

**archive.today**

webpage capture

**Saved from**<https://medium.com/data-science-collective/document-similarity-using-cosine-simila>

search

28 Oct 2025 10:19:21 UTC

**Redirected from**<https://sandanisesanika.medium.com/document-similarity-using-cosine-similarity-bdc>

no other snapshots from this url

**All snapshots from host** [medium.com](https://medium.com)**from host** [sandanisesanika.medium.com](https://sandanisesanika.medium.com)

no other snapshots from this url

**Webpage****Screenshot**

share

download .zip

report bug or abuse

Buy me a coffee

**Medium**

Search



Write

Sign up

Sign in

**Data Science Collective**

Member-only story

# Document Similarity Using Cosine Similarity



Sandani Fernando

Follow

3 min read · Sep 3, 2022



7



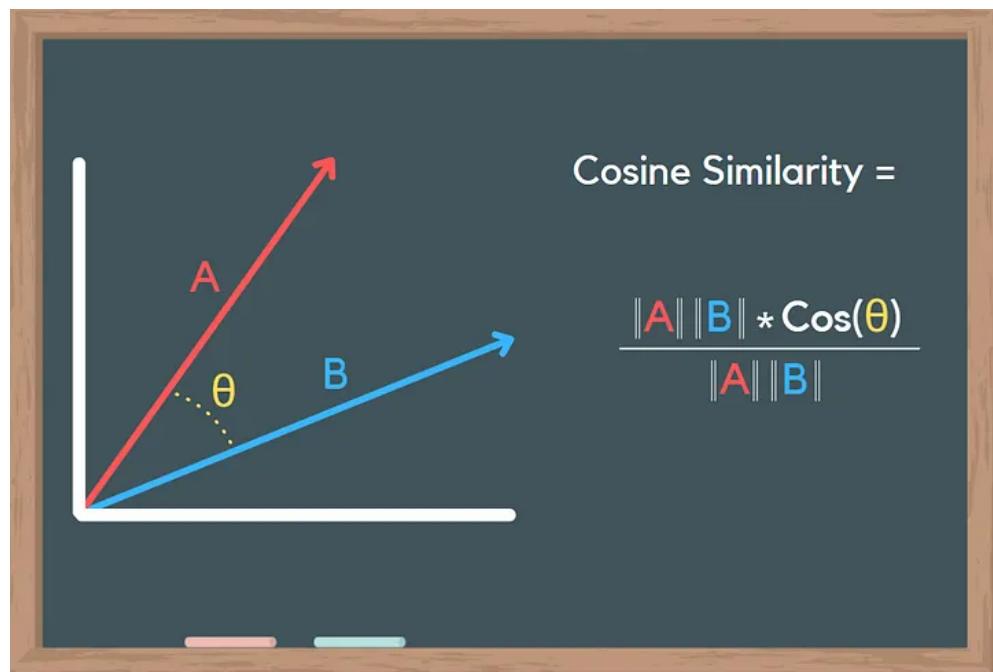
Here I considered 8 text documents are a set of news articles related to three 3 different news topics namely, Hurricane Gilbert Heads Toward Dominican Coast, IRA terrorist attack, and McDonald's Opens First Restaurant in China.

So, I plan to determine Document Similarity; how similar two or more documents are concerning each other in this document collection.

## Cosine Similarity

Cosine Similarity is a measurement that quantifies the similarity between two or more vectors. The cosine similarity is the cosine of the angle between vectors. The vectors are typically non-zero and are within an inner product space.

The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the Euclidean norms or magnitude of each vector.



The first step to this is preprocessing the data in the 8 documents.

1. Removal of stop words. (These are the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc.) and do not add much information to the text.)
2. Removal of numbers and special characters. (Also, you can convert the numbers into 'num' and then continue)
3. Convert all the letters in the documents to lowercase letters.

