****

**Department of CSE**

**Mini Project Proposal**

**Course Name :** Machine Learning

**Course code :** CSE475

**Section** : 03

**Date of Submission : 3 August 2021**

**Submitted to :**

**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 :***

1. ***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.

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 are 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:

1. ***Count Vectorizer***
2. ***Tf-Idf Vectorizer****.*
3. **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 :

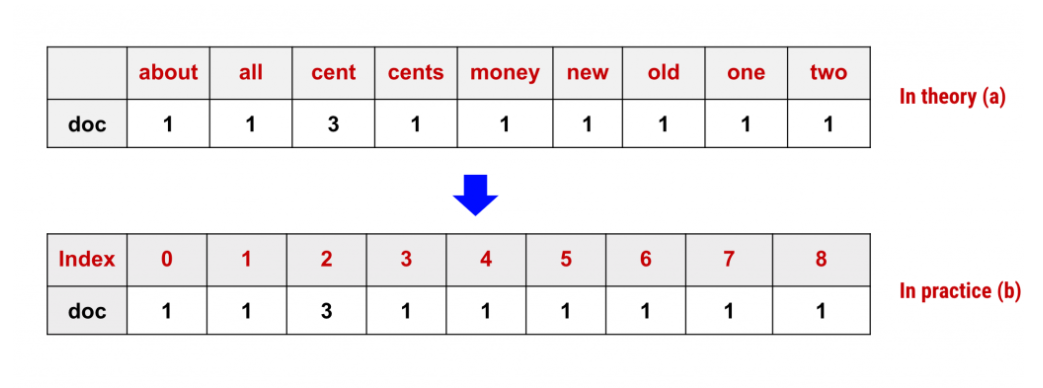


Figure 1:Representation of words in count vectorizer

1. **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.

**The calculation of TF-IDF can be explained :**

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))*

**Our Idea :**

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:

1. 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 on the basis of structural similarity.
2. 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.

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/>