DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

Course Title: AIES

Course Code: CT-361

GROUP MEMBERS:

- ZAINAB SHARIF **(CT-22002)**
- LAIBA RAZA RIZVI **(CT-22003)**
- FARWA ABBAS **(CT-22014)**

TABLE OF CONTENTS

TITLE : SENTIMENT ANALYSIS ON REVIEWS

1. Problem Definition

The exponential growth of user-generated content on the internet has made it crucial to extract meaningful insights from textual data. One such insight is **sentiment**, which reflects a user's opinion, emotion, or attitude toward a subject. This project focuses on **binary sentiment classification** of reviews as either *positive* or *negative*, using machine learning techniques.

The key challenge lies in accurately interpreting natural language, which is often ambiguous and context-dependent. To solve this, we leverage foundational principles of **Natural Language Processing (NLP)** and **probabilistic machine learning**, specifically the **Naive Bayes classifier**, due to its suitability for high-dimensional text data.

2. Conceptual Foundation and Research

2.1 Theoretical Foundations

Our solution is grounded in core computing concepts:

- **Bayesian Inference**: Based on Bayes' Theorem, which allows us to compute the posterior probability of a sentiment given observed words.
- **Bag of Words (BoW)** and **TF-IDF**: Techniques to convert unstructured text into structured numerical vectors.
- **Natural Language Processing**: Preprocessing pipelines that normalize, tokenize, stem, and clean the input text.
- **Supervised Learning**: Training a model on labeled data to predict outcomes for unseen instances.

2.2 Why Naive Bayes?

- It assumes **feature independence**, simplifying the joint probability model.
- It performs well in **text classification**, even with the naive independence assumption.
- It is **computationally efficient**, scalable, and interpretable.

2.3 Literature and Domain Review

Previous research and industrial applications demonstrate Naive Bayes's competitiveness in baseline text classification tasks. When combined with **TF-IDF weighting** and proper **preprocessing**, it can achieve high accuracy on moderately sized datasets with rapid training times.

3. Methodology and Implementation

3.1 Dataset

We used a structured dataset of labeled reviews. Each review is associated with a binary sentiment label:

- 1: Positive
- 0: Negative

The dataset is split into training and testing sets to evaluate generalization performance.

3.2 Modular Design Overview

Module 1: Data Preprocessing

- **Goal**: Clean and normalize the input text.
- **Steps**:
  - Lowercasing
  - Removing punctuation and stopwords
  - Tokenization
  - Stemming (using PorterStemmer)

Module 2: Feature Extraction

- **Goal**: Transform preprocessed text into numerical features.
- **Technique**: TfidfVectorizer from sklearn.feature_extraction.text
- This captures word frequency while reducing the weight of common but less informative words.

Module 3: Model Training

- **Model**: MultinomialNB from sklearn.naive_bayes
- **Input**: TF-IDF vectors
- **Training**: Fit the model using the training subset.
- **Underlying Theory**: Applies Bayes' theorem assuming independence of terms.

Module 4: Model Evaluation

- **Techniques**: Accuracy, Confusion Matrix, Precision, Recall, F1-Score
- **Tools**: classification_report from sklearn.metrics

4. Results and Evaluation

	precision	recall	f1-score	support
0	0.84	0.88	0.86	3966
1	0.88	0.84	0.86	4034
accuracy			0.86	8000
macro avg	0.86	0.86	0.86	8000
weighted avg	0.86	0.86	0.86	8000

Analysis

- **True Positives**: Clearly positive reviews (e.g., “amazing”, “fantastic”) are well-identified.
- **False Negatives**: Reviews with sarcasm or nuanced language are harder to detect.
- **Conclusion**: The model performs well for basic sentiment classification, validating the choice of Naive Bayes.

5. Computing Principles in Practice

This project demonstrates the application of fundamental AI principles:

- **Probability and Statistics**: Core to Naive Bayes theory.
- **Text Mining**: Converting human language into machine-interpretable features.
- **Modular Design**: Clean separation of concerns enhances reusability and maintenance.

- **Algorithm Efficiency**: Naive Bayes scales well with data volume, making it ideal for real-time systems.

CODE:

```
import pandas as pd

# Load the dataset
data = pd.read_csv("/content/movie.csv")
print(data.columns)  # Check column names
```

```
Index(['text', 'label'], dtype='object')
```

```
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

def preprocessing(text):
    text = text.lower()
    text = re.sub(f'[{string.punctuation}]', '', text)
    text = re.sub(r'\d+', '', text)
    tokens = text.split()
    tokens = [stemmer.stem(word) for word in tokens if word not in stop_words]
    return ' '.join(tokens)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
data['text'] = data['text'].apply(preprocessing)
```

```python
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['label'], test_size=0.2, random_state=42)

# Define pipeline
nb_model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('nb', MultinomialNB())
])

# Train the model
nb_model.fit(X_train, y_train)

# Evaluate
y_pred = nb_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.85925
```

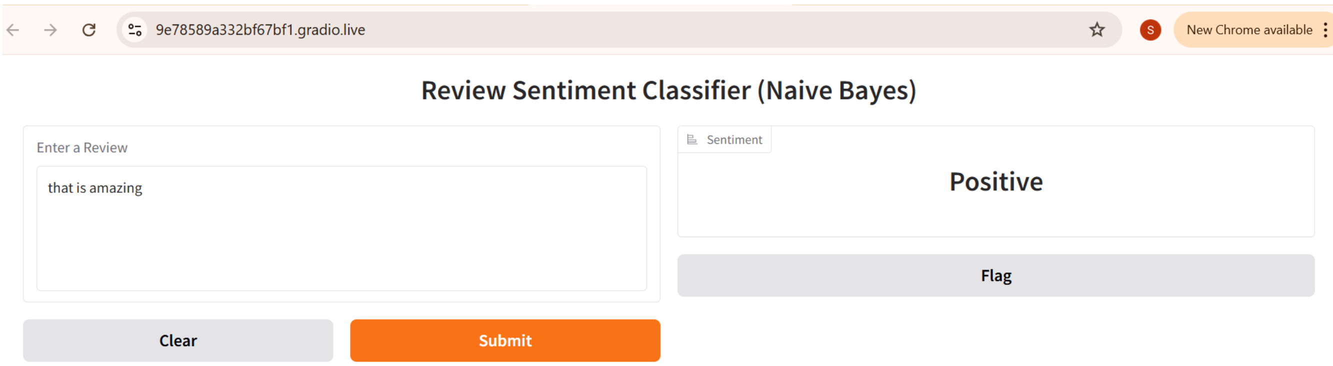
```python
import gradio as gr

def classify_review(text):
    text = preprocessing(text)
    prediction = nb_model.predict([text])[0]
    return "Positive" if prediction == 1 else "Negative"

gr.Interface(
    fn=classify_review,
    inputs=gr.Textbox(lines=5, label="Enter a Review"),
    outputs=gr.Label(label="Sentiment"),
    title="Review Sentiment Classifier (Naive Bayes)"
).launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://50c035a39dde439f46.gradio.live
```

OUTPUT:



9e78589a332bf67bf1.gradio.live
New Chrome available
Review Sentiment Classifier (Naive Bayes)
Enter a Review
that is amazing
Sentiment
Positive
Flag
Clear
Submit

9e78589a332bf67bf1.gradio.live
New Chrome available
Review Sentiment Classifier (Naive Bayes)
Enter a Review
that was disappointing!!!
Sentiment
Negative
Flag
Clear
Submit

6. Conclusion

This project successfully applies machine learning principles to perform sentiment analysis on reviews. It showcases conceptual clarity, domain-relevant methodology, and a clean modular structure. Naive Bayes provides a reliable baseline for sentiment classification, and the entire system is extendable and practical.

7. References

- Scikit-learn documentation: https://scikit-learn.org/
- NLTK documentation: https://www.nltk.org/

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
BACHELORS OF SCIENCE IN COMPUTER SCIENCE

Complex Computing Problem Assessment Rubrics

Course Code: CT-361		Course Title: Artificial Intelligence & Expert System	
Criteria and Scales			
Excellent (3)	**Good (2)**	**Average (1)**	**Poor (0)**
Criterion 1: Understanding the Problem: How well the problem statement is understood by the student			
Understand the problem clearly and identify the underlying issues and functionalities.	Adequately understands the problem and identifies the underlying issues and functionalities.	Inadequately defines the problem and identifies the underlying issues and functionalities.	Fails to define the problem adequately and does not identify the underlying issues and functionalities.
Criterion 2: Research: The amount of research that is used in solving the problem			
Contains all the information needed for solving the problem	Good research leads to a successful solution	Mediocre research which may or may not lead to an adequate solution	No apparent research
Criterion 3: Code: How complete the code is along with the assumptions and selected functionalities			
Complete the code according to the selected functionalities of the given case with clear assumptions	Incomplete code according to the selected functionalities of the given case with clear assumptions	Incomplete code according to the selected functionalities of the given case with unclear assumptions	Wrong code and naming conventions
Criterion 4: Report: How thorough and well-organized is the solution			
All the necessary information is organized for easy use insolving the problem	Good information organized well could lead to a good solution	Mediocre information which may or may not lead to a solution	No report provided

Total Marks: ____________

Teacher's Signature:________________