

Magnum Opus Task - 2 Cosine Similarity

Name – Tejas Jambhale

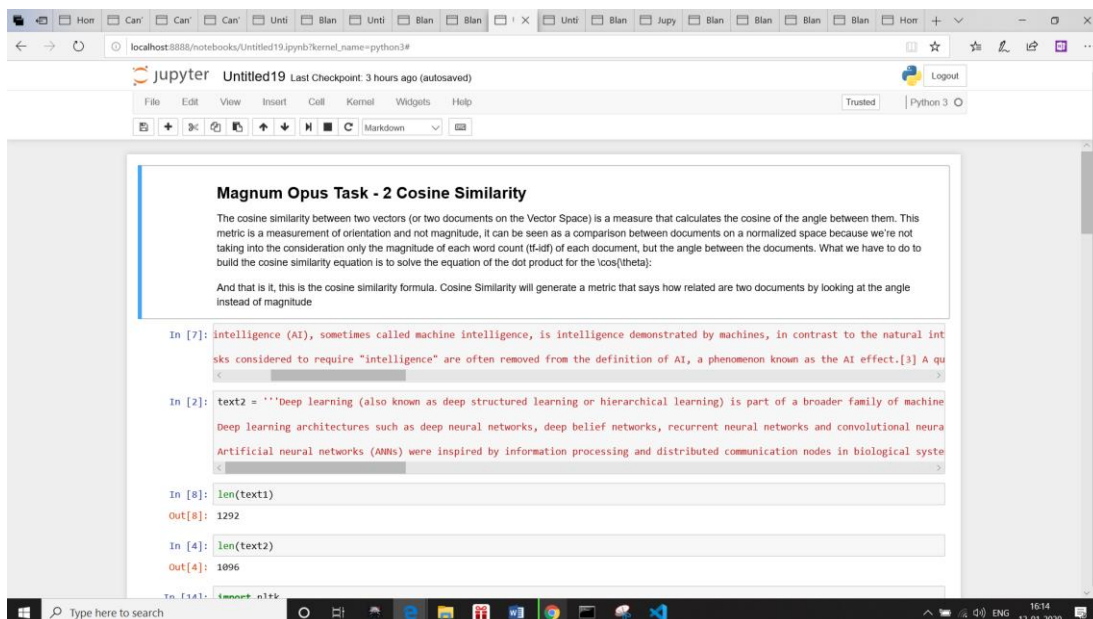
Reg No. – 17BCE0861

The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space because we're not taking into the consideration only the magnitude of each word count (tf-idf) of each document, but the angle between the documents. What we have to do to build the cosine similarity equation is to solve the equation of the dot product for the $\cos \theta$:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

And that is it, this is the cosine similarity formula. Cosine Similarity will generate a metric that says how related are two documents by looking at the angle instead of magnitude



```
localhost:8888/notebooks/Untitled19.ipynb?kernel_name=python3#Magnum-Opus-Task--2-Cosine-Similarity

Jupyter Untitled19 Last Checkpoint: 3 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [14]: import nltk
from nltk.corpus import stopwords

words1 = nltk.word_tokenize(text1) #The string of text is converted in to a List pf tokens
words2 = nltk.word_tokenize(text2)

stop_words = set(stopwords.words('english'))
filtered_words1 = [w for w in words1 if not w in stop_words] #StopWords are removed
filtered_words2 = [w for w in words2 if not w in stop_words]

final1 = " ".join(filtered_words1)
final2 = " ".join(filtered_words2)

In [35]: documents=(final1,final2)
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents) #Vector space is created
print (tfidf_matrix.shape)

from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)

(2, 164)

Out[35]: array([[1.          , 0.07846781]])

This above output shows that the similarity between the two texts is 0.07
```

```
localhost:8888/notebooks/Untitled19.ipynb?kernel_name=python3#

Jupyter Untitled19 Last Checkpoint: 3 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

This above output shows that the similarity between the two texts is 0.07

In [36]: import math
# This was already calculated on the previous step, so we just use the value
cos_sim = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)[0,1]
angle_in_radians = math.acos(cos_sim)
print(math.degrees(angle_in_radians))

85.49949917866428

The cosine angle of similarity is approximately 85 degrees

In [29]: tfidf_matrix

Out[29]: <2x164 sparse matrix of type '<class 'numpy.float64''
         with 176 stored elements in Compressed Sparse Row format>

In [50]: l = len(documents) - 1
for i in range(l):
    minimum = (1, None)
    minimum = min((spatial.distance.cosine(tfidf_matrix[i].todense(), tfidf_matrix[i + 1].todense()), i), minimum)
for i in minimum:
    print(1-i)

0.87846780976059198
1

The above results comes from using Scipy library to find the cosine similarity
```

