ABSTRACT

We examined a similarity measure between text documents clustering. Data mining is a challenging field with more research and application areas. Text document clustering, which is a subset of data mining helps groups and organizes a large quantity of unstructured text documents into a small number of meaningful clusters. An algorithm which works better by calculating the degree of closeness of documents using their document matrix was used to query the terms/words in each document. We also determined whether a given set of text documents are similar/different to the other when these terms are queried. We found that, the ability to rank and approximate documents using matrix allows the use of Singular Value Decomposition (SVD) as an enhanced text data mining algorithm. Also, applying SVD to a matrix of a high dimension results in matrix of a lower dimension, to expose the relationships in the original matrix by ordering it from the most variant to the lowest.

Keywords: Data mining; similarity; term frequency; singular value decomposition; clustering.
1. INTRODUCTION

In the past few years, text mining has become one of the few areas that has seen an abrupt spurt in an attempt to derive meaningful information from text data. Text data mining which results in Information Retrieval or Extraction (IR or IE) appeared in the middle of the 1980s and involved a manual way of data extraction which made it very labor intensive [1]. The field of text data mining encompasses areas like natural language processing, statistics and machine learning. A survey proved that unstructured text data occupies about 80% of digital space where as 20% of digital space is made up of structured [2]. This implies that, most relevant information can be found in unstructured text data and thus need urgent attention. Obviously, the urgency that can be attached to the information hidden in the unstructured text data, is its extraction or retrieval [3], how it can be done, and how accurate the extracted information would be.

The challenge is to ensure that unstructured text-based data or natural language should convey exactly the same meaning. If a statement in an unstructured data set is transformed numerically before analyzing, it is feared that its actual meaning would be completely lost if not partially.

In order to extract information from any document, the text content of that document must be classified (assigning keywords to documents), remove stop words and clustered (grouping similar documents per the subject of discussion). How best can the classification and clustering be done in order to retrieve accurate information is now the problem [1,4]. The main objective of this paper is to use Singular Value Decomposition as similarity measure algorithm for text document clustering. The specific objectives are: To investigate and analyze the workings of the existing text data mining algorithms and identify its drawbacks in literature. To implement a text data mining algorithm on an unstructured text data that would be used to run a series of tests by computing Term Frequency – Inverse Term Frequency (TF-ITF).

To implement a text data mining algorithm on text data that will eliminate the drawbacks of the existing text data mining algorithms. Lastly, to use Cosine similarity measure to query for similar terms in a set of documents.

This paper is organized as follows. A formal literature review of text data mining is presented. Section 3, is made up of materials and methods used to retrieve the similarity of text data (documents). In section 4, the results and discussions of the experiments were presented. Section 5 ends the paper with a conclusion.

2 RELATED WORKS

Text mining is obviously affecting the world in a highly positive manner. Due to this, work done so far in the related field of study are discussed in this section. Several methods on document similarity measurements would also be reviewed to be able to refine the already existing methods in other to produce new viable solutions.

2.1 Theoretical Background of Data Mining

Text mining involves the intellectual means of discovering unknown patterns and connections among unrelated collection of digital data. It has helped in the sieving of useful patterns in various unstructured texts [5]. Based on the state-of-the-art of text data retrieval, several techniques have come together to enable smooth discovery of useful and interesting knowledge. Such techniques thus proposed IR and IE, conceptual structure, artificial intelligence, among others. Text mining has aided in large volume research among others. This leads to the creation of new related facts and information that can be used in an inspired manner. For instance, a new idea relating to financial stocks can be achieved by intellectually discovering new facts from old documents containing stock related texts. This could be achieved by trying to find some sort of similarities among the documents involved.

The authors [6] studied the sequence and process of text mining and discovery. These process were used to avoid the challenges of high dimensionality in clustering of words with ambiguities such as synonyms, noisy data, abbreviations, spelling mistakes in texts. The authors [7], studied Natural Language Processing (NLP), which looks at the analysis of human language so that computers can understand natural language the way humans do and retrieval of information in text data, this was used and applied on how to search for information. The authors of [8,9] mentioned some existing supervised learning algorithms, developed to deal with automatic text classification. This makes the user to choose one among the several available text mining
algorithms [5]. The authors discuss the significance of clustering as a useful technique that organizes large quantities of unordered text documents into smaller number of meaningful and coherent groups [10]. The author [11] studied the limitation of some classifier like Naive Bayes Classifier, Genetic Algorithm, and Decision Tree. Example; the Naive Bayes classifier uses the maximum a posteriori estimation. It assumes that the occurrence of each word in a document is conditionally independent of all other words in that document given its class. Although the Naive Bayes works well [11] it requires a large number of training documents for learning accurately. The authors [12], used documents representation to transform documents from the full text version to a document vector. Usually, one has a collection of documents which is represented by word-by-word document Matrix. This is used to reduce the complexity of the documents and make them easier to handle, using vector space model [13]. The idea of this work is to retrieve similarities in a set of documents like the well-known Jaccard, Euclidean k-means algorithm and Naive Bayes [14]. The algorithm aims to create a vector space model for the text data retrieval in this paper.

