment label.')
```

2. **User Input Field**

The main interaction point for the user is the text input field. This field allows users to enter a Marathi sentence. The input field is designed to handle simple sentences, ensuring that users can quickly submit text for analysis without confusion.

```
user_input = st.text_input('Enter Sentence:', '')
```

3. **Submit Button**

Next to the input field, a Submit button is provided to allow users to submit their sentence for sentiment analysis. Upon clicking the button, the system processes the text and predicts its sentiment using the trained machine learning model.

Action: When the user clicks the "Submit" button, the system triggers the backend function that processes the text, vectorizes it, and classifies it using the Multinomial Naive Bayes model.

Validation: If no input is provided, the system prompts the user to enter a valid sentence before submitting.

```
if st.button("Submit"):
    if user_input:
        sentiment = predict_sentiment(user_input)
        # Output sentiment prediction
    else:
        st.write("Please enter a sentence.")
```

4. Displaying Sentiment Prediction

Once the user submits a sentence, the system outputs the predicted sentiment. The result is displayed on the same page under the input field. The sentiment is classified into one of the following categories:

- Positive
- Negative
- Neutral

This provides immediate feedback to the user. The interface ensures that the result is displayed in a user-friendly format, making it clear which sentiment has been predicted for the input text.

```
if sentiment == 1:
    sentiment_label = "Positive"
elif sentiment == -1:
    sentiment_label = "Negative"
else:
    sentiment_label = "Neutral"

st.write(f'Predicted Sentiment: {sentiment_label}')
```

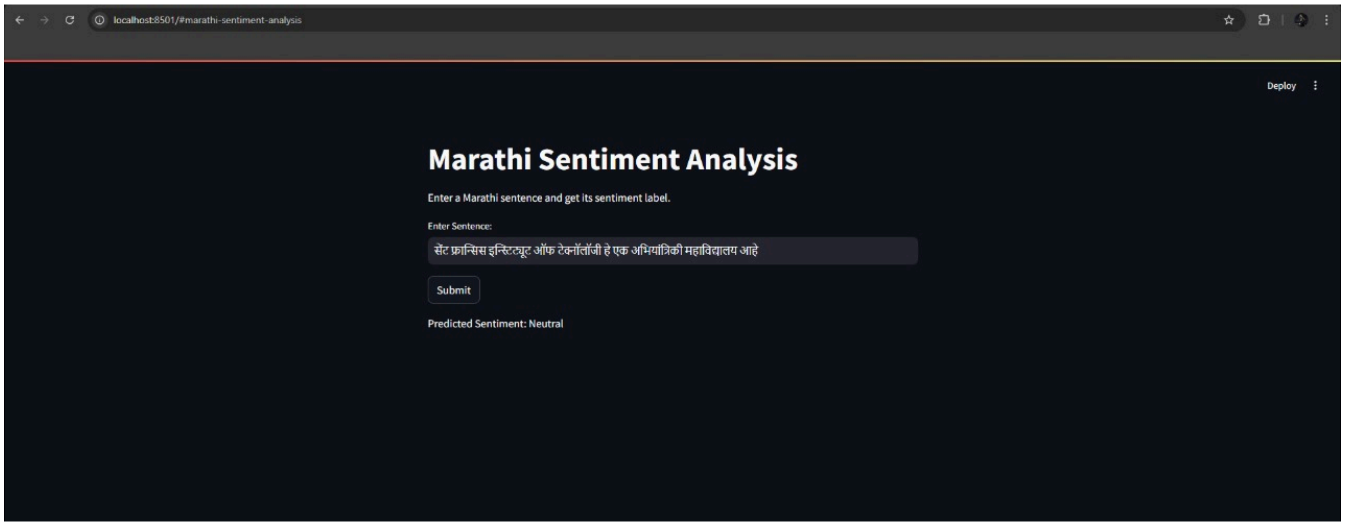
5. Error Handling and Feedback

The interface includes basic error handling to ensure that users enter valid text. If the input field is left blank and the user clicks "Submit," a message is displayed, prompting the user to input a valid sentence. This prevents any confusion and enhances the user experience by ensuring proper input validation.

