Munish Patwa

SENTIMENT_Industry_Project_Report_(Review_2)-1.pdf

Assignment 7

Research_1

Ganpat University

Document Details

Submission ID

trn:oid:::1:3238895406

Submission Date

May 4, 2025, 11:31 PM GMT+5:30

Download Date

May 4, 2025, 11:38 PM GMT+5:30

File Name

SENTIMENT_Industry_Project_Report__Review_2_-1.pdf

File Size

447.4 KB

27 Pages

3,008 Words

18,598 Characters

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

31 Not Cited or Quoted 10%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

7% Internet sources
5% Publications
0% Submitted works (Student Papers)

Match Groups

31 Not Cited or Quoted 10%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

7% Internet sources
5% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	Swati Raj, Jhalak Dutta, Saikat Bandopadhyay, Rajesh Prasad. "Chapter 18 Featur... 1%
2	Publication	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challeng... 1%
3	Internet	www.jetir.org <1%
4	Internet	fastercapital.com <1%
5	Publication	Narasimha Rao Vajjhala, Kenneth David Strang. "Cybersecurity in Knowledge Ma... <1%
6	Internet	github.com <1%
7	Internet	www.readkong.com <1%
8	Internet	interoncof.com <1%
9	Publication	Howard Anderson, Sharon Yull, Bruce Hellingsworth. "Higher National Computin... <1%
10	Internet	repository.tudelft.nl <1%

11	Internet	
www.coursehero.com		<1%
12	Internet	
core.ac.uk		<1%
13	Internet	
simeononsecurity.ch		<1%
14	Internet	
store.steampowered.com		<1%
15	Publication	
Beksultan Sagyndyk, Dilyara Baymurzina, Mikhail Burtsev. "Chapter 39 DeepPavl...		<1%
16	Internet	
epn.org		<1%
17	Internet	
medium.com		<1%
18	Internet	
www.ijeat.org		<1%
19	Internet	
www.nerc.com		<1%
20	Publication	
"Evolutionary Artificial Intelligence", Springer Science and Business Media LLC, 20...		<1%
21	Internet	
platforce.io		<1%
22	Publication	
H.L. Gururaj, Francesco Flammini, J. Shreyas. "Data Science & Exploration in Artifi...		<1%

INDEX

12

CHAPTER: 1 INTRODUCTION

CHAPTER 1 INTRODUCTION

In today's digital age, the vast amount of textual data generated through social media, product reviews, customer feedback, and online discussions has created an urgent need for effective sentiment analysis. Businesses, organizations, and researchers rely on sentiment analysis to extract meaningful insights from user-generated content, allowing them to understand public opinion, customer satisfaction, and market trends. Manually analyzing this data is time-consuming, inefficient, and prone to human bias. Therefore, leveraging artificial intelligence (AI) and machine learning (ML) for sentiment analysis provides a scalable & automated approach to processing and interpreting text data.

This project's goal is to develop a Sentiment Analysis Demo Application that enables users to input text reviews and obtain sentiment classification, keyword extraction, and relevant insights. The system utilizes a MLmodel trained on a Kaggle dataset to predict sentiment polarity (positive, negative, or neutral). The trained model is unified into a Flask-based web application, providing an interactive platform where users can analyze textual data effortlessly.

The core of this project revolves around Natural Language Processing (NLP), which involves techniques for text preprocessing,sentiment classification and feature extraction. The model is designed to capture sentiment-related features in textual data and make predictions according to the learned patterns. The application can be useful for businesses looking to analyze customer feedback, brands monitoring social media sentiment, and researchers studying opinion trends.

Tools & Technologies Used in This Project:

- **Flask: A lightweight web framework used to build the application's backend.**
- **Python: The primary programming language for model training, text processing, and API development.**
- **Natural Language Toolkit (NLTK) & SpaCy: NLP libraries used for text preprocessing, tokenization, stopword removal, and lemmatization.**
- **Scikit-learn: Machine learning library used for training and evaluating sentiment classification models.**
- **TensorFlow/Keras: Frameworks used for implementing deep learning models to enhance sentiment analysis accuracy.**
- **Pandas & NumPy: Libraries used for data manipulation, analysis, and feature extraction.**
- **Kaggle Dataset: A real-world dataset used to train the sentiment analysis model.**
- **Jupyter Notebook: Environment for experimenting with data, testing models, and fine-tuning parameters.**

19

CHAPTER: 2 PROJECT SCOPE

CHAPTER 2 PROJECT SCOPE

The scope of this project is to create an AI-powered Sentiment Analysis Demo Application that utilizes ML and (NLP) techniques to analyze user-generated text and classify it into sentiment categories. The system is designed to assist businesses, researchers, and analysts in extracting meaningful information from textual data, such as customer reviews, social media comments, and feedback. By automating sentiment classification, the application eliminates the need for manual analysis and ensures accurate and consistent sentiment evaluation.

