Plagiarism Checker

SWE4001: Artificial Intelligence - Review 2

Points of Discussion

These are the broad topics we are going to cover in this session.

01 Introduction

02 Methodolgy

03 Pseudocodes

04 Implementation

Introduction on Plagiarism

Destroyed Student Reputation

Destroyed Professional Reputation

Destroyed Academic Reputation

Legal Repercussions

Massive Financial Penalties

Monetary Repercussions

Cosine Similarity Methodology

Measures similarity between two vectors

01

What is cosine similarity?

In NLP, Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it calculates the cosine of the angle between two vectors projected in a multi-dimensional space.

02

How does cosine similarity work?

- Cosine Similarity is a value that is bound by a constrained range of 0 and 1
- Suppose the angle between the two vectors was 90 degrees. In that case, the cosine similarity will have a value of 0; this means that the two vectors are orthogonal or perpendicular to each other.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Pseudocode...

Pseudocode for the Comparison Algorithm

```
Document[] docs = readDocsFromDisk();

for each Document, i, in docs {

for each document,j, following i in docs{

compareSentences(docs[i], docs[j]); }}

compareSentences(Document doc1, Document doc2){

for each sentence,i, in doc1{

for each sentence,j, in doc2 {

int common = number of shared words;

int score = similarityScore(i,j,common);

if(score > SIM THRESHOLD ||

common > COM THRESHOLD)

storeLink(sent1, sent2, score); }}}}
```

Implementation

- Retrieve all files
- Get the content for each file
- Check cosine similarity value among each and every file contents
- Conversion of dictionary into JSON format using API
- Display the results

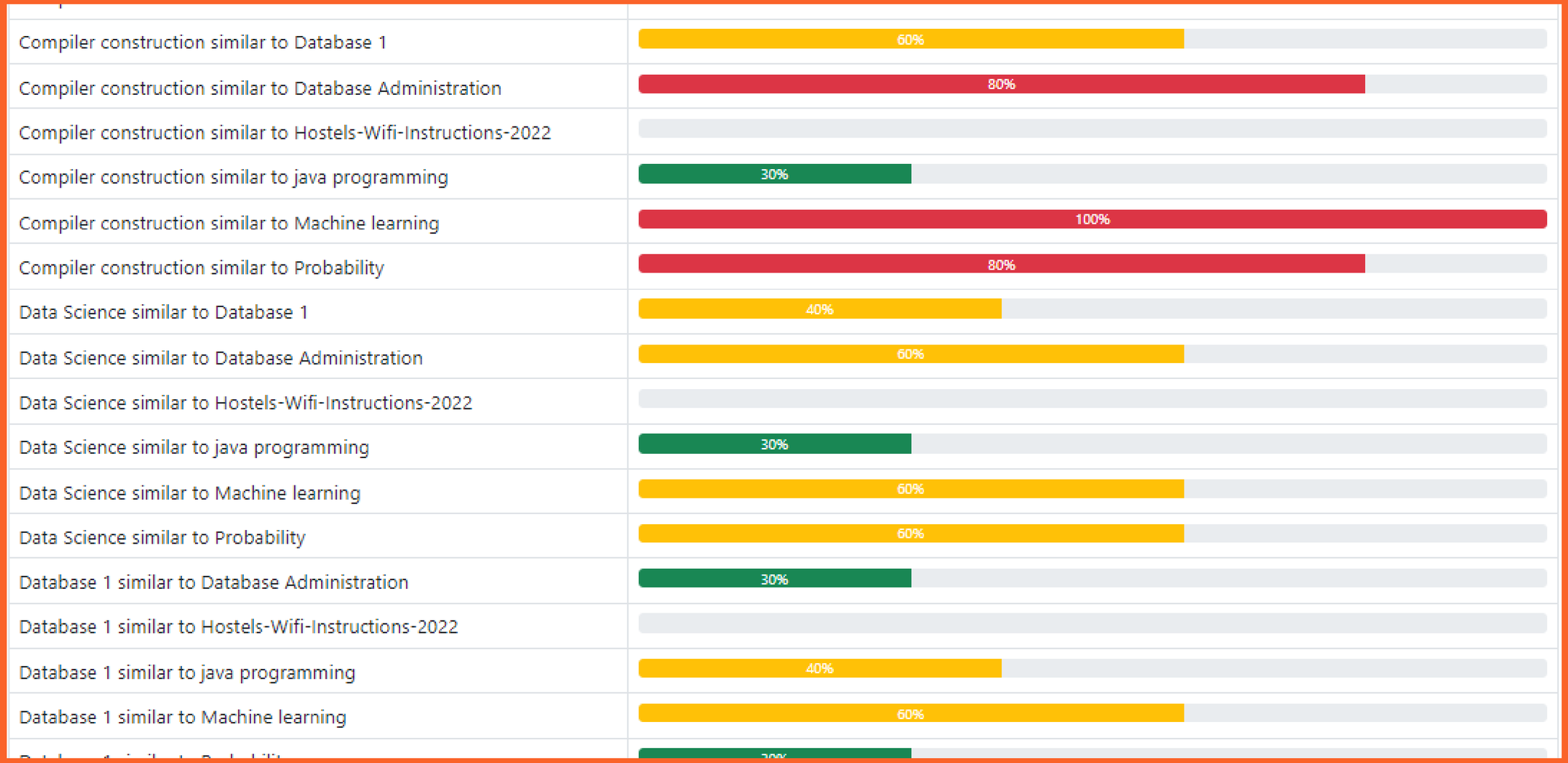
```
student_files = [doc for doc in os.listdir() if doc.endswith('.txt') or doc.endswith('.pdf')]
student_notes = [open(_file, encoding='utf-8', errors='ignore').read() for _file in student_files]

def vectorize(Text): return TfidfVectorizer().fit_transform(Text).toarray()
def similarity(doc1, doc2): return cosine_similarity([doc1, doc2])

vectors = vectorize(student_notes)
s_vectors = list(zip(student_files, vectors))

 def check_plagiarism():
    plagiarism_results = {}
    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]
            if(sim_score > 0):
                sim_score = round(sim_score, 1)
                student_pair = sorted(
                    (os.path.splitext(student_a)[0], os.path.splitext(student_b)[0]))
                res = (student_pair[0]+' similar to ' + student_pair[1])
                plagiarism_results[res] = sim_score
    api = json.dumps(plagiarism_results)
    return api
```

Test Result


Compiler construction similar to Database 1
60%
Compiler construction similar to Database Administration
80%
Compiler construction similar to Hostels-Wifi-Instructions-2022
Compiler construction similar to java programming
30%
Compiler construction similar to Machine learning
100%
Compiler construction similar to Probability
80%
Data Science similar to Database 1
40%
Data Science similar to Database Administration
60%
Data Science similar to Hostels-Wifi-Instructions-2022
Data Science similar to java programming
30%
Data Science similar to Machine learning
60%
Data Science similar to Probability
60%
Database 1 similar to Database Administration
30%
Database 1 similar to Hostels-Wifi-Instructions-2022
Database 1 similar to java programming
40%
Database 1 similar to Machine learning
60%

Thank You !

20MIS1072 Ganasala Sri Sai Prasanna

20MIS1043 Abhishek B

20MIS1048 Alan Prince Richwin