Paper SM07

Code Compare -

A Machine Learning Based Framework for Code Plagiarism Analysis and Detection

Saroj Kumar Jena, IQVIA Netherlands B.V., Amsterdam, NL

ABSTRACT

This paper introduces an innovative framework for detecting programming code plagiarism by integrating cutting-edge tools and techniques. The system leverages R Shiny as an intuitive user interface and Python-Flask as the backend API to ensure seamless interaction and robust functionality. At the core of the model, the TF-IDF (Term Frequency-Inverse Document Frequency) method is utilized to quantify the significance of terms within the code, while cosine similarity measures the angle-based distance between vectorized code samples for efficient comparison. Together, these algorithms ensure precise and computationally efficient identification of code similarities, empowering developers to identify potential instances of plagiarism or similarity. With its ability to compare coding practices across different environments, such as SAS-to-SAS or R-to-R scripts, the framework showcases its adaptability for real-world programming scenarios. With its comprehensive approach to code comparison, this work provides a valuable resource for software development teams striving to uphold ethical coding practices.

INTRODUCTION

CODE PLAGIARISM IN CLINCAL TRIAL PROGRAMMING

Code plagiarism is a growing concern in clinical trial programming, where reproducibility, traceability, and regulatory compliance are paramount. In environments where SAS and R are used to generate and validate statistical outputs, ensuring originality between main and QC programs is critical to maintaining data integrity and audit readiness.

ETHICAL CODING PRACITCES

Maintaining integrity in code development is essential, not only for intellectual property protection but also for fostering trust within programming teams and regulatory bodies. Ethical coding ensures accountability and promotes a culture of transparency, especially when multiple programmers contribute to validation workflows.

EXISTING SOILUTIONS AND LIMITATIONS

Traditional plagiarism detection tools often rely on superficial string matching or manual review, which can be time-consuming and error prone. Many lack the flexibility to handle domain-specific programming languages like SAS or R, and struggle with structural variations that occur between main and QC programs in clinical trial settings.

UNIQUE CONTRIBUTION OF THIS FRAMEWORK

The proposed framework introduces a machine learning-based approach that goes beyond syntax to analyze semantic similarities. By integrating TF-IDF and cosine similarity within a scalable architecture, it offers a robust, language-agnostic solution for detecting code duplication platforms like SAS and R. This domain-aware design ensures accurate detection even when formatting or variable names differ, making it highly adaptable for clinical trial programming workflows.

METHODLOGY

ARCHITECTURE OVERVIEW

The framework is designed with a modular architecture to ensure scalability and ease of integration across clinical trial programming workflows:

- **Frontend**: Built using **R Shiny**, the interface allows users to upload code files and view similarity results in an intuitive, interactive dashboard.
- **Backend**: Powered by **Python Flask**, the API layer handles data preprocessing, vectorization, and similarity computations efficiently.