The ultimate goal of this model is to construct a robust sentiment analysis model trained on a Kaggle dataset that can efficiently process user reviews and classify them as negative, positive or neutral. The system is integrated into a Flask-based web application, enabling users to input text and receive instantaneous results. The project also focuses on enhancing sentiment detection through keyword extraction and NLP-based feature engineering to provide deeper insights into the text.

Key Components of the System:

- Data Collection and Preprocessing: The sentiment analysis model is trained using a dataset obtained from Kaggle. Preprocessing techniques such as tokenization, stopword removal, stemming, and lemmatization are applied to clean and normalize the text data.
- Machine Learning Model: The project employs traditional ML models like Random Forest, Logistic Regression and Naïve Bayes as well as deep learning techniques using LSTMs or Transformers, to improve sentiment classification accuracy.
- Feature Engineering: NLP-based techniques, such as TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings, are utilized to extract valuable features out of the text, improving the model's ability to understand context.
- Flask API for Deployment: The trained model is deployed using Flask, allowing users to send text input via a web interface and receive real-time sentiment predictions.
- User Interface (UI): A simple and intuitive web-based UI enables users to enter text, view sentiment classification results, and analyze extracted keywords.

This project aims to provide an automated, scalable, and user-friendly sentiment analysis solution, helping stakeholders gain actionable insights from textual data. The system is designed to streamline sentiment evaluation, making it applicable in areas such as product review sentiment tracking, social media monitoring and customer feedback analysis.

CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

Minimum Hardware Requirements

Component	Specification
Processor	Intel Core i3 / AMD Ryzen 3 (or equivalent)
RAM	4GB
HDD	30GB

Table 3.1 Minimum Hardware Requirements

Minimum Software Requirements

Component	Specification
Operating System	Windows, macOS, or Linux
Web Browser	Chrome, Firefox, Edge, or Safari (latest versions recommended)
Python	Python 3.7 or above
Machine Learning Libraries	Scikit-learn, TensorFlow/Keras, Pandas, NumPy (for model training and evaluation)
Database	MySQL (for storing hotel records and querying the data)
IDE/Notebook	Jupyter Notebook or any preferred IDE (e.g., VSCode) for model development and experimentation

Table 3.2 Minimum Software Requirements

CHAPTER: 4 PROCESS MODEL

CHAPTER 4 PROCESS MODEL

This project follows a structured workflow to ensure accurate sentiment analysis of textual data. The system processes user-provided text, applies NLP techniques, and utilizes a trained machine learning model to classify sentiment and extract key insights. The workflow includes data collection, preprocessing, feature extraction, model training, and sentiment prediction. Below is a step-by-step breakdown of the process:

1. Data Collection

The dataset used for training the sentiment analysis model is obtained from Kaggle, containing user reviews and their corresponding sentiment labels (negative,positive or neutral). The dataset consists of a diverse set of text samples from different domains such as product reviews, social media comments and customer feedback

2. Data Preprocessing

Before training the model, the collected text data is cleaned and processed to enhance accuracy. The following preprocessing techniques are applied:

- Text Cleaning: Removing special characters, numbers, and unnecessary symbols.
- Tokenization: Dividing text into individual words or phrases.
- Stopword Removal: Eliminating commonly used words that do not add innate value to sentiment analysis (e.g., "is," "the," "and").
- Stemming & Lemmatization: Transforming words to their original form to maintain consistency (e.g., "running" → "run").

3. Feature Extraction & Selection

To improve sentiment classification accuracy, key features are extracted using (NLP) techniques such as:

- TF-IDF (Term Frequency-Inverse Document Frequency): Decides the importance/value of words in a document relative to the entire dataset.
- Word Embeddings (Word2Vec, GloVe): Captures contextual meaning and relationships between words.

- N-grams: Identifies patterns in sequences of words to improve sentiment detection.