```
else:
    st.write("Please enter a sentence.")
```

User Experience (UX) Considerations

- Simplicity: The interface is minimalist, ensuring users can interact with it easily without any distractions. Only essential elements such as input fields, buttons, and result displays are included.
- Real-time Feedback: The sentiment prediction is displayed instantly after the user submits the sentence, ensuring a smooth and interactive experience.
- Responsive Design: Since the interface is built using Streamlit, it is responsive and works across various devices, including desktops and mobile browsers.



Marathi Sentiment Analysis

Enter a Marathi sentence and get its sentiment label.

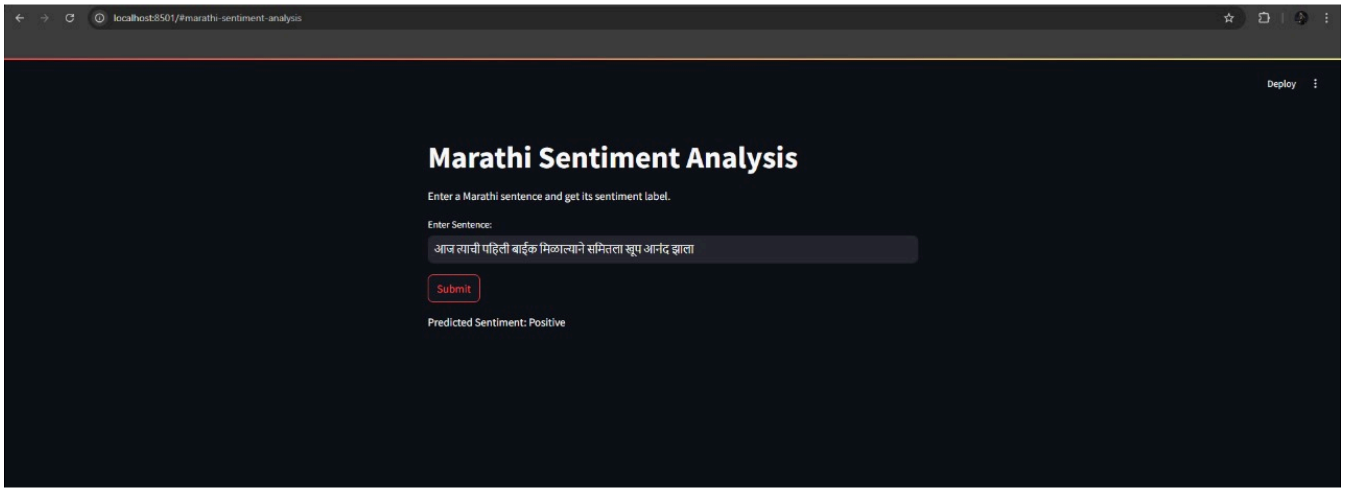
Enter Sentence:

सेंट फ्रान्सिस इन्स्टिट्यूट ऑफ टेक्नॉलॉजी हे एक अभियांत्रिकी महाविद्यालय आहे

Predicted Sentiment: Neutral

Figure 5: Neutral Sentiment Prediction

The UI allows users to input Marathi text, such as "सेंट फ्रान्सिस इन्स्टिट्यूट ऑफ टेक्नॉलॉजी हे एक अभियांत्रिकी महाविद्यालय आहे", (St. Francis Institute of Technology is an Engineering College) and upon clicking "Submit" button, it displays the predicted sentiment as "Neutral".



Marathi Sentiment Analysis

Enter a Marathi sentence and get its sentiment label.

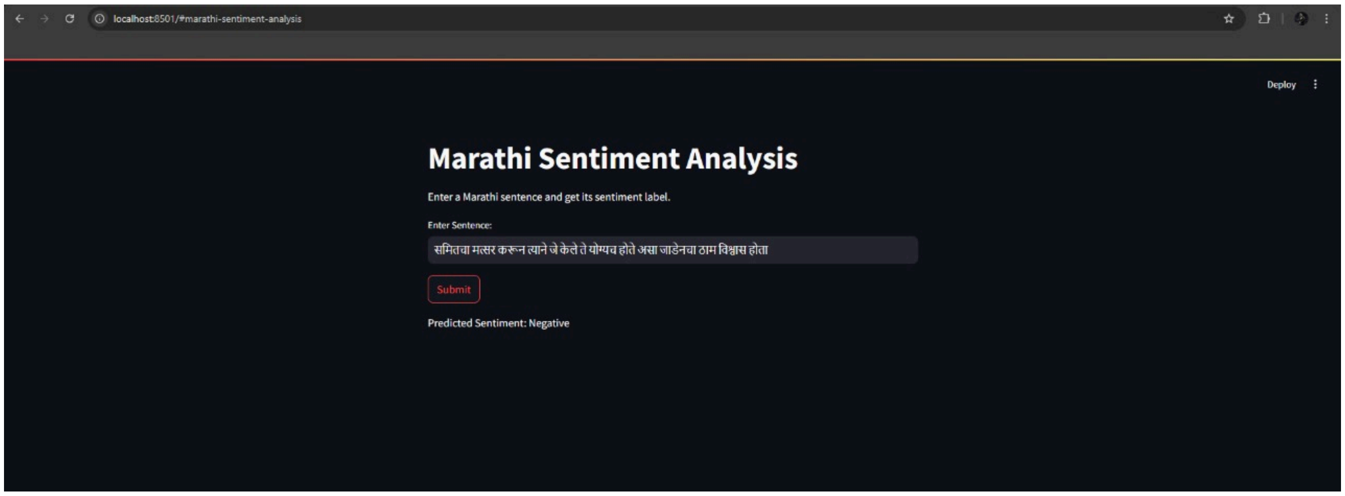
Enter Sentence:

आज त्याची पहिली बाईक मिळाल्याने समितला खूप आनंद झाला

Predicted Sentiment: Positive

Figure 6: Positive Sentiment Prediction

The UI allows users to input Marathi text, such as "आज नवीन बाईक मिळाल्यानंतर समित खूप खुश झाला होता," (After getting his new bike today Samit was very happy) and upon clicking the "Submit" button, it displays the predicted sentiment as "Positive."



Marathi Sentiment Analysis

Enter a Marathi sentence and get its sentiment label.

Enter Sentence:

समितचा मत्सर करून त्याने जे केले ते योग्यच होते असा जाडेनचा ठाम विश्वास होता

Predicted Sentiment: Negative

Figure 7: Negative Sentiment Prediction

The UI allows users to input Marathi text, such as "समितचा हेवा करून आपण चांगलं केलं, असा जेडनचा ठाम विश्वास होता," (Jaden was of firm belief that he did good by being envious of Samit) and upon clicking the "Submit" button, it displays the predicted sentiment as "Negative."

Chapter 6

Implementation

6.1 Algorithms

Multinomial Naive Bayes Algorithm

