Assignment 01

S14490

D.S.P.A.U.M. PEIRIS

1. Screenshots with an explanation of the tools you used for the above-mentioned Document Similarity implementation.

Libraries

In [10]:
```
import pandas as pd
import numpy as np
import seaborn as sns; sns.set_theme()
import seaborn as sns
import re

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import euclidean_distances

import os
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
nltk.download('punkt')
nltk.download('stopwords')
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
```

```
[nltk_data] Downloading package punkt to C:\Users\Anjula
[nltk_data]     Umesh\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to C:\Users\Anjula
[nltk_data]     Umesh\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Pandas (Create a data frame from the text documents)

Numpy (Transform vectors and perform dot products between vectors)

Seaborn (Visualiza the cosine similarity matrix)

Nltk.corpus (Remove stop words from the document)

Nltk.stem.porter (function to stem the words to their base form)

Sklearn.feature_extraction.text (To convert documents to vectors)

Sklearn.metrics.pairwise (To measure cosine similarity

Re (Regex library to remove special characters from the text)

2. Brief explanation of the pre-processing steps you followed.

Read all the files and constructed a pandas data frame and cleaned the data by removing stop words and special characters

3. List of .txt documents related to each news topic.

The heat maps depicts that the following titles and documents show similarity. This conclusion was atteined after considering the cosine similarity measure

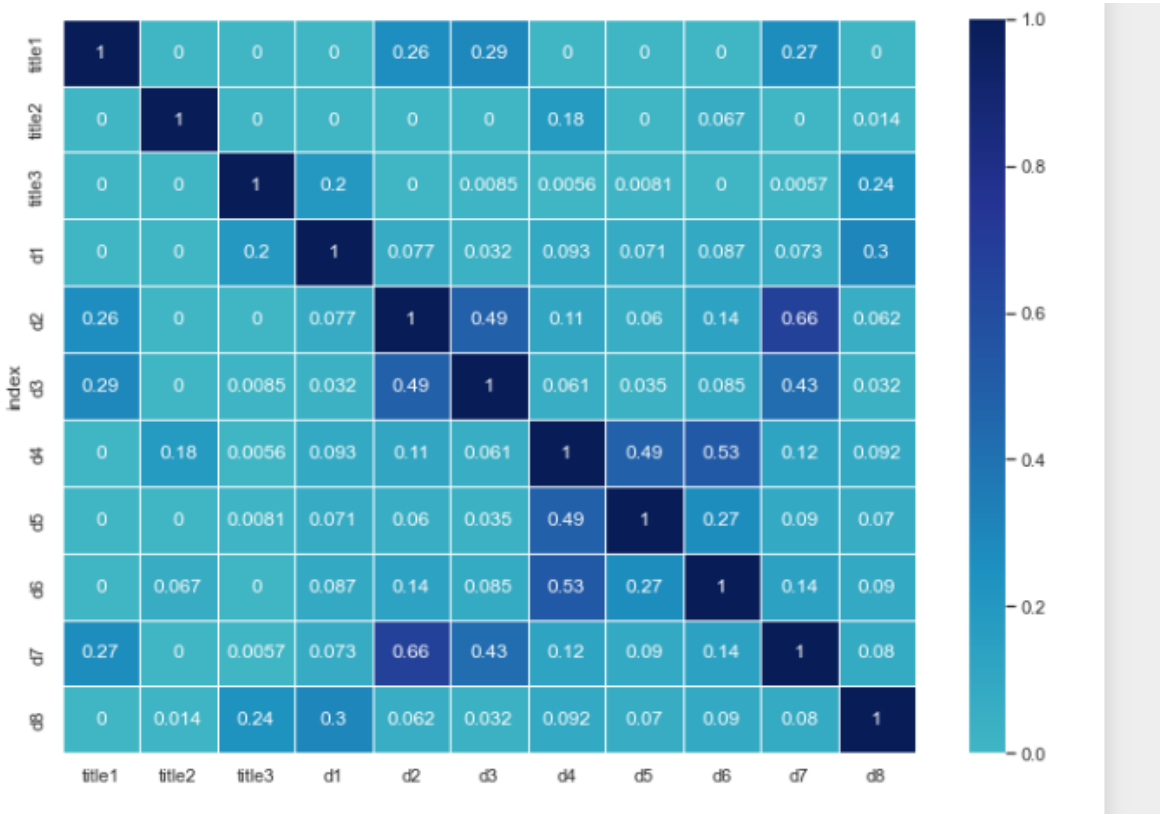
Title 01 = {doc 2, doc 3, doc 7}
Title 02 = {doc 4, doc6}
Title 03 = {doc 1, doc, 8}

Document 05 refrain from showing any similarities with the any of the 3 titles.
But preveours results dictates that document 05 showed some similarity to document 4 and document 6

Final Assumptions

- Title 1 = "Hurricane Gilbert Heads Toward Dominican Coast"
- Title 2= "IRA terrorist attack"
- Title 3= "McDonald's Opens First Restaurant in China"

- Title 1= {doc 2, doc 3, doc 7}
- Title 2= {doc 4, doc 5, doc6}
- Title 3= {doc 1, doc, 8}



4. <u>Append your full code lines at the end of the PDF file</u>

Libraries

In [10]:
```
import pandas as pd
import numpy as np
import seaborn as sns; sns.set_theme()
import seaborn as sns
import re