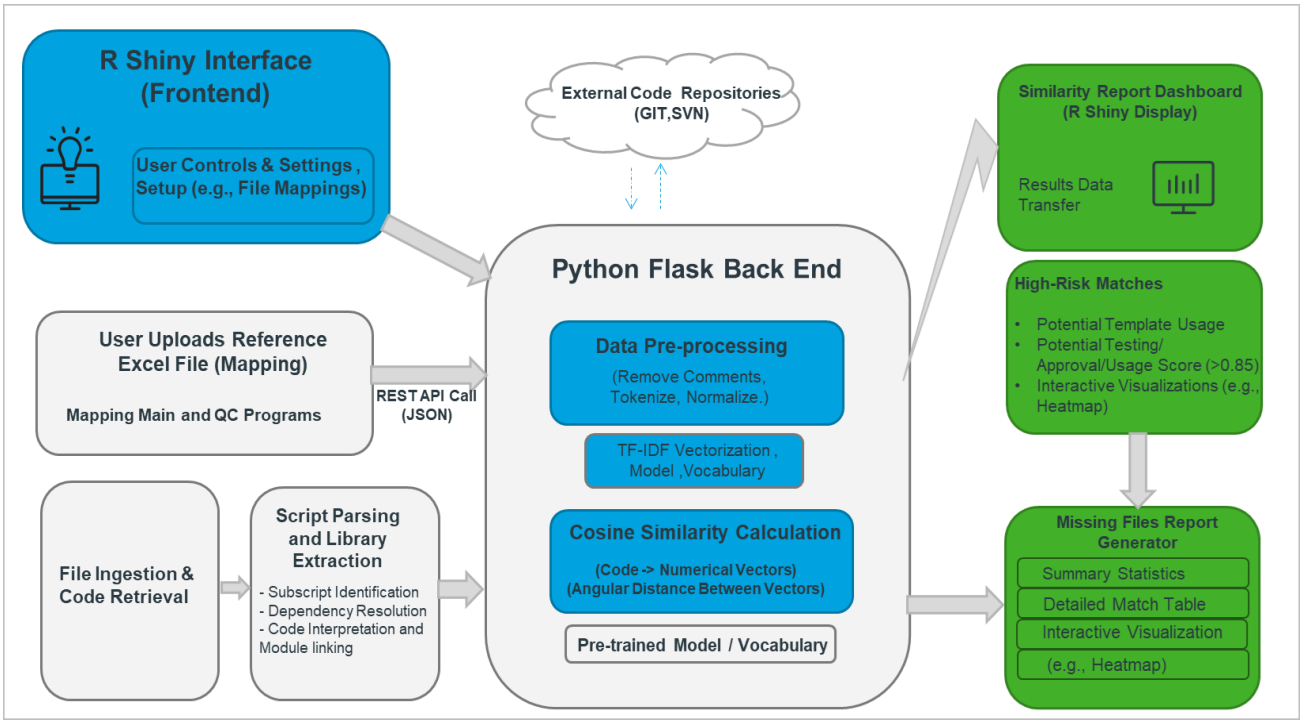

Figure 1 - System Architecture Overview

DATA PREPROCESSING

Uploaded code undergoes a cleaning process to remove comments, whitespace, and non-essential tokens. The system then applies **tokenization** and **normalization** to prepare the code for vector-based analysis, ensuring consistency across different coding styles.

TF-IDF FOR CODE ANALYSIS

The **TF-IDF (Term Frequency–Inverse Document Frequency)** algorithm transforms code into numerical vectors by evaluating the relative importance of each token within the corpus. This highlights distinctive constructs and reduces noise from commonly used syntax.

COSINE SIMILARITY

Cosine similarity measures the angular distance between two TF-IDF vectors, providing a robust metric for detecting semantic and structural similarities. This approach is resilient to superficial changes such as variable renaming or formatting differences, making it ideal for comparing SAS main and QC programs or R scripts.

SUPPORTED ENVIORNMENTS

The framework currently supports:

- SAS-to-SAS comparisons
- R-to-R comparisons

Future enhancements may include **cross-language comparisons**, such as Python-to-R, to broaden applicability across diverse programming ecosystems.

IMPLEMENTATION

TECHNICAL STACK

- **Frontend**: R Shiny
- **Backend**: Python Flask
- **Libraries**: scikit-learn (TF-IDF, cosine similarity), pandas, NumPy, Flask-RESTful

SAMPLE WORKFLOW

- The user uploads a **reference Excel file** via the R Shiny interface. This file contains mappings between main and QC programs across multiple folders and subfolders for the entire study.
- The system automatically locates and retrieves the corresponding SAS or R code files based on the Excel mappings.
- During preprocessing, the framework also identifies and includes any **referenced programs** such as %include statements in SAS or source() calls in R, ensuring a comprehensive comparison.
- All code files are sent to the Flask API for cleaning, tokenization, and normalization.
- TF-IDF vectors are generated and compared using cosine similarity to detect semantic and structural similarities.
- Results are returned to the frontend, including:
    - **Similarity scores and matched segments** between main and QC programs
    - A **summary table** showing:
        - Total number of main programs
        - Total number of QC programs
        - Number of matched pairs
        - Number of unmatched or missing files
    - A **missing file report** listing any main or QC programs that could not be located or matched.