4. Model Training

The processed data is used to train ML models for sentiment classification. Different models are evaluated, including:

- Logistic Regression & Naïve Bayes: Traditional models for text classification.
- Random Forest & Support Vector Machines (SVM): More advanced models improving accuracy.
- Deep Learning (LSTMs, Transformer-based models like BERT): Enhances sentiment detection using contextual learning.

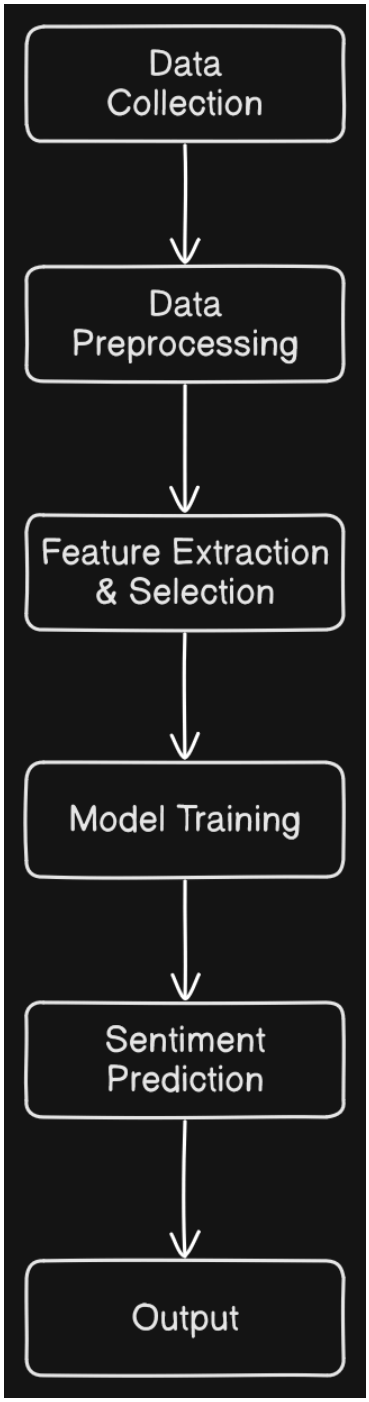
5. Sentiment Prediction

Once the model is trained, users can enter text data into the Flask-based web application. The model processes the text and predicts its sentiment category (positive, negative, or neutral), along with keyword extraction for deeper insights.

6. Output

The predicted sentiment and extracted keywords are displayed to the user, providing valuable insights into the emotional tone and key aspects of the text.

This structured workflow ensures an efficient and accurate sentiment analysis system, making it applicable for various industries, including customer feedback evaluation, brand sentiment monitoring, and social media trend analysis.


Data Collection
Data Preprocessing
Feature Extraction & Selection
Model Training
Sentiment Prediction
Output

CHAPTER: 5 PROJECT PLAN

CHAPTER 5 PROJECT PLAN

5.1 List of Major Activities

The development of the **Sentiment Analysis Demo Application** follows a structured workflow consisting of several key phases. Each phase ensures systematic progress, from research and data collection to model development, testing, and deployment. Below is the list of major activities involved in the project:

1. Research & Exploration

Understanding the problem domain of sentiment analysis and its real-world applications. Exploring different deep learning and machine learning models for sentiment classification (e.g., Naïve Bayes, Random Forest, LSTMs, BERT).Reviewing **NLP techniques** for text processing and feature extraction.

2. Data Collection & Preparation

Obtaining a sentiment-labeled dataset from **Kaggle** or other sources.Cleaning and normalizing text data by applying **preprocessing techniques** (removing stopwords, stemming, lemmatization, etc.).Splitting the dataset into **validation, training and test sets** for model evaluation.

3. Feature Engineering & Selection

Extracting relevant text features using **TF-IDF, word embeddings (Word2Vec, GloVe), and N-grams**.Identifying and selecting key linguistic patterns that influence sentiment classification.

4. Model Development & Training

Implementing different **ML models** (Logistic Regression, Naïve Bayes, Random Forest, SVM).Developing and training **deep learning models** (LSTMs, Transformer-based models like BERT).Comparing models to select the most **precise and efficient** approach.

5. Model Evaluation & Optimization

Evaluating model performance employing metrics such as **Recall, Precision, Accuracy and F1-score**.Fine-tuning hyperparameters to **improve accuracy and reduce overfitting**.Testing model generalization across **different datasets**.