Multinomial Naive Bayes (MNB) is a supervised learning algorithm based on the application of Bayes' Theorem. It is particularly well-suited for classification problems involving discrete features, especially in text classification tasks where word frequencies are used as features. The Multinomial Naive Bayes algorithm is an extension of the basic Naive Bayes classifier, specifically designed to handle multiclass classification where the features represent counts or frequency of occurrences (such as words in a document).

1. Bayes' Theorem

At the core of the Naive Bayes algorithm is Bayes' Theorem, which calculates the probability of a class given a feature vector:

$$P(C_k|X) = \frac{P(X|C_k) \cdot P(C_k)}{P(X)}$$

Where:

- $P(C_k|X)$ is the posterior probability of class C_k given the feature vector *X*
- $P(X|C_k)$ is the likelihood of the feature vector *X* given class C_k
- $P(C_k)$ is the prior probability of class C_k
- $P(X)$ is the evidence or the total probability of the feature vector.

Naive Bayes is considered "naive" because it assumes that all features are conditionally independent of one another given the class. Despite this strong assumption of independence, Naive Bayes has been shown to work very well in practice, especially in text classification problems.

2. Multinomial Naive Bayes for Text Classification

In the context of text classification, the Multinomial Naive Bayes algorithm models the data as a multinomial distribution, where the features represent word counts or term frequencies in a document.

Features (X): In text classification, the features $X = (x_1, x_2, ..., x_n)$ represent the frequency of each word (or token) in the document. The word counts are obtained from the preprocessed text, which may be converted into a term frequency vector using methods like CountVectorizer or TF-IDF (Term Frequency-Inverse Document Frequency).

Classes (C): The classes represent the sentiment categories that the model is trying to predict (for example, positive, neutral, and negative sentiment).

3. The Multinomial Distribution

In the Multinomial Naive Bayes algorithm, the probability of a document (feature vector) given a class is modeled as a multinomial distribution:

$$P(X|C_k) = \prod_{i=1}^{n} P(x_i|C_k)^{x_i}$$

Where:

- $P(x_i|C_k)$ is the probability of the word x_i occurring in class C_k
- x_i is the count of word i in the document.
- The likelihood of the entire document is the product of the probabilities of all words in that document, given the class C_k

