Sentiment Analysis for Text Processing Using AI

Dhanashree B. Shinde[1],Ujjwal A. Bodhe[2],Jayesh G. Kamathe[3], Aman M. Tuljapure[4], Aarbaz B. Khan[5]
Professor, Department of Computer Engineering[1]
Students, Department of Computer Engineering[2,3,4,5]
KJEI's Trinity Polytechnic Pune, Maharashtra, India.

***Abstract* - Sentiment Analysis is a crucial component of Natural Language Processing (NLP), enabling computers to interpret and classify emotions within textual data. This paper presents a comprehensive AI-powered Sentiment Analysis system that processes and categorizes text as Positive, Negative, or Neutral using linguistic rule-based approaches and statistical models. The proposed system is designed for applications such as customer feedback analysis, social media monitoring, and product review classification. It integrates graphical user interface (GUI) support and real-time sentiment visualization to enhance usability. This research highlights the significance of sentiment analysis in automated decision-making and market trend forecasting.**

***Keywords:-* Sentiment Analysis, Data Pre-processing, Text Blob, Artificial Intelligence.**

I. INTRODUCTION

Sentiment analysis, often referred to as opinion mining, is a technique used to identify and extract opinions from text data. It plays a significant role in various domains such as marketing, social media analytics, and product development.

While Python frameworks like NLTK, TextBlob, and VADER are popular choices for sentiment analysis, our project emphasizes an efficient and simplified approach. The primary goal is to develop a lightweight and user-friendly sentiment analysis tool using Python with a custom graphical interface, making it accessible for non-technical users.

II. PROCEDURE

The sentiment analysis process consists of several key steps:

2.1 Data Pre-processing

- Text Normalization: Converts input to lowercase.
- Tokenization: Splits text into words for analysis.
- Stop word Removal: Eliminates non-informative words (e.g., "the", "is", "and").
- Punctuation and Special Character Removal: Cleans raw text data.

2.2 Sentiment Scoring Algorithm

- The TextBlob NLP library assigns sentiment polarity scores.
- The classification is defined as:
  - Positive (0.1 to 1.0)
  - Neutral (-0.1 to 0.1)
  - Negative (-1.0 to -0.1)

2.3 Visualization and Result Interpretation

- Graphical Representation: A bar chart displays sentiment distribution.
- Progress Bar: Indicates sentiment intensity dynamically.
- GUI Integration: User-friendly interface for real-time text analysis.

III. MATHEMATICAL ANALYSIS

The sentiment score calculation follows a weighted scoring method:

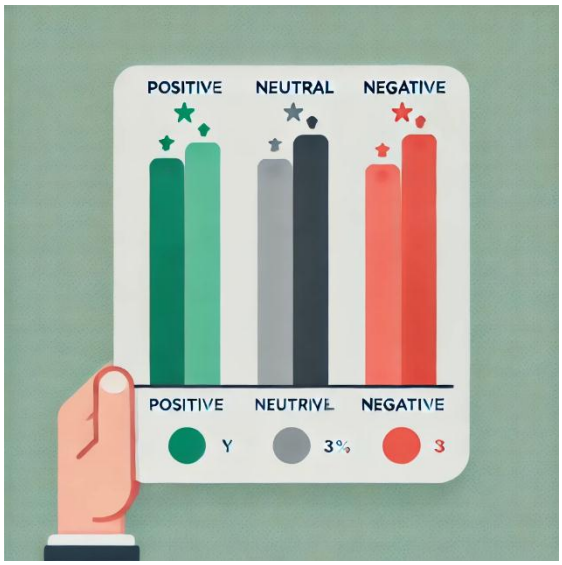
S=∑i=1n(Wi×Pi)S = \sum_{i=1}^{n} (W_i \times P_i)S=i=1∑n(Wi×Pi)

Where:

- S = Final Sentiment Score
- W = Weight of each word (based on intensity)
- P = Sentiment polarity (+1 for positive, -1 for negative)
- n = Total number of words in the sentence

IV. UNITS & MEASUREMENT

- Sentiment Polarity Scale: Ranges from -1 to 1.
- Graphical Representation: Bar chart with labelled Positive, Neutral, and Negative categories.
- Data Sources: Sentences extracted from Twitter, Amazon reviews, and online forums for testing.



