Plagiarism Detection System Using Advanced Data Structures

Hashmat Raza
Syed Ali Hassan Shirazi

Contents

Abstract

This project implements a sophisticated plagiarism detection system using Qt framework and C++. The system employs multiple string matching algorithms and advanced data structures to provide accurate plagiarism detection across various document formats. The implementation features a modern graphical user interface, secure user authentication, and robust document comparison capabilities.

1 Introduction

In the digital age, maintaining academic integrity and protecting intellectual property has become increasingly challenging. Our plagiarism detection system addresses this need by providing an efficient and user-friendly tool for comparing documents and identifying potential instances of content duplication. The system uses a combination of different algorithms to ensure high accuracy in plagiarism detection.

2 Problem Statement

Academic institutions and content creators face significant challenges in:

- Detecting copied content across multiple documents
- Processing various document formats efficiently
- Providing accurate similarity metrics
- Handling large documents without performance degradation
- Maintaining user data security and privacy

3 Proposed Solution and Methodology

Our solution implements a multi-layered approach to plagiarism detection:

1. **Document Processing**
    - Text extraction from multiple formats

   - String cleaning and normalization
   - Tokenization and stopword removal

2. **Similarity Detection Algorithms**
   - TF-IDF (Term Frequency-Inverse Document Frequency)
   - N-gram Analysis
   - Cosine Similarity
   - KMP (Knuth-Morris-Pratt) Pattern Matching
   - Boyer-Moore Algorithm

3. **Data Structures**
   - AVL Trees for efficient data storage and retrieval
   - Hash Maps for frequency analysis
   - Vectors for dynamic data management

4 Features

1. **User Authentication System**
   - Secure registration and login
   - Data persistence using CSV storage
   - Input validation and error handling

2. **Document Management**
   - Multiple document upload support
   - Target document selection
   - Various format support (TXT, PDF, DOC)

3. **Plagiarism Analysis**
   - Multiple algorithm combination
   - Percentage-based similarity scoring
   - Detailed results visualization
   - Color-coded verdict system

5 UI/UX Implementation Details

5.1 Splash Screen

- Animated logo display
- Fade-in and fade-out transitions
- Clean, minimalist design

5.2 Registration Interface

- Form-based user input
- Real-time validation
- Clear error messaging
- Professional styling with modern colors

5.3 Login System

- Simple two-field interface
- Secure credential verification
- Error handling with user feedback

5.4 Main Plagiarism Interface

- Three-button layout for main functions
- Results table with color coding
- Clear instruction labels
- Professional typography and spacing

6 Technical Implementation Details

6.1 String Matching Algorithms

- **KMP Algorithm**: Efficient pattern matching with $O(m + n)$ complexity
- **Boyer-Moore Algorithm**: Bad character rule implementation
- **N-gram Analysis**: Weighted scoring with variable n-gram sizes (3, 5, 7)

6.2 TF-IDF Implementation

- Term frequency calculation
- Inverse document frequency computation
- Cosine similarity measurement

7 Challenges Faced

- **Performance Optimization**: Large document processing, memory management
- **UI Response**: Maintaining responsiveness during processing, progress feedback
- **Algorithm Integration**: Combining multiple detection methods, weight balancing

8 Project Limitations

- Limited document format support
- Memory constraints for very large documents
- No real-time collaboration features
- Basic document text extraction

- Local storage system limitations

9 Use of Built-in APIs and Libraries

- **Qt Framework**: QApplication, QMainWindow, QWidget, QFile handling
- **STL**: Vectors, Maps, Sets, String operations
- **Qt GUI Components**: QLabel, QPushButton, QLineEdit, QTableWidget

10 DSA Concepts Used

10.1 Frontend (GUI) Data Structures

- Queue for managing document processing order
- Priority Queue for urgent comparisons
- Stack for undo/redo operations
- Trie for auto-complete suggestions

10.2 Backend Data Structures

- AVL Trees for balanced storage
- Hash Maps for $O(1)$ lookups
- Sets for unique word collections
- Bloom Filters for duplicate checking

10.3 Algorithms

- String matching (KMP, Boyer-Moore, Rabin-Karp)
- Sorting and searching algorithms
- Tree balancing and pattern matching

11 Conclusion

The project successfully implements a comprehensive plagiarism detection system using advanced data structures and algorithms. The combination of AVL trees, efficient string matching algorithms, and modern UI design creates a powerful tool for document similarity analysis. The system demonstrates practical application of theoretical DSA concepts in solving real-world problems.

12 References

1. Qt Documentation: `https://doc.qt.io/`
2. Knuth, Donald E. *The Art of Computer Programming, Volume 3: Sorting and Searching*
3. Cormen, Leiserson, Rivest, and Stein. *Introduction to Algorithms*
4. Boyer-Moore Algorithm Documentation
5. Research paper: *TF-IDF Based Document Similarity Metrics*
6. Qt Style Sheets Reference Documentation
7. *Modern C++ Programming with Test-Driven Development*
8. *Data Structures and Algorithm Analysis in C++*