.

The model learns the conditional probabilities $P(x_i|C_k)$ by calculating the relative frequency of each word in the training set for each class. Smoothing techniques like Laplace Smoothing (also called add-one smoothing) are often used to handle zero probabilities when a word does not appear in the training data for a particular class.

4. Classification Process

Once the model is trained, it predicts the class of a new document by applying Bayes' theorem to compute the posterior probability for each class C_k , and assigns the document to the class with the highest posterior probability:

$$\hat{C} = \arg\max_{C_k} P(C_k|X) = \arg\max_{C_k} P(X|C_k) \cdot P(C_k)$$

Where:

- $P(C_k)$ is the prior probability of class C_k (estimated from the training data as the proportion of documents in each class).
- $P(X|C_k)$ is the likelihood of the document belonging to class C_k , calculated from the product of word probabilities conditioned on the class.

5. Application in This Project

In this project, the Multinomial Naive Bayes algorithm is applied to classify Marathi text (tweets) into one of three sentiment classes: positive, neutral, or negative. The algorithm is trained on the L3CubeMahaSent dataset, which contains manually labeled Marathi tweets. The preprocessed text is transformed into word frequency vectors using CountVectorizer, and the model predicts the sentiment by computing the likelihood of the text belonging to each sentiment class.

Given the large size of the dataset and the nature of text data, Multinomial Naive Bayes is an ideal choice due to its efficiency and suitability for handling sparse feature matrices, which are common in text classification tasks.

6.2 Working of the project

The sentiment analysis system for Marathi text is implemented using a combination of data preprocessing techniques, text vectorization, and the Multinomial Naive Bayes machine learning model. The following steps describe how the project is structured and the workflow for training and evaluating the model.

Step 1: Loading the Dataset

The first step involves loading the datasets for training, validation, and testing. The dataset used contains Marathi tweets that are labeled with sentiment values. The training dataset is used to train the model, the validation dataset is used for model evaluation during training, and the test dataset is reserved for final performance testing.