The above results comes from using Scipy library to find the cosine similarity

```
In [65]: stemmer = nltk.stem.porter.PorterStemmer() #Initialize the porter stemmer and stem each word in the two texts
wordstem1 = [stemmer.stem(token) for token in filtered_words1]
wordstem2 = [stemmer.stem(token) for token in filtered_words2]
finalstem1 = " ".join(wordstem1)
finalstem2 = " ".join(wordstem2)
documents=(finalstem1,finalstem2)
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
print (tfidf_matrix.shape)

cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)

cos_sim = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[:,1])
angle_in_radians = math.acos(cos_sim)
print(math.degrees(angle_in_radians))

(2, 22)

Out[65]: array([[1.          , 0.17383606]])
79.98906864712323
```

As we can see after performing stemming the similarity values more than doubles to 0.17 and angle reduces to 7.99. Thus here stemming can be justified

```
In [67]: text1 = 'NLP comprises models, techniques and strategies to help us understand how the language we use influences the way we th
```

Here we take 4 different definitions of NLP to compare the similarity and compare the results of before and after stemming

```
In [67]: text1 = 'NLP comprises models, techniques and strategies to help us understand how the language we use influences the way we th
text2 = 'natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken'
text3 = 'natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artific
text4 = 'natural language processing (NLP) is a discipline which is interested in how human languages, and, to some extent, the

import nltk
from nltk.corpus import stopwords

words1 = nltk.word_tokenize(text1)
words2 = nltk.word_tokenize(text2)
words3 = nltk.word_tokenize(text3)
words4 = nltk.word_tokenize(text4)

stop_words = set(stopwords.words('english'))
filtered_words1 = [w for w in words1 if w not in stop_words]
filtered_words2 = [w for w in words2 if w not in stop_words]
filtered_words3 = [w for w in words3 if w not in stop_words]
filtered_words4 = [w for w in words4 if w not in stop_words]

final1 = " ".join(filtered_words1)
final2 = " ".join(filtered_words2)
final3 = " ".join(filtered_words3)
final4 = " ".join(filtered_words4)

documents=(final1,final2,final3,final4)
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
print (tfidf_matrix.shape)

from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)

wordstem1 = [stemmer.stem(token) for token in filtered_words1]
wordstem2 = [stemmer.stem(token) for token in filtered_words2]
wordstem3 = [stemmer.stem(token) for token in filtered_words3]
wordstem4 = [stemmer.stem(token) for token in filtered_words4]
finalstem1 = " ".join(wordstem1)
finalstem2 = " ".join(wordstem2)
finalstem3 = " ".join(wordstem3)
finalstem4 = " ".join(wordstem4)
documents=(finalstem1,finalstem2,finalstem3,finalstem4)

tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
print (tfidf_matrix.shape)

cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
```

localhost:8888/notebooks/Untitled19.ipynb?kernel_name=python3#

jupyter Untitled19 Last Checkpoint: 3 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
wordstem = [word_tokenize(word) for word in filtered_documents]
finalstem1 = " ".join(wordstem1)
finalstem2 = " ".join(wordstem2)
finalstem3 = " ".join(wordstem3)
finalstem4 = " ".join(wordstem4)
documents=(finalstem1,finalstem2,finalstem3,finalstem4)

tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
print (tfidf_matrix.shape)

cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
```

(4, 46)

Out[67]: array([[1. , 0.15612086, 0.04356603, 0.04898682]])

(4, 41)

Out[67]: array([[1. , 0.15612086, 0.05646589, 0.07332736]])

The above output gives an array which is the similarity of the first with all other sentences. When checking without stemming we can see above the first and second sentence have the most cosine similarity

After stemming there is no difference between similarity of first 2 sentences but the cosine similarity visibly differs in the other two sentence value when all sentences are compared the first one

In []:

Type here to search 16:20 12-01-2020