3 MATERIALS AND METHODS

3.1 Preliminaries

This section is devoted to discuss a brief step by step process used in determining the similarity measure of text data. Singular Value Decomposition (SVD) method is put to the test by using it to analyze three documents. Three separate experiments were carried out, each time by changing the content of the documents and also altering the term frequency computation method. The main term (word) frequency used are the raw and the $tf-idf$ frequency.

Three sections of the write-up of this paper were chosen at random and copied into three separate plain text files which were labeled doc1, doc2 and doc3. The length of each document was varied so that the number of words or terms in them were not necessarily the same to foster diversity. In the first two sets of the experiments, all the three documents were made to have the same content. It is not very effective to use the plain text documents as they are. The terms in each sentence of the documents must be tokenized. Tokenization is essential because it breaks each sentence into simpler components which is stored in an array. The array below shows the tokenized portion of doc1:

After this process then comes the removal of stop words. There may be lots of stop words and some other non-ascii characters which may hinder the process. Due to this, document preprocessing was inevitable. Stop words such ['the','that','what','with','this','is','from','your'] were all pruned from the documents. Added to the above, words which are not stop words but are less than 3 characters were also removed from the documents.

| Experiment | Documents       | Frequency Type | Status          |
|------------|----------------|----------------|-----------------|
| 1          | doc1, doc2 and doc3 | Raw term count | The same content |
| 2          | doc1, doc2 and doc3 | TF-IDF         | The same content |
| 3          | doc1, doc2 and doc3 | Raw term count | Varying content  |

Fig. 1. Python Screenshot for generating table 3 – 4
3.2 Frequency Measurements

3.2.1 Term frequency (tf)

The measure of the number of times a word appears in a given document is what is termed as its term frequency. Let a set of documents be defined as

\[ D = \{d_1, d_2, ..., d_n\} \] (1)

where \( D \) is the document array and \( d_1, d_2, ..., d_n \) are the distinct documents in the array. Each \( d_1, d_2, ..., d_n \) is a multiset in \( D \) which contains a collection of words which are not necessarily distinct.

Notationally, \( d_i = \{w_1, w_2, ..., w_m\} \). It is very important to note that each document in the array must be distinct \( d_1 \neq d_2 \neq \cdots \neq d_n \).

Let the term frequency of a given word in a document be \( tf(d, w) \), then the document can be represented in a vector form as

\[ d_i = \left( tf(d_i, w_1), tf(d_i, w_2), ..., tf(d_i, w_m) \right) \] (2)

where \( i = 1, 2, ..., n \). Let the raw count of words in the document be \( f_{d,w} \in f \), then the term frequency of a document in its naive state is given by

\[ tf(d, w) = f_{d,w} \] (3)

Words with higher term frequencies are presumed to have more importance to the document than those with low term frequency values. But this measure is misleading when you take stop words e.g. “the”, “that”, “is” etc into consideration.

3.2.2 Inverse document frequency (IDF)

The \( idf \) of a given word in a document, measures the level of information the word provides to the document. Where: \( D, w \) have their usual meanings

\[ \cdot N \] is the total number of documents in \( D \)
\[ \cdot |\{d \in D : w \in d\}| \] is the number of documents in \( D \) where the word appears.

\[ idf(D, w) = \log \left( \frac{N}{|\{d \in D : w \in d\}|} \right) \] (4)

3.2.3 Term frequency-inverse document frequency (TF-IDF)

The \( tf - idf \) is a numerical value which tells how important a term or word is, to a document. The \( tf idf \) value is computed as follows:

\[ tfidf(w, d, D) = tf(d, w) \times idf(D, w) \] (5)

\[ = tf(d, w) \times \log \left( \frac{N}{|\{d \in D : w \in d\}|} \right) \] (6)

When its value is 1, the \( \log \) term will be 0 hence \( tfidf(w, d, D) = 0 \). By definition, a word whose \( tf - idf \) is 0 appears in all predefined documents. Consider the term frequency Table 3.3 with two document collections, Documents 1 and 2.

| Document 1 | Document 2 |
|------------|------------|
| Word       | Word Count | Word       | Word Count |
| Smooth     | 3          | Higher     | 1          |
| Higher     | 2          | Low        | 1          |
| This       | 2          | That       | 4          |
| Low        | 1          | Google     | 2          |

We compute the \( tf - idf \) of the word Higher in document 2.

\[ tfidf(\text{Higher}, d, 2) = tf(d, w) \times idf(D, w) \]

\[ = tf(d_1, \text{Higher}) \times \log \left( \frac{N}{|\{d \in D : w \in d\}|} \right) \]

\[ = 1 \times \log \left( \frac{2}{3} \right) \]

\[ = 1 \times \log (1) \]

\[ = 0 \]

This implies that, because there are two documents, each word that appears in both will attain a \( tf - idf = 0 \). The same argument can be concluded on the word Low. Let compute that for the word Google which appear two times in document 2.
\[ tfidf(\text{Google}, d_2, 2) = tf(d, w) \times idf(D, w) \]
\[ = tf(d_2, \text{Google}) \times \log \left( \frac{N}{|\{d \in D : w \in d\}|} \right) \]
\[ = 2 \times \log \left( \frac{2}{1} \right) \]
\[ = 2 \times 0.3010 \]
\[ = 0.6020 \]

So the \( tf - idf \) of the word “Google” in document 2 is 0.6020. Supposing the word “Google” appeared 4 times in the document, its \( tf - idf \) will be 1.204. Hence the word “Google” is very important in document 2 and also its \( tf - idf \) increased because its \( tf \) also increased.

