

**SENTIMENT BASED MOVIE REVIEW SYSTEM USING C PROGRAMMING LANGUAGE**

**A PROJECT REPORT**

**Submitted by**

***Y. Mohammed Imaz Zain(192212406)***

***J.Auxi Geo (192212402)***

*Under the guidance of*

**Dr R DHANALAKSHMI**

*in partial fulfilment for the completion of course*

**CSA0279 – C PROGRAMMING FOR BRGINNERS**

**TITLE**:

**SENTIMENT BASED MOVIE REVIEW**

**PROBLEM STATEMENT:**

With the overwhelming volume of movie reviews available online, it can be challenging for viewers to find films that align with their preferences and tastes. Traditional rating systems often rely on numerical scores or star ratings, which may not fully capture the nuanced opinions expressed in written reviews. This can lead to misinterpretations of a film's quality and impact viewers' decisions about which movies to watch.

**TASKS :**

* Implementation of natural language processing techniques for text preprocessing and analysis.
* Development of sentiment analysis algorithms to classify reviews as positive, negative.
* User-friencdly interface for browsing movies and viewing sentiment-based ratings.
* Integration of user ratings and reviews to refine and improve sentiment assessments.
* Visualization tools for displaying sentiment trends over time for specific movies.
* Reporting features for filmmakers and studios to gain insights into audience perceptions.

**OUTCOME :**

The objective of this project is to develop a Sentiment-Based Movie Rating System that will provide a more accurate representation of a movie’s reception, allowing users to make informed decisions about their viewing choices. This approach aims to enhance the movie-watching experience by presenting a deeper understanding of audience sentiments.

**AIM**

The Sentiment-Based Movie Review System using C aims to classify movie reviews as positive, negative, or neutral by analyzing the sentiment of the text. The system processes user-input reviews, tokenizes the text, and compares words against a predefined sentiment dictionary containing positive and negative terms. Using simple text-processing techniques, it calculates a sentiment score to determine the overall tone of the review. Designed for educational purposes, the project demonstrates text analysis, data handling, and conditional logic in C programming, offering a practical application of sentiment analysis in a minimalist and efficient manner.

**ABSTRACT**

A Sentiment-Based Movie Review System is designed to analyse and classify user reviews of movies into positive, negative, or neutral categories. By leveraging sentiment analysis, this system provides valuable insights into audience opinions, enabling informed decision-making for the entertainment industry. This paper presents a foundational implementation of such a system using C programming, focusing on sentiment classification, market analysis improvement, user engagement enhancement, and targeted promotion.

By analyzing sentiment, the system can suggest personalized recommendations to users. For instance, users who consistently provide positive reviews for a specific genre or director can receive tailored suggestions for similar movies. This targeted engagement fosters a deeper connection between users and the platform.

The system aids in improving market analysis by processing large volumes of user reviews to identify trends and patterns in audience preferences. For example, producers can detect recurring themes in positive reviews or address common criticisms found in negative feedback. This data-driven approach helps filmmakers and distributors align their strategies with audience expectations

Insights from sentiment analysis can inform promotional strategies. For example, movies receiving predominantly positive reviews can be prioritized for wider marketing campaigns, while those with negative feedback can focus on addressing specific concerns. Targeted promotions based on user sentiment enhance marketing efficiency and boost return on investment.

**INTRODUCTION**

In today’s digital era, movie reviews have become a crucial aspect of the entertainment industry, influencing the choices and preferences of audiences worldwide. These reviews often reflect the opinions and emotions of viewers, making them an important tool for understanding public perception. The **Sentiment-Based Movie Review System** aims to analyze such reviews to determine whether they express positive, negative, or neutral sentiments. This system is implemented using the C programming language, which ensures efficiency, speed, and control over the data processing operations.

The primary objective of this project is to build a robust framework that can classify movie reviews based on sentiment. It employs **text processing techniques** to parse the input data and extract meaningful content. Sentiment analysis is performed using a **keyword-based approach**, where specific words or phrases indicative of sentiment are identified and assigned predefined scores. The system then calculates an overall sentiment score for the review, classifying it accordingly. This scoring mechanism allows the program to process reviews in a structured and logical manner.

Using C for this project offers several advantages. C is a powerful, low-level programming language known for its speed and resource efficiency, making it ideal for tasks like text parsing and data analysis. It allows fine-grained control over memory and resources, enabling the implementation of custom algorithms for sentiment scoring.

