Code Understanding Report

Generated: 2025-05-05 23:56:12

This report presents automated insights based on large language models and code analysis tools.

File: app.py

Summary

- This code defines a function vectorize that takes in a list of text, converts each text in this list into a TF-IDF representation, and returns these vectors as a numpy array. The TfidfVectorizer is a popular method used to convert text to numerical data.
- In the given code, a function "similarity" is defined to calculate the cosine similarity between two documents (doc1, doc2). Cosine similarity is a measure of similarity between two non-zero-length vectors. In other words, it measures the cosine of the angle between the two vectors.

The cosine_similarity function is used from the sklearn.metrics module, which calculates the cosine similarity between two sets of data. Here, the function is used to calculate the similarity between two documents.

In the function signature, `doc1` and `doc2` are - This python function is looking at text similarity to detect plagiarism among a set of student's submissions, using cosine similarity to measure similarity. The function creates a set of all pairs of students, computes the similarity of two texts using the cosine similarity function, and adds the resultant pair and similarity score to a dictionary. If a pair of students' text pairs are semantically similar to each other, the function will add this pair with its similarity score to a plagiarism result set.

The code snippet provided appears to be part of a larger program for detecting plag

Docstring

- ### Code: def vectorize(Text): return TfidfVectorizer().fit_transform(Text).toarray()

Docstring:

This function takes a list of text documents and returns a matrix of TF-IDF features.

Parameters: - Text: A list of strings, where each string is a document.

Returns: - A sparse - ### Code: def similarity(doc1, doc2): return cosine_similarity([doc1, doc2])

Docstring:

The function `similarity` takes two documents as input and returns the cosine similarity between them.

The `cosine_similarity` function is a part of the `sklearn.metrics.pairwise` module, - ### Code: def check*plagiarism(): global s*vectors for student*a, text*vector*a in s*vectors: new*vectors = s*vectors.copy() current*index = new*vectors.index((student*a, text*vector*a)) del* *new*vectors[current*index] for student*b, text*vector*b in new*vectors: sim*score = similarity(text*vector*a, text*vector*b)[0][1] student*pair = sorted((student*a, student*b)) score = (student*pair[0], student*pair[1], sim*score) plagiarism*results.add(score) return plagiarism*results

Docstring:

This function checks for plagiarism in a list of text vectors. It does this by comparing each text vector with every other text vector in the list. The similarity between the two vectors is calculated using the `similarity` function,

Code Quality

Tool: `pylint`
Issues: 0`

```
text ************* Module tmph1gabp1n C:
\Users\nmoha\AppData\Local\Temp\tmph1gabp1n.py:21:8: W0602: Using
global for 's_vectors' but no assignment is done (global-
variable-not-assigned)
```

Conclusion

This codebase is about document vectors and similarity calculation between documents. It uses the TF-IDF (Term Frequency-Inverse Document Frequency) method to transform text into numerical data, which can then be used for similarity calculation between documents. The vectorized and similarity-calculated representations are stored in a numpy array, which can be very useful for further analyses or predictions.