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import euclidean_distances

import os
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
nltk.download('punkt')
nltk.download('stopwords')
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
```

```
[nltk_data] Downloading package punkt to C:\Users\Anjula
[nltk_data]     Umesh\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to C:\Users\Anjula
[nltk_data]     Umesh\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Uploading data

In [11]:
```
doc1 = open("doc 1.txt", "r")
doc2 = open("doc 2.txt" , "r")
doc3 = open("doc 3.txt" , "r")
doc4 = open("doc 4.txt" , "r")
doc5 = open("doc 5.txt" , "r")
doc6 = open("doc 6.txt" , "r")
doc7 = open("doc 7.txt" , "r")
doc8 = open("doc 8.txt" , "r")
```

In [12]:
```
doc_1 = doc1.read()
doc_2 = doc2.read()
doc_3 = doc3.read()
doc_4 = doc4.read()
doc_5 = doc5.read()
doc_6 = doc6.read()
doc_7 = doc7.read()
doc_8 = doc8.read()

docs = [doc_1 , doc_2,doc_3 , doc_4,doc_5 , doc_6,doc_7 , doc_8]
```

In [13]:
```
doc_1
```

Out[13]:
```
"Thousands of queue-hardened Soviets on Wednesday cheerfully lined up to get a tas
te of ''gamburgers'', ''chizburgers'' and ''Filay-o-feesh'' sandwiches as McDonal
d's opened in the land of Lenin for the first time.\nThe world's largest version o
f the landmark American fast-food chain rang up 30,000 meals on 27 cash registers,
breaking the opening-day record for McDonald's worldwide, officials said.\nThe Sov
iets, bundled in fur coats and hats, seemed unfazed, lining up before dawn outside
the 700 seat restaurant, the first of 20 planned across the Soviet Union. \nThe cr
ush of customers was so intense the company stayed open until midnight, two hours
later than planned.\nI only waited an hour and I think they served thousands befor
e me, said a happy middle-aged woman who works at an aluminum plant.\nAnd it was o
nly 10 rubles for all this, she said. I'm taking it back for the girls at the fact
ory to try.\nBig Macs were priced at 3.75 rubles and double cheeseburgers at 3 rub
les about two hours' pay for a starting McDonald's staffer or the average Soviet,
but much cheaper than other private restaurants that have sprung up recently.\nThe
official exchange rate is 1.59 dollar per ruble but foreign visitors can buy ruble
s for 16 cents each, about what the currency is worth on the black market.\nHalf t
he day's sales were donated to the Soviet Children's Fund, which provides medical
care and assistance to orphans and disadvantaged children, Gary Reinblatt, senior
vice president of McDonald's Canada, said from Toronto.\nThe restaurant, built by
the company in a joint venture with the city of Moscow that began 14 years ago, br
ought to 52 the number of countries where McDonald's operates.\nThe previous openi
ng-day record for sales was in Budapest, company officials said. Besides its resta
urants in the United States, the leading number of McDonald's are in Canada and Ja
