Experiment No.6

Title: Applying similarity measures on the textual datasets

Batch:A2 **Roll No.:16010421063** **Experiment No.: 6**

Aim: Applying similarity measures on the textual datasets using cosine distance and BERT.

Resources needed: Python

Theory:

BERT: High-performance semantic similarity with BERT

Bidirectional Encoder Representations from Transformers is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others. BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models.

Sentence similarity is one of the clearest examples of how powerful highly-dimensional magic can be. The logic is like this:

- Take a sentence, convert it into a vector.
- Take many other sentences, and convert them into vectors.
- Find sentences that have the smallest distance (Euclidean) or smallest angle (cosine similarity) between them.
- We now have a measure of semantic similarity between sentences.

BERT, as mentioned — is the MVP of NLP. And a big part of this is down to BERTs ability to embed the meaning of words into densely packed vectors. We call them dense vectors because every value within the vector has a value and has a reason for being that value — this is in contrast to sparse vectors, such as one-hot encoded vectors where the majority of values are 0. BERT is great at creating these dense vectors, and each encoder layer (there are several) outputs a set of dense vectors.

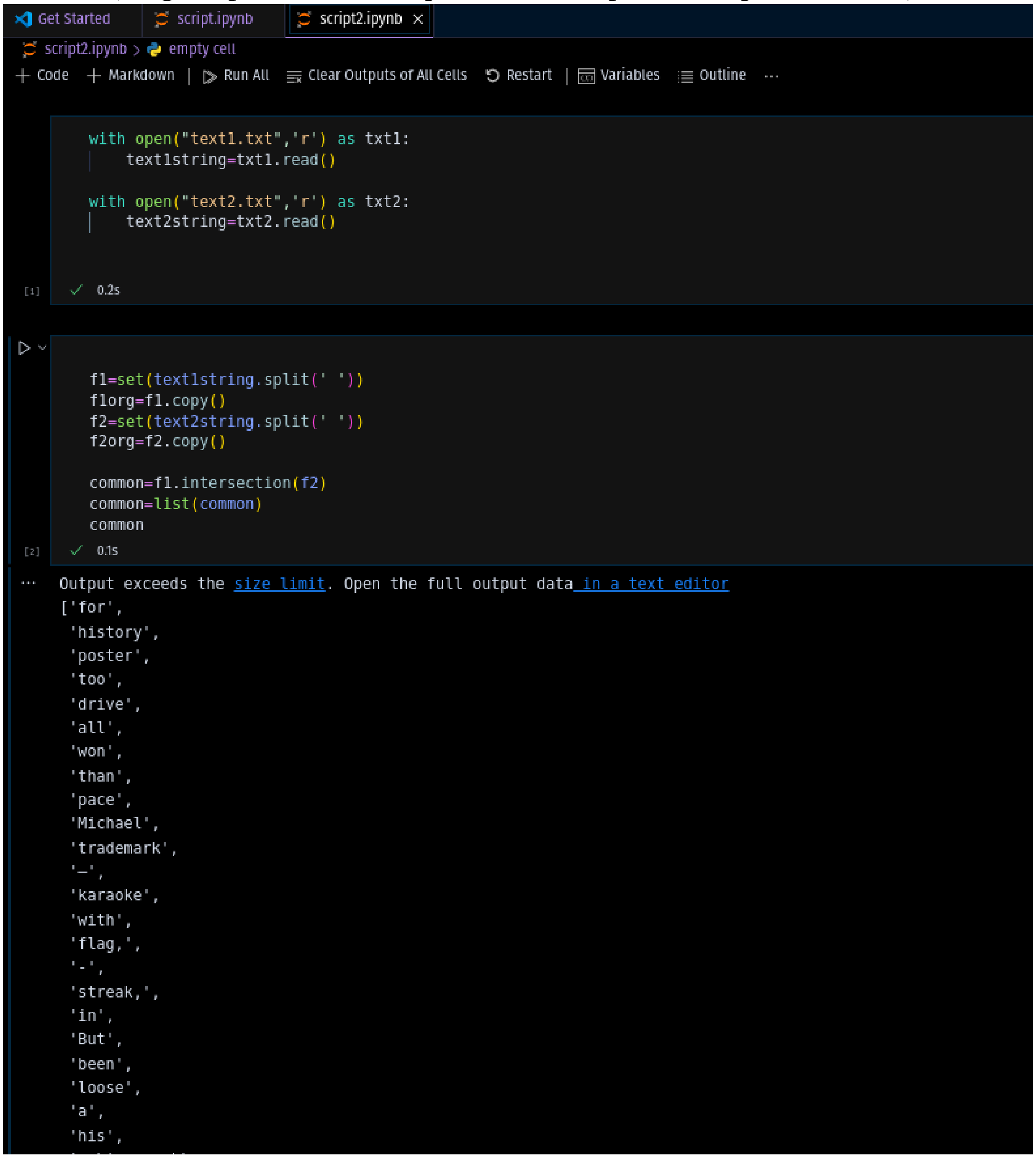
Code: Please use the following link to find cosine distance between different statements using BERT in python:

https://www.youtube.com/watch?v=Ey81KfQ3PQU&t=102s

Procedure / Approach /Algorithm / Activity Diagram:

1. Convert some given set of statements into dense vectors and calculate the cosine distance between them using BERT model in python.

Results: (Program printout with output / Document printout as per the format)


Get Started
script.ipynb
script2.ipynb
script2.ipynb > empty cell
+ Code + Markdown | Run All Clear Outputs of All Cells Restart | Variables Outline ...

```
with open("text1.txt",'r') as txt1:
    text1string=txt1.read()

with open("text2.txt",'r') as txt2:
    text2string=txt2.read()
```

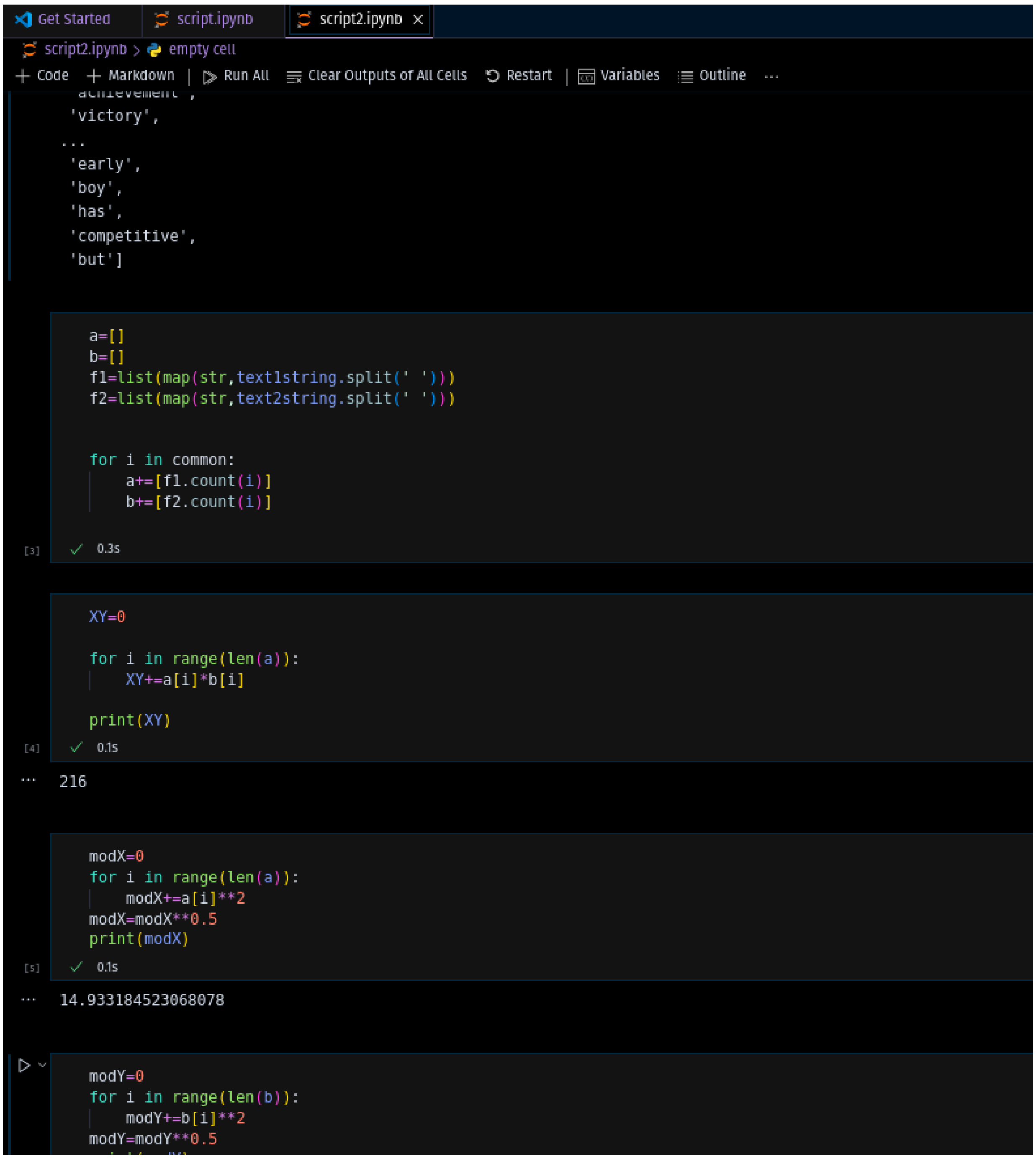
[1] ✓ 0.2s

```
f1=set(text1string.split(' '))
f1org=f1.copy()
f2=set(text2string.split(' '))
f2org=f2.copy()

common=f1.intersection(f2)
common=list(common)
common
```