```
import pandas as pd

# Load datasets
train_df = pd.read_csv('train.csv')
val_df = pd.read_csv('valid.csv')
test_df = pd.read_csv('test.csv')

# Display the first few rows of each dataset
print(train_df.head())
print(val_df.head())
print(test_df.head())
```

Step 2: Analyzing the Dataset

Before proceeding with model training, it is important to analyze the datasets for any missing values and to understand the data types in each dataset. This step ensures that the data is clean and ready for processing.

```
# Check for missing values and data types in each dataset
print(train_df.info())
print(val_df.info())
print(test_df.info())
```

Step 3: Loading the Marathi Stopwords List

A list of custom Marathi stopwords is loaded from an external text file. These stopwords are words that do not carry significant meaning for sentiment analysis (such as common words like "आहे", "होतो") and need to be removed during text preprocessing to reduce noise.

```
# Load custom Marathi stopwords from the text file
def load_stopwords(file_path):
    with open(file_path, 'r', encoding='utf-8') as f:
        stopwords = f.read().splitlines()
    return set(stopwords)

marathi_stopwords = load_stopwords('marathi_stopwords.txt')
```

Step 4: Text Preprocessing

To prepare the raw text for machine learning, several preprocessing steps are applied. These include:

- Punctuation Removal: Removing punctuation from the text.
- Number Removal: Removing numbers from the text.
- Tokenization: Splitting the text into individual words.
- Stopword Removal: Removing Marathi stopwords to focus on sentiment-related words.

```
import re
import nltk

# Download necessary NLTK resources
nltk.download('punkt')

# Define a function for preprocessing text
def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    text = re.sub(r'\d+', '', text)  # Remove numbers
    tokens = nltk.word_tokenize(text)  # Tokenization
    tokens = [word for word in tokens if word not in marathi_stopwords]  #
Remove stopwords
    return ' '.join(tokens)

# Apply preprocessing to each dataset
train_df['cleaned_text'] = train_df['tweet'].apply(preprocess_text)
val_df['cleaned_text'] = val_df['tweet'].apply(preprocess_text)
test_df['cleaned_text'] = test_df['tweet'].apply(preprocess_text)
```

Step 5: Text Vectorization

After preprocessing, the text data is transformed into numerical form using CountVectorizer. This step converts the cleaned text into a bag-of-words representation, where each word in the dataset is treated as a feature. This vectorized format is essential for machine learning models to process text data.

```
from sklearn.feature_extraction.text import CountVectorizer

# Initialize CountVectorizer
vectorizer = CountVectorizer()

# Vectorize training data
X_train_vectorized = vectorizer.fit_transform(train_df['cleaned_text'])
y_train = train_df['label']

# Vectorize validation and testing data
X_val_vectorized = vectorizer.transform(val_df['cleaned_text'])
y_val = val_df['label']
X_test_vectorized = vectorizer.transform(test_df['cleaned_text'])
y_test = test_df['label']
```

Step 6: Model Training

The Multinomial Naive Bayes model is chosen as the machine learning algorithm for this project. The model is trained on the vectorized training data (word frequency vectors) and corresponding sentiment labels.

```
from sklearn.naive_bayes import MultinomialNB

# Initialize and train the model
model = MultinomialNB()
model.fit(X_train_vectorized, y_train)
```

Step 7: Prediction and Evaluation

Once the model is trained, it is evaluated using the validation and test datasets. Predictions are made, and the model's performance is measured using a classification report and confusion matrix. The classification report provides detailed metrics such as precision, recall, F1-score, and accuracy for each sentiment class (positive, neutral, negative).

```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Predict on validation set
y_val_pred = model.predict(X_val_vectorized)

# Classification report for validation set
print("Validation Classification Report:")
print(classification_report(y_val, y_val_pred))

# Confusion matrix for validation set
cm_val = confusion_matrix(y_val, y_val_pred)
sns.heatmap(cm_val, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(y_train), yticklabels=np.unique(y_train))
plt.title('Confusion Matrix - Validation Set')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# Predict on test set
y_test_pred = model.predict(X_test_vectorized)

# Classification report for test set
print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))

# Confusion matrix for test set
cm_test = confusion_matrix(y_test, y_test_pred)
sns.heatmap(cm_test, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(y_train), yticklabels=np.unique(y_train))
plt.title('Confusion Matrix - Test Set')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

Step 8: Sentiment Prediction

Finally, a sentiment prediction function is created to classify user-input text. The function preprocesses the text, vectorizes it, and predicts its sentiment using the trained model. This allows the model to be used for real-time sentiment classification.

```
def predict_sentiment(text):
    cleaned_text = preprocess_text(text)
    vectorized_text = vectorizer.transform([cleaned_text])
    prediction = model.predict(vectorized_text)
    return prediction[0]
```

Chapter 7
Conclusion

This project successfully demonstrates the application of sentiment analysis for Marathi text, a low-resource language, using a combination of Natural Language Processing (NLP) techniques and a machine learning model. By utilizing the L3CubeMahaSent dataset, the largest publicly available dataset for Marathi sentiment analysis, and implementing the Multinomial Naive Bayes algorithm, the system effectively classifies Marathi tweets into positive, neutral, and negative sentiment categories.