pan, the officials said.\nSoviets got a first-hand look at such alien concepts as
efficiency and fast, friendly service. Normally dour citizens broke into grins as
they caught the infectious cheerful mood from youthful Soviet staffers hired for t
heir ability to smile and work hard.\nAccordions played folk songs and women in tr
aditional costumes danced with cartoon characters, including Mickey Mouse and Baba
Yaga, a witch of Russian fairy tales.\nOne Muscovite, accustomed to clerks who sna
rl if they say anything at all, asked for a straw and was startled when a smiling
young Soviet woman found him one and popped it straight into his drink.\nFor most
customers, it was their first experience with a hamburger. Sandwiches were served
in the familiar bag marked with the golden arches, but were packed in wrappers bea
ring Cyrillic letters, approximating ``gamburger.''\nThey tried them one-handed.Th
ey picked their sandwiches apart to examine the contents. One young woman finally
squashed her ``Beeg Mak'' to fit her lips around it. \n''It tasted great!'' a 14 y
ears old boy said.\nIt's a lot different from a stolovaya,'' he continued with a s
mile, referring to the much cheaper but run down dirty cafeterias that slop rice a
nd fat or boiled sausage.\nUnder the sign of the golden arches, accented by the So
viet hammer and sickle flag, hundreds lined up for the long awaited grand opening
at 10 am on Pushkin Square, reaching out excitedly for McDonald's flags and pins a
s the hamburger chain's army fulfilled the Soviet penchant for souvenirs with West
ern logos.\nPublicity conscious managers had the staff shout ''Good morning, Ameri
ca!'' in English and Russian, for an American TV network.\nMcDonald's of Canada Ch
airman George Cohon, the man behind the deal, said many people were buying multipl
e orders and the restaurant served 15,000 to 20,000 people in just the first five
hours of operation.\nThe restaurant limited purchases to 10 Big Macs per customer
in hopes of preventing burger scalping.\nMcDonald's built its own factory, includi
ng bakery, dairy, meat processing plant and even potato storage yard, to provide i
ts own guaranteed supplies in a country where up to 25 percent of the harvest rots
en route to the consumer.\nOne McDonald's associate said the company wound up impo
rting wooden crates from Finland for storing potatoes because when they went to bu
ild crates, they found there was no wood, and no nails.\nThey found you need a per
mit to buy nails."
```

In [14]:
```
df = pd.DataFrame()
df['documents'] = docs
df.head(4)
```

Out[14]:

	documents
0	Thousands of queue-hardened Soviets on Wednesd...
1	Hurricane Gilbert, packing 110 mph winds and t...
2	Hurricane Gilbert swept toward the Dominican R...
3	An explosion today flattened a military barrac...

In [15]:
```
stop_words_l=stopwords.words('english')
df['cleaned_docs']=df.documents.apply(lambda x: " ".join(re.sub(r'[^a-zA-Z]',' ',w
```

