CODE SIMILARITY ANALYSIS REPORT

Files Compared:

- code_1.py
- code_5.py

Overall Similarity: 53.5%

Report Generated: September 14, 2025 at 11:27:17

Analysis Category: Medium

Code Similarity Analysis Report

Professional Code Comparison & Analysis

Executive Summary

Project: Code Similarity Analysis
Date/Time: September 14, 2025 at 11:27:17
Primary Method: DIFFLIB
Threshold: 70%

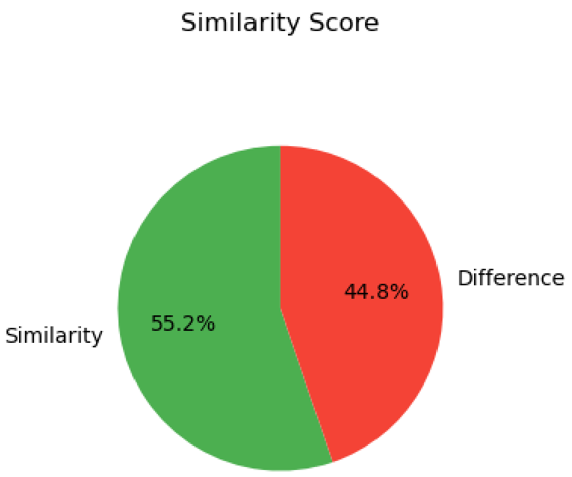
Algorithm Results Summary

Algorithm	Score (%)	Category	Status
Difflib (Sequence Matching)	55.2%	Medium	* Normal
TF-IDF Cosine Similarity	49.8%	Medium	* Normal
AST (Abstract Syntax Tree)	79.1%	High	! High
Jaccard Similarity	30.0%	Low	* Low
AVERAGE SIMILARITY	**53.5%**	**Medium**	*** Normal**

Similarity Visualizations

The following charts provide visual representation of the similarity analysis:

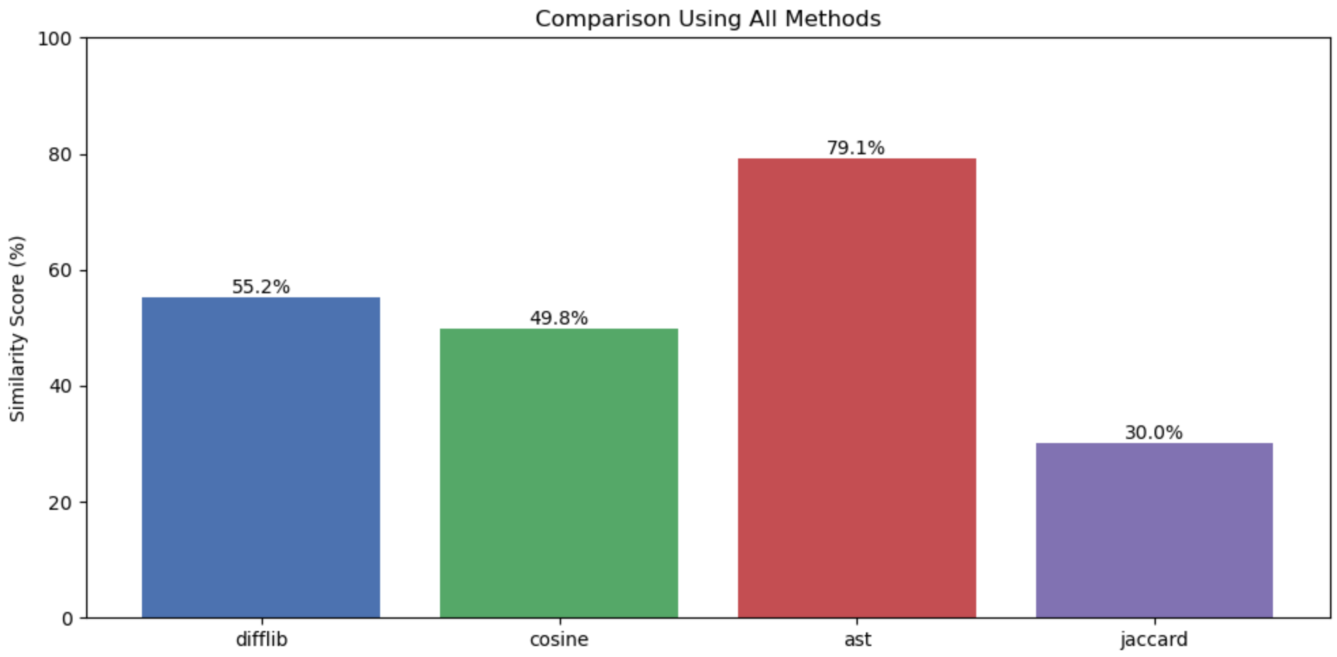
Selected Algorithm (DIFFLIB) Result:



Comprehensive Multi-Algorithm Comparison:

Code Similarity Analysis Report

Professional Code Comparison & Analysis



Detailed Algorithm Analysis

Difflib (Sequence Matching): 55.2%

Compares code as sequences of characters, identifying longest common subsequences. Effective for detecting exact matches and minor modifications.

Result: Medium similarity detected. Some similarities found.

TF-IDF Cosine Similarity: 49.8%

Converts code to TF-IDF vectors and calculates cosine similarity. Good for detecting structural similarities regardless of variable names.

Result: Medium similarity detected. Some similarities found.

AST (Abstract Syntax Tree): 79.1%

Analyzes the syntactic structure of code by comparing Abstract Syntax Trees. Most effective for detecting structural plagiarism.

Result: High similarity detected. Moderate similarity detected.

Jaccard Similarity: 30.0%

Measures similarity as the ratio of common tokens to total unique tokens. Useful for detecting copied code with minor additions.

Result: Low similarity detected. Minimal similarity detected.

Professional Analysis & Interpretation

Overall Assessment: LOW SIMILARITY - Some common patterns detected but mostly different. High similarity detected by: AST (Abstract Syntax Tree).

Conclusion

Based on the comprehensive analysis using multiple algorithms, the codes show MODERATE similarity (>=40%). Some common patterns exist but with significant differences.

Technical Implementation Details

Preprocessing Pipeline:

1. Comment Removal: All single-line and multi-line comments removed

2. Identifier Normalization: Variables, functions, and classes renamed systematically

3. Whitespace Standardization: Consistent formatting applied

4. Language-Specific Processing: Custom handling for different programming languages

Code Sample Analysis

File 1: code_1.py

```
def add(a, b):
    return a + b

print(add(2, 3))
```

File 2: code_5.py

```
def divide(a, b):
    if b == 0:
        return "Division by zero not allowed"
    return a / b

print(divide(10, 2))
```

Key Differences Identified

```
--- file1
+++ file2
@@ -1,3 +1,5 @@
 def f0(p0, p1):
-    return a + b
-print(add(2, 3))
+    if b == 0:
+        return 'Division by zero not allowed'
+    return a / b
+print(divide(10, 2))
```