COLUMBIA UNIVERSITY

Plagiarism Detector Project

Group 5: Tongfei Guo, Yijing Fei, Xinran Han, Yuqing Xu, Xingyue Yan, Celeste Chen

Python 5210 Fall 2023

We aim to:

- Identify similarities and overlaps in code snippets
- Provide a tool for educators and professionals to maintain code integrity
- Enhance fairness and ethical standards in the programming domain

How our Model Works?

Input

Student work

Database

Detector

Similarity Score

The Arsenal of Algorithms

- **Jaccard Index**: Measures similarity between sets by comparing the intersection and union of code tokens.
- **MinHash**: Efficiently estimates the similarity between datasets by leveraging randomized hash functions and minimum hash values.
- **difflib**: Python's built-in library for comparing sequences, useful for identifying differences between code fragments.
- **CountVector**: Converts code snippets into numerical vectors, enabling quantitative analysis of code structure.
- **TF-IDF** (Term Frequency-Inverse Document Frequency): Evaluates the importance of code terms in relation to the entire codebase.
- **Fuzzywuzzy** (fuzz.ratio): Uses Levenshtein distance to measure the similarity between two strings.

Jaccard Index Algorithms

It also known as the Jaccard similarity coefficient, is a statistic used in understanding the similarities and difference between sample sets, by comparing the intersection and the union of the sets.

The formula for the jaccard index is $J(A,B)=|A \cap B|/|A \cup B|$
The jaccard Index ranges from 0 to 1:

- A Jaccard Index of 1 means that the two sets are identical.
- A Jaccard Index of 0 means that the two sets are completely different.

Pros :

- Effective for Sparse Data
- Robust to Overlap Variance
- Size Independent

Cons :

- Limited Contextual understanding
- Not Ideal for Continuous Data
- Insensitive to Frequency

MinHash algorithm

Implement in conjunction with the datasketch package, efficiently addresses the problem by creating concise dataset signatures, enabling swift estimation of similarities and effectively identifying patterns within large datasets.

Pros -

- Handling large-scale datasets,
- Quick similarity estimation,
- Identification of patterns and similarities.

Cons -

- Influenced by the number of hash functions and sampling,
- Limited interpretability of the generated hash signatures

```
1. The MinHash similarity score between notebooks 0101.ipynb and 0102.ipynb is 1.0, indicating potential plagiarism.
2. The MinHash similarity score between notebooks 0101.ipynb and 0103.ipynb is 0.9609375, indicating potential plagiarism.
3. The MinHash similarity score between notebooks 0102.ipynb and 0103.ipynb is 0.9609375, indicating potential plagiarism.

[('0101.ipynb', '0102.ipynb', 1.0),
 ('0101.ipynb', '0103.ipynb', 0.9609375),
 ('0102.ipynb', '0103.ipynb', 0.9609375)]
```

Difflib

it provides a weighted measure that considers both term frequency and document frequency, making it suitable for capturing the uniqueness of certain code snippets.

Pros -

- Weighted representation,
- Handles common stopwords,
- Considers word importance

Cons -

- Sparsity,
- Limited semantic understanding

CountVector

It prioritizes simplicity and efficiency, particularly for concise code snippets, by offering a straightforward count-based representation that is both easy to implement and computationally efficient; however, it may not account for semantic relationships between terms or their order within the document.

Pros -

- Simple and intuitive,
- easy to implement,
- efficient for short texts

Cons -

- Ignores word order,
- equal weight for all terms.

Future improvement

combining CountVector with contextual embeddings (Word2Vec)

TF-IDF (Term Frequency-Inverse Document Frequency)

it provides a weighted measure that considers both term frequency and document frequency, making it suitable for capturing the uniqueness of certain code snippets.

Pros -

- Weighted representation,
- Handles common stopwords

Cons -

- Sparsity,
- Limited semantic understanding

Token-based Similarity Algorithm

By focusing on tokens (the smallest units of meaning in the code), this method provides a detailed and nuanced comparison, making it suitable for applications like plagiarism detection or similarity assessment in coding assignments.

Pros:

- Robust Text Comparison
- Efficient
- Wide applicability

Cons:

- Low sensitivity to synonyms
- Inability to handle complex linguistic features

FuzzyWuzzy

The code focuses on comparing the code cells within Jupyter notebooks. Jupyter notebooks contain various types of cells, such as markdown cells and code cells. This detector only considers code cells for comparison. It ignores other types like markdown or raw cells.

Pros :

- Focused on Code Similarity
- Use of Fuzzy Matching
- Automation
- Customizable Threshold

Cons :

- No Contextual Analysis
- Ignore Non-Code Similarities

Demo

THANK YOU

Appendix:

- Jaccard Index:
  - Measures the similarity between two sets by comparing the intersection and union of their elements.
  - Well-suited for comparing the similarity of code snippets.
- Needleman-Wunsch:
  - A global alignment algorithm commonly used in bioinformatics but adapted for code similarity here.
  - Allows for the identification of similarities in code structures, even with variations.
- difflib:
  - A Python library that provides tools for comparing sequences.
  - Utilizes Ratcliff/Obershelp algorithm to find the longest contiguous matching subsequence.
- CountVector:
  - Represents each document (code snippet) as a vector of word frequencies.
  - Enables the comparison of code based on the occurrences of distinct terms.
- TF-IDF (Term Frequency-Inverse Document Frequency):
  - Assigns weights to terms based on their frequency in a document relative to their frequency in the entire corpus.
  - Effective in identifying important terms in code and measuring their significance.