**SYNAPSE TASK 1:-**Text Similarity with Cosine Similarity and Word Embeddings.

There are different sections in which the task of finding similarity between various texts/documents is divided:-

* Data Pre-processing.
* Feature Extraction .
* Vector Similarity.
* Decision Function.

**1)Data Pre-processing:-**

It is the first task that is performed on our documents.Over here,we remove all the unnecessary stuffs from the text like:

* Punctuations or special characters.
* Any HTML tags.
* Stop words (like of, how, on, what etc.).
* Any non-ascii values if present.

**Lemmatization:**

* Lemmatizing our text means converting a word to its basic/root form.
* for eg. words like lives , lived , living are converted to live.

To further ensure uniformity in the data, the entire text document is converted to lower case.

For eg.

Input -"Hello, my name is Manan.I along with my family are living in Mumbai, which is the Financial Capital of India."

after pre-processing might look somewhat like this:

Output-"hello name manan family live mumbai financial capital india"

**2) Feature Extraction:-**

In this step,we convert our pre-processed data into a numerical vector using either of the various Word Embeddings.

There are basically two types of word embedding techinques :-

* Frequency based Embeddings.
* Prediction based Embeddings.
* Here, we will just talk about Frequency based Embeddings.

**Frequency based Embeddings:**

There are several types of vectorizations that fall under this category. We will be discussing only about 2 of its types over here. We can use either of the methods for frequency extraction.

i)Count Vectorization:

* Let's say we have a corpus consisting of D documents and we find a total of N unique words out of those. These N words will be part of a dictionary .Therefore,we will have a matrix of DxN.
* The row labels will consist of different document names in the corpus, whereas each column header will be a unique word identified in the corpus.
* Then each cell/matrix position will be filled by a number according to the frequency of that word in that particular document.
* For. eg, Let us understand this using a simple example.

D1: He is a lazy boy. She is a lazy girl.

D2: Both are lazy people.

The dictionary created will be a list of unique words in the corpus

= [‘He’,’She’,’lazy’,’boy’,’girl’,’Both’,’people’].

Here, D=2, N=7

The count matrix M of size 2 X 7 will be represented as –

He She lazy boy girl Both people

D1 1 1 2 1 0 0 0

D2 0 0 1 0 1 1 1

* Here,He occurs once in document D1,hence cell 1 has value 1. Similarly,all other cells are filled by corresponding frequency values of different words.
* Generally, if the documents are large and have, let’s say millions of unique words,not all the words are taken into formation of the matrix. A few, say 10000-20000 are taken based on their frequencies.

ii)TF-IDF Vectorization:

* TF-IDF vectorization is a bit different form count vectorization as it takes into account the entire corpus.
* This method focusses more on important words that are document specific rather than the common words which appear to be in many documents.
* It has 2 parts.TF and IDF.TF stands for the Term Frequency and IDF stands for Inverted Document Frequency.
* TF=n/Dn ,where n is number of times the word occurs in the document and Dn is the total number of words in document.
* IDF=log(N/ND),where N is the total number of documents whereas ND stands for number of documents where the particular word appears.
* For eg.

Let D1-'This is Lionel Messi.'

and D2-'This is Cristiano Ronaldo.'

D1 D2

word count word count

This 1 This 1

is 1 is 1

Lionel 1 Cristiano 1

Messi 1 Ronaldo 1

TF for 'This'=1/4 ,while TF for 'Messi'=1/4.

IDF for 'This'=log (2/2) =0, while IDF for 'Messi'=log (2/1) =0.303

Therefore, TF-IDF for 'This'= (1/4)\*0=0

and for 'Messi'= (1/4)\*0.303=0.075.

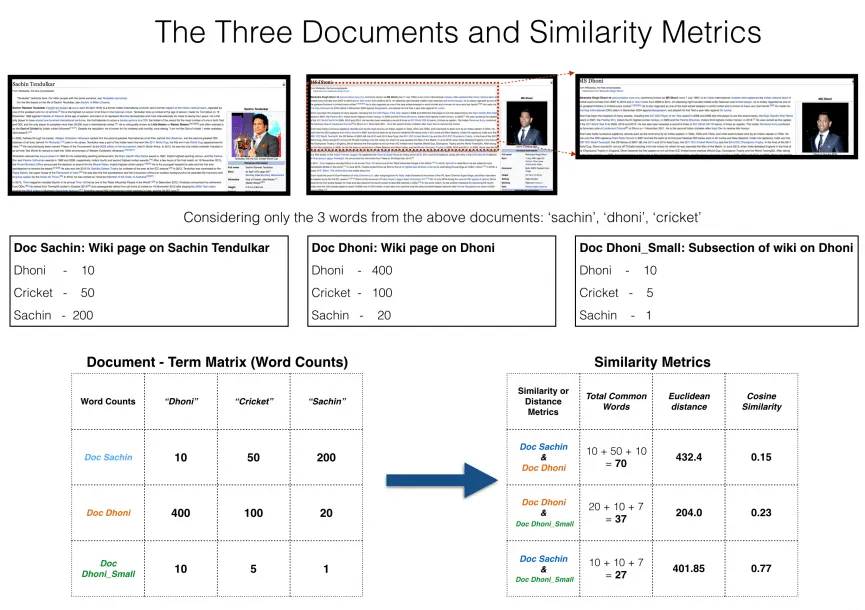
* Hence, we see that for a more important/document specific words, TF-IDF score is more.

**3)Vector Similarity:-**

Finally, after vectorizing the corpus, we need to compare these vectorized documents in order to find the similarity score or the amount of similarity between them. Out of all, one of the techniques is Cosine Similarity which we are going to use here.

**Cosine Similarity:**

* It is one of the techniques used to determine similarity between two documents.
* In this method, we measure the cosine of the angle between the vectors formed by the 2 vectorized documents.
* 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.

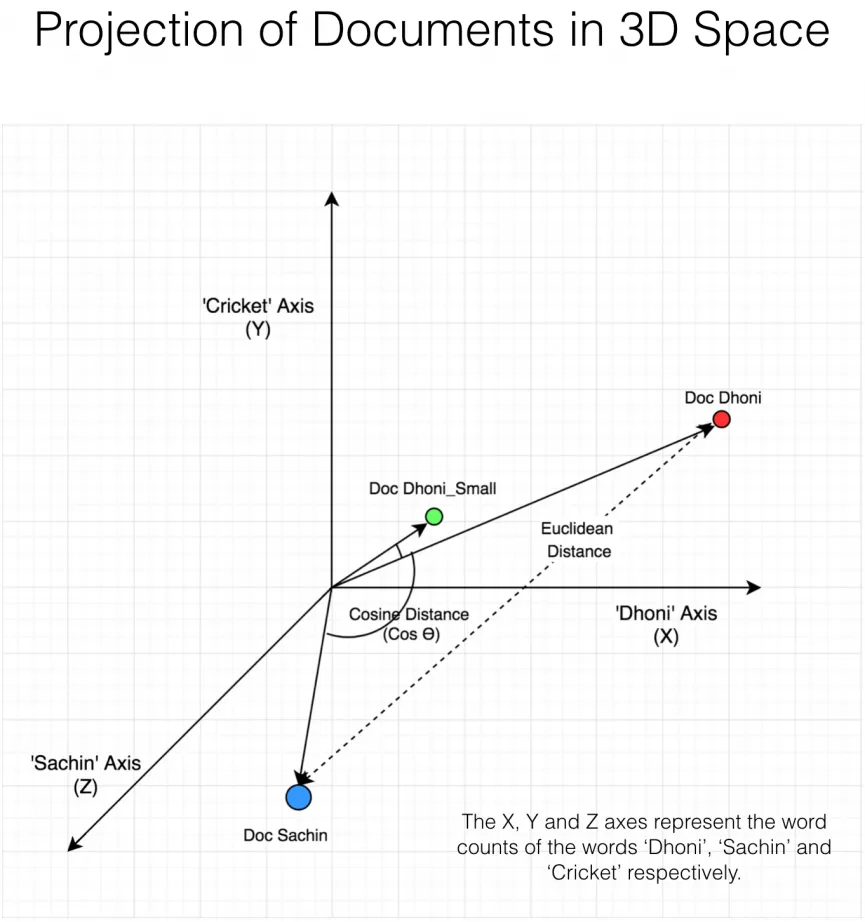


All the three documents are connected by a common theme – the game of Cricket. We have considered the top 3 common words between the documents: ‘Dhoni’, ‘Sachin’ and ‘Cricket’.

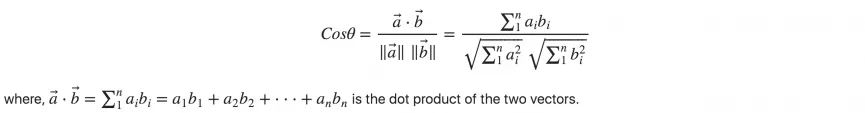
If we go by the principle of counting of the number of common words, the first two documents would have higher number of common words than the third one due to their large lengths. Hence, documents 1 and 2 would be judged more similar than documents 2 and 3,which would be wrong and it’s the thing that we want to avoid.

Here is the scenario where Cosine Similarity pitches in.

We 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.



Cosine Similarity Formula is:



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 magnitude.

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

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.

**4)Decision Function:-**

* This is the final step where we set a threshold value as to finally conclude whether the similarity score obtained from above is apt to classify two documents as similar.
* For eg.,if the threshold value is 0.8 and we have a similarity score of 0.75,then we consider the two documents as not to be similar whereas if its above 0.85,they are considered similar.