## TfidfVectorizer

Transforms text to feature vectors that can be used as input to the estimator. `vocabulary_` is a dictionary that converts each token (word) to a feature index in the matrix, each unique token gets a feature index.

```
[1] # Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import math
import io
import sklearn

[2] from sklearn.feature_extraction.text import TfidfVectorizer
```

Importing the Libraries

```
from google.colab import files
uploaded = files.upload()

Choose Files 8 files
• doc 1.txt(text/plain) - 4366 bytes, last modified: 8/3/2022 - 100% done
• doc 2.txt(text/plain) - 4894 bytes, last modified: 8/3/2022 - 100% done
• doc 3.txt(text/plain) - 1960 bytes, last modified: 8/3/2022 - 100% done
• doc 4.txt(text/plain) - 4626 bytes, last modified: 8/3/2022 - 100% done
• doc 5.txt(text/plain) - 2567 bytes, last modified: 8/3/2022 - 100% done
• doc 6.txt(text/plain) - 2467 bytes, last modified: 8/3/2022 - 100% done
• doc 7.txt(text/plain) - 4046 bytes, last modified: 8/3/2022 - 100% done
• doc 8.txt(text/plain) - 3606 bytes, last modified: 8/3/2022 - 100% done

Saving doc 1.txt to doc 1.txt
Saving doc 2.txt to doc 2.txt
Saving doc 3.txt to doc 3.txt
Saving doc 4.txt to doc 4.txt
Saving doc 5.txt to doc 5.txt
Saving doc 6.txt to doc 6.txt
Saving doc 7.txt to doc 7.txt
Saving doc 8.txt to doc 8.txt
```

Uploading Files

Now, we can define the documents in the following way.

```
[5] d1 = open("doc 1.txt").read()
    d2 = open("doc 2.txt").read()
    d3 = open("doc 3.txt").read()
    d4 = open("doc 4.txt").read()
    d5 = open("doc 5.txt").read()
    d6 = open("doc 6.txt").read()
    d7 = open("doc 7.txt").read()
    d8 = open("doc 8.txt").read()
```

Reading the files

Next, convert the collection of raw documents to a matrix of TF-IDF features.

```
[6] response=tfidf.fit_transform([d1,d2,d3,d4,d5,d6,d7,d8])

[7] tfidf.get_feature_names_out()
array(['ability', 'absolutely', 'accented', ..., 'yugoslavia',
       'yugoslavs', 'zone'], dtype=object)

[8] print(len(tfidf.vocabulary_))
1220

[9] tfidf.vocabulary_
{'tell': 1093,
 'scotland': 950,
 'forensic': 422,
 'team': 1087,
 'antiterrorist': 37,
 'squad': 1033,
 'called': 150,
 'help': 498,
 'investigate': 553,
 'gave': 442,
 'took': 1113,
 'suffering': 1070,
 'burns': 142,
 'head': 487,
 'fractures': 424,
 'sort': 1021,
 'injuries': 540,
 'expect': 360,
 'buckland': 131,
 'hospital': 512,
 'dover': 309,
 'gas': 439,
 'investigators': 554,
```

```
▶ from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(response)

array([[1.          , 0.08478265, 0.03724656, 0.09173213, 0.05683063,
       0.07873028, 0.06678192, 0.32589712],
       [0.08478265, 1.          , 0.50371969, 0.10770526, 0.05514628,
       0.12236688, 0.67489687, 0.06879077],
       [0.03724656, 0.50371069, 1.          , 0.05704943, 0.02799075,
       0.08077183, 0.44647155, 0.03188128],
       [0.09173213, 0.10770526, 0.05704943, 1.          , 0.47136054,
       0.51744755, 0.10181042, 0.08550804],
       [0.05683063, 0.05514628, 0.02799075, 0.47136054, 1.          ,
       0.2618378 , 0.06858955, 0.06015473],
       [0.07873028, 0.12236688, 0.08077183, 0.51744755, 0.2618378 ,
       1.          , 0.1261857 , 0.07749121],
       [0.06678192, 0.67489687, 0.44647155, 0.10181042, 0.06858955,
       0.1261857 , 1.          , 0.06270039],
       [0.32589712, 0.06879077, 0.03188128, 0.08550804, 0.06015473,
       0.07749121, 0.06270039, 1.          ]])
```

Cosine Similarities

According to cosine similarity, article 1 is most similar to article 8. Articles 2,3 and 7 are most similar to each other. And articles 4,5,6 are most similar to each other.

Let's meet with another interesting topic later!