### 3.3 Singular Value Decomposition (SVD)

The SVD is a method of transforming a rectangular matrix into three separate matrices which tend to explain or expose certain key relationships that exist within the original matrix [15]. The reduced method is most often used in Latent Semantic Indexing (LSI) which is a type of document retrieval and term similarity. In using SVD, for LSI, a word-document matrix is constructed by representing the key words in the documents as the rows and the documents as the columns.

LSI is typically used as a dimension reduction or noise reducing technique.

Each entry of the constructed matrix is the frequency of a word in a particular document. SVD of a rectangular matrix, \( A_{mn} \), is given by

\[ A_{mn} = U_{nm}D_{mn}V_{nm}^T \quad (7) \]

\( m \): number of rows in matrix A.
\( n \): number of columns in matrix A.
\( U_{nm} \): An orthogonal matrix with columns as orthonormal eigenvectors from \( AA^T \).
\( V_{nm}^T \): The transpose of an orthogonal matrix with columns as orthonormal eigenvectors from \( A^TA \).
\( D_{mn} \): A diagonal matrix whose elements are the square roots of eigenvalues from \( U_{nm} \) or \( V_{mn}^T \) and are arranged from the highest to lowest.

![Fig. 2. Screenshot of showing how LSI reduces the dimension of a document matrix](image)

#### 3.3.1 Computing SVD of a given nxn matrix

An SVD for a \( 5 \times 5 \) matrix is elaborated. This will be a sequel to applying the method to a real world document analysis problem in the next section. Let \( W \) be a word-document matrix which has 5 columns that represent documents and 5 rows that represent words or terms.
It should be well noted that $w w^T$ is the dot product of each document in the array. Now the eigenvalues (singular values) of $w w^T$ are $\lambda_1 = 1284.27, \lambda_2 = 920.68, \lambda_3 = 50.80, \lambda_4 = 15.74, \lambda_5 = 0.48$ and the columns of $U$ are eigenvectors corresponding to the eigenvalues arranged from the largest to the lowest from left to right.

In practice the rows of $U$ depict words which are linearly independent. Similarly, $V$ can be also be computed as shown:
Now $V$ has the following columns consisting of the eigenvectors of the eigenvalues arranged from the largest to the lowest from left to right. The rows of $V$ are actually documents and so its transpose would have the documents as columns.

$$
V = \begin{bmatrix}
-0.4646 & 0.0215 & 0.8685 & 0.0008 & -0.1713 \\
-0.0701 & -0.7600 & -0.0631 & -0.6013 & -0.2278 \\
-0.7351 & 0.0088 & -0.2840 & -0.2235 & 0.5650 \\
-0.4844 & 0.0254 & -0.3899 & 0.3327 & -0.7035 \\
-0.0650 & -0.6415 & 0.0443 & 0.6912 & 0.3233 \\
\end{bmatrix}
$$

$$
V^T = \begin{bmatrix}
-0.4646 & -0.0701 & -0.7351 & -0.4844 & -0.0650 \\
0.0215 & -0.7600 & 0.0088 & 0.0254 & -0.6415 \\
0.8685 & -0.0631 & -0.2840 & -0.3899 & 0.0443 \\
0.0088 & -0.6013 & -0.2235 & 0.3327 & -0.7035 \\
-0.1713 & -0.2278 & 0.5650 & 0.3233 \\
\end{bmatrix}
$$

The last thing left to compute is the diagonal matrix, $D$, which contains the square root of the eigenvalues as its elements ordered from largest to lowest.

$$
D = \begin{bmatrix}
35.8367 & 0 & 0 & 0 & 0 \\
0 & 30.3427 & 0 & 0 & 0 \\
0 & 0 & 7.1280 & 0 & 0 \\
0 & 0 & 0 & 3.0685 & 0 \\
0 & 0 & 0 & 0 & 0.6991 \\
\end{bmatrix}
$$

In a similar manner, word similarity can be obtain using the matrix $UD$ and comparing the resulting rows also using vector norms.

4. RESULTS AND DISCUSSION

Experiment 1: Same Content in Three Documents using TF.

Here, all the three documents have the same content. The aim is to determine whether the SVD could actually detect the similarity between the three documents. Tables 3 to 4 show the terms or words used in the experiment and their term frequencies.