6. Backend Development *(Optional, if required)*

Implementing a **Flask-based API** to handle text input and return sentiment predictions.Ensuring seamless integration with the trained **machine learning model**.

7. Frontend Development *(Optional, if required)*

Designing a **simple UI** where users can manually input text and view sentiment analysis results.Displaying **extracted keywords** and **sentiment classification** in an intuitive manner.

8. System Integration & Testing *(Optional, if required)*

Connecting the **frontend, backend, and machine learning model** for a seamless user experience.Conducting extensive testing to ensure stability, accuracy, and performance.

9. Final Deployment & Documentation

Deploying the system on a suitable platform (e.g., **Heroku, AWS, or a local server**).
Preparing project reports, **technical documentation, and user guides**.

5.2 Estimated Time Duration in Days

Activity	Estimated Duration (Days)
Research & Exploration	10 Days
Data Collection & Database Setup	8 Days
Data Preprocessing & Feature Engineering	12 Days
Model Development & Training	14 Days
Model Evaluation & Tuning	10 Days
Backend Development (Optional)	12 Days
Frontend Development (Optional)	12 Days
System Integration (Optional)	8 Days
Testing & Debugging (Optional)	10 Days
Final Deployment & Documentation (Optional)	6 Days

Total Estimated Duration: ~2.5 Months

Table 5.1 Task Completion Estimated Time Duration in Days

CHAPTER: 6 IMPLEMENTATION DETAILS

turnitin Page 21 of 31 - Integrity Submission Submission ID trn:oid:::1:3238895406

CHAPTER 6 IMPLEMENTATION DETAIL

This chapter provides an in-depth explanation of the implementation of the Sentiment Analysis Demo Application. It covers the system architecture, workflow, integration of technologies, and a step-by-step breakdown of each module involved.

6.1 System Architecture

The Sentiment Analysis system follows a client-server architecture, where the frontend, backend, and machine learning model work together to analyze user reviews and determine their sentiment. The main components of the system include:

Frontend (React.js)

- The frontend is developed using React.js, providing a simple and interactive user interface.
- Users enter text-based reviews, which are then sent to the backend for sentiment analysis.
- The results (sentiment score and keywords) are displayed to the user.

Backend (Flask API)

- The backend is implemented using Flask, which handles user requests and model predictions.
- It receives input from the frontend and preprocesses the text before passing it to the trained ML model.
- The backend returns sentiment analysis results to the frontend.

Machine Learning Model (Trained Sentiment Model)

- A pretrained sentiment analysis model (such as a Naïve Bayes Classifier, LSTM, or Transformer-based model) is used.
- It classifies the input review into positive, negative, or neutral sentiment and extracts important keywords.

6.2 Workflow Implementation

The sentiment analysis workflow involves multiple stages, from user input to model prediction and result display.

6.2.1 User Input & Frontend Processing

- The user enters a text review into the React-based frontend.
- The input is structured into a JSON request and sent to the Flask backend through an API call.

6.2.2 Text Preprocessing (Backend Processing)

- The backend receives the text and performs preprocessing, which includes:
    - Removing stopwords, special characters, and punctuation.
    - Converting the text to lowercase.
    - Tokenization and lemmatization.
- The cleaned text is then passed to the sentiment analysis model.

6.2.3 Sentiment Prediction Using Machine Learning Model

- The trained sentiment model analyzes the input text and predicts whether the sentiment is negative, positive or neutral.
- The model also extracts key terms that influenced the sentiment classification.

6.2.4 Query Execution & Result Fetching

- The backend processes the model's output and formats it into a structured JSON response.
- If necessary, the backend may fetch additional reference data (e.g., predefined sentiment scores for certain words).

6.2.5 Displaying Results to the User

- The frontend receives the sentiment prediction and displays:
    - The sentiment category (Positive/Negative/Neutral).
    - Confidence score of the prediction.
    - Important keywords that influenced the sentiment.

6.3 Machine Learning Model Integration and Prediction Workflow

This section describes how the machine learning model interacts with the backend and frontend to provide sentiment predictions.

6.3.1 Feature Engineering & Preprocessing

- The raw text input undergoes preprocessing to ensure better accuracy.
- Text cleaning techniques (tokenization, lemmatization, and stopword removal) are applied.
- TF-IDF, word embeddings (Word2Vec, GloVe), or transformer embeddings may be used to convert text into a numerical format.