TF IDF

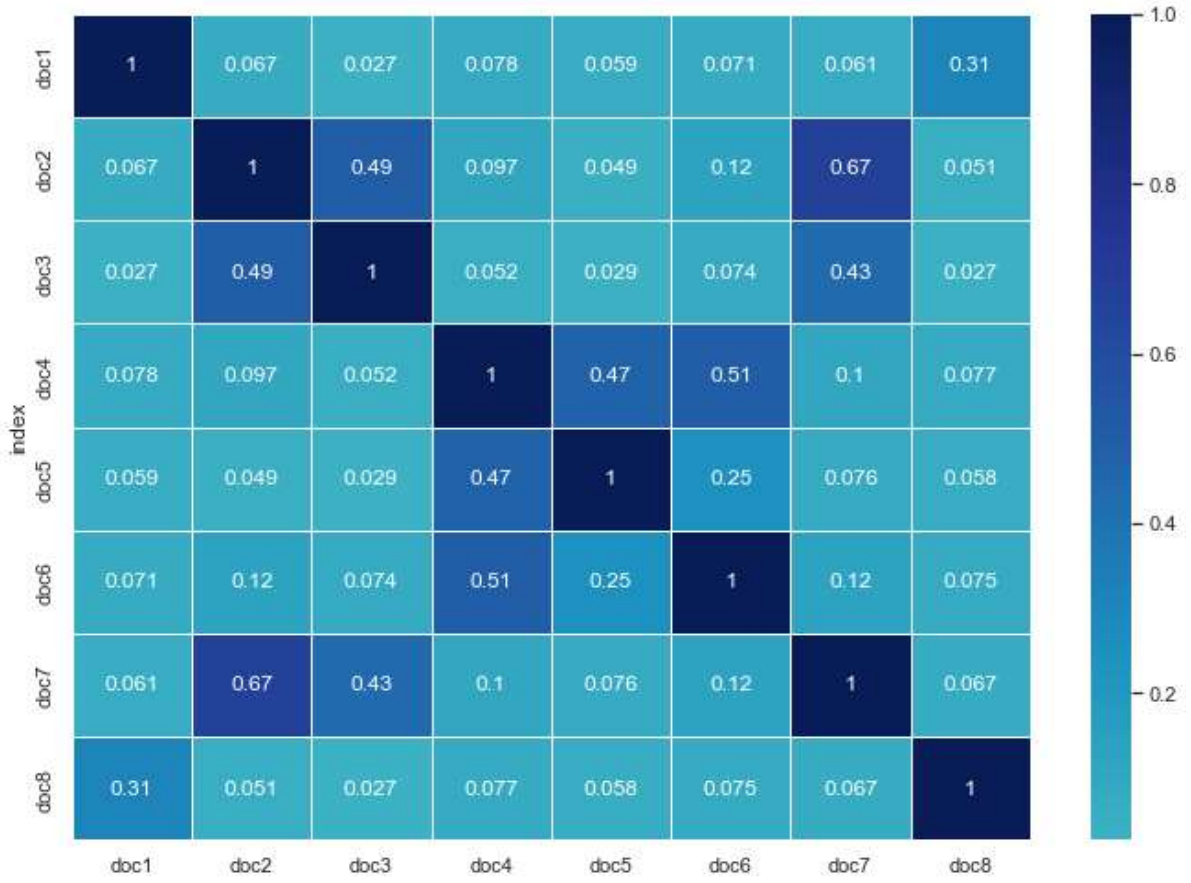
In [25]:
```
tfidfvectoriser=TfidfVectorizer()
tfidfvectoriser.fit(df.cleaned_docs)
tfidf_vectors=tfidfvectoriser.transform(df.cleaned_docs)

pairwise_similarities=np.dot(tfidf_vectors,tfidf_vectors.T).toarray()
pairwise_differences=cosine_similarity(tfidf_vectors)
```

In [32]:
```
#Cosine similarity between each other documents

df_tfidf = pd.DataFrame(pairwise_similarities, columns = ['doc1','doc2','doc3','doc
df_tfidf['index'] = ['doc1','doc2','doc3','doc4','doc5','doc6','doc7','doc8']
df_tfidf.index = df_tfidf["index"]
df_tfidf.drop('index', inplace=True, axis=1)
print(df_tfidf)
```

```
           doc1      doc2      doc3      doc4      doc5      doc6      doc7  \
index
doc1   1.000000  0.066642  0.026974  0.078243  0.058865  0.071022  0.060783
doc2   0.066642  1.000000  0.491084  0.096656  0.049404  0.119074  0.666062
doc3   0.026974  0.491084  1.000000  0.052050  0.028986  0.073928  0.433640
doc4   0.078243  0.096656  0.052050  1.000000  0.472237  0.506724  0.100712
doc5   0.058865  0.049404  0.028986  0.472237  1.000000  0.245422  0.075643
doc6   0.071022  0.119074  0.073928  0.506724  0.245422  1.000000  0.124801
doc7   0.060783  0.666062  0.433640  0.100712  0.075643  0.124801  1.000000
doc8   0.307001  0.051172  0.027002  0.077030  0.057951  0.074719  0.066800

           doc8
index
doc1   0.307001
doc2   0.051172
doc3   0.027002
doc4   0.077030
doc5   0.057951
doc6   0.074719
doc7   0.066800
doc8   1.000000
```