[2] ✓ 0.1s

```
Output exceeds the size limit. Open the full output data in a text editor
['for',
 'history',
 'poster',
 'too',
 'drive',
 'all',
 'won',
 'than',
 'pace',
 'Michael',
 'trademark',
 '—',
 'karaoke',
 'with',
 'flag,',
 '-',
 'streak,',
 'in',
 'But',
 'been',
 'loose',
 'a',
 'his',
```



```
 'victory',
...
 'early',
 'boy',
 'has',
 'competitive',
 'but']
```

```
a=[]
b=[]
f1=list(map(str,text1string.split(' ')))
f2=list(map(str,text2string.split(' ')))


for i in common:
    a+=[f1.count(i)]
    b+=[f2.count(i)]
```

[3] ✓ 0.3s

```
XY=0

for i in range(len(a)):
    XY+=a[i]*b[i]

print(XY)
```

[4] ✓ 0.1s

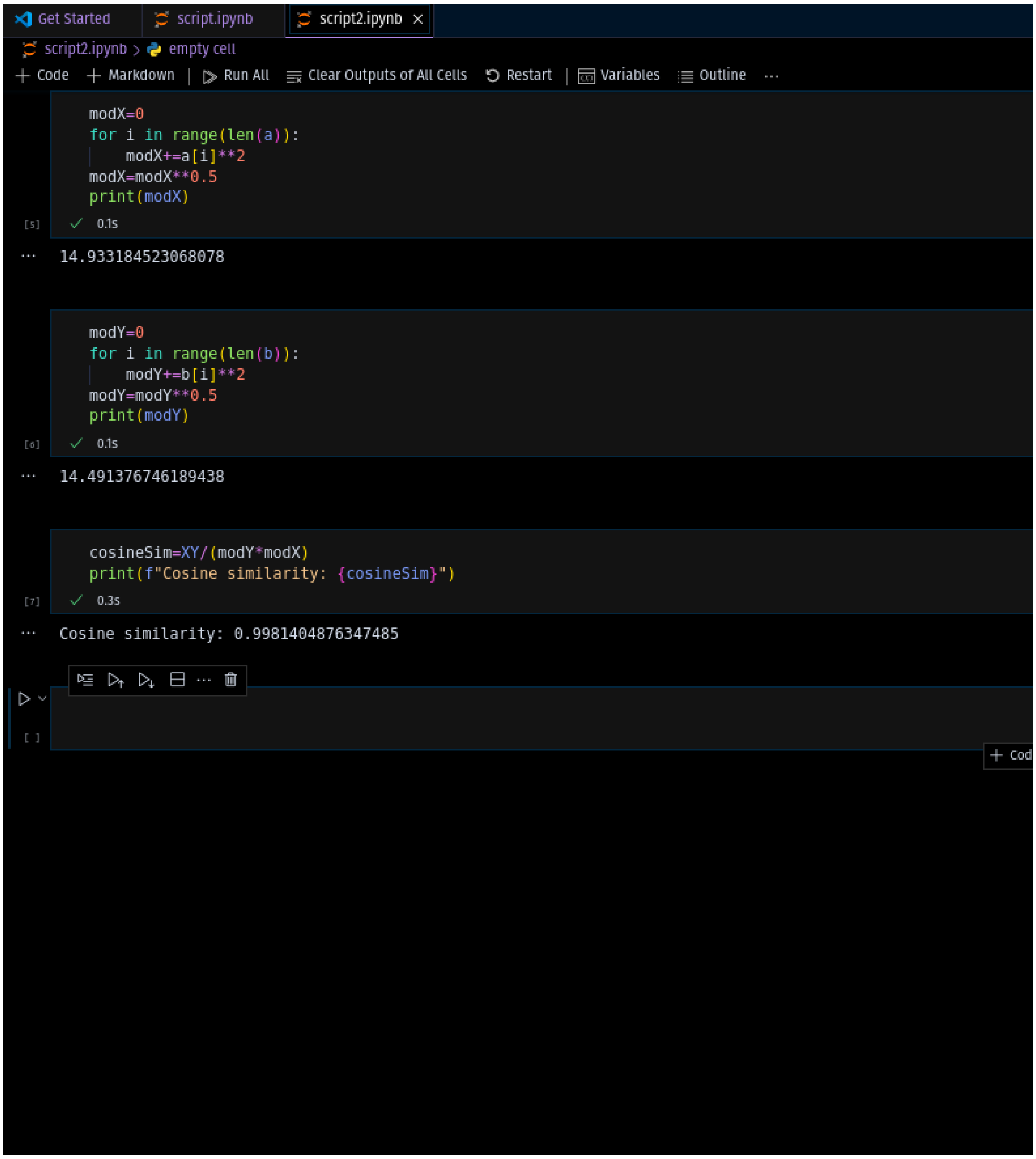
```
216
```

```
modX=0
for i in range(len(a)):
    modX+=a[i]**2
modX=modX**0.5
print(modX)
```

[5] ✓ 0.1s

```
14.933184523068078
```

```
modY=0
for i in range(len(b)):
    modY+=b[i]**2
modY=modY**0.5
```



```
modX=0
for i in range(len(a)):
    modX+=a[i]**2
modX=modX**0.5
print(modX)
```

[5] ✓ 0.1s

```
14.933184523068078
```

```
modY=0
for i in range(len(b)):
    modY+=b[i]**2
modY=modY**0.5
print(modY)
```

[6] ✓ 0.1s

```
14.491376746189438
```

```
cosineSim=XY/(modY*modX)
print(f"Cosine similarity: {cosineSim}")
```

[7] ✓ 0.3s

```
Cosine similarity: 0.9981404876347485
```

[ ]

Questions:

1. Define cosine similarity with an example.

In data analysis, cosine similarity is a measure of similarity between two sequences of numbers. For defining it, the sequences are viewed as vectors in an inner product space, and the cosine similarity is defined as the cosine of the angle between them, that is, the dot product of the vectors divided by the product of their lengths. It follows that the cosine similarity does not depend on the magnitudes of the vectors, but only on their angle. The cosine similarity always belongs to the interval [-1,1]. For example, two proportional vectors have a cosine similarity of 1, two orthogonal vectors have a similarity of 0, and two opposite vectors have a similarity of -1. The cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

2. What are similarity measures applicable to textual texts other than cosine distance. List and explain at least 3 of them.

a. Bag of words

b. count-vectorizer and document-term-matrix
c. levenshtein distance

Outcomes:

CO2 : Comprehend descriptive and proximity measures of data

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

Completely understood application of similarity measures on textual dataset

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3nd Edition
2. Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. Introduction to data mining. Pearson Education India, 2016.