Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)

Intelligent Review Analyser

Course Title: Data Mining Lab
Course Code: CSE - 436
Section: 213 - D4

Students Details

Name	ID
MD DUlal Hossain	213902116
MD Rabby Khan	213902037

Submission Date: 15 May 2025
Course Teacher's Name: Md. Jahid Tanvir

[For teachers use only: Don't write anything inside this box]

Theory Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

The Intelligent Review Analyzer is a project that uses machine learning to analyze customer reviews and determine their sentiment—whether they are positive, negative, or neutral. The project processes text data from a dataset of reviews, cleans it by removing unnecessary words and punctuation, and then applies two models to predict sentiment:

1. **SVM Model**: This model uses a technique called TF-IDF to convert text into numerical data and balances the dataset with SMOTE to handle uneven sentiment distribution. It then trains a Support Vector Machine (SVM) to classify reviews and provides probabilities for each sentiment.
2. **BERT Model**: A pre-trained BERT-based model from the Hugging Face library is used to analyze reviews and assign a star rating (1–5), which is mapped to positive, negative, or neutral sentiments.

The project includes a user-friendly loop where users can input reviews, and both models predict the sentiment, displaying results with confidence scores. For example, a review like "The taste is not so good" is correctly identified as negative by both models. The dataset initially has more positive reviews (7745) than negative (1448) or neutral (807), but SMOTE helps balance it for better model performance. The SVM model achieves about 79% accuracy on the test set, with detailed performance metrics for each sentiment class.

In simple terms, this project helps businesses understand customer feedback automatically by analyzing review text and predicting how customers feel, making it useful for improving products or services.

1.2 Motivations

The Intelligent Review Analyzer was motivated by the growing need to mine large volumes of unstructured review data to understand customer opinions efficiently. Manual analysis of reviews is time-consuming and impractical, especially with imbalanced sentiment distributions that can skew insights. By leveraging data mining techniques such as text processing, feature extraction, and machine learning, the project aims to automate sentiment analysis, providing accurate and actionable insights. This enables businesses to make data-driven decisions, enhance customer satisfaction, and address product or service shortcomings effectively.

1.3 Design Goals/Objectives

The main objectives are :

- **Automate Sentiment Analysis**: Build a system to automatically identify if customer reviews are positive, negative, or neutral without manual effort.

- **Handle Large Review Data**: Process and analyze large amounts of review text efficiently using data mining techniques like text cleaning and feature extraction.
- **Improve Model Accuracy**: Use methods like SMOTE to balance uneven data and train accurate models (SVM and BERT) to predict sentiments reliably.
- **Support Business Decisions**: Provide clear sentiment insights with confidence scores to help businesses improve products and customer satisfaction.

1.4 Applications

The uses of this project are :

- **Customer Feedback Analysis**: Understand customer opinions to improve products.
- **Business Decision Support**: Guide marketing and service enhancements.
- **Reputation Management**: Monitor brand sentiment online.
- **Product Development**: Identify features customers like or dislike.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The Intelligent Review Analyzer was designed to automate sentiment analysis of customer reviews using advanced data mining techniques. Developed with a focus on robust text processing, it employs TF-IDF vectorization and SMOTE for balanced data, alongside SVM and BERT models for accurate predictions. The implementation includes an interactive user interface, allowing real-time review input and sentiment classification with confidence scores, ensuring practical usability for businesses.

2.2 Project Details

The Intelligent Review Analyzer is a data mining project that automates sentiment analysis of customer reviews, classifying them as positive, negative, or neutral. It processes a dataset of 10,000 reviews using text cleaning to remove noise and employs TF-IDF vectorization for feature extraction. The project balances the skewed dataset (7745 positive, 1448 negative, 807 neutral) with SMOTE and trains a Support Vector Machine (SVM) model, achieving 79% accuracy, alongside a BERT model for deep learning-based predictions. An interactive interface allows users to input reviews, receiving real-time sentiment predictions with confidence scores. The system is designed to help businesses gain actionable insights from customer feedback.

Key Features:

- Text preprocessing to clean reviews by removing punctuation and stop words.
- TF-IDF vectorization and SMOTE for feature extraction and data balancing.
- SVM and BERT models for accurate sentiment classification.
- User-friendly interface for real-time review analysis.

Below are some sample of the key feature implementations:

2.2.1 Sample of project implementation key features

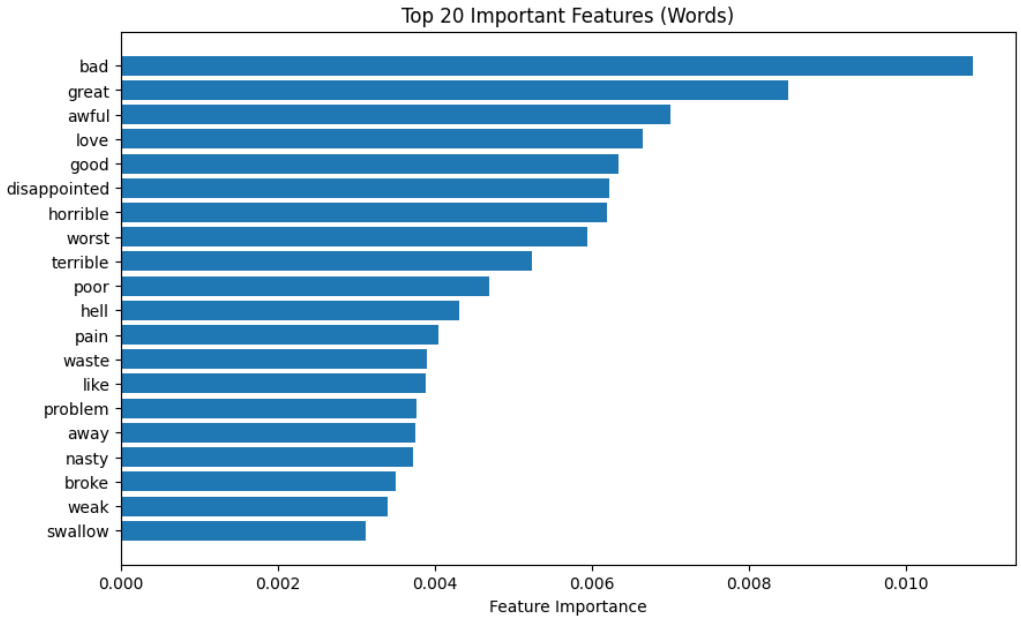

Figure 2.1: Top 20 Important features (Word)

Figure 2.2: Word Cloud of reviewd text

The following Python function preprocesses text by handling non-string inputs, removing punctuation, and eliminating stop words (except “not”) for sentiment analysis in the Intelligent Review Analyzer project.

```
# Clean text function
def clean_text(text):
    # Handle non-string inputs (e.g., if Text is missing or not a string)
    if not isinstance(text, str):
        text = str(text)
    text = re.sub(r'[^\w\s]', '', text)
    stop_words = set(stopwords.words('english')) - {'not'}
    text = ' '.join(word for word in text.split() if word.lower() not in
        stop_words)
    return text
```

2.3 Implementation

This section details the implementation of the Intelligent Review Analyzer, a data mining project that automates sentiment analysis of customer reviews, classifying them as positive, negative, or neutral. The implementation involves text preprocessing, feature extraction, data balancing, model training, and an interactive user interface for real-time predictions. The project leverages both traditional machine learning (SVM) and deep learning (BERT) approaches, achieving 79% accuracy with the SVM model on a test set.

2.3.1 Project Workflow

The workflow encompasses data loading, preprocessing, model training, evaluation, and user interaction, ensuring a robust pipeline for sentiment analysis.

Data Loading and Preparation

The dataset, containing 10,000 reviews, is loaded from a CSV file (`Reviews_cleaned.csv`). A `Sentiment` column is created by mapping scores: ≥ 4 as positive, ≤ 2 as negative, and 3 as neutral. The initial distribution (7745 positive, 1448 negative, 807 neutral) indicates imbalance, addressed later with SMOTE.

Text Preprocessing

Reviews are cleaned to remove noise, such as punctuation and stop words, while preserving sentiment-relevant words like "not." The cleaned text is used for feature extraction.

Feature Extraction and Data Balancing

TF-IDF vectorization converts text into numerical features, with a maximum of 5000 features and bigrams. SMOTE balances the training data to address the skewed sentiment distribution.

Model Training and Evaluation

Two models are trained: an SVM with a linear kernel on balanced TF-IDF features and a pre-trained BERT model for deep learning-based analysis. The SVM achieves 79% accuracy, with detailed metrics for each sentiment class.

User Interaction

An interactive loop allows users to input reviews, receiving sentiment predictions and confidence scores from both models in real-time.

2.3.2 Tools and Libraries

The project relies on a suite of Python libraries and tools to implement the sentiment analysis pipeline.

Python and Jupyter Notebook

Python 3 is used as the primary programming language, executed within a Jupyter Notebook on Google Colab, leveraging its GPU support for BERT model inference.

Data Processing Libraries

- `pandas`: For loading and manipulating the review dataset.
- `numpy`: For numerical operations, such as handling TF-IDF matrices.

- `re`: For regular expression-based text cleaning (e.g., removing punctuation).

Natural Language Processing Libraries

- `nltk`: Provides stop words for text preprocessing, excluding sentiment-relevant terms like "not."
- `scikit-learn`: Supplies `TfidfVectorizer` for feature extraction, `SVC` for SVM training, and `classification_report` for evaluation.
- `transformers`: From Hugging Face, provides the BERT-based sentiment analysis pipeline (`nlptown/bert-base-multilin`

Data Balancing and Model Persistence

- `imblearn`: Implements SMOTE (`imblearn.over_sampling.SMOTE`) to balance the dataset.
- `joblib`: Saves the TF-IDF vectorizer and SVM model for reuse.

2.3.3 Implementation Details

This subsection provides in-depth details of key components, including code snippets and example outputs, reflecting the project's functionality.

Text Preprocessing

The `clean_text` function preprocesses reviews by handling non-string inputs, removing punctuation, and filtering stop words. Below is the implementation:

```
# Clean text function
def clean_text(text):
    # Handle non-string inputs (e.g., if Text is missing or not a string)
    if not isinstance(text, str):
        text = str(text)
    text = re.sub(r'[^\w\s]', '', text)
    stop_words = set(stopwords.words('english')) - {'not', 'no', 'never'}
    text = ' '.join(word for word in text.split() if word.lower() not in
        stop_words)
    return text
```

This function is applied to the dataset to create a `Text_Cleaned` column. For example, the review "The taste is not so good" is cleaned to "taste not good," as shown in the output:

```
Cleaned review: taste not good
```

Feature Extraction and Data Balancing

The TF-IDF vectorizer is initialized with 5000 features and bigrams, transforming cleaned text into numerical features:

```
tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
X = tfidf.fit_transform(df['Text_Cleaned'])
y = df['Sentiment'].map({'positive': 1, 'negative': 0, 'neutral': 2})
```

SMOTE balances the training set, equalizing the number of samples across sentiments (6200 each for negative, positive, neutral):

```
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
```

The balanced distribution is verified:

```
Sentiment Distribution After SMOTE (Training Set):
0    6200
2    6200
1    6200
Name: count, dtype: int64
```

Model Training and Evaluation

The SVM model is trained on balanced data with a linear kernel and probability estimates:

```
svm = SVC(kernel='linear', probability=True, random_state=42)
svm.fit(X_train_balanced, y_train_balanced)
```

The model is evaluated on the test set, achieving 79% accuracy:

```
SVM Model Performance on Test Set:
              precision    recall  f1-score   support
    negative       0.54      0.62      0.58       287
    positive       0.91      0.87      0.89      1545
     neutral       0.29      0.35      0.32       168
    accuracy                           0.79      2000
   macro avg       0.58      0.61      0.60      2000
weighted avg       0.81      0.79      0.80      2000
```

The BERT model uses a pre-trained pipeline for sentiment analysis:

```
sentiment_analyzer = pipeline("sentiment-analysis", model="nlptown/bert-base-
    multilingual-uncased-sentiment")
```

User Interface for Real-Time Predictions

An interactive loop enables users to input reviews, with both models predicting sentiments and confidence scores:

```
while True:
    user_review = input("\nPlease enter a review (or type 'exit' to quit): ")
    if user_review.lower() == 'exit':
        print("Exiting the program. Goodbye!")
        break
    if not user_review.strip():
        print("Please enter a valid review.")
        continue
    predict_sentiment_svm(user_review)
    predict_sentiment_bert(user_review)
```

For the review "The taste is not good," the outputs are:

```
SVM Prediction for Review: The taste is not good
SVM Sentiment: negative
SVM Probabilities: {'negative': 0.9441443183028706, 'positive': 0.005039482886536689,

'neutral': 0.05081619881059254}

BERT Prediction for Review: The taste is not good
BERT Sentiment: negative
BERT Confidence Score: 0.4313 (Star Rating: 2)
```

Model Persistence

The TF-IDF vectorizer and SVM model are saved for future use:

```
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
joblib.dump(svm, 'svm_model.pkl')
```

Chapter 3

Conclusion

3.1 Discussion

The Intelligent Review Analyzer project successfully implemented a data mining pipeline to automate sentiment analysis of customer reviews, classifying them as positive, negative, or neutral, using a dataset of 10,000 reviews with an initial distribution of 7745 positive, 1448 negative, and 807 neutral sentiments. The workflow involved text preprocessing to remove punctuation and stop words, TF-IDF vectorization for feature extraction, and SMOTE to balance the dataset, enabling a linear SVM model to achieve 79% accuracy on the test set, with precision and recall metrics highlighting stronger performance for positive sentiments (0.91 precision, 0.87 recall) but lower for neutral (0.29 precision, 0.35 recall). A pre-trained BERT model complemented the SVM, providing deep learning-based predictions, as demonstrated by accurate negative sentiment classification for reviews like "The taste is not good" (SVM: 94.4% negative probability, BERT: 2-star rating with 0.4313 confidence). The interactive user interface allowed real-time review analysis, making the system practical for business applications such as customer feedback analysis and product improvement. Observations indicate that while the SVM excelled with balanced data, the BERT model offered nuanced sentiment detection, though both struggled with neutral sentiments due to their underrepresentation, suggesting future improvements in data augmentation or model tuning to enhance overall performance.

References

[1] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

[3] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, O'Reilly Media, 2009.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.