The project's success lies in its carefully designed preprocessing pipeline, which includes tokenization, stop-word removal, and normalization, all of which are tailored to the linguistic intricacies of Marathi. The use of CountVectorizer for feature extraction further enabled the model to capture the frequency of words and their impact on sentiment classification. The Multinomial Naive Bayes algorithm, chosen for its efficiency and strong performance in text classification tasks, proved to be effective in handling the sparse nature of word frequency data.

Through thorough training and evaluation using the training, validation, and test splits of the L3CubeMahaSent dataset, the system demonstrated reliable sentiment classification performance. The use of well-established evaluation metrics, such as accuracy, precision, recall, and F1-score, ensured a comprehensive understanding of the model's strengths and areas for improvement.

In conclusion, this project contributes to the growing field of Natural Language Processing for low-resource languages like Marathi. It highlights the potential of traditional machine learning algorithms in achieving high accuracy and scalability for sentiment analysis. Future work could focus on exploring more advanced models such as transformers, expanding the dataset, or further refining the preprocessing techniques to enhance performance. This system provides a foundational approach that can be built upon for more sophisticated sentiment analysis systems and real-world applications in Marathi language processing.

References

[1] M. Abbas et al., “Multinomial Naive Bayes Classification Model for Sentiment Analysis,” IJCSNS International Journal of Computer Science and Network Security, vol. 19, no. 3, p. 62, 2019, Available: http://paper.ijcsns.org/07_book/201903/20190310.pdf

[2] A. Pingle, A. Vyawahare, I. Joshi, R. Tangsali, and R. Joshi, “L3Cube-MahaSent-MD: A Multi-domain Marathi Sentiment Analysis Dataset and Transformer Models,” *arXiv (Cornell University)*, Jan. 2023, doi: https://doi.org/10.48550/arxiv.2306.13888.

[3] C. Dewi, R.-C. Chen, Henoch Juli Christanto, and F. Cauteruccio, “Multinomial Naïve Bayes Classifier for Sentiment Analysis of Internet Movie Database,” *Vietnam Journal of Computer Science*, vol. 10, no. 04, pp. 485–498, Aug. 2023, doi: https://doi.org/10.1142/s2196888823500100.

[4] K. Rakshitha, R. H M, M. Pavithra, A. H D, and M. Hegde, “Sentimental analysis of Indian regional languages on social media,” *Global Transitions Proceedings*, vol. 2, no. 2, pp. 414–420, Nov. 2021, doi: https://doi.org/10.1016/j.gltp.2021.08.039.

[5] “Applying Multinomial Naive Bayes to NLP Problems - GeeksforGeeks,” *GeeksforGeeks*, Jan. 11, 2019. https://www.geeksforgeeks.org/applying-multinomial-naive-bayes-to-nlp-problems/

Acknowledgements

A project is always a coordinated, guided and scheduled team effort aimed at realizing a common goal. We are grateful and gracious to all those people who have helped and guided us through this project and make this experience worthwhile. We wish to sincerely thank our Principal Dr. Sincy George and our CMPN HOD Dr. Kavita Sonawane for giving us this opportunity to prepare a project in the Final Year of Computer Engineering. We are highly indebted to our institute, St. Francis Institute of Technology and the Department of Computer Engineering for providing us with this learning opportunity with the required resources to accomplish our task so far. We would also like to express our deep gratitude to our assigned mentor, Ms. Pradnya Sawant, who constantly guided and supervised us, and also furnished essential information concerning the project. This work would not have been possible without her necessary insights and intellectual suggestions that have helped us achieve so much. We would like to thank our teacher Ms. Pradnya Sawant who approved the topic for our NLP mini project and gave us some valuable suggestions to make our project better. We also take the opportunity to thank all teaching and non-teaching staff for their endearing support and cooperation.