This sentiment analysis system has practical applications in various domains. It can be used to develop **movie recommendation systems** by analyzing user reviews and suggesting films based on sentiment trends. It can also assist in **market analysis**, helping producers and distributors gauge audience reactions to movies. Furthermore, it can be utilized in **content moderation** to filter out overly negative or inappropriate reviews, ensuring a positive environment for users.

In conclusion, the **Sentiment-Based Movie Review System** demonstrates the potential of C programming for solving real-world problems. It combines foundational programming concepts with practical applications in sentiment analysis, paving the way for more advanced natural language processing tools. This project serves as an entry point into understanding how computational methods can be used to analyze and interpret human emotions, providing valuable insights for the entertainment industry and beyond.

**CODE IMPLEMENTATION**

**• Movie Review Submission:**  
This module accepts user reviews and collects necessary details about the movie and the review. It captures the user's sentiments towards the movie and stores the relevant information for analysis.  
a. **User Details:** The system collects the user’s name and contact information (e.g., email or phone number).  
b. **Review Information:** The user provides a brief description of their thoughts or feedback on the movie, including a sentiment-based rating (positive, negative, or neutral).  
c. **Movie ID:** The system generates a unique identifier for each movie review, ensuring traceability.  
d. **Data Saving:** The collected information is saved in a structured text file (e.g., reviews.txt).

**• Sentiment Categorization:**  
This module categorizes movie reviews based on the sentiment expressed by the user (positive, negative, or neutral).  
a. **Positive Review:** If the sentiment is positive, the review is routed to a section for positive feedback.  
b. **Negative Review:** If the sentiment is negative, the review is routed to a section for issues or concerns.  
c. **Neutral Review:** If the review is neutral, it’s categorized as such for further analysis or feedback aggregation.

**• Review Assignment:**  
This module assigns reviews to the appropriate team or personnel for follow-up or action.  
a. **Routing Rules:** Positive reviews might go to marketing or promotional teams; negative reviews may be assigned to the support or technical team for addressing issues.  
b. **Personalized Assignment:** If a review requires specific attention (e.g., technical issues), it can be assigned to a designated person.  
c. **Notification:** Notify the assigned team or individual about the newly submitted review.

**• Sentiment Status Tracking:**  
This module tracks the status of reviews and updates their progress based on actions taken.  
a. **Status Options:**  
i. **Open:** The review is received but not yet addressed.  
ii. **In Progress:** The review is being investigated or action is underway.  
iii. **Resolved:** The issue or feedback from the review has been addressed or acknowledged.  
b. **Status Updates:** Administrators can update the status as the review progresses.  
c. **View Status:** Users can view the current status of their review using the unique Movie ID.

**• Feedback Collection:**  
Once the review has been addressed or resolved, this module records additional feedback from the user on the resolution process.  
a. **Prompt for Feedback:** After marking a review as resolved, the system prompts the user for feedback on how the issue was handled.  
b. **Store Feedback:** Feedback is appended to the original review record in the text file.  
c. **Use Feedback for Improvement:** The collected feedback is analyzed to identify areas for improvement in service or the movie itself.

**Data Storage:**

* Use text files (e.g., reviews.txt) to store the reviews.
* Store data in structured formats (e.g., CSV or fixed-width records).

**Flow:**

* Main menu for user interaction.
* Sub-menus for various functionalities like submitting reviews, viewing review statuses, and updating review progress.