| Term       | Document 1 | Document 2 | Document 3 |
|------------|------------|------------|------------|
| abrupt     | 1.0        | 1.0        | 1.0        |
| advancement| 1.0        | 1.0        | 1.0        |
| algorithms | 1.0        | 1.0        | 1.0        |
| appeared   | 1.0        | 1.0        | 1.0        |
| applications| 1.0      | 1.0        | 1.0        |
| areas      | 3.0        | 3.0        | 3.0        |
| attempt    | 1.0        | 1.0        | 1.0        |
| automated  | 1.0        | 1.0        | 1.0        |
| become     | 1.0        | 1.0        | 1.0        |
| believed   | 1.0        | 1.0        | 1.0        |
| blogs      | 1.0        | 1.0        | 1.0        |
| change     | 1.0        | 1.0        | 1.0        |
| comments   | 1.0        | 1.0        | 1.0        |
| commercial| 1.0        | 1.0        | 1.0        |
| complaints | 1.0        | 1.0        | 1.0        |
| Term                | Document 1 | Document 2 | Document 3 |
|---------------------|------------|------------|------------|
| constantly          | 1.0        | 1.0        | 1.0        |
| could               | 1.0        | 1.0        | 1.0        |
| coupled             | 1.0        | 1.0        | 1.0        |
| customer            | 1.0        | 1.0        | 1.0        |
| derive              | 1.0        | 1.0        | 1.0        |
| digital             | 2.0        | 2.0        | 2.0        |
| easily              | 1.0        | 1.0        | 1.0        |
| encompasses         | 1.0        | 1.0        | 1.0        |
| engineering         | 1.0        | 1.0        | 1.0        |
| example             | 1.0        | 1.0        | 1.0        |
| extracted           | 1.0        | 1.0        | 1.0        |
| extraction          | 3.0        | 3.0        | 3.0        |
| facebook            | 1.0        | 1.0        | 1.0        |
| Field               | 3.0        | 3.0        | 3.0        |
| generated           | 1.0        | 1.0        | 1.0        |
| include             | 1.0        | 1.0        | 1.0        |
| information         | 2.0        | 2.0        | 2.0        |
| instruments         | 1.0        | 1.0        | 1.0        |
| retrieval           | 1.0        | 1.0        | 1.0        |
| intensive           | 1.0        | 1.0        | 1.0        |
| involved            | 1.0        | 1.0        | 1.0        |
| labour              | 1.0        | 1.0        | 1.0        |
| language            | 1.0        | 1.0        | 1.0        |
| learning            | 1.0        | 1.0        | 1.0        |
| machine             | 1.0        | 1.0        | 1.0        |
| manual              | 1.0        | 1.0        | 1.0        |
| meaningful          | 1.0        | 1.0        | 1.0        |
| media               | 2.0        | 2.0        | 2.0        |
| middle              | 1.0        | 1.0        | 1.0        |
| mining              | 4.0        | 4.0        | 4.0        |
| natural             | 1.0        | 1.0        | 1.0        |
| number              | 1.0        | 1.0        | 1.0        |
| occupies            | 1.0        | 1.0        | 1.0        |
| places              | 1.0        | 1.0        | 1.0        |
| print               | 2.0        | 2.0        | 2.0        |
| processing          | 1.0        | 1.0        | 1.0        |
| proved              | 1.0        | 1.0        | 1.0        |
| related             | 1.0        | 1.0        | 1.0        |
| space               | 2.0        | 2.0        | 2.0        |
| spurt               | 1.0        | 1.0        | 1.0        |
| statistics          | 1.0        | 1.0        | 1.0        |
| structured          | 1.0        | 1.0        | 1.0        |
| survey              | 1.0        | 1.0        | 1.0        |
| technological       | 1.0        | 1.0        | 1.0        |
| todays              | 1.0        | 1.0        | 1.0        |
| tremendous          | 1.0        | 1.0        | 1.0        |
| unstructured        | 1.0        | 1.0        | 1.0        |
| value               | 1.0        | 1.0        | 1.0        |
| various             | 1.0        | 1.0        | 1.0        |
| world               | 1.0        | 1.0        | 1.0        |
| years               | 1.0        | 1.0        | 1.0        |
The matrix generated from the tables above resulted in a $69 \times 3$ matrix. Each column of the matrix stands for a document and each row is the word found in the document. The entries or elements of the matrix are the frequencies of each word in a specific document.

Now the matrix $DD^T$ is computed from $D$. It should be well noted that $DD^T$ is the results of the dot product of the occurrence of a term in all the documents and all the words in each of the documents in succession. By so doing each word is projected unto all the other words in each of the documents. This algebraic process results in a $69 \times 69$ square matrix where each row and column stands for a particular word. Due to this $DD^T$ is referred to as the term-term (word-word) matrix. For instance in $DD^T$, the value in row 1 and column 2 of $DD^T_{11} = 3$ which means that, the words abruptly maps to itself three times in all the documents combined. In the same analogy, the words abrupt maps unto the word advancement three times.

The next stage is to compute all the eigenvalues and the corresponding eigenvectors, $U$, of the matrix $DD^T$. The eigenvalues arranged in ascending order are presented below.

$$\lambda_1 = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}$$

$$D = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\vdots & \vdots & \vdots \\
1 & 1 & 1
\end{bmatrix}$$

$$DD^T = \begin{bmatrix}
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
3 & 3 & 3 & 3 & 9 & \ldots & \ldots & 3 & 3 & 3 \\
\end{bmatrix}$$
The matrix $V = DD^T$ is a $3 \times 3$ matrix. Here each document is being projected onto the other. Matrix $V$ forms what we call the document-document matrix. Computation of all the eigenvalues and the corresponding eigenvectors, $\lambda$, of the matrix $D^T D$ arranged in ascending order are presented below.

\[ D^T D = \begin{bmatrix} 123 & 123 & 123 \\ 123 & 123 & 123 \\ 123 & 123 & 123 \end{bmatrix} \]

\[ \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} 369.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.0000 \end{pmatrix} \]

$s$ is the square root of all the diagonal elements of $\lambda v$ for $D^T D$. The first two eigenvalues are the largest amongst the set of the eigenvalues.

\[ S = \begin{pmatrix} 19.2094 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.0000 \end{pmatrix} \]

\[ S_2 = \begin{pmatrix} 19.2094 & 0.0000 \\ 0.0000 & 0.0000 \end{pmatrix} \]
Each row of the matrix $u_2$ corresponds to a term (word) in the documents. As shown, the term abrupt is mapped to the row $[0.0902 \ -0.0653]$ in the matrix. It is very interesting to note that the experiment here is made up of three documents, represented by the columns of $v_2^T$.

$$v_2^T = \begin{bmatrix} 0.5774 & 0.5774 & 0.5774 \\ 0.8162 & -0.4266 & -0.3896 \end{bmatrix}$$

The term and document matrices $U_2$ and $V_2^T$ respectively are scaled by multiplying them by the diagonal matrix $S_2$ to achieve the following results:

$$\hat{U}_2 = U_2 S_2$$

$$\hat{V}_2^T = S_2 V_2^T = \begin{bmatrix} 11.0915 & 11.0915 & 11.0915 \\ 0 & 0 & 0 \end{bmatrix}$$

Now the first 6 terms in the document can be written in a matrix form as follows:

$$\text{abrupt} = \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix} \quad \text{advancement} = \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix} \quad \text{algorithms} = \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix} \quad \text{appeared} = \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix}$$

$$\text{applications} = \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix} \quad \text{areas} = \begin{bmatrix} 5.1962 \\ 0 \end{bmatrix}$$

It is obvious to see that all the three documents are similar with respect to their vector entries. To A query, $q$ consisting of the terms abrupt and appeared is computed as:

$$\text{CosineSimilarity} = \frac{q d_i}{|d_i| |q|} \quad (8)$$

$$q = \frac{1}{3} \left( \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix} + \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix}$$

The value of the cosine similarity rule proofs that, the two terms abrupt and appeared are both in the document 1 document 2 and 3. The similarity values obtained 1.000 shows that, the terms (abrupt and appeared) are in all the three documents and that all the documents are the same. When a system containing these three documents is queried with the terms abrupt and appeared, all the three documents would be retrieved. Now let us compute the terms applications and areas are in the documents. Using the Cosine Similarity rule in equation 8 we obtained.
The same Cosine Similarity values are achieved for document 1, documents 2 and document 3. Hence the terms applications and areas are in the three documents. Based on these results it could be deduced that all the words are in the three documents leading to the assertion that the three documents are very similar.

Experiment 2: Same Content in Three Documents using TF-IDF

In this experiment the TF-IDF of each word in the documents are used. The matrix generated in this regard is $69 \times 3$ but its entries are a little bit different from that of the previous matrix. It should be noted that the terms used in this experiment are the same to that of experiment 1.

\[
\begin{bmatrix}
    0.2483 & 0.2483 & 0.2483 \\
    0.2483 & 0.2483 & 0.2483 \\
    0.2483 & 0.2483 & 0.2483 \\
    \vdots & \vdots & \vdots \\
    0.2483 & 0.2483 & 0.2483 \\
    0.2483 & 0.2483 & 0.2483 \\
    0.2483 & 0.2483 & 0.2483 \\
\end{bmatrix}
\]

The eigenvalues and their corresponding eigenvectors, $U$, of the matrix $DD^T$ are computed and arranged in ascending order as in the previous experiment. The computations carried out so far shows that using the TF-IDF reduces the numerical values of the entries of the various generated matrices. The highest eigenvalue for the TF-IDF is $31.28342$ where as that of the TF computations is $369.00000$. As already discussed, matrix $U$ is a projection of the terms unto themselves. In sequel, the document-document projection is computed as $DTD$ which results in a $3 \times 3$ matrix since the documents are only three. The eigenvalues, eigenvectors and $S$ which is the square root of all the diagonal elements of $\lambda v$ for the matrix $DTD$ are computed and arranged in ascending order of magnitude are presented below.

\[
\lambda_v = \begin{pmatrix}
    \lambda_1 \\
    \lambda_2 \\
    \lambda_3 
\end{pmatrix} =
\begin{pmatrix}
    31.28312213879992 \\
    0.000000000000005 \\
    0.000000000000003 
\end{pmatrix}
\]

\[
V = \begin{pmatrix}
    0.5774 & 0.7887 & 0.2113 \\
    0.5774 & -0.2113 & -0.7887 \\
    0.5774 & 0.5774 & 0.5774 
\end{pmatrix}
\]

\[
S = \begin{pmatrix}
    5.5932 & 0.0000 & 0.0000 \\
    0.0000 & 0.0000 & 0.0000 \\
    0.0000 & 0.0000 & 0.0000 
\end{pmatrix}
\]

We than compute $S_2$ which constitute the first two eigenvalues from $S$ and their corresponding columns from $U$, to get $U_2$. 

19
Note that the experiment here is made up of three documents represented by their columns;

\[
U_2 = \begin{bmatrix}
  \text{abrupt} & \text{advancement} & \text{algorithms} & \text{appeared} & \text{applications} \\
  0.0891 & -0.1504 & 0.0891 & -0.1506 & 0.0891 \\
  0.0891 & -0.1506 & 0.0891 & -0.1506 & 0.0891 \\
  0.2673 & 0.3666 & 0.2673 & 0.3666 & 0.2673 \\
  \vdots & \vdots & \vdots & \vdots & \vdots \\
  \text{years} & \text{world} & \text{areas} & \text{various} & \text{various} \\
  0.0891 & 0.05277 & 0.0891 & 0.05705 & 0.0891 & 0.2320
\end{bmatrix}
\]

Both \( U_2 \) and \( V_T \) are scaled by multiplying them by the diagonal matrix \( S_2 \).

\[
V_2^T = S_2 V_2^T = \begin{bmatrix}
  0.7887 & 0.5774 & 0.5774 \\
  0.5774 & 0.7887 & 0.5774 \\
  0.5774 & 0.5774 & 0.7887 \\
  0.2113 & -0.0000 & -0.0000 \\
  -0.5774 & -0.0000 & -0.0000 \\
\end{bmatrix}
\]

Now the first 4 terms in the document can be written in a matrix form as follows:

\[
\begin{bmatrix}
  \text{abrupt} \\
  \text{advancement} \\
  \text{algorithms} \\
  \text{appeared}
\end{bmatrix} = \begin{bmatrix}
  0.4983 \\
  -0.0000 \\
  0.4983 \\
  -0.0000
\end{bmatrix}
\]

The documents can be expressed in a similar manner as follows:

\[
\begin{align*}
document1 &= \begin{bmatrix}
  3.2292 \\
  0.0000
\end{bmatrix} & \document2 &= \begin{bmatrix}
  3.2292 \\
  -0.0000
\end{bmatrix} & \document3 &= \begin{bmatrix}
  3.2292
\end{bmatrix}
\end{align*}
\]

At this stage a query, \( q \), consisting of the words algorithms and applications is submitted to the three documents to verify if the words are indeed in them:
Computing the Cosine Similarity using equation 8 for each document: document 1, document 2, and document 3

$$q = \frac{1}{2} \left( \begin{bmatrix} 0.0891 \\ -0.1506 \end{bmatrix} + \begin{bmatrix} 0.4983 \\ -0.0000 \end{bmatrix} \right) = \begin{bmatrix} 0.2937 \\ -0.0753 \end{bmatrix}$$

In this experiment, the TF-IDF count in conjunction with the SDV algorithm was used to determine whether three documents are similar based on the words or terms that are in them. In conclusion, the value of the cosine similarity rule depicts that there is a 97% certainty that the two terms algorithm and application are both in the document 1. If this experimental result is compared to the work done by [16] we will conclude that the similarity between documents can be depicted using the TF-ID.

**Experiment 3: Varying Content in Three Documents using TF**

The last experiment that we performed utilized three documents with varying content. In all, 30 terms were used which has one term repeating in only two documents. Here the term frequency measure was utilized. The computations in experiment one and two were repeated in experiment three. Here we expect that the resulting values of the Cosine Rule would approach 0 because the documents had varying content.

| Term          | Document 1 | Document 2 | Document 3 |
|---------------|------------|------------|------------|
| Abrupt        | 1          | 0          | 0          |
| Advancement   | 1          | 0          | 0          |
| Algorithms    | 1          | 0          | 1          |
| Appeared      | 1          | 0          | 0          |
| Applications  | 1          | 0          | 0          |
| Areas         | 3          | 0          | 0          |
| Attempt       | 1          | 0          | 0          |
| Automated     | 1          | 0          | 0          |
| Become        | 1          | 0          | 0          |
| Believed      | 1          | 0          | 0          |
| Another       | 0          | 1          | 0          |
| automatically | 0          | 1          | 0          |
| Available     | 0          | 1          | 0          |
| Based         | 0          | 1          | 0          |
| Called        | 0          | 1          | 0          |
| Chaos         | 0          | 1          | 0          |
| Cleaned       | 0          | 1          | 0          |
| Combined      | 0          | 1          | 0          |
| Concerned     | 0          | 1          | 0          |
| Corpus        | 0          | 1          | 0          |
| Appropriate   | 0          | 0          | 1          |
| Classified    | 0          | 0          | 1          |
| Clustered     | 0          | 0          | 1          |
| Clustering    | 0          | 0          | 1          |
| Collected     | 0          | 0          | 1          |
| Databases     | 0          | 0          | 1          |
| Exists        | 0          | 0          | 1          |
| Files         | 0          | 0          | 1          |
| Information   | 0          | 0          | 1          |
The document-term matrix \( D \) generated from the tables above resulted in a \( 29 \times 3 \) matrix, we had a zero entry when that term is not found in that document. \( DD^T \) was computed, the eigenvalues, \( \lambda u \) and their corresponding eigenvectors, \( U \), of the matrix \( DD^T \) was computed and arranged in ascending order of magnitude.

\[
D = \begin{pmatrix}
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{pmatrix}
\]

\[
U = \begin{pmatrix}
0.0092 & -0.0638 & 0.2765 & -0.0579 & 0.0380 & -0.3534 & -0.2101 & -0.1030 & -0.0211 & 0.0334 & 0.9086 \\
-0.0092 & 0.0638 & -0.2765 & 0.0579 & -0.0380 & 0.3534 & 0.2101 & 0.1030 & 0.0211 & -0.0334 & -0.9086 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000
\end{pmatrix}
\]

The matrix \( V = D^T D \) which will give a \( 3 \times 3 \) matrix. Matrix \( V \) forms what we call the document to document matrix. Now \( S \) which is the square root of all the diagonal elements of \( \lambda v \) for the matrix \( D^T D \)

\[
V = \begin{pmatrix}
0.1222 & 0 & 0.9925 \\
0 & -1.0000 & 0 \\
-0.9925 & 0 & 0.1222
\end{pmatrix}
\]

\[
S_3 = \begin{pmatrix}
4.2571 & 0.0000 & 0.0000 \\
0.0000 & 3.1623 & 0.0000 \\
0.0000 & 0.0000 & 3.1428
\end{pmatrix}
\]

Taking the first two columns which constitute the first two eigenvalues from \( \lambda \), to form \( U_3 \) and picking their corresponding columns from \( U \), also forming \( U_3 \).

\[
U_3 = \begin{pmatrix}
\text{abrupt} & -0.2331 & 0.0000 & 0.0389 \\
\text{advancement} & -0.2331 & 0.0000 & 0.0389 \\
\text{algorithm} & 0.2331 & 0.0000 & 0.0389 \\
\text{appeared} & -0.2331 & 0.0000 & 0.0389 \\
\text{applications} & -0.2331 & 0.0000 & 0.0389 \\
\text{areas} & 0.6094 & 0.0000 & 0.1166 \\
\text{exists} & 0.0287 & -0.0000 & -0.3158 \\
\text{files} & -0.0287 & 0.0000 & -0.3158
\end{pmatrix}
\]

Each row of the matrix \( U_3 \) corresponds to a term (word) in the documents. The term abrupt and files corresponds to the row \([-0.2331 0.0000 0.0389]\) and \([-0.0287 0.0000 -0.3158]\) in \( U_3 \) respectively. The documents are represented by the columns of

22
Both $U_3$ and $V_3^T$ are scaled by multiplying them with the diagonal matrix $S_3$ to achieve the following results:

$$V_3^T = S_3V_3^T = \begin{bmatrix} 4.2252 & 0 & 0.5201 \\ 0 & -3.1623 & 0 \\ 0.3840 & 0 & -3.1192 \end{bmatrix}$$

Now the first 4 terms in the document can be written in a matrix form as follows:

At this stage a query, $\square$, consisting of the words algorithms and information is submitted to the three documents to verify if the words are indeed in them. The query then becomes

$$q = \frac{1}{2} \begin{bmatrix} -1.1147 \\ 0.0000 \\ -0.8703 \end{bmatrix} + \begin{bmatrix} -0.1222 \\ 0.0000 \\ -0.9925 \end{bmatrix} = \begin{bmatrix} -0.6184 \\ 0.0000 \\ -0.9314 \end{bmatrix}$$

For document 1

$$\text{CosineSimilarity}_1 = \frac{1}{(1.1180)(4.2426)} \begin{bmatrix} 4.2252 \\ -0.6184 \\ 0.0000 \\ 0.0000 \end{bmatrix} \begin{bmatrix} -0.6184 \\ 0.0000 \\ -0.9314 \\ 0.3840 \end{bmatrix}$$

$$= 0.6263$$
The rule yielded 0.6263 when document one was queried for the terms algorithms and information. Evidently only one of the terms in document one hence the value 0.6263.

**For document 2**

\[
\text{CosineSimilarity}_2 = \frac{1}{(1.1180)(3.1623)} \begin{bmatrix} 0.6184 & 0.0000 & -0.9314 \\ -3.1623 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -3.1623 \end{bmatrix}
\]

\[
= 0
\]

**For document 3**

\[
\text{CosineSimilarity}_3 = \frac{1}{(1.1180)(3.1623)} \begin{bmatrix} 0.6184 & 0.0000 & -0.9314 \\ -3.1623 & 0 \end{bmatrix} \begin{bmatrix} 0.5201 \\ 0 \end{bmatrix}
\]

\[
= 0.7308
\]

In this experiment; A query containing the terms algorithms and information were generated. Document 1, a similarity index of 0.6263, it means one of the two terms is in document 1. This claim is very true since table 4 shows that the term algorithms is in document 1 whereas information is not. As far as document 2 is concerned, both terms are not in it hence the similarity index is 0. This can also be seen in table 4. Document 3 had both terms hence its measure was 0.7308. The mean of these values are computed in other to obtain the overall similarity of the documents. The mean similarity value is 0.4524 which signifies that the three documents are not similar. The authors [16] [17] [18] [19] used SVD to reduce the dimension of a given matrix, produces good classification results. [17] Used SVD for feature reduction on image to analysis a give dataset. SVD was used to by [20] [17] to check the occurrence of a give dataset. The average of the occurrence and similarity between a given documents was analyses by these authors for the purpose of data clustering. Taking into consideration the work carried out by these authors and this paper, SVD is an algorithm to approximate a matrix and get out the important features by removing noise and insignificant term form a give documents.

**4 CONCLUSIONS**

Data mining involves the process of analyzing data from different point of views and concluding it in different views. In this paper several algorithms for text/data mining and the way each algorithm works was discussed in literature. We focused on using the Latent Semantic Indexing (LSI), which is a type of Singular Value Decomposition (SVD) to retrieve, compute and identify the similarity between documents. The experimental result proves that SDV can be used to improve similarity measure for text document clustering. In a detailed description, the Singular Value Decomposition, which was an enhanced algorithm brings into existence a significant clustering and similarity measure solutions. Using SVD in conjunction with cosine similarity measure, comes with the advantages of; computational efficiency, predicting similarity between documents and an efficient and reliable document similarity. It will ease data Miners and researchers when searching for reports or document with same term frequency and similarity comparison.

**COMPETING INTERESTS**

Authors have declared that no competing interests exist.

**REFERENCES**

1. Ghosh S, Roy S, Bandyopadhyay SK. A Tutorial Review on Text Mining Algorithms. International Journal of Advanced Research in Computer and Communication Engineering. 2012;1(4):223-233.

2. Chakraborty M, Pagolu K. Analysis of Unstructured Data: Applications of Text Analytics and Sentiment Mining. SAS Publishing. 2014;1(14):1288-2014.
3. Mooney RJ, Nahm UY. Text mining with information extraction- multilingualism and electronic language management, Proc. 4th Internat. MIDP Colloquium, September 2003, Bloemfontein, South Africa, W Daelemans, T du Plessis, C Snyman and L Teck (eds) VanSchaiik Pub., South Africa; 2003.

4. Kamruzzaman SM, Farhana H, Ahmed RH. Text Classification Using Data Mining. In: Proceedings of the International Conference on Information and Communication Technology in Management (ICITM). Multimedia University, Melaka, Malaysia; 2005.

5. Han J, Kamber M, Data Mining: Concepts and techniques. San Francisco: Morgan Kaufman Publishers, Elsevier. 2000;214-239.

6. Berry MW, & Castellanos M. Survey of Text Mining: Clustering, Classification, and Retrieval, Second Edition, CA: Springe. 2007;246-254.

7. Monett D, Lewis CW, & Thórisson KR, Introduction to the JAGI Special Issue “On Defining Artificial Intelligence”, Commentaries and Author’s Response. Journal of Artificial General Intelligence. 2020;11(2):1-100.

8. Yang Y, Zhang J, Kisiel B. A scalability analysis of classifiers in text categorization. 2003;2

9. Canasai K, Chuleerat J. A Parallel Learning Algorithm for Text Classification. In: proceedings of The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). Canada; 2002.

10. Jiawei H, Jian P, & Micheline K. “Data Mining: Concepts and Techniques,” Morgan Kaufmann Publisher; CA: 2012;1-5.

11. Han J, Kamber M. Data Mining: Concepts and techniques. CA: Morgan Kaufma Publishers, Elsevier. 2001;14-26.

12. Dasgupta A, Drineas P, Harb B. Feature Selection Methods for Text Classification. In: Proceedings of the 13th ACM SIGKDD International conference on Knowledge discovery and data mining, San Jose, CA; 2007; USA.

13. Qi X, Davison BD. “Web Page Classification: Features and Algorithms”. ACM Computing Surveys. 2009; 41(2). DOI.ORG/10.1145/1459352.1459357

14. Aggarwal CC, Zhai C. A Survey of Text Clustering Algorithms in mining Text data.New York: Springer-Verlag. 2012;77-128.

15. Baker K. Singular Value Decomposition Tutorial. 2013; Available:https://datajobs.com/data-science-repo/SVD-Tutorial-%5bKirk-Baker%5d.pdf [Accessed on 15th June, 2020 1:10 pm]

16. Nikita S, Recommender Systems with Python — Part III: Collaborative Filtering (Singular Value Decomposition); 2019. Available:https://heartbeat.fritz.ai/recommender-systems-with-python-part-iii-collaborative-filtering-singular-value-decomposition-5b5dc83f242b [Accessed on 23th August, 2020 2:10 pm]

17. Rhonda D. Phillips, Layne T. Watson, Randolph HW, Christine EB. Feature reduction using a singular value decomposition for the iterative guided spectral class rejection hybrid classifier, ISPRS Journal of Photogrammetry and Remote Sensing. 2009;64(1):107-116,ISSN 0924-2716. Available:https://doi.org/10.1016/j.isprsjprs.2008.03.004

18. Michael WB, Susan T D, Todd A L. Computational methods for intelligent information access; 1995 Available:https://doi.org/10.1145/224170.285669

19. Tapan K. Sarkar; Magdalena Salazar-Palma; Ming Da Zhu; Heng Chen, "Mathematical Principles Related to Modern System Analysis," in Modern Characterization of Electromagnetic Systems and its Associated Metrology, IEEE. 2021;1-20 DOI: 10.1002/9781119076230.ch1.

20. Eric N. In Data Mining Applications with R; 2014.

© 2021 Adu et al.; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here: https://www.sdiarticle4.com/review-history/73195