Cosine Similarity

In [39]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
ax = sns.heatmap(df_tfidf,annot=True, center=0, linewidth=0.5, cmap="YlGnBu")
```



Clusters based on similarity measures:

- Doc 01 and Doc 08
- Doc 02, Doc 03, Doc 07
- Doc 04, Doc 05, Doc 06

Above matrix visualize the documents that are similar to each other

Checking articles relating to following topics

- Hurricane Gilbert Heads Toward Dominican Coast
- IRA terrorist attack
- McDonald's Opens First Restaurant in China

In [40]:
```
t1 = "Hurricane Gilbert Heads Toward Dominican Coast"
t2 = "IRA terrorist attack"
t3 = "McDonald's Opens First Restaurant in China"
```

In [41]:
```
docs = [t1,t2,t3,doc_1 , doc_2,doc_3 , doc_4,doc_5 , doc_6,doc_7 , doc_8]
```

In [42]:
```
#Creating a DF
df = pd.DataFrame()
df['documents'] = docs
df.head(4)
```

Out[42]:

	documents
0	Hurricane Gilbert Heads Toward Dominican Coast
1	IRA terrorist attack
2	McDonald's Opens First Restaurant in China
3	Thousands of queue-hardened Soviets on Wednesd...

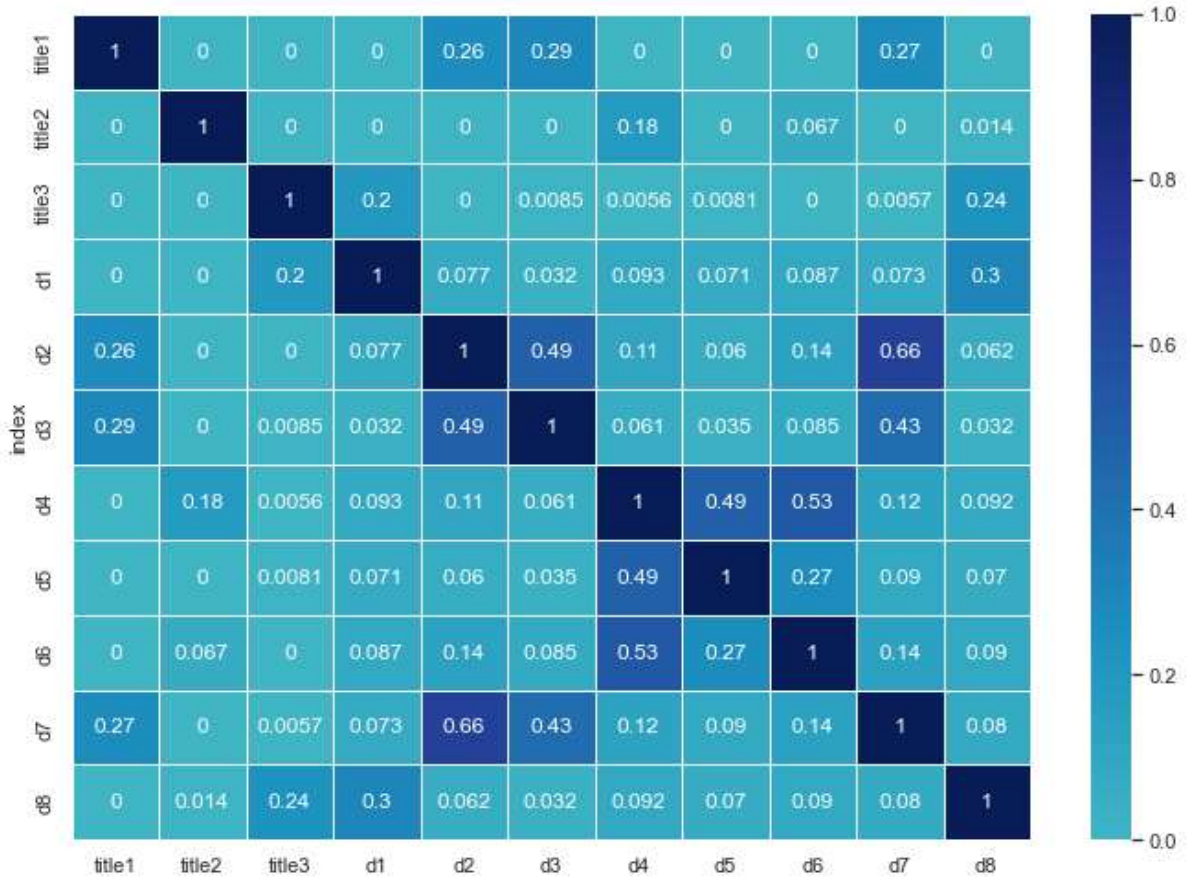
In [43]:
```
stop_words_l=stopwords.words('english')
df['cleaned_docs']=df.documents.apply(lambda x: " ".join(re.sub(r'[^a-zA-Z]',' ',w
```

In [44]:
```
tfidfvectoriser=TfidfVectorizer()
tfidfvectoriser.fit(df.cleaned_docs)
tfidf_vectors=tfidfvectoriser.transform(df.cleaned_docs)