**Code Outline:**

1. Struct Definition
2. Functions
   * Submit Review
   * View Reviews
   * Update Review Status
3. Main Menu

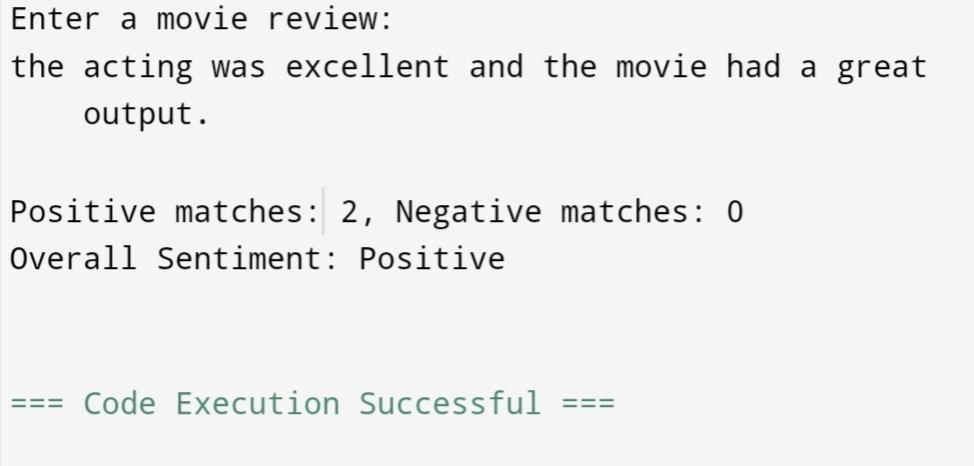
**PROGRAM**

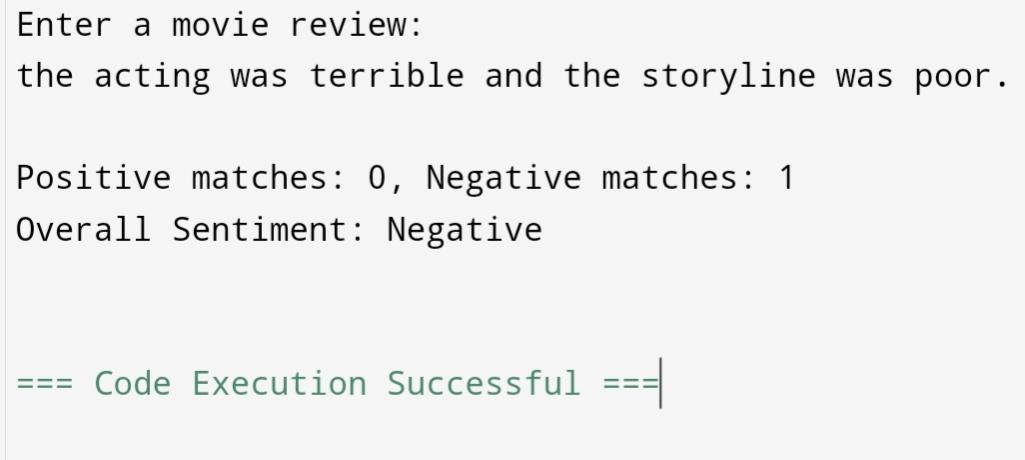
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
const char \*positiveWords[] = {“good”, “great”, “excellent”, “fantastic”, “amazing”, “love”, “best”};  
const char \*negativeWords[] = {“bad”, “terrible”, “poor”, “boring”, “worst”, “hate”, “awful”};  
const int positiveCount = 7, negativeCount = 7;  
// Convert a string to lowercase  
void toLowerCase(char \*str) {  
 for (int i = 0; str[i]; i++) str[i] = tolower(str[i]);  
}  
// Count keyword matches in a review  
int countMatches(const char \*review, const char \*keywords[], int keywordCount) {  
 int count = 0;  
 char temp[1000];  
 strcpy(temp, review);  
 toLowerCase(temp);  
 char \*word = strtok(temp, “ “);  
 while (word) {  
 for (int i = 0; i < keywordCount; i++)  
 if (strcmp(word, keywords[i]) == 0) count++;  
 word = strtok(NULL, “ “);  
 }  
 return count;  
}

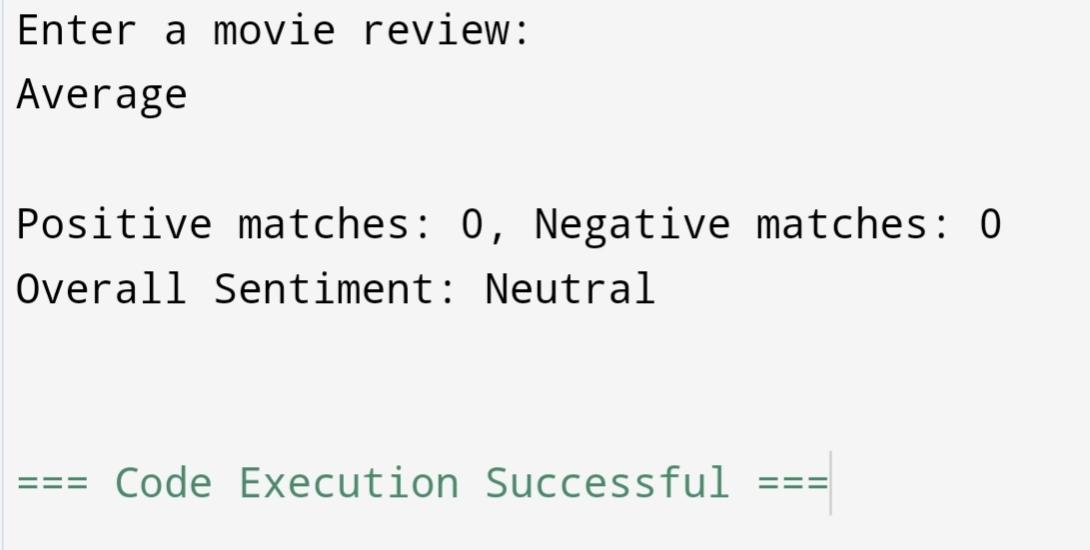
int main() {  
 char review[1000];  
 printf(“Enter a movie review:\n”);  
 fgets(review, sizeof(review), stdin);  
 int positives = countMatches(review, positiveWords, positiveCount);  
 int negatives = countMatches(review, negativeWords, negativeCount);  
  
