Text Sentiment Analysis From Social edia

Project submitted to the
SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Masters of Technology

In

Computer Science and Engineering
School of Engineering and Sciences

Submitted by

G. Gowthami
(AP24122040018)

Under the Guidance of

(Dr.Ashu Abdul)

SRM University–AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh – 522 240
April, 2025

Certificate

Date: 25-04-2025

This is to certify that the work present in this Project entitled " **Text Sentiment Analysis From Social Media**" has been carried out by **G. Gowthami (AP24122040018)** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)
Prof. / Dr. Ashu Abdul

[Designation]
[Affiliation]

Contents

List of Figures

Abstract

The proliferation of user-generated content on social media, review sites, and blogs has created a vast source of opinionated text data. Manually analyzing the sentiment expressed in this data is impractical due to its volume and the time required. This project presents a simple yet effective Sentiment Analysis tool developed using Python. It leverages the VADER (Valence Aware Dictionary and sEntiment Reasoner) model, accessed via the NLTK library, for sentiment scoring. A graphical user interface (GUI) built with Tkinter allows users to input text easily. The system analyzes the input and classifies the sentiment as Positive, Negative, or Neutral, providing immediate feedback with color-coding and emojis. This tool serves as a fast and accessible solution for basic sentiment evaluation, suitable for educational purposes or quick analysis of informal text.

Abbreviations

API Application Programming Interface

BERT Bidirectional Encoder Representations from Transformers

GUI Graphical User Interface

NLTK Natural Language Toolkit

NLP Natural Language Processing

VADER Valence Aware Dictionary and sEntiment Reasoner

Chapter 1

Introduction

1.1 Background and Motivation

In the digital age, individuals constantly express their opinions, emotions, and experiences online through various platforms like social media, product reviews, and personal blogs. Understanding the underlying sentiment or emotional tone in this text data is crucial for businesses, researchers, and individuals alike. Sentiment analysis, also known as opinion mining, is the computational study of opinions, sentiments, and emotions expressed in text. Manually sifting through and analyzing large volumes of text for sentiment is inefficient, time-consuming, and often subjective. Automated sentiment analysis tools offer a scalable solution.

This project focuses on developing a desktop application for sentiment analysis. It utilizes Python, a versatile programming language popular in data science and NLP, along with the NLTK (Natural Language Toolkit) library. Specifically, it employs the VADER sentiment analysis tool, which is particularly well-suited for text from social media due to its sensitivity to nuances like capitalization, punctuation, and negations. To make the tool user-friendly, a simple GUI is created using Tkinter, Python's standard GUI library.

1.2 Problem Statement

The sheer volume of text data generated online presents a significant challenge for understanding public opinion and customer feedback in real-time. Manual analysis is not scalable, is prone to human bias, and incurs substantial time and cost. While sophisticated deep learning models exist, there is also a need for simple, fast, and accessible tools for basic sentiment analysis tasks, especially for users without extensive technical expertise or computational resources. Such a tool can be valuable for quick checks, learning NLP concepts, or preliminary analysis of informal text like tweets or short comments. Therefore, the problem is to create an easy-to-use desktop application that performs rule-based sentiment analysis quickly and provides clear, understandable results.

1.3 Objectives

The primary objectives of this project are:

- To develop a functional desktop application for sentiment analysis using Python.
- To implement sentiment classification (Positive, Negative, Neutral) using the VADER lexicon.
- To create an intuitive Graphical User Interface (GUI) using Tkinter for user input and output display.
- To provide clear visual feedback including color-coding and emojis based on the detected sentiment.
- To demonstrate a practical application of NLP techniques for basic opinion mining.

1.4 Scope

The scope of this project includes the design, implementation, and basic testing of a sentiment analysis tool. It focuses on using the VADER lexicon accessed via NLTK for analyzing English text input through a Tkinter GUI. The analysis provides a classification into Positive, Negative, or Neutral categories based on the VADER compound score. The project does not include implementing advanced machine learning models, handling multiple languages, or analyzing sentiment from real-time data streams via APIs. It serves as a proof-of-concept and educational tool for rule-based sentiment analysis.