6.3.2 Model Processing & Sentiment Prediction

- The preprocessed text is passed into the trained sentiment analysis model.
- The model predicts:
    - Sentiment label (Positive/Negative/Neutral).
    - Confidence score (e.g., 85% positive sentiment).
    - Top keywords contributing to the sentiment.

6.3.3 Response Evaluation & Data Processing

- The backend formats the prediction output into a structured response.
- The final result is returned to the frontend.

6.4 Backend Implementation - Flask API

- Handles API requests from the frontend.
- Processes text input and forwards it to the model.
- Returns predictions and extracted keywords.

6.5 Frontend Implementation - React.js

- Captures user input (text reviews).
- Displays sentiment results in a user-friendly format**.**

6.6 Conclusion

This chapter provides a comprehensive overview of the implementation details of the Sentiment Analysis Demo Application, highlighting the current progress and outlining the next steps required to complete the project. While key components, such as the sentiment analysis model and preprocessing pipeline, have been implemented, several aspects of the system are still under development.

Key Areas of Development

The following components require further work to achieve full functionality:

- Backend Development: The Flask-based backend, responsible for handling user inputs, processing text, and interacting with the machine learning model, is still being refined. Additional API endpoints for improved data handling may be introduced.
- Frontend Development: The React-based user interface, which enables users to input text and view sentiment analysis results, is in progress and will be further optimized for better user experience.
- Error Handling & Robustness: Implementing comprehensive error handling to manage invalid inputs, API failures, and unexpected outputs from the model is a key next step.
- Performance Optimization: Enhancing the speed of model inference, API response times, and frontend rendering will be addressed to ensure seamless user interactions.
- Security Enhancements: Implementing data validation, input sanitization, and API security measures to prevent vulnerabilities.
- Cloud Deployment: Currently, the system runs locally, and future deployment to a cloud platform (e.g., AWS, GCP, or Heroku) will be considered for scalability and accessibility.

10

CHAPTER: 7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

This project aims to develop a Sentiment Analysis Demo Application that leverages machine learning techniques to analyze textual data and determine sentiment polarity. The system integrates Natural Language Processing (NLP) models with a user-friendly web interface, allowing users to input text and receive sentiment predictions along with relevant insights.

So far, we have successfully:

- Developed and trained a sentiment analysis model using a labeled dataset from Kaggle.
- Implemented data preprocessing techniques, including text cleaning, tokenization, and vectorization.
- Designed an initial Flask backend to process user input, pass it through the model, and return predictions.
- Created the basic frontend framework using React for user interaction.

Although full-stack integration is still in progress, the key machine learning components—model training, evaluation, and prediction workflow—have been successfully implemented, proving the feasibility of the approach. Once fully developed, the system will provide a seamless and efficient tool for real-time sentiment analysis, benefiting users in various domains such as customer feedback analysis, social media monitoring, and business insights.

7.2 Future Work

To enhance the Sentiment Analysis Application, several key improvements and expansions are planned:

1. Full-Stack Integration:
    - Establish seamless communication between the React frontend and the Flask backend using REST APIs.
    - Enable real-time sentiment analysis, allowing users to input text dynamically and receive instant feedback.
2. Advanced Feature Extraction & Model Enhancement:

- Implement more advanced NLP techniques, such as word embeddings (Word2Vec, GloVe, or BERT), to improve model accuracy.
- Experiment with alternative models, including transformer-based models (BERT, RoBERTa), to compare their performance with the current approach.

3. Improved Error Handling & Security Measures:
   - Add input validation to handle missing or inappropriate text inputs.
   - Implement role-based access control (RBAC) to restrict unauthorized access.
4. User Interface Optimization & Testing:
   - Conduct user testing to gather feedback and refine the UI for a more intuitive experience.
   - Implement visual elements (e.g., graphs, confidence scores) to improve data interpretation.
5. System Deployment & Scalability:
   - Deploy the application to a cloud platform (AWS, Heroku, or GCP) for accessibility and scalability.
   - Optimize backend performance to handle high volumes of user requests efficiently.

CHAPTER: 8 REFERENCES

1. Pandas Documentation (read_sql method)
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_sql.html
2. FastAPI Documentation
https://fastapi.tiangolo.com/
3. React.js Documentation
https://reactjs.org/docs/getting-started.html
4. Dataset
https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews
5. Flask Documentation
https://flask.palletsprojects.com/en/stable/