

Modern Education Society's
College of Engineering, Pune

NAME OF STUDENT	CLASS
SEMESTER/YEAR	ROLL NO
DATE OF PERFORMANCE	DATE OF SUBMISSION
EXAMINED BY	EXPERIMENT NO

Assignment No-1

Title: Write a program to Compute Similarity between two text documents.

Objectives: To Compute Similarity between two text documents.

Problem Statement: Write a program to Compute Similarity between two text documents.

Outcomes: Student can understand how to compute similarity between two text.

Tools Required:

Hardware:

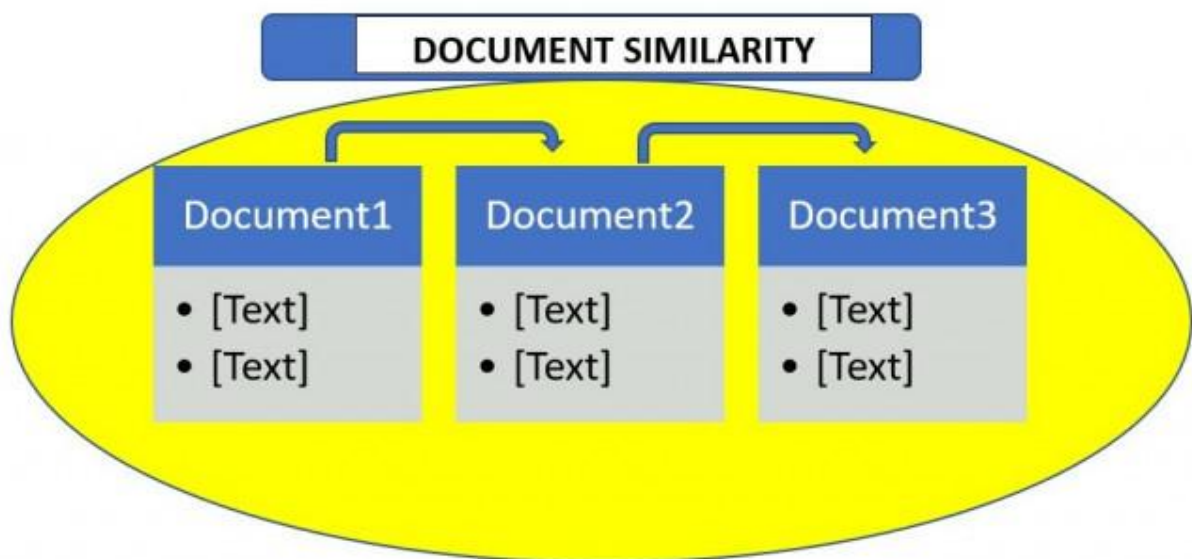
Software: Open source operating system

Theory:

Introduction:

Document similarity, as the name suggests determines how similar are the two given documents. By “documents”, we mean a collection of strings. For example, an essay or a .txt file. Many organizations use this principle of document similarity to check plagiarism. It is also used by many exams conducting institutions to check if a

student cheated from the other. Therefore, it is very important as well as interesting to know how all of this works.



Document similarity is calculated by calculating document distance. Document distance is a concept where words(documents) are treated as vectors and is calculated as the angle between two given document vectors. Document vectors are the frequency of occurrences of words in a given document. Let's see an example:

Say that we are given two documents **D1** and **D2** as:

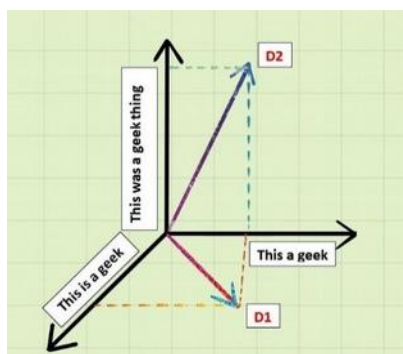
D1: "This is a geek"

D2: "This was a geek thing"

The similar words in both these documents then become:

"This a geek"

If we make a 3-D representation of this as vectors by taking D1, D2 and similar words in 3 axis geometry, then we get:



Now if we take dot product of **D1** and **D2**,

```
1.D2 = "This"."This"+"is"."was"+"a"."a"+"geek"."geek"+"thing".0
```

```
D1.D2 = 1+0+1+1+0
```

```
D1.D2 = 3
```

Now that we know how to calculate the dot product of these documents, we can now calculate the angle between the document vectors:

```
cos d = D1.D2/|D1||D2|
```

Here d is the document distance. It's value ranges from 0 degree to 90 degrees. Where 0 degree means the two documents are exactly identical and 90 degrees indicate that the two documents are very different.

Now that we know about document similarity and document distance, let's look at a Python program to calculate the same:

Document similarity program:

Our algorithm to confirm document similarity will consist of three fundamental steps:

- Split the documents in words.
- Compute the word frequencies.
- Calculate the dot product of the document vectors.

For the first step, we will first use the `read()` method to open and read the content of the files. As we read the contents, we will split them into a list. Next, we will calculate the word frequency list of the read in the file. Therefore, the occurrence of each word is counted and the list is sorted alphabetically.

Steps:

Note: Install modules with the help of pip

Pip install nltk

pip install numpy

`pip install stopwords`

Steps:

1) Create 2 text files on your desktop.

2) For example, let's create 2 text files as given below

Text1.txt

Hello IR

Text2.txt

Hello IR

3) And now copy paste the python code & note : Save that file on Desktop.

Copy and paste the code.

4) Then execute it.

If documents are exactly same then the value returned will be 1.0 else some floating point value will be returned based on how many similar terms are there in both documents

Program:

```
from nltk.corpus import stopwords
```

```
from nltk.tokenize import word_tokenize
```

```
import numpy as np
```

```
import nltk
```

```
nltk.download("punkt")
```

```
nltk.download("stopwords")
```

```
def process(file):
```

```
    raw=open (file).read()
```

```
    tokens=word_tokenize(raw)
```

```
    words=[w.lower() for w in tokens]
```

```
    porter= nltk.PorterStemmer()
```

```
    stemmed_tokens=[porter.stem(t) for t in words]
```

```
    # removing stop words
```

```
    stop_words=set(stopwords.words('english'))
```

```
    filtered_tokens=[w for w in stemmed_tokens if not w in  
stop_words]
```

```
    #count words
```

```
    count=nltk.defaultdict(int)
```

```
    for word in filtered_tokens:
```

```
        count[word]+=1
```

```
    return count;
```

```
def cos_sim(a,b):  
    dot_product=np.dot(a,b)  
    norm_a=np.linalg.norm(a)  
    norm_b=np.linalg.norm(b)  
    return dot_product/(norm_a * norm_b)
```

```
def getSimilarity(dict1,dict2):  
    all_words_list=[]  
    for key in dict1:  
        all_words_list.append(key)  
    for key in dict2:  
        all_words_list.append(key)  
    all_words_list_size=len(all_words_list)  
  
    v1=np.zeros(all_words_list_size,dtype=np.int)  
    v2=np.zeros(all_words_list_size,dtype=np.int)  
    i=0  
    for (key) in all_words_list:  
        v1[i]=dict1.get(key,0)  
        v2[i]=dict2.get(key,0)  
        i=i+1  
    return cos_sim(v1,v2)
```

```
if __name__ == '__main__':  
    dict1=process("text1.txt")  
    dict2=process("text2.txt")  
    print("Similarity between two text  
documents",getSimilarity(dict1,dict2))
```

Conclusion:After completion of this experiment.We have studied how to Compute Similarity between two text documents.

Questions:

Q.1)How to compute similarity between two text document?