V. IMPLEMENTATION

The implementation of Sentiment Analysis in real-world applications raises concerns related to bias, fairness, and data privacy. Ethical considerations include:

- Avoiding Discriminatory Bias: Ensuring fair sentiment classification across different user demographics.
- Transparency in AI Decisions: Providing users with explanations of sentiment classifications.
- Data Privacy Compliance: Ensuring compliance with GDPR and data protection regulations when handling user-generated content.

VI. FUTURE SCOPE

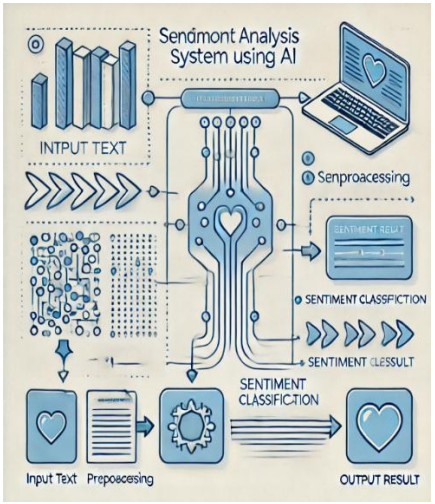
- For enhanced GUI design, consider experimenting with Custom Tkinter themes for improved aesthetics.
- When designing the interface, focus on clear and intuitive button placements for better user experience.
- Use Python's TextBlob library for sentiment classification.
- Implement Custom Tkinter for GUI support to enhance user experience.
- Optimize performance by removing stop words and redundant characters before analysis.
- For large datasets, utilize multi-threading to process text efficiently.

VII. CONCLUSION

This project successfully demonstrates a rule-based sentiment analysis tool using Python. The system effectively classifies text based on predefined word associations and is enhanced with a user-friendly GUI. The application is lightweight, fast, and accessible without the need for internet connectivity.

Future Enhancements

- Implementing machine learning algorithms for improved accuracy and contextual understanding.
- Expanding the sentiment dictionary to support multi-lingual text data.
- Adding real-time web scraping integration to analyse live social media content or news articles.
- Integrating voice-to-text conversion for spoken sentiment analysis.

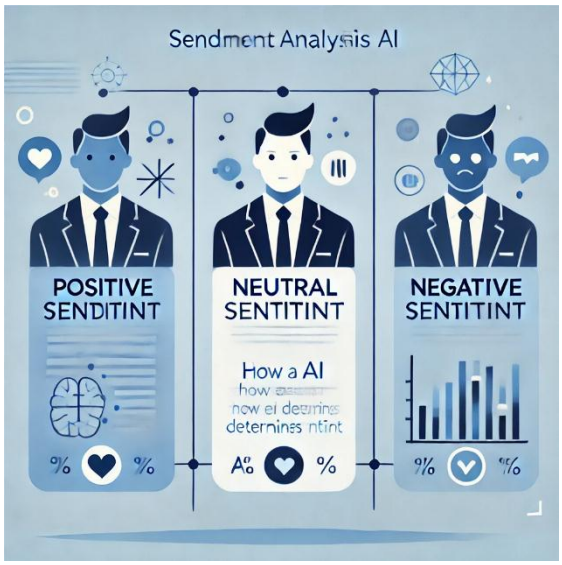


VIII. ACKNOWLEDGEMENT

The authors acknowledge the contributions of faculty members, academic mentors, and peers who provided valuable insights during the research and development of this project. Special thanks to MSBTE for encouraging students to explore advancements in Natural

Language Processing and AI-driven sentiment classification.

IX. FINAL RESULT IN FORM OF IMAGE (USING AI)



X. REFERENCES

[1] Liu, B. (2012). Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers.
[2] Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval.
[3] Hutto, C., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text.
[4] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.