Cosine Similarity – Understanding the math and how it works (with python codes)

by Selva Prabhakaran (https://www.machinelearningplus.com/author/selva86/) |

[/#facebook] [/#twitter] [/#whatsapp] [/#linkedin] [/#reddit] [/#google_bookmarks] [/#google_gmail]
[https://www.addtoany.com/share#url=https%3A%2F%2Fwww.machin
similarity%2F&title=Cosine%20Similarity%20%E2%80%93%20Understandin

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

By the end of this tutorial you will know:

1. What is cosine similarity is and how it works?
2. How to compute cosine similarity of documents in python?
3. What is soft cosine similarity and how its different from cosine similarity?
4. When to use soft cosine similarity and how to compute it in python?

Cosine Similarity – Understanding the math and how it works. Photo by Matt Lamers

Search here..

Upcoming Posts

spaCy - Complete Natural Laguage Processing Guide (NEW) (https://www.machinelearningplus.com/spacy-nlp-guide)
101 NLP Exercises (using modern libraries) (NEW) (https://www.machinelearningplus.com/nlp/nlp-exercises/)
How to train spaCy to autodetect new entities (NER) (NEW) (https://www.machinelearningplus.com/nlp/training-custom-ner-model-in-spacy/)
Support Vector Machines Algorithm from Scratch
Creating Plots in Julia
Julia DataFrames (NEW) (https://www.machinelearningplus.com/julia/datafram in-julia/)
101 Julia Practice Exercises
Python SQLite - Must Read Guide
Linear Regression with Julia (NEW) (https://www.machinelearningplus.com/linear-regression-in-julia/)
Waterfall Plot in Python (NEW) (https://www.machinelearningplus.com/waterfall-plot-in-python)
Python JSON - Guide (https://www.machinelearningplus.com/python-json-guide/)
Logistic Regression in Julia
Probability Theory - Beginners Guide
Graph Theory
Gentle Introduction to Markov Chain (NEW) (https://www.machinelearningplus.com/?page_id=3489&preview=true)
Logistic Regression from Scratch
Python Collections - Guide and Examples (NEW) (https://www.machinelearningplus.com/python-collections-guide/)

1. Introduction

A commonly used approach to match similar documents is based on counting the maximum number of common words between the documents.

But this approach has an inherent flaw. That is, as the size of the document increases, the number of common words tend to increase even if the documents talk about different topics.

The cosine similarity helps overcome this fundamental flaw in the 'count-the-common-words' or Euclidean distance approach.

2. What is Cosine Similarity and why is it advantageous?

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size.

Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents.

As a similarity metric, how does cosine similarity differ from the number of common words?

When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude. If you want the magnitude, compute the Euclidean distance instead.

The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance because of the size (like, the word 'cricket' appeared 50 times in one document and 10 times in another) they could still have a smaller angle between them. Smaller the angle, higher the similarity.

3. Cosine Similarity Example

Recent Posts

Subplots Python (Matplotlib) (https://www.machinelearningplus.com/plots/subplots-python-matplotlib/)

101 NLP Exercises (using modern libraries) (https://www.machinelearningplus.com/nlp/nlp-exercises/)

How to Train spaCy to Autodetect New Entities (NER) [Complete Guide] (https://www.machinelearningplus.com/nlp/training-custom-ner-model-in-spacy/)

For-Loop in Julia (https://www.machinelearningplus.com/julia/for-loop-in-julia/)

DataFrames in Julia (https://www.machinelearningplus.com/julia/datafram in-julia/)

Matplotlib Line Plot (https://www.machinelearningplus.com/plots/matplotl line-plot/)

K-Means Clustering Algorithm from Scratch (https://www.machinelearningplus.com/predictive-modeling/k-means-clustering/)

While-loop in Julia (https://www.machinelearningplus.com/julia/while-loop-in-julia/)

Function in Julia (https://www.machinelearningplus.com/julia/function-in-julia/)

Python Scatter Plot (https://www.machinelearningplus.com/plots/python-scatter-plot/)

Julia – Programming Language (https://www.machinelearningplus.com/julia/julia-programming-language/)

Requests in Python (Guide) (https://www.machinelearningplus.com/python/reques in-python/)

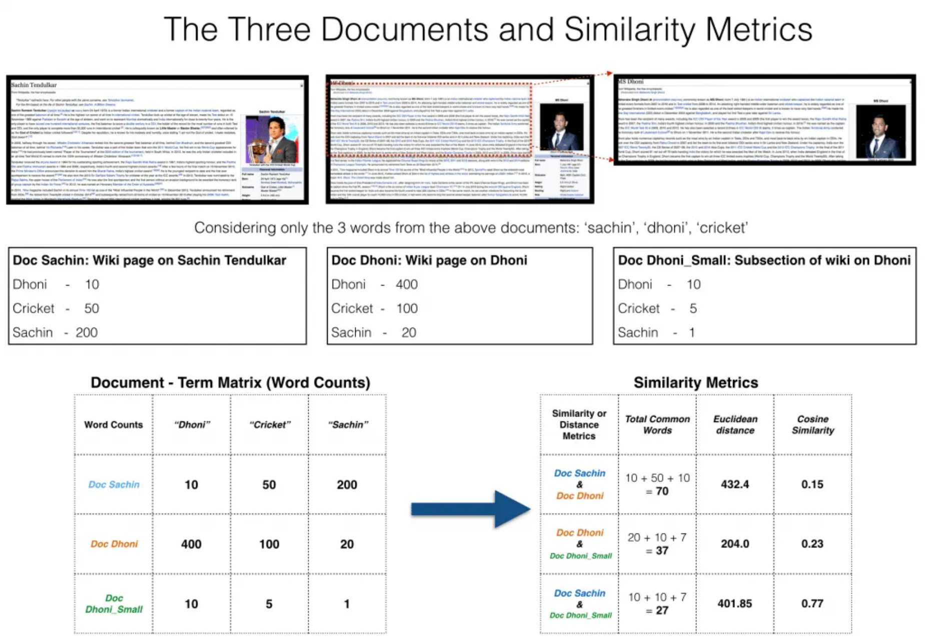
Matplotlib Pyplot (https://www.machinelearningplus.com/plots/matplotl pyplot/)

Python Boxplot (https://www.machinelearningplus.com/plots/python-boxplot/)

Bar Plot in Python (https://www.machinelearningplus.com/plots/bar-plot-in-python/)

data.table in R – The Complete Beginners Guide (https://www.machinelearningplus.com/data-manipulation/datatable-in-r-complete-guide/)

Let's suppose you have 3 documents based on a couple of star cricket players – Sachin Tendulkar and Dhoni. Two of the documents (A) and (B) are from the wikipedia pages on the respective players and the third document (C) is a smaller snippet from Dhoni's wikipedia page.


The Three Documents and Similarity Metrics

Considering only the 3 words from the above documents: 'sachin', 'dhoni', 'cricket'

Doc Sachin: Wiki page on Sachin Tendulkar	Doc Dhoni: Wiki page on Dhoni	Doc Dhoni_Small: Subsection of wiki on Dhoni
Dhoni - 10	Dhoni - 400	Dhoni - 10
Cricket - 50	Cricket - 100	Cricket - 5
Sachin - 200	Sachin - 20	Sachin - 1

Document - Term Matrix (Word Counts)

Word Counts	"Dhoni"	"Cricket"	"Sachin"
Doc Sachin	10	50	200
Doc Dhoni	400	100	20
Doc Dhoni_Small	10	5	1

Similarity Metrics

Similarity or Distance Metrics	Total Common Words	Euclidean distance	Cosine Similarity
Doc Sachin & Doc Dhoni	10 + 50 + 10 = 70	432.4	0.15
Doc Dhoni & Doc Dhoni_Small	20 + 10 + 7 = 37	204.0	0.23
Doc Sachin & Doc Dhoni_Small	10 + 10 + 7 = 27	401.85	0.77

[https://www.machinelearningplus.com/wp-content/uploads/2018/10/the_three_documents.png]

[https://www.machinelearningplus.com/wp-content/uploads/2018/10/the_three_documents.png] The Three Documents

As you can see, all three documents are connected by a common theme – the game of Cricket.

Our objective is to quantitatively estimate the similarity between the documents.

For ease of understanding, let's consider only the top 3 common words between the documents: 'Dhoni', 'Sachin' and 'Cricket'.

You would expect `Doc A` and `Doc C`, that is the two documents on Dhoni would have a higher similarity over `Doc A` and `Doc B`, because, `Doc C` is essentially a snippet from `Doc A` itself.

However, if we go by the number of common words, the two larger documents will have the most common words and therefore will be judged as most similar, which is exactly what we want to avoid.

The results would be more congruent when we use the cosine similarity score to assess the similarity.

Let me explain.

Let's project the documents in a 3-dimensional space, where each dimension is a frequency count of either: 'Sachin', 'Dhoni' or 'Cricket'. When plotted on this space, the 3 documents would appear something like this.

Augmented Dickey Fuller Test (ADF Test) – Must Read Guide (https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/)

KPSS Test for Stationarity (https://www.machinelearningplus.com/time-series/kpss-test-for-stationarity/)

101 R data.table Exercises (https://www.machinelearningplus.com/data-manipulation/101-r-data-table-exercises/)

P-Value – Understanding from Scratch (https://www.machinelearningplus.com/statistics/p-value/)

Top Posts & Pages

ARIMA Model - Complete Guide to Time Series Forecasting in Python (https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/)

Parallel Processing in Python - A Practical Guide with Examples (https://www.machinelearningplus.com/python/paralle processing-python/)

Machine Learning Better Explained! (https://www.machinelearningplus.com/)

Time Series Analysis in Python - A Comprehensive Guide with Examples (https://www.machinelearningplus.com/time-series/time-series-analysis-python/)

Cosine Similarity - Understanding the math and how it works (with python codes) (https://www.machinelearningplus.com/nlp/cosine-similarity/)

Topic Modeling with Gensim (Python) (https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/)

Top 50 matplotlib Visualizations - The Master Plots (with full python code) (https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/)

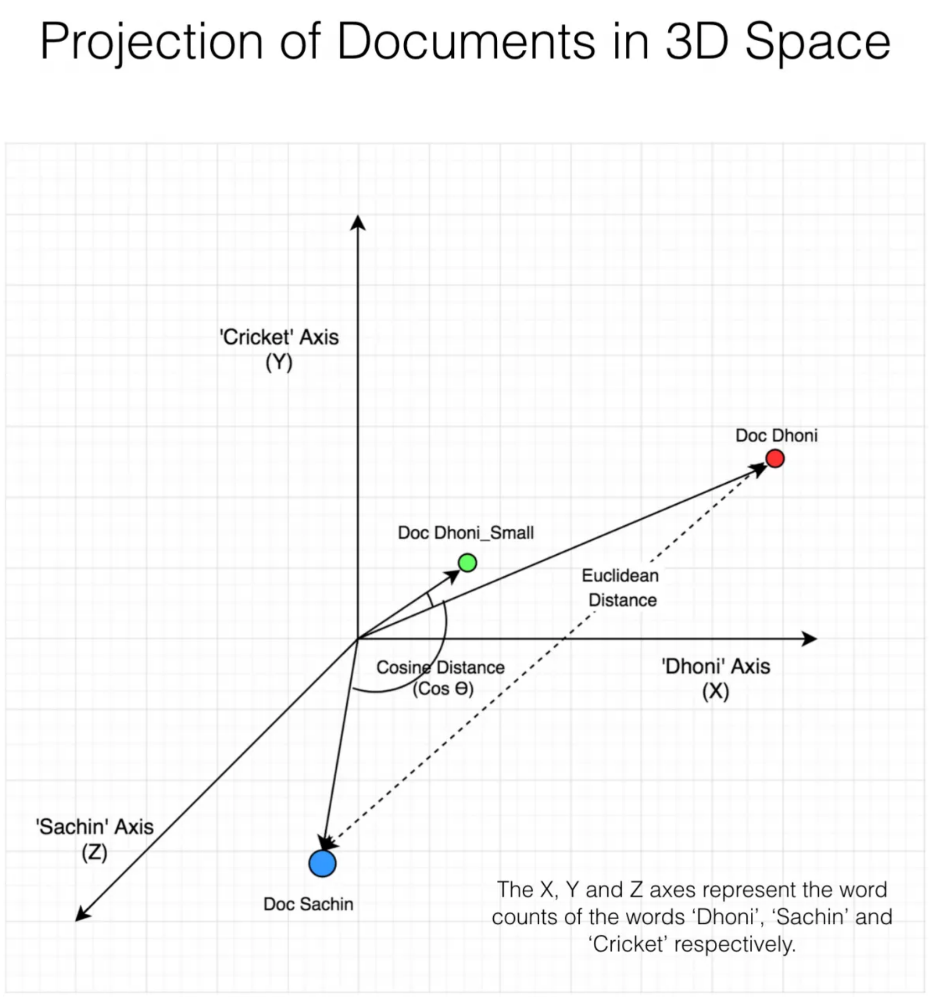
101 Pandas Exercises for Data Analysis (https://www.machinelearningplus.com/python/101-pandas-exercises-python/)

Matplotlib Histogram - How to Visualize Distributions in Python (https://www.machinelearningplus.com/plots/matplotl histogram-python-examples/)

Vector Autoregression (VAR) - Comprehensive Guide with Examples in Python (https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/)

Tags

Classification (https://www.machinelearningplus.com/tag/class data.table (https://www.machinelearningplus.com/tag/data-table/) Data Manipulation (https://www.machinelearningplus.com/ta


Projection of Documents in 3D Space
'Cricket' Axis (Y)
Doc Dhoni
Doc Dhoni_Small
Euclidean Distance
Cosine Distance (Cos ϴ)
'Dhoni' Axis (X)
'Sachin' Axis (Z)
Doc Sachin
The X, Y and Z axes represent the word counts of the words 'Dhoni', 'Sachin' and 'Cricket' respectively.

(https://www.machinelearningplus.com/wp-content/uploads/2018/10/3d_projection.png)

(https://www.machinelearningplus.com/wp-content/uploads/2018/10/3d_projection.png) 3d Projection

As you can see, `Doc Dhoni_Small` and the main `Doc Dhoni` are oriented closer together in 3-D space, even though they are far apart by magnitiude.

It turns out, the closer the documents are by angle, the higher is the Cosine Similarity (Cos theta).

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

(https://www.machinelearningplus.com/wp-content/uploads/2018/10/Cosine-Similarity-Formula-1.png)

(https://www.machinelearningplus.com/wp-content/uploads/2018/10/Cosine-Similarity-Formula-1.png) Cosine Similarity Formula

As you include more words from the document, it's harder to visualize a higher dimensional space. But you can directly compute the cosine similarity using this math formula.

Enough with the theory. Let's compute the cosine similarity with Python's scikit learn.

4. How to Compute Cosine Similarity in Python?

We have the following 3 texts:

manipulation/) Debugging (https://www.machinelearningplus.com/tag/debugging/) Doc2Vec (https://www.machinelearningplus.com/tag/doc2vec/) Evaluation Metrics (https://www.machinelearningplus.com/tag/evaluation-metrics/) FastText (https://www.machinelearningplus.com/tag/fasttext/) Feature Selection (https://www.machinelearningplus.com/tag/feature-selection/) Gensim (https://www.machinelearningplus.com/tag/gens HuggingFace (https://www.machinelearningplus.com/tag/huggingface/) Julia (https://www.machinelearningplus.com/tag Julia Packages (https://www.machinelearningplus.com/tag/julia-packages/) LDA (https://www.machinelearningplus.com/tag/lda/) Lemmatization (https://www.machinelearningplus.com/tag/lemmatization/) Linear Regression (https://www.machinelearningplus.com/tag/linear-regression/) Logistic (https://www.machinelearningplus.com/tag/logistic/) Loop (https://www.machinelearningplus.com/tag/loop/) LSI (https://www.machinelearningplus.com/tag/lsi/) Machine Learning (https://www.machinelearningplus.com/tag/m learning/) Matplotlib (https://www.machinelearningplus.co NLP (https://www.machinelearningplus.com/ta NLTK (https://www.machinelearningplus.com/tag/nltk/) Numpy (https://www.machinelearningplus.com/tag/numpy/) P-Value (https://www.machinelearningplus.com/tag/p-value/) Pandas (https://www.machinelearningplus.com/tag/pandas/) Phraser (https://www.machinelearningplus.com/tag/phraser/) plots (https://www.machinelearningplus.com/tag/plots Practice Exercise (https://www.machinelearningplus.com/tag/practice-exercise/) Python (https://www.machinelearni R (https://www.machinelearningplus.com/ Regex (https://www.machinelearningplus.com/tag/regex/) Regression (https://www.machinelearningplus.com/tag/regressio Residual Analysis (https://www.machinelearningplus.com/tag/residual-analysis/) Scikit Learn (https://www.machinelearningplus.com/tag/scikit-learn/) Significance Tests (https://www.machinelearningplus.com/tag/significan tests/) Soft Cosine Similarity (https://www.machinelearningplus.com/tag/soft-cosine-similarity/) spaCy (https://www.machinelearningplus.com/tag/spac Stationarity (https://www.machinelearningplus.com/tag/stationari TextBlob

/

Doc Trump (A) : Mr. Trump became president after winning the political election. Though he lost the support of some republican friends, Trump is friends with President Putin.

Doc Trump Election (B) : President Trump says Putin had no political interference is the election outcome. He says it was a witchhunt by political parties. He claimed President Putin is a friend who had nothing to do with the election.

Doc Putin (C) : Post elections, Vladimir Putin became President of Russia. President Putin had served as the Prime Minister earlier in his political career.

Since, Doc B has more in common with Doc A than with Doc C, I would expect the Cosine between A and B to be larger than (C and B).

```
 # Define the documents
doc_trump = "Mr. Trump became president after winning the political election. Though he lost th

doc_election = "President Trump says Putin had no political interference is the election outcom

doc_putin = "Post elections, Vladimir Putin became President of Russia. President Putin had ser

documents = [doc_trump, doc_election, doc_putin]
```

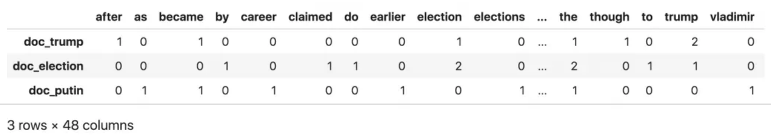
To compute the cosine similarity, you need the word count of the words in each document. The `CountVectorizer` or the `TfidfVectorizer` from scikit learn lets us compute this. The output of this comes as a `sparse_matrix`.

On this, am optionally converting it to a pandas dataframe to see the word frequencies in a tabular format.

```
 # Scikit Learn
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Create the Document Term Matrix
count_vectorizer = CountVectorizer(stop_words='english')
count_vectorizer = CountVectorizer()
sparse_matrix = count_vectorizer.fit_transform(documents)

# OPTIONAL: Convert Sparse Matrix to Pandas Dataframe if you want to see the word frequencies.
doc_term_matrix = sparse_matrix.todense()
df = pd.DataFrame(doc_term_matrix,
                  columns=count_vectorizer.get_feature_names(),
                  index=['doc_trump', 'doc_election', 'doc_putin'])
df
```



	after	as	became	by	career	claimed	do	earlier	election	elections	...	the	though	to	trump	vladimir
doc_trump	1	0	1	0	0	0	0	0	1	0	...	1	1	0	2	0
doc_election	0	0	0	1	0	1	1	0	2	0	...	2	0	1	1	0
doc_putin	0	1	1	0	1	0	0	1	0	1	...	1	0	0	0	1

3 rows × 48 columns

(https://www.machinelearningplus.com/tag/textblob/) TextSummarization (https://www.machinelearningplus.com/tag/textsummarization/) TFIDF (https://www.machinelearningplus.com/tag/tfidf/) Time Series (https://www.machinelearningplus.com/tag/ti series/) Topic Modeling (https://www.machinelearningplus.com/tag/topic-modeling/) Visualization (https://www.machinelearningplus.com/tag/visualizat Word2Vec (https://www.machinelearningplus.com/tag/word2vec/)

(https://www.machinelearningplus.com/wp-content/uploads/2018/10/doc-term-matrix.png)

(https://www.machinelearningplus.com/wp-content/uploads/2018/10/doc-term-matrix.png) Doc-Term Matrix

Even better, I could have used the TfidfVectorizer() instead of CountVectorizer(), because it would have downweighted words that occur frequently across docuemnts.

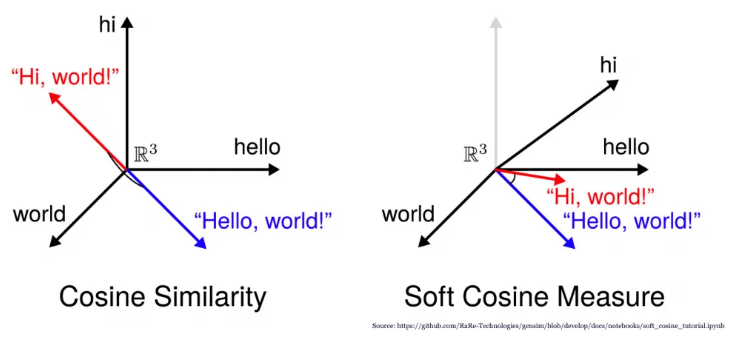
Then, use cosine_similarity() to get the final output. It can take the document term matri as a pandas dataframe as well as a sparse matrix as inputs.

```
 # Compute Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(df, df))
#> [[ 1.          0.48927489  0.37139068]
#>  [ 0.48927489  1.          0.38829014]
#>  [ 0.37139068  0.38829014  1.        ]]
```

5. Soft Cosine Similarity

Suppose if you have another set of documents on a completely different topic, say 'food', you want a similarity metric that gives higher scores for documents belonging to the same topic and lower scores when comparing docs from different topics.

In such case, we need to consider the semantic meaning should be considered. That is, words similar in meaning should be treated as similar. For Example, 'President' vs 'Prime minister', 'Food' vs 'Dish', 'Hi' vs 'Hello' should be considered similar. For this, converting the words into respective word vectors, and then, computing the similarities can address this problem.



(https://www.machinelearningplus.com/wp-content/uploads/2018/10/soft-cosine.png)

(https://www.machinelearningplus.com/wp-content/uploads/2018/10/soft-cosine.png)
Soft Cosines

Let's define 3 additional documents on food items.

```
# Define the documents
doc_soup = "Soup is a primarily liquid food, generally served warm or hot (but may be cool or c

doc_noodles = "Noodles are a staple food in many cultures. They are made from unleavened dough

doc_dosa = "Dosa is a type of pancake from the Indian subcontinent, made from a fermented batte

documents = [doc_trump, doc_election, doc_putin, doc_soup, doc_noodles, doc_dosa]
```

To get the word vectors, you need a word embedding model. Let's download the FastText model using gensim's downloader api.

```
import gensim
# upgrade gensim if you can't import softcossim
from gensim.matutils import softcossim
from gensim import corpora
import gensim.downloader as api
from gensim.utils import simple_preprocess
print(gensim.__version__)
#> '3.6.0'

# Download the FastText model
fasttext_model300 = api.load('fasttext-wiki-news-subwords-300')
```

To compute soft cosines, you need the dictionary (a map of word to unique id), the corpus (word counts) for each sentence and the similarity matrix.

```
# Prepare a dictionary and a corpus.
dictionary = corpora.Dictionary([simple_preprocess(doc) for doc in documents])

# Prepare the similarity matrix
similarity_matrix = fasttext_model300.similarity_matrix(dictionary, tfidf=None, threshold=0.0,

# Convert the sentences into bag-of-words vectors.
sent_1 = dictionary.doc2bow(simple_preprocess(doc_trump))
sent_2 = dictionary.doc2bow(simple_preprocess(doc_election))
sent_3 = dictionary.doc2bow(simple_preprocess(doc_putin))
sent_4 = dictionary.doc2bow(simple_preprocess(doc_soup))
sent_5 = dictionary.doc2bow(simple_preprocess(doc_noodles))
sent_6 = dictionary.doc2bow(simple_preprocess(doc_dosa))

sentences = [sent_1, sent_2, sent_3, sent_4, sent_5, sent_6]
```

If you want the soft cosine similarity of 2 documents, you can just call the softcossim() function

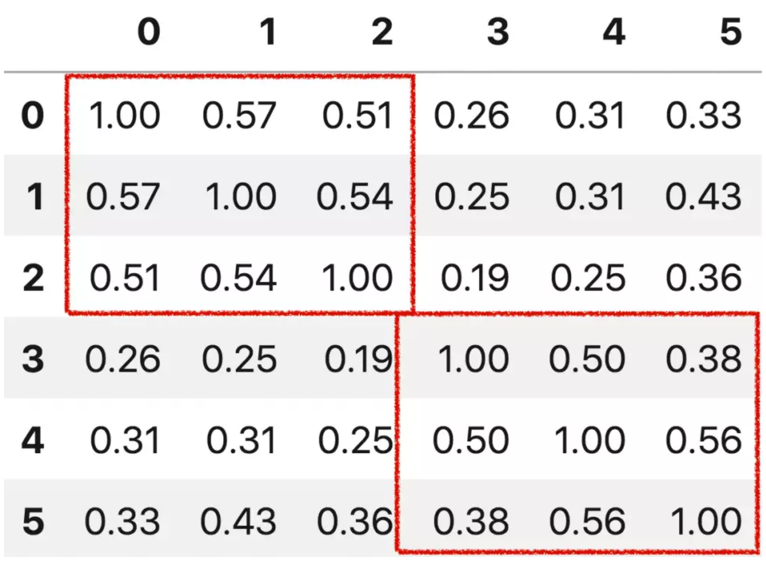
```
# Compute soft cosine similarity
print(softcossim(sent_1, sent_2, similarity_matrix))
#> 0.567228632589
```

But, I want to compare the soft cosines for all documents against each other. So, create the soft cosine similarity matrix.

```
import numpy as np
import pandas as pd

def create_soft_cossim_matrix(sentences):
    len_array = np.arange(len(sentences))
    xx, yy = np.meshgrid(len_array, len_array)
    cossim_mat = pd.DataFrame([[round(softcossim(sentences[i],sentences[j], similarity_matrix)
    return cossim_mat

soft_cosine_similarity_matrix(sentences)
```



	0	1	2	3	4	5
0	1.00	0.57	0.51	0.26	0.31	0.33
1	0.57	1.00	0.54	0.25	0.31	0.43
2	0.51	0.54	1.00	0.19	0.25	0.36
3	0.26	0.25	0.19	1.00	0.50	0.38
4	0.31	0.31	0.25	0.50	1.00	0.56
5	0.33	0.43	0.36	0.38	0.56	1.00

[https://www.machinelearningplus.com/wp-content/uploads/2018/10/soft-cosine-similarity-matrix.png]

[https://www.machinelearningplus.com/wp-content/uploads/2018/10/soft-cosine-similarity-matrix.png] Soft cosine similarity matrix

As one might expect, the similarity scores amongst similar documents are higher (see the red boxes).

6. Conclusion

Now you should clearly understand the math behind the computation of cosine similarity and how it is advantageous over magnitude based metrics like Euclidean distance.

Soft cosines can be a great feature if you want to use a similarity metric that can help in clustering or classification of documents.

If you want to dig in further into natural language processing, the gensim tutorial (https://www.machinelearningplus.com/nlp/gensim-tutorial/) is highly recommended.

[(/#facebook)] [(/#twitter)]
[(/#whatsapp)] [(/#linkedin)] [(/#reddit)]
[(/#google_bookmarks)]

[(/#google_gmail)](/#google_gmail)

+ [(https://www.addtoany.com/share#url=https%3A%2F%2Fwww.machin
similarity%2F&title=Cosine%20Similarity%20%E2%80%93%20Understandin

ALSO ON **MACHINELEARNINGPLUS.COM**

2 years ago • 23 comments
A Comprehensive Guide with …

9 months ago • 2 comments
Principal Component Analysis …

9 r
A D (/

What do you think?
57 Responses

Upvote Love

Comments **Community** **Privacy Policy** 1 **Login**

Recommend 2 Tweet Share **Sort by Newest**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Cefas Garcia Pereira • 18 days ago
Is there anyway to make the api.load('fasttext-wiki-news-subwords-300') faster?
• Reply • Share ›

Suhas Gajendra • 8 months ago
Thanks for the clear explanation.

Am facing the below error,
DeprecationWarning: Call to deprecated `similarity_matrix` (Method will be removed in 4.0.0, use gensim.models.keyedvectors.WordEmbeddingSimilarityI
instead). [ipykernel_launcher.py:1]
• Reply • Share ›

Ihor Konovalenko • a year ago
Article is very interesting. But, author wrote

> You would expect Doc A and Doc C, that is the two documents on Dhoni would have a higher similarity over Doc A and Doc B, because, Doc C is essentially a snippet from Doc A itself.

May be it would be more correct to say "You would expect Doc **B** and Doc C, that is the two documents on Dhoni... ...because, Doc C is essentially a snippet from Doc **B** itself".

It based on author's previous explanation that

> Two of the documents (A) and (B) are from the wikipedia pages on the respective players and the third document (C) is a smaller snippet from Dhoni's wikipedia page.

1 • Reply • Share ›

Laveena • a year ago

Very impressive explanation, can you please explain how you have calculated "total_common_words" in Similarity Metrics? It doesn't actually matches my calculation for e.g. Doc Sachin & Doc Dhoni = (Dhoni)10 + (Cricket)50 + (Sachin)20

^ | ⌄ • Reply • Share ›

Saikam Rama Krishna Reddy • a year ago

can you explain how similarity matrix derived from word counts matrix?

^ | ⌄ • Reply • Share ›

Sandeep Hooda • a year ago

Your explanation style is very impressive. You might need few corrections in documents I guess. For example consider this "You would expect Doc A and Doc C, that is the two documents on Dhoni would have a higher similarity over Doc A and Doc B". But in you example you have taken doc B and C on Dhoni and doc A on sachin. Similarly in Trump and putin example "Since, Doc B has more in common with Doc A than with Doc C, I would expect the Cosine between A and B to be larger " Cosine should be small for A and B , right?

^ | ⌄ • Reply • Share ›

Luca Nannini • a year ago

```
---------------------------------------------------------
------------------
MemoryError Traceback (most recent call last)
<ipython-input-12-e9b5f1dc9c0b> in <module>
----> 1 print(softcossim(sent_1, sent_2,
similarity_matrix))

~\Anaconda3\lib\site-packages\gensim\matutils.py in
softcossim(vec1, vec2, similarity_matrix)
814 vec1 = np.array([vec1[i] if i in vec1 else 0 for i in
word_indices], dtype=dtype)
815 vec2 = np.array([vec2[i] if i in vec2 else 0 for i in
word_indices], dtype=dtype)
--> 816 dense_matrix = similarity_matrix[[[i] for i in
word_indices], word_indices].todense()
817 vec1len = vec1.T.dot(dense_matrix).dot(vec1)[0, 0]
818 vec2len = vec2.T.dot(dense_matrix).dot(vec2)[0, 0]

~\Anaconda3\lib\site-packages\scipy\sparse\csc.py in
__getitem__(self, key)
168 # Things that return a sequence of values.
169 else:
--> 170 return self.T[col, row]
171
172 def nonzero(self):

~\Anaconda3\lib\site-packages\scipy\sparse\csr.py in
__getitem__(self, key)
348 csr_sample_values(self.shape[0], self.shape[1],
```