pairwise_similarities=np.dot(tfidf_vectors,tfidf_vectors.T).toarray()
pairwise_differences=cosine_similarity(tfidf_vectors)
```

In [45]:
```
#Cosine similarity between other documents

df_tfidf_title = pd.DataFrame(pairwise_similarities, columns = ['title1','title2',
df_tfidf_title['index'] = ['title1','title2','title3','d1','d2','d3','d4','d5','d6
df_tfidf_title.index = df_tfidf_title["index"]
df_tfidf_title.drop('index', inplace=True, axis=1)
print(df_tfidf_title)
```

```
          title1    title2    title3        d1        d2        d3        d4  \
index
title1  1.000000  0.000000  0.000000  0.000000  0.263809  0.293651  0.000000
title2  0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.181938
title3  0.000000  0.000000  1.000000  0.201713  0.000000  0.008479  0.005627
d1      0.000000  0.000000  0.201713  1.000000  0.077438  0.031598  0.093115
d2      0.263809  0.000000  0.000000  0.077438  1.000000  0.489123  0.111991
d3      0.293651  0.000000  0.008479  0.031598  0.489123  1.000000  0.060587
d4      0.000000  0.181938  0.005627  0.093115  0.111991  0.060587  1.000000
d5      0.000000  0.000000  0.008062  0.070757  0.059728  0.034569  0.491804
d6      0.000000  0.066815  0.000000  0.086775  0.136341  0.084562  0.525264
d7      0.266087  0.000000  0.005665  0.073067  0.664209  0.427281  0.118622
d8      0.000000  0.013903  0.239132  0.297913  0.061502  0.032470  0.091599

              d5        d6        d7        d8
index
title1  0.000000  0.000000  0.266087  0.000000
title2  0.000000  0.066815  0.000000  0.013903
title3  0.008062  0.000000  0.005665  0.239132
d1      0.070757  0.086775  0.073067  0.297913
d2      0.059728  0.136341  0.664209  0.061502
d3      0.034569  0.084562  0.427281  0.032470
d4      0.491804  0.525264  0.118622  0.091599
d5      1.000000  0.267175  0.089735  0.070105
d6      0.267175  1.000000  0.144274  0.090439
d7      0.089735  0.144274  1.000000  0.080153
d8      0.070105  0.090439  0.080153  1.000000
```

In [47]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
ax = sns.heatmap(df_tfidf_title,annot=True, center=0, linewidth=0.5, cmap="YlGnBu"
```



The heat maps depicts that the following titles and documents show similarity. This conclusion was atteined after considering the cosine similarity measure

Title 01 = {doc 2, doc 3, doc 7} Title 02 = {doc 4, doc6} Title 03 = {doc 1, doc, 8}

Document 05 refrain from showing any similarities with the any of the 3 titles. But preveours results dictates that document 05 showed some similarity to document 4 and document 6

Final Assumptions

- Title 1 = "Hurricane Gilbert Heads Toward Dominican Coast"
- Title 2= "IRA terrorist attack"
- Title 3= "McDonald's Opens First Restaurant in China"

- Title 1= {doc 2, doc 3, doc 7}
- Title 2= {doc 4, doc 5, doc6}
- Title 3= {doc 1, doc, 8}

In [ ]: