Department of CSE

Mini Project Report

Course Name: Machine Learning

Course code: CSE475

Section: 03

Date of Submission: 5 September 2021

Submitted to:

Dr. Md Samiullah

Group Member Information

Group Member Name	Student ID
Sumona Yasmin	2018-2-60-062
Kasfia Saif	2018-2-60-001
Nazia Afrin	2018-2-60-023

1 Introduction:

Cosine Similarity:

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis. 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. When we try to match or find the similarity between documents there can a problem which 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. In other words, Cosine similarity is a metric used to determine how similar the documents are irrespective of their size.

2 Background:

a) How Cosine Similarity works:

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 we are talking about are arrays containing the word counts of two documents. A document can be represented by thousands of attributes, each recording the frequency of a particular word (such as a keyword) or phrase in the document. Thus, each document can be addressed as term frequency vector. Cosine similarity is a measure of similarity that can be used to compare documents or, say, give a ranking of documents with respect to a given vector of query words. Let x and y be two vectors for comparison.

$$\text{sim}(x, y) = x * y /\|x\| * \|y\|$$

which means both the vector x and y will be divided by their respective Euclidian norm of vectors.

b) Feature extraction:

In order to understand the process of transformation of text to numeric values we must shed some light to the concept of vectorizing of documents or text also known as feature extraction. Each word of the document is represented by a numeric value. The transformation of this text to numeric is called word vectorization or feature extraction. Vectorization is essential because we need vectors representation of words to work with cosine similarity. For this project we have user two types of vectorizes implemented from scratch:

i. ***Count Vectorizer***

ii. ***Tf-Idf Vectorizer***.

i. Count Vectorizer:

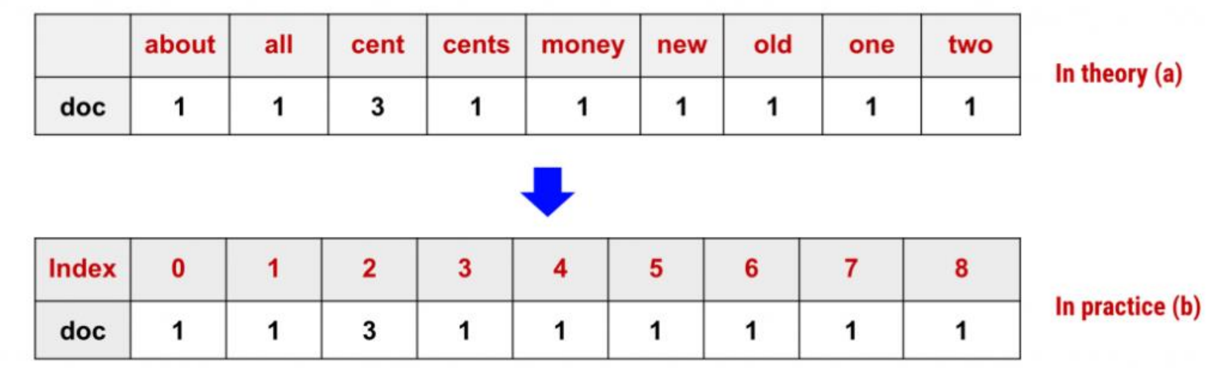
Count Vectorizer if a form of vectorization scheme that takes the count value of each occurring word and transforms them into vectors. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. Count Vectorizer is used to convert a collection of text documents to a vector of term/token counts.

The following example is simple enough to grasp the concept of count vectorizer and how it represents textual data:

Example input document:

doc= ["One Cent, Two Cents, Old Cent, New Cent: All About Money"]

Representation in Count Vectorizer:



	about	all	cent	cents	money	new	old	one	two
doc	1	1	3	1	1	1	1	1	1

In theory (a)

Index	0	1	2	3	4	5	6	7	8
doc	1	1	3	1	1	1	1	1	1

In practice (b)

Figure 1:Representation of words in count vectorizer

ii. Tf-IDF Vectorizer:

TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. **TF-IDF (term frequency-inverse document frequency)** is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. TF-IDF was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word.

The **term frequency** of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document.

The **inverse document frequency** of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

<u>The calculation of TF-IDF can be explained:</u>

First, we calculate the term-frequency of each word given the formula:

- *df(t) = occurrence of t in documents*

The IDF value if calculated by dividing the total number of documents with the term-frequency value:

- *idf(t) = N/df (N = Number of documents)*

In case of a large corpus, the IDF value Also becomes huge, to avoid the effect we take the log of idf:

- *idf(t) = log (N/ (df + 1))*

Finally, we get the value of TF-IDF by multiplying the term frequency and Inverse Term frequency together:

- *tf-idf (t, d) = tf(t, d) * log(N/(df + 1))*

<u>Our idea:</u>

As cosine similarity is an excellent way to find out the non-contextual similarity of two different documents, we wanted to use these vectorization techniques to learn and explore how words can be represented in numeric values to find out the similarity of documents using machine learning. And thusly we implemented the vector representations both count based and TI-IDF from scratch to get to know how the Machine learn about the words. The main focus of our implementation was to be able to identify and experiment with different datasets and to learn how similar given datasets are.

Implementation:

In order to implement this project, we have taken few steps these include:

a) **User input:** We uses python ‘sys’ module to take user input. We enabled user to input (.pdf) files to check weather these (.pdf) are similar or not based on structural similarity.

b) **Dataset Preprocess:** In order to convert the pdf file, we need to extract the text data from it and thusly we did some preprocess so we can work with the data.

c) **Vectorizations:** After extracting text data we took each document and vectorized then in two different ways. Count Vectorizer and Tf-IDF vectorizer. The vectorization process include:

1) ***Tokenization of each word***: We take each word and make them unique tokens, append them as corpus and remove duplicates.

2) ***Removing stop words****:* The stop words are very common in documents, and they are almost everywhere. The stop words are removed manually from the documents.

After the words are preprocessed, we handle them in two distinctive ways. Frist, we count each word and their frequency which is known as the Count vectorizer.
Or we take each word and follow the formula of Tf-IDF to make the vectorizations.

d) **Calculating cosine similarity:** Each vectorized representation of documents are then sent to the function to calculate the cosine similarity of texts. For that we follow the simple cosine similarity formula: $sim(x, y) = x \ast y /||x|| \ast ||y||$ where x and y are documents to be tested.

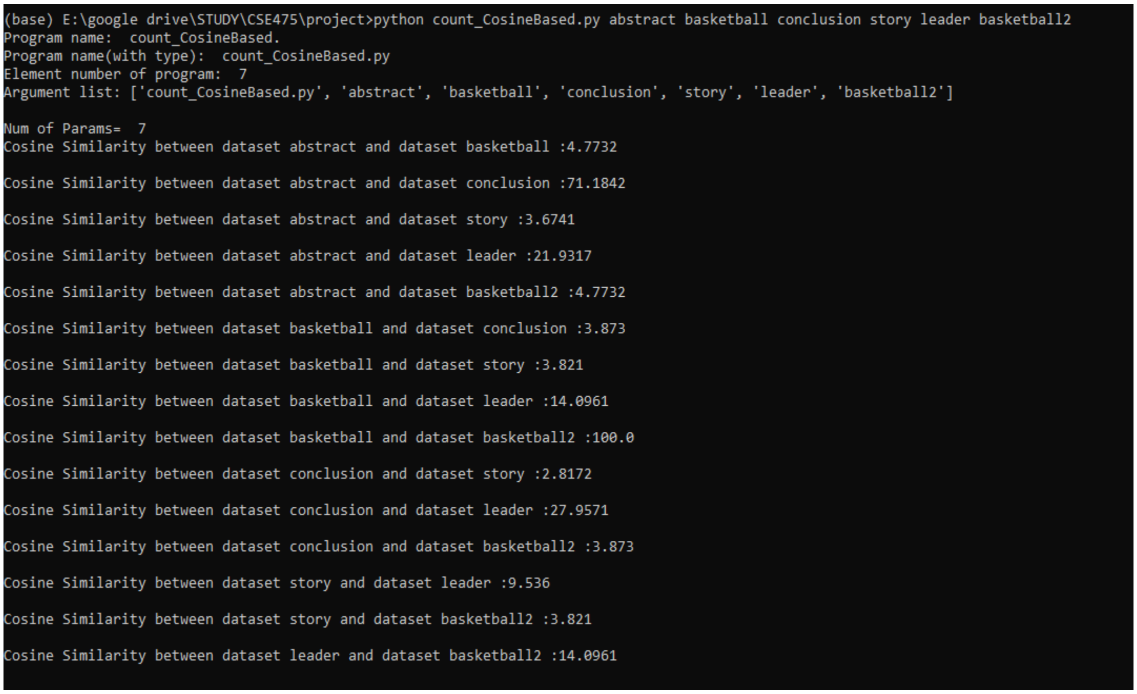
5) ***Experimentation*:**

For the experimentation part we took a bunch of datasets and ran through out code in order to find the cosine similarity using both the count vectorizer method and Tf-IDF method. In order to make sure out code is working we compared every dataset with each other meaning every dataset and their corresponding cosine value is being compared with the other dataset present there. If two datasets show structural similarity the cosine value tends to be closer to 100 and the less the similarity the value is less also.

In the first program the datasets are taken, pre-processed and vectorized according to count vectorizer and then are run through cosine similarity functions.

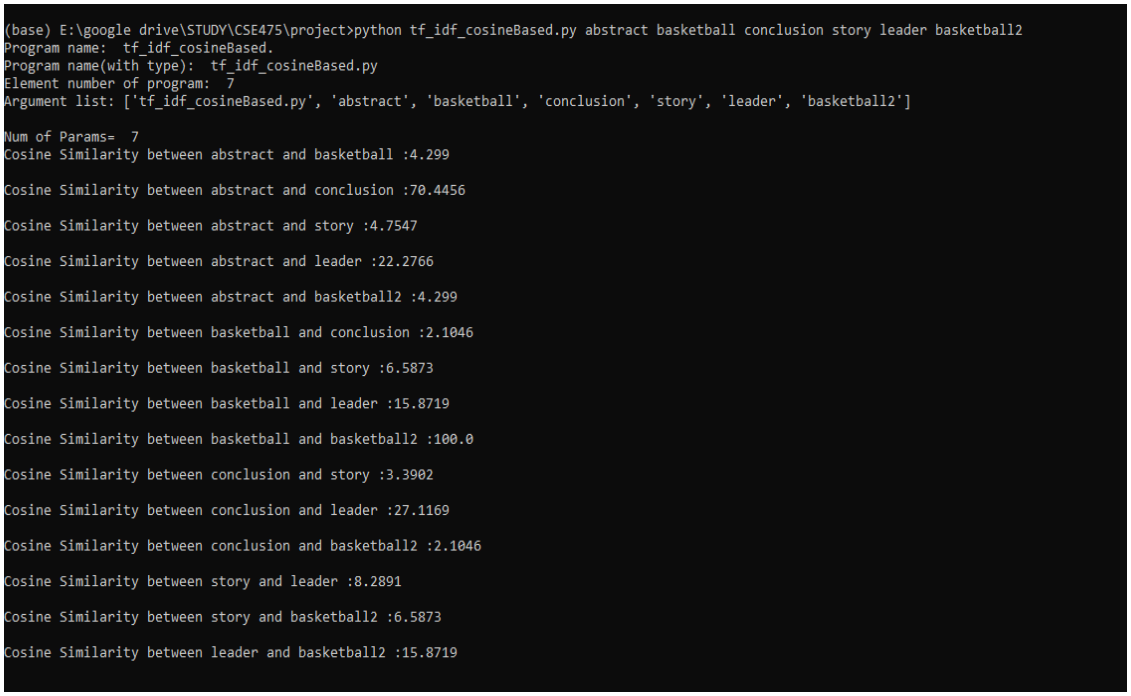
Similarly in the second program the dataset is taken and pre-processed and are transformed into numeric values using TF-IDF vectorizer, finally we check the similarity of all these datasets comparing with each other

The following figures show the output of Structural similarity among different datasets.



```
(base) E:\google drive\STUDY\CSE475\project>python count_CosineBased.py abstract basketball conclusion story leader basketball2
Program name:  count_CosineBased.
Program name(with type):  count_CosineBased.py
Element number of program:  7
Argument list: ['count_CosineBased.py', 'abstract', 'basketball', 'conclusion', 'story', 'leader', 'basketball2']

Num of Params=  7
Cosine Similarity between dataset abstract and dataset basketball :4.7732

Cosine Similarity between dataset abstract and dataset conclusion :71.1842

Cosine Similarity between dataset abstract and dataset story :3.6741

Cosine Similarity between dataset abstract and dataset leader :21.9317

Cosine Similarity between dataset abstract and dataset basketball2 :4.7732

Cosine Similarity between dataset basketball and dataset conclusion :3.873

Cosine Similarity between dataset basketball and dataset story :3.821

Cosine Similarity between dataset basketball and dataset leader :14.0961

Cosine Similarity between dataset basketball and dataset basketball2 :100.0

Cosine Similarity between dataset conclusion and dataset story :2.8172

Cosine Similarity between dataset conclusion and dataset leader :27.9571

Cosine Similarity between dataset conclusion and dataset basketball2 :3.873

Cosine Similarity between dataset story and dataset leader :9.536

Cosine Similarity between dataset story and dataset basketball2 :3.821

Cosine Similarity between dataset leader and dataset basketball2 :14.0961
```

Figure 2: Cosine similarity of documents using count vectorizer



```
(base) E:\google drive\STUDY\CSE475\project>python tf_idf_cosineBased.py abstract basketball conclusion story leader basketball2
Program name:  tf_idf_cosineBased.
Program name(with type):  tf_idf_cosineBased.py
Element number of program:  7
Argument list: ['tf_idf_cosineBased.py', 'abstract', 'basketball', 'conclusion', 'story', 'leader', 'basketball2']

Num of Params=  7
Cosine Similarity between abstract and basketball :4.299

Cosine Similarity between abstract and conclusion :70.4456

Cosine Similarity between abstract and story :4.7547

Cosine Similarity between abstract and leader :22.2766

Cosine Similarity between abstract and basketball2 :4.299

Cosine Similarity between basketball and conclusion :2.1046

Cosine Similarity between basketball and story :6.5873

Cosine Similarity between basketball and leader :15.8719

Cosine Similarity between basketball and basketball2 :100.0

Cosine Similarity between conclusion and story :3.3902

Cosine Similarity between conclusion and leader :27.1169

Cosine Similarity between conclusion and basketball2 :2.1046

Cosine Similarity between story and leader :8.2891

Cosine Similarity between story and basketball2 :6.5873

Cosine Similarity between leader and basketball2 :15.8719
```

Figure 3:Cosine similarity of documents using tf-idf vectorizer

6) **Conclusion:** Machine Learning has made a tremendous impact improving our lives. The possibility of things that can achieved by ML is unbound. Out project and experiments to how machine recognized actual text and find similarity on a structural level gave us immense opportunity to explore and understand the core concept of how a machine can understand human language and represent then as their own. This project has given us a chance to understand and experiment with document similarity with the help of algorithms and math. Upon this project and experiments we discovered vectorization of document text is the fundamental key part to work with textual data, moreover how a document is structurally similar to another document can depend on how we have vectorized the data or text. Throughout our experimentations we learnt that applying the Tf-IDF vectorization makes the cosine similarity a bit improved then the Count vectorizer.

Reference:

1) https://www.studytonight.com/post/scikitlearn-countvectorizer-in-nlp
2) https://kavita-ganesan.com/how-to-use-countvectorizer/#.YS0QzI4zZ9A
3) https://www.sciencedirect.com/topics/computer-science/cosine-similarity
4) https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e
5) https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a
6) https://monkeylearn.com/blog/what-is-tf-idf/