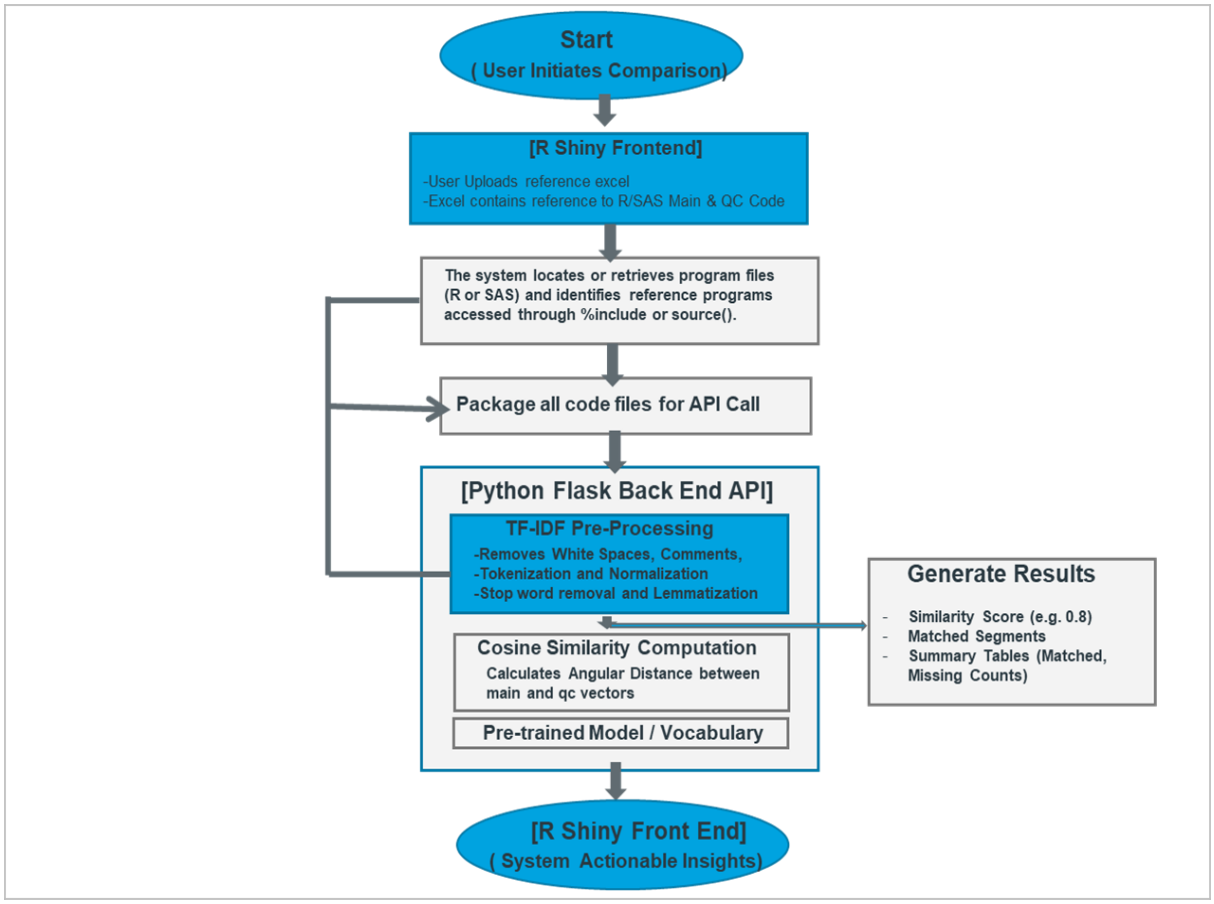

Figure 2 - "Code Compare" Framework Workflow

RESULTS

PERFORANCE METRICS

Initial testing of the framework on a curated dataset of SAS and R scripts revealed high precision in identifying similar code segments. Cosine similarity scores above 0.85 were consistently associated with known cases of plagiarism, while scores below 0.5 indicated distinct implementations.

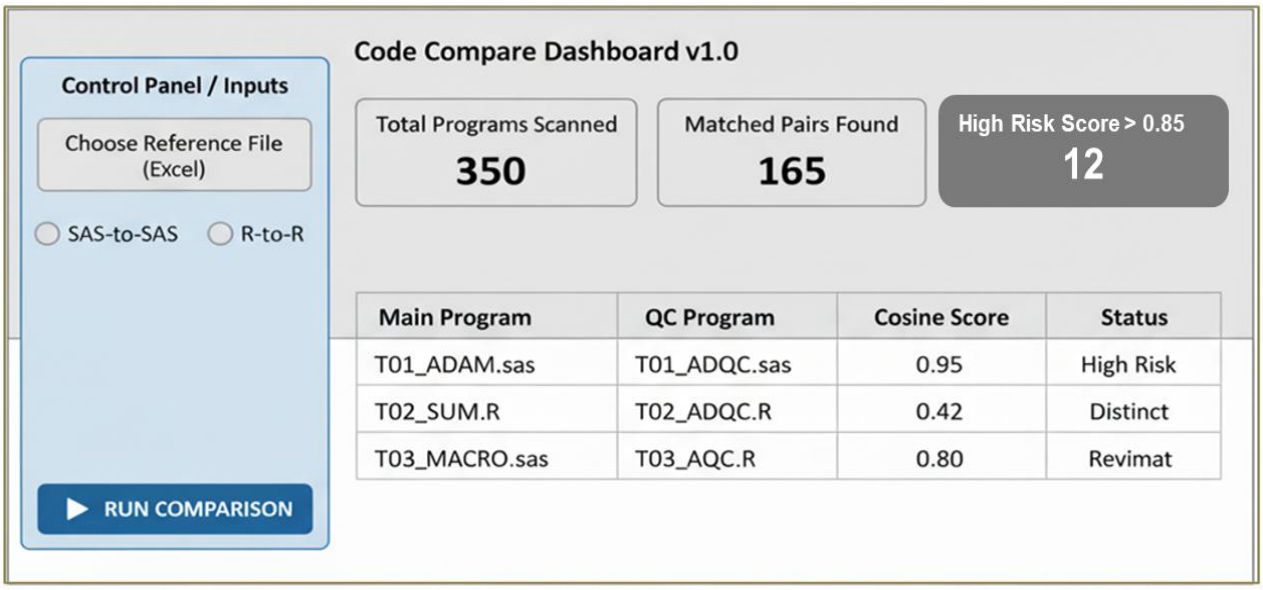

Figure 3 - R Shiny User Interface and Comparison Dashboard

CASE STUDIES

- **Internal Duplication Scenario:** Two SAS programs intended for different endpoints in the same study (Program A: Main; Program B: QC) were flagged with a similarity score of 0.92. Manual review confirmed structural and logical duplication: the programmer had copied the Main program and only made minor variable renaming changes for the QC version.

- **Template Usage Scenario:** SAS macros from two different teams working on separate studies showed a similarity score of 0.78. Further analysis revealed shared, approved template usage, not plagiarism—highlighting the tool's ability to prompt deeper review (i.e., the "**Revimat**" status, signifying **Review and Mandatory Verification**) and to distinguish approved shared code from unauthorized duplication.

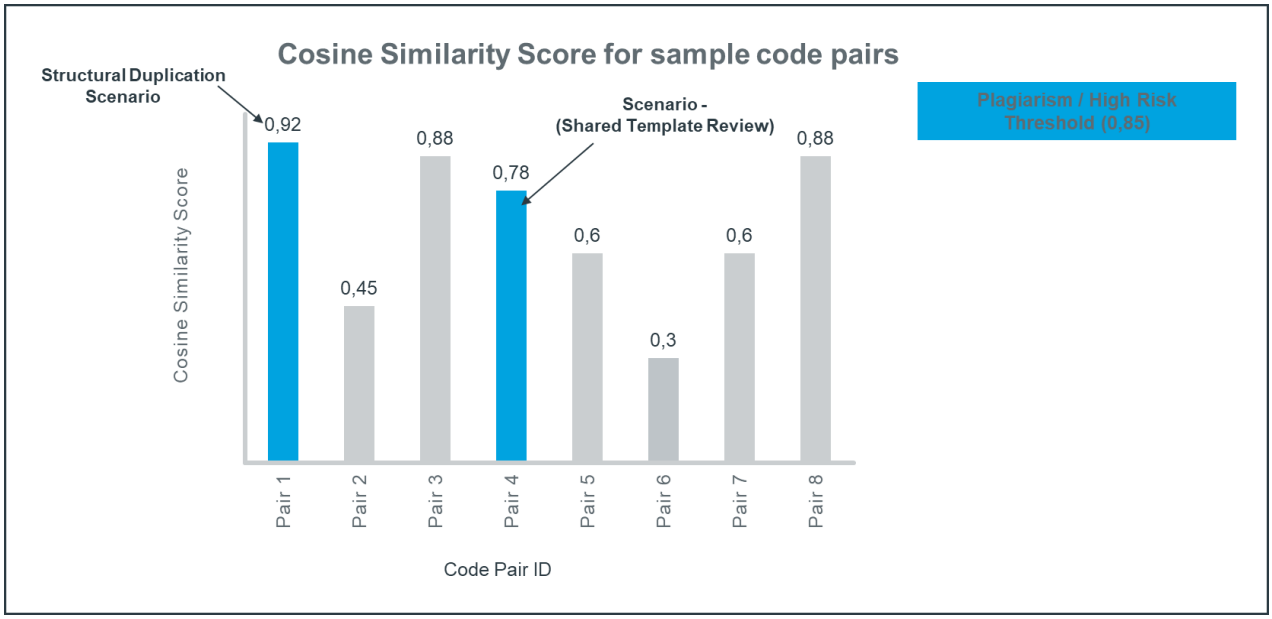

Figure 4 - Cosine Similarity Score Visualization with Plagiarism Threshold

TOOL COMPARISON

Beyond high precision (Figure 4), our framework significantly outperforms traditional string-matching solutions across key operational metrics:

- **Robustness to Code Variation:** Demonstrated better tolerance for renamed variables and reordered logical blocks, successfully identifying semantic similarity where string-matching failed.
- **Scalability & Speed:** Achieved significantly faster processing time when deployed on large organizational codebases.
- **Usability:** Provided more intuitive, actionable, and visual results via the R Shiny comparison dashboard.

DISCUSSION

STREGTH OF THE APPROACH

✓ **Language-Agnostic Design** supports multiple environments (SAS-to-SAS, R-to-R).

✓ **Semantic-Level Comparison** leveraging TF-IDF and Cosine Similarity, which is robust to superficial changes.

✓ **Modular Architecture** allowing easy integration with existing programming workflows.

✓ **User-Friendly Interface** (R Shiny) encourages adoption across all technical skill levels.

✓ **Directly Supports Validation Programming** by providing objective evidence of code originality between Main and QC programs.

✓ **Enhances Audit Readiness** by documenting the absence of unauthorized code reuse and providing clear audit trails.

✓ **Facilitates Sponsor-CRO Collaboration** by establishing objective metrics for quality assurance and consistent standards.

LIMITATIONS AND CHALLENGES

➢ Current model does not support cross-language comparisons (e.g., Python-to-R)

➢ Highly templated or boilerplate code may yield false positives.

➢ Lack of contextual understanding (e.g., intent or comments) may limit interpretability.

FUTURE ENHANCEMENTS

- ❖ Expand support to Python, SQL, and Java.
- ❖ Incorporate deep learning models (e.g., code embeddings via transformers).
- ❖ Add explainability features to highlight matched logic blocks.
- ❖ Enable batch processing and integration with version control systems.
- ❖ Potential for integration with CDISC standards or automated compliance checks.
- ❖ Leverage alternative algorithms such as Rabin-Karp combined with cosine similarity, or AI-based LLM models for enhanced semantic detection. These implementations have been evaluated but intentionally excluded from the current release to maintain tool simplicity and performance.

CONCLUSION

This framework presents a scalable, intelligent solution for detecting code plagiarism within clinical trial programming environments. By combining TF-IDF and cosine similarity within a user-friendly R Shiny interface, it empowers programming teams to uphold ethical coding standards, improve traceability, and streamline validation workflows. Its ability to compare SAS main and QC programs, as well as R-to-R scripts, makes it especially valuable for clinical research settings where reproducibility and audit readiness are critical.

With its modular design and extensibility, the framework offers a practical and adaptable tool for both sponsor and CRO teams working across diverse studies.

Its adaptability and precision make it an asset for clinical trial programming teams striving to uphold ethical standards and streamline validation processes.

REFERENCES

- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research.
- Flask Documentation: https://flask.palletsprojects.com
- R Shiny Documentation: https://shiny.rstudio.com
- SAS Macro Language Reference: https://documentation.sas.com

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Saroj Kumar Jena
Company: IQVIA Netherlands B.V.
Address: Herikerbergweg 314, 1101 CT Amsterdam, Netherlands
Email: saroj.jena@iqvia.com
Website: