Plagiarism Checker

Anshika Arjariya

November 29, 2025

Abstract

This project presents a simple plagiarism detection system developed in C. The program reads two text files, preprocesses their content by removing punctuation and converting the text to lowercase, and then compares the words between the files to compute a similarity percentage. The system uses basic string operations, tokenization, and pattern matching to identify similarity. The output is a plagiarism percentage along with a classification of similarity level. This project demonstrates modular programming concepts, file handling, string processing, and logical algorithm implementation using C.

CONTENTS

1. Problem Definition

1.1 Overview

Plagiarism in academic writing is a major concern, especially when comparing multiple student submissions.

1.2 Objectives

The objective of this project is to design a simple plagiarism checker that:

- Reads two text files provided by the user.
- Removes punctuation marks for clean comparison.
- Converts the text to lowercase to avoid case-based mismatches.
- Splits the text into words and compares them.
- Calculates similarity percentage based on matched words.

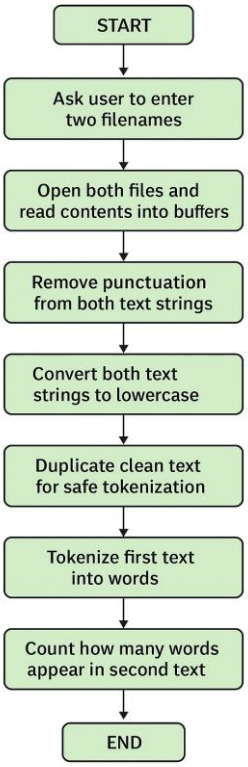
The output helps determine whether two files share low, moderate, or high similarity.

2. System Design

2.1 Algorithm

1. Start program.
2. Ask user to enter two filenames.
3. Open both files and read contents into buffers.
4. Remove punctuation from both strings.
5. Convert all characters to lowercase.
6. Duplicate clean text for safe tokenization.
7. Tokenize the first text into words.
8. Count how many words appear in the second text.
9. Calculate similarity percentage = (matched words / total words) × 100.
10. Display similarity percentage and category.
11. End program.

2.2 Flowchart



3. Implementation Details

3.1 Key features

1. File Handling.
2. Text Preprocessing.
3. String Processing.
4. Tokenization of Words.
5. Word Matching Logic.
6. Similarity Percentage Calculation.
7. Categorized Output.
8. Error Handling.
9. Large Buffer Handling

3.2 Code snippets

- **Remove punctuation**

```
void removePunctuation(char *str) {
   char temp[10000];
   int j = 0;
   for (int i = 0; str[i] != '\0'; i++) {
      if (isalpha(str[i]) || isspace(str[i])) {
         temp[j++] = str[i];
      }
   }
   temp[j] = '\0';
   strcpy(str, temp);
}
```

- **Convert to lowercase**

```
void toLowerCase(char *str) {
   for (int i = 0; str[i]; i++)
      str[i] = tolower(str[i]); }
```

- **Read file content**

```
 void readFileContent(char *filename, char *buffer) {
    FILE *fp = fopen(filename, "r");
    if (!fp) {
       printf("Error: Could not open file % ", filename);
       exit(1);
    }
    char line[500];
    while (fgets(line, sizeof(line), fp)) {
       strcat(buffer, line);
    }
    fclose(fp); }
```

- **Calculate similarity**

```
float calculateSimilarity(char *text1, char *text2) {
    int totalWords = 0, matchedWords = 0;
    char *word;

    word = strtok(text1, " ");
    while (word != NULL) {
        totalWords++;
        if (strstr(text2, word)) {
            matchedWords++;
        }
        word = strtok(NULL, " ");
    }
    if (totalWords == 0) return 0.0;
    return ((float)matchedWords / totalWords) * 100;
}
```

4. Testing & Results

4.1 Test Case 1: Identical Files

- File A: "Hello world"
- File B: "Hello world"
  Output: 100% similarity — Highly Similar

Test Case 2: Similar Sentences

- A: "The cat sat on the mat"
- B: "A cat was sitting on the mat"
  Output: ~60% similarity — Moderately Similar

Test Case 3: Completely Different Texts

Output: Low similarity (<20%)

5. Conclusion & Future Work

5.1 Conclusion

This project successfully demonstrates a basic plagiarism checking tool using C. The system performs text normalization and word-based matching to compute similarity. Although simple, it serves as a strong foundation for string processing and file handling.

5.2 Future Work

- Add sentence-level similarity detection.
- Use more advanced text comparison techniques.
- Detect Paraphrasing.
- Highlight copied words.
- Compare multiple files in one run.

6. References

- UPES C Major Project Guidelines
- C Standard Library Documentation (stdlib.h, string.h, ctype.h)
- Online resources