Implementation of Text Similarity using Word Frequency and Cosine Similarity in Python

Ahmad Farhan AlShammari
Department of Computer and Information Systems
College of Business Studies, PAAET
Kuwait

ABSTRACT
The goal of this research is to develop a text similarity program using word frequency and cosine similarity in Python. The purpose of text similarity is to measure the similarity between texts. The word frequency is used to measure the word importance in the text, and cosine similarity is used to measure the similarity between texts. The basic steps of text similarity are explained: preprocessing text, creating list of words, creating bag of words, creating word frequency, calculating cosine similarity, and printing similarity score. The developed program was tested on an experimental text from Wikipedia. The program successfully performed the basic steps of text similarity and provided the required results.

Keywords
Artificial Intelligence, Machine Learning, Natural Language Processing, Text Mining, Text Similarity, Word Frequency, Cosine Similarity, Python, Programming.

1. INTRODUCTION
The rapid development of Information and Communications Technology (ICT) is enabling the volume of data to grow very fast. Processing large amounts of data has become a crucial issue. Computer systems need more powerful methods to process data, analyze it, and extract information. Actually, machine learning is playing a key role in processing data more quickly and efficiently.

Machine learning (ML) is a branch of Artificial Intelligence (AI) that is focused on the study of computer algorithms to improve the performance of computer programs.

Text similarity is one of the important applications of machine learning. It is a common field between ML and Natural Language Processing (NLP). It applies both the techniques of NLP and the methods of ML to process human language.

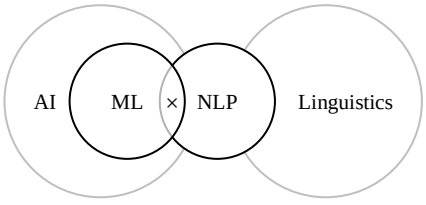

Fig 1: Field of Text Similarity

2. LIREATURE REVIEW
The review of literature revealed the major contributions in the field of text similarity [1-11]. The research started in the late fifties. In 1958, Hans Luhn [12] published the first paper in text summarization. He used the "word frequency" to measure the importance of words and sentences in the text. Then, the sentences of high scores are selected and added to the summary.

The research was originally focused on Information Retrieval (IR) [4]. Gerard Salton was the leading scientist in the field of information retrieval, and was called the father of IR.

Salton and his research group at Cornell University developed the first "automatic" information retrieval system SMART (System for the Mechanical Analysis and Retrieval of Text) [13-14].

Salton, Wong, and Yang [15] introduced the Vector Space Model (VSM) to easily represent text in the vector form. The text can be represented as a vector of numbers or weights as shown in the following view:

$$Text = (weight_1, weight_2, \ldots, weight_n)$$

Where: $weight_i$ is the weight of word ($word_i$) in the text.

Text vectorization helped to easily perform numerical calculations on text.

Over time, researchers continued to develop new weighting methods. For example, Karen Sparck-Jones [22,23,24-26] suggested the Inverse Document Frequency (IDF) to overcome the limitation of word frequency.

Later, Salton developed the Term Frequency-Inverse Document Frequency (TF-IDF) method [16-21]. It is the most widely used weighting method in retrieval systems, search engines, and digital libraries [6].

Salton also introduced the cosine similarity method to measure the similarity between texts. The concept of cosine similarity was originally used to measure the angle between vectors.

The fundamental concepts of text similarity are explained in the following section:

Text Similarity:
Text similarity is the process of measuring the similarity between texts to show if they are similar or not.

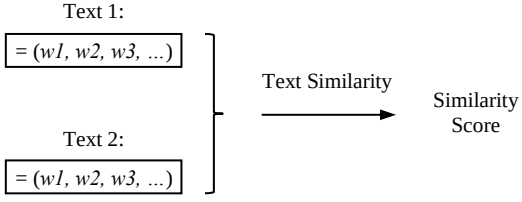

Fig 2: Concept of Text Similarity

Text similarity is used in many applications, for example: searching, ranking, recommendation, clustering, and classification.

Methods of Text Similarity:

There are different methods used to measure the text similarity such as: Euclidean distance, Manhattan distance, Jaccard distance and cosine similarity. They are used for different purposes, and have their advantages and disadvantages.

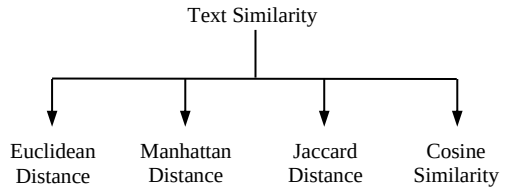

Fig 3: Methods of Text Similarity

A brief explanation for the methods of text similarity is shown in the following section:

The Euclidean distance is the direct distance between two points.

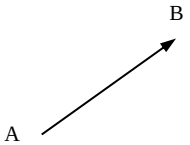

Fig 4: Explanation of Euclidean Distance

The Manhattan distance is the sum of horizontal and vertical distances between two points.

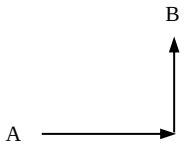

Fig 5: Explanation of Manhattan Distance

The Jaccard distance is the number of intersection items divided by the number of union items.

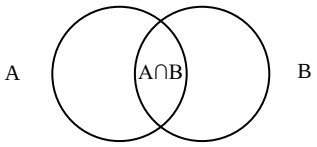

Fig 6: Explanation of Jaccard Distance

The cosine similarity is the cosine of the angle between two vectors.

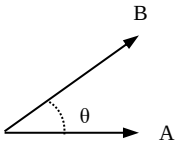

Fig 7: Explanation of Cosine Similarity

Cosine similarity is the most widely used method in text similarity because of its simplicity and efficiency.

In this research, the cosine similarity method is applied.

Text Similarity System:

In the text similarity system; the input is the two texts. Then, the system will process the two texts and apply the cosine similarity method to calculate the similarity between them. Finally, the output is the similarity score.

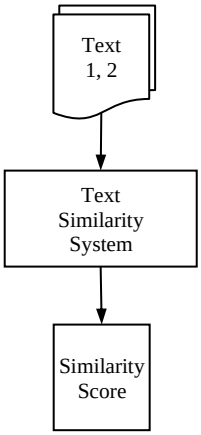

Fig 8: Diagram of Text Similarity System

Preprocessing Text:

The raw text should be "cleaned" from the unwanted characters and words such as punctuation symbols and stopwords.

List of Words:

The text is "tokenized" or split into words. The result of word tokenization is the list of words as shown in the following view:

$$List\ of\ Words = [word_1, word_2, \ldots, word_n]$$

Bag of Words:

Bag of words (BoW) is the set of words without repetition.

$$Bag\ of\ Words = (word_1, word_2, \ldots, word_m)$$

Word Frequency:

Word frequency is the number of times a word occurs in the text divided by the number of words in the text. It is calculated by the following formula:

$$freq(w_i) = \frac{Nw_i}{Nw} \qquad (1)$$

Where: Nw_i is the number of times the word (w_i) occurs in the text, and Nw is the total number of words in the text.

Cosine Similarity:

Cosine similarity is a mathematical method used to measure the similarity between texts. The concept of cosine similarity was originally used in mathematics to measure the angle between vectors.

For example, consider the two vectors *A* and *B* in the plane, as shown in the following