Chapter 2

Literature Survey

The foundation of this project relies on established work in sentiment analysis, particularly rule-based approaches suitable for social media text. Key literature includes:

- **Hutto, C. J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text [1].** This seminal paper introduced VADER (Valence Aware Dictionary and sEntiment Reasoner). The authors presented VADER as a lexicon and rule-based sentiment analysis tool specifically attuned to sentiments expressed in microblogging contexts like Twitter. It uses a combination of a sentiment lexicon (a list of words with associated polarity scores) and heuristic rules that consider factors like punctuation (e.g., '¡), capitalization (e.g., 'GREAT'), intensifiers (e.g., 'very'), and negation. They demonstrated VADER's effectiveness, showing it often outperformed other benchmarks on social media data. Its parsimonious nature means it doesn't require training data, making it computationally efficient. This work directly forms the basis for the sentiment analysis method used in this project.

- **Veena, G., Vinayak, A., & Nair, A. J. (2021). Sentiment Analysis using Improved VADER and Dependency Parsing [2].** This more recent work explores using VADER, potentially in an improved or combined fashion. The study investigated combining VADER's lexicon-based approach with dependency parsing to potentially enhance accuracy by considering grammatical relationships between words within the sentence structure. This research highlights the continued relevance of VADER as a baseline or component in sentiment analysis systems, even as researchers explore ways to augment its capabilities by integrating it with more complex linguistic analysis techniques.

These studies validate the choice of VADER for this project, establishing it as a well-regarded, efficient, and effective tool for analyzing the sentiment of informal, social media-style text, which aligns with the project's goal of creating a simple and fast analyzer using its core principles and lexicon.

Chapter 3

Methodology

The sentiment analysis process in this application follows a structured workflow, integrating user interaction via the GUI, text processing using Python and NLTK, and sentiment classification based on the VADER lexicon.

3.1 Workflow

The methodology involves the following steps:

1. **Input Collection:** The user provides English text input through the text area widget in the Tkinter GUI.

2. **Preprocessing:** The input text undergoes basic preprocessing. The implemented code converts the text to lowercase and tokenizes it into individual words using regular expressions (`re.findall(r'\w+', text)`). This step simplifies the text and prepares it for lexicon lookup by implicitly removing punctuation.

3. **Valence Lookup:** Each word token is looked up in the VADER sentiment lexicon, accessed via `nltk.sentiment.vader.SentimentIntensityAnalyzer().lexicon`. Every word is assigned a valence score based on the lexicon, typically ranging from -4 (most negative) to +4 (most positive). Words not present in the lexicon are assigned a neutral score of 0.0.

4. **Score Aggregation:** The valence scores of all identified words in the input text are summed arithmetically to obtain a raw aggregate sentiment score for the entire text.

5. **Normalization (Compound Score Calculation):** The aggregated raw score is normalized to produce a single 'compound' score, constrained between -1 (most negative) and +1 (most positive). The normalization formula employed is consistent with VADER's approach:

$$\text{compound} = \frac{\text{score}}{\sqrt{\text{score}^2 + \alpha}} \tag{3.1}$$

   Where *score* is the aggregated raw score, and α is a normalization constant (set to 15 in the implementation), which helps moderate the influence of extreme raw scores.

6. **Classification:** The calculated compound score determines the final sentiment classification using standard VADER thresholds:
   - Sentiment is **Positive** if compound score ≥ 0.05.
   - Sentiment is **Negative** if compound score ≤ -0.05.
   - Sentiment is **Neutral** if $-0.05 <$ compound score < 0.05.
7. **Output Presentation:** The determined sentiment classification (Positive, Negative, or Neutral), along with supporting details (analyzed words with scores, total raw score, normalized compound score), is displayed back to the user within the Tkinter GUI's result label. The output is enhanced with color-coding (green for positive, red for negative, blue for neutral) and relevant emojis (, ,).

3.2 VADER Model

The core sentiment analysis logic is based on the principles and lexicon of VADER (Valence Aware Dictionary and sEntiment Reasoner). Key aspects of the VADER model relevant to this project include:

- **Rule-Based & Lexicon-Based:** Operates on a pre-defined dictionary (lexicon) of words with sentiment scores (valences) and associated grammatical rules, eliminating the need for model training.
- **Social Media Focus:** Specifically designed and validated for the nuances of social media text, including informal language.
- **Compound Score:** VADER's primary output metric for overall sentiment is the compound score, calculated via normalization (Equation 3.1), which is used for classification in this tool.
- *Note:* While the full VADER model includes sophisticated heuristics for intensifiers, negation, capitalization, etc., the implementation in this project uses a simplified approach focusing on direct lexicon lookup and summation followed by normalization, leveraging the core VADER lexicon.

3.3 System Architecture

The application employs a simple client-side architecture comprising two main layers:

1. **User Interface (UI) Layer (Tkinter):** Manages user interaction through standard Tkinter widgets (`tk.Label`, `tk.Text`, `tk.Button`). It captures user input and displays the analysis results, running the main event loop (`app.mainloop()`) to remain responsive.
2. **Backend Logic Layer (Python Script):** Contains the core processing logic.

- Handles the button click event (`analyze_sentiment` function).
- Retrieves and preprocesses the input text.
- Executes the sentiment analysis using the `vader_like_manual_analyzer` function, which performs lexicon lookup, score aggregation, normalization, and classification.
- Updates the UI's result label with the formatted output string and appropriate color.

Libraries Used:

- **Tkinter:** For building the GUI.
- **NLTK:** For accessing the VADER sentiment lexicon.
- **re:** For text tokenization using regular expressions.

```
+-----------------------------------------------------------+
|                   User Interface Layer                     |
|                   (Tkinter GUI)                            |
|                                                            |
|  +-----------------------+     +----------------------+    |
|  | tk.Label (Title)      |     | tk.Label (Results)   |    |
|  +-----------------------+     +----------------------+    |
|  | tk.Text (Input Box)   |                               |
|  +-----------------------+                               |
|  | tk.Button (Analyze)   |                               |
|  +-----------------------+                               |
|                 |                                        |
|                 v (Event Trigger: Button Click)          |
+-----------------------------------------------------------+
                  |
                  v
+-----------------------------------------------------------+
|                 Backend Logic Layer (Python)               |
|                                                            |
|  Function: analyze_sentiment()                             |
|    |- Retrieves text from the input widget                 |
|    |- Preprocesses text                                    |
|    |- Calls vader_like_manual_analyzer()                   |
|    |    |- Lexicon Lookup                                  |
|    |    |- Score Aggregation                               |
|    |    |- Score Normalization                             |
|    |    |- Sentiment Classification                        |
|    |- Updates UI Result Label with score + color           |
+-----------------------------------------------------------+
```

Figure 3.1: System Architecture Diagram showing UI, Backend Logic, and Library interaction.

3.4 Implementation Details

The application is implemented as a single Python script (see Appendix **??**). The NLTK library is used to download and access the VADER lexicon. The `tkinter` library provides the GUI components. The core analysis logic resides in the `vader_like_manual_analyzer` function, which encapsulates preprocessing, scoring, normalization, and classification. Error handling is included for empty input using `tkinter.messagebox`. The final output provides not only the sentiment category but also a breakdown of the process for transparency.

Chapter 4

Results and Discussion

The developed application provides a functional and interactive tool for basic sentiment analysis, successfully meeting the project objectives.

4.1 Tool Interface

The GUI, built using Tkinter, presents a clean and simple interface. It consists of:

- An instruction label.
- A multi-line text entry box for user input.
- An "Analyze Sentiment" button to trigger the analysis.
- A result display area where the classification, detailed scores, and color-coded feedback appear.

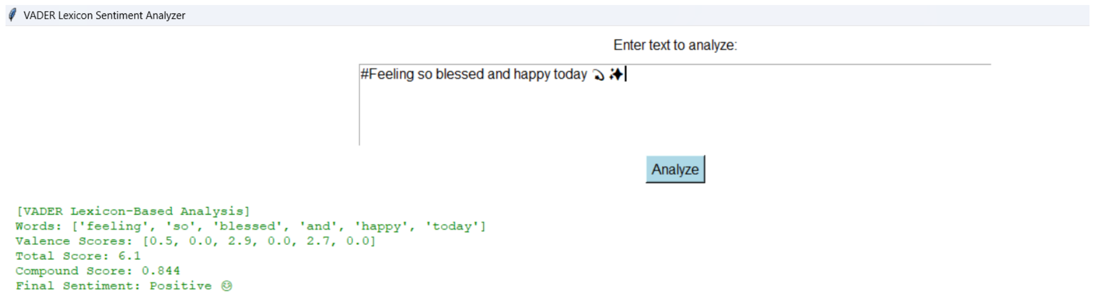
The interface is intuitive and requires no prior technical knowledge to operate.

4.2 Sample Outputs

Testing with various inputs demonstrates the tool's functionality, aligning with the expected VADER-based classification:

- **Positive Input:** For text like *"The service was amazing, and I had a great time"*, the tool correctly identifies the strong positive valence ("amazing", "great") and classifies the sentiment as **Positive** (displayed in green). (See Figure 4.1).
- **Negative Input:** For text like *"This is the worst product I have ever used. Completely disappointed."*, the negative keywords ("worst", "disappointed") lead to a classification of **Negative** (displayed in red). (See Figure 4.2).
- **Neutral Input:** For objective text lacking strong sentiment cues, such as *"I went to the store and bought some groceries."*, the tool accurately classifies it as **Neutral** (displayed in blue). (See Figure 4.3).

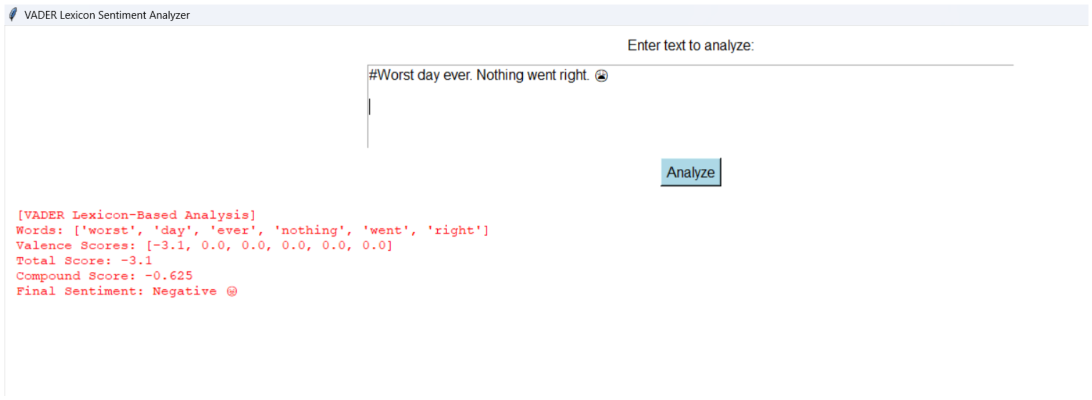
The inclusion of analyzed words with their scores, the total raw score, and the normalized compound score in the output provides valuable insight into how the classification was reached.



```
#Feeling so blessed and happy today 😊✨
```

```
[VADER Lexicon-Based Analysis]
Words: ['feeling', 'so', 'blessed', 'and', 'happy', 'today']
Valence Scores: [0.5, 0.0, 2.9, 0.0, 2.7, 0.0]
Total Score: 6.1
Compound Score: 0.844
Final Sentiment: Positive 😊
```

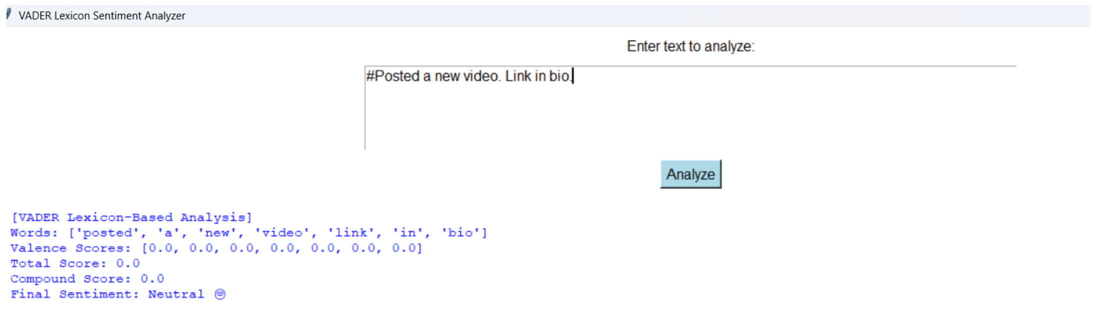
Figure 4.1: GUI Screenshot - Positive Sentiment Example.



```
#Worst day ever. Nothing went right. 😞
```

```
[VADER Lexicon-Based Analysis]
Words: ['worst', 'day', 'ever', 'nothing', 'went', 'right']
Valence Scores: [-3.1, 0.0, 0.0, 0.0, 0.0, 0.0]
Total Score: -3.1
Compound Score: -0.625
Final Sentiment: Negative 😞
```

Figure 4.2: GUI Screenshot - Negative Sentiment Example.



```
#Posted a new video. Link in bio
```

```
[VADER Lexicon-Based Analysis]
Words: ['posted', 'a', 'new', 'video', 'link', 'in', 'bio']
Valence Scores: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Total Score: 0.0
Compound Score: 0.0
Final Sentiment: Neutral 😐
```

Figure 4.3: GUI Screenshot - Neutral Sentiment Example.

4.3 Performance and Limitations

Performance:

- **Speed:** The tool is fast and provides near real-time analysis for typical input lengths due to the efficiency of the rule-based VADER lexicon lookup.

- **Suitability:** It is well-suited for basic sentiment analysis tasks, educational purposes, and quick checks of informal text.

Limitations:

- **Complexity Handling:** The simplified VADER implementation (without full heuristics) and the inherent limitations of rule-based systems mean it may struggle with sarcasm, irony, complex sentence structures, and context-dependent sentiment.
- **Scalability:** Being a desktop Tkinter application, it is designed for individual, interactive use and is not suitable for batch processing large volumes of data.
- **Domain Specificity:** Accuracy may decrease for highly specialized domains with jargon not present in the general VADER lexicon.
- **Language:** The current implementation is limited to English text analysis.

Despite these limitations, the tool effectively demonstrates the core concepts of sentiment analysis using a widely recognized lexicon and provides a practical, user-friendly interface.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This project successfully developed and implemented a desktop application for sentiment analysis using Python, the VADER lexicon via NLTK, and a Tkinter-based graphical user interface. The application effectively takes user text input, analyzes it based on word valence scores, and classifies the overall sentiment as Positive, Negative, or Neutral using VADER's compound score thresholds. The results are presented clearly to the user with color-coding and emojis for immediate understanding.

The tool serves as a practical demonstration of applying NLP techniques for opinion mining. It is fast, lightweight, and particularly adept at handling the informal language often found in social media contexts, aligning with VADER's design principles. While possessing limitations inherent to rule-based systems and its scope, the application fulfills its objective as a straightforward, real-time sentiment analysis tool suitable for educational exploration and basic sentiment evaluation tasks.

5.2 Future Scope

Several avenues exist for enhancing the capabilities and utility of this sentiment analysis tool:

1. **Full VADER Heuristics:** Implement the complete set of VADER's rules (handling intensifiers, negation scope, punctuation, capitalization) for potentially improved accuracy that matches the standard VADER analyzer more closely.

2. **Real-time Data Integration:** Integrate the tool with social media APIs (e.g., Twitter, Reddit) to fetch and analyze live data streams, enabling real-time monitoring of public sentiment.

3. **Advanced Model Integration:** Explore replacing or complementing the VADER approach with more sophisticated machine learning or deep learning models (e.g., Naive Bayes, SVM, BERT) trained on larger datasets, which could potentially handle more complex linguistic nuances like sarcasm and context, albeit at a higher computational cost.

4. **Multilingual Support:** Extend functionality to analyze text in languages other than English by incorporating multilingual sentiment lexicons or models.

5. **Web/Mobile Deployment:** Re-engineer the application as a web service (using frameworks like Flask or Django) or a mobile application to broaden accessibility and eliminate the need for local Python/NLTK setup.

6. **Aspect-Based Sentiment Analysis (ABSA):** Enhance the tool to identify sentiment towards specific entities or aspects mentioned within the text (e.g., analyzing sentiment towards 'battery life' and 'camera' separately in a phone review).

These future developments could transform the tool from a basic demonstrator into a more robust and versatile sentiment analysis platform.

Bibliography

[1] Hutto, C. J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media (ICWSM-14)*, pp. 216–225.

[2] Veena, G., Vinayak, A., & Nair, A. J. (2021). Sentiment Analysis using Improved VADER and Dependency Parsing. In *2021 2nd Global Conference for Advancement in Technology (GCAT)* (pp. 1–6). IEEE. doi: 10.1109/GCAT52182.2021.9587779.

[3] Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc. (Reference for NLTK library)

[4] Python Software Foundation. Tkinter — Python interface to Tcl/Tk. Retrieved from https://docs.python.org/3/library/tkinter.html (Reference for Tkinter - typically cited via Python documentation)