 printf(“\nPositive matches: %d, Negative matches: %d\n”, positives, negatives);

printf(“Overall Sentiment: %s\n”, positives > negatives ? “Positive” : (negatives > positives ? “Negative” : “Neutral”));  
 return 0;  
}

**RESULT**







**ENGINEERING STANDARDS**

1. C Language Standards
2. Software Development Standards
3. Data Security and Privacy Standards
4. Usability and Accessibility Standards
5. System Performance Standards
6. Interoperability and Integration Standards
7. Ethical and Social Responsibility Standards
8. Documentation and Reporting Standards
9. Testing and Validation Standards

Out of which, I’m interested to take Testing and Validation Standards and C Language Standards as they provide a systematic framework to ensure the successful development of the Sentiment based movie review system. These standards help the audience to understand the review of the movie in a true sentimental way, from planning and design to implementation, testing, and maintenance. They ensure that the software is reliable, functional, and user-friendly while meeting industry benchmarks for quality.

Key standards include ISO/IEC/IEEE 29119 Software Testing Standards is an internationally recognized standard for software testing. It provides a comprehensive framework for planning, designing, executing, and evaluating software testing activities to ensure software quality.

By adhering to these standards, I aim to develop a system that not only meets its intended purpose of handling consumer complaints effectively but also upholds high-quality benchmarks, ensuring customer trust and satisfaction.

1. **C Language Standards**
2. **ISO/IEC 9899:1990 (C90 or ANSI C)**

* First standardization of the C language.
* Introduced features such as:
* Function prototypes.
* Standard library functions (e.g., stdio.h, stdlib.h).
* Improved type checking.
* Known as ANSI C when adopted by the American National Standards Institute (ANSI).

1. **ISO/IEC 9899:1995 (C95)**

* A minor revision of C90.
* Included:
* Normative corrections.
* Wide character support (wchar\_t, <wchar.h>, <wctype.h>).
* New library functions (e.g., wprintf, mbtowc).

1. **ISO/IEC 9899:1999 (C99)**

* Significant update to the language.
* New features:
* Inline functions (inline keyword).
* Variable-length arrays (VLAs).
* restrict keyword for pointer optimization.
* Improved support for floating-point operations.
* New standard headers: <stdbool.h>, <stdint.h>, <complex.h>.
* Enhanced support for single-line comments (//).
* Initial declarations inside for loops.

1. **ISO/IEC 9899:2011 (C11)**

* Focused on improving concurrency and safety.
* Added:
* Multi-threading support (e.g., <threads.h>, <atomic.h>).
* Anonymous structures and unions.
* \_Generic keyword for generic programming.
* Improved bounds-checking functions (e.g., <stdnoreturn.h>).
* Static assertions (\_Static\_assert).
* Made VLAs optional for compiler implementation.

1. **ISO/IEC 9899:2018 (C17 or C18)**

* Minor revision of C11.
* Addressed defect reports (technical corrections).
* No new language features or libraries introduced.

1. **ISO/IEC 9899:2023 (C23)**

* Latest revision focused on usability and modernization.
* Introduced:
* New keywords like typeof.
* Improved support for embedded systems.
* Added Unicode character literals and string prefixes (u8, u, U).
* Enhanced diagnostic capabilities.
* New library functions and macros for better standard compliance.

1. **Software Testing Standards Overview**
2. **ISO/IEC/IEEE 29119-1: Concepts and Definitions**

* Provides the foundational terms, concepts, and principles of software testing.
* Establishes a common language for testing professionals.
* Key principles:
* Testing is a lifecycle activity.
* Testing provides stakeholders with information about software quality.
* Testing aims to detect and prevent defects.

1. **ISO/IEC/IEEE 29119-2: Test Processes**

* Defines a process model for software testing, including:
* Organizational Test Process:
* Establishing test policies, strategies, and reusable assets.
* Test Management Process:
* Planning, monitoring, and controlling test activities.
* Dynamic Test Process:
* Designing, implementing, and executing test cases.
* Test Evaluation Process:
* Analyzing and reporting test results to assess software quality.

1. **ISO/IEC/IEEE 29119-3: Test Documentation**

* Specifies templates for test documentation:
* Test Plan:
* Outlines scope, objectives, resources, and schedule for testing.
* Test Design Specification:
* Details test conditions, cases, and data requirements.
* Test Case Specification:
* Defines individual test cases, including input, expected results, and procedures.
* Test Procedure Specification:
* Describes steps to execute test cases.
* Test Report:
* Summarizes testing outcomes, defects, and overall quality assessment.

1. **ISO/IEC/IEEE 29119-4: Test Techniques**

* Describes test design techniques for:
* Specification-Based Testing:
* Based on system requirements, including equivalence partitioning and boundary value analysis.
* Structure-Based Testing:
* Focuses on code structure, such as branch and condition coverage.
* Experience-Based Testing:
* Relies on tester expertise, including exploratory and error-guessing methods.

1. **ISO/IEC/IEEE 29119-5: Keyword-Driven Testing**

* Provides guidelines for using keyword-driven testing approaches:
* Keywords represent high-level actions (e.g., "Login", "Search").
* Enables reusable and modular test automation**.**

**FUTURE SCOPE**

The **Sentiment-Based Movie Review System** implemented in C holds significant potential for future enhancements and broader applications. As technology evolves and the demand for more sophisticated text analysis grows, this project can be further developed to address the limitations of the current system and extend its capabilities. Below are some key areas for future scope:

**1. Integration of Machine Learning Algorithms**

* Incorporate machine learning models to improve accuracy in sentiment classification by learning from large datasets of labeled reviews.
* Use algorithms like Naive Bayes, Support Vector Machines (SVM), or Neural Networks to make the system adaptive and scalable.

**2. Natural Language Processing (NLP) Techniques**

* Employ advanced NLP libraries or frameworks to analyze the contextual meaning of words and phrases.
* Handle complex linguistic structures, sarcasm, and idiomatic expressions, which are challenging for simple keyword-based approaches.

**3. Multi-Language Support**

* Extend the system to support reviews in multiple languages, enabling a more inclusive analysis of global audiences.
* Use language-specific dictionaries or integrate external translation APIs for preprocessing.

**4. Real-Time Sentiment Analysis**

* Enhance the system to process and classify reviews in real-time, enabling its use in live applications such as social media sentiment tracking.

**5. User-Friendly Interface**

* Develop a graphical user interface (GUI) for better user interaction, allowing users to input reviews and view results easily.
* Integrate the system with web or mobile applications to make it more accessible.

**6. Dynamic Keyword and Sentiment Dictionary**

* Create a dynamic dictionary that updates sentiment scores based on user feedback or trends in the data.
* Include synonyms and related words to improve the accuracy of sentiment detection.

**7. Advanced Text Preprocessing**

* Implement techniques like stemming, lemmatization, and stop-word removal for more effective text parsing.
* Handle noisy data, such as typos or slang, using robust preprocessing algorithms.

**8. Big Data Integration**

* Scale the system to analyze large volumes of reviews using distributed computing frameworks like Hadoop or Spark.
* Apply data visualization techniques to present insights from big datasets effectively.

**CONCLUSION**

The Sentiment-Based Movie Review System is a foundational project that demonstrates the potential of computational methods in analyzing and understanding human emotions expressed through text. Implemented using the C programming language, this system leverages basic text processing and keyword-based sentiment analysis to classify movie reviews as positive, negative, or neutral. Its simplicity, combined with the power of C, makes it both efficient and educational, providing an excellent starting point for more advanced sentiment analysis systems.

While the current implementation focuses on basic sentiment detection, the project opens doors for numerous future enhancements. By integrating modern technologies like machine learning, natural language processing, and big data, the system can be made more accurate, adaptive, and capable of handling large-scale, real-world applications. Multi-language support, real-time analysis, and user-friendly interfaces can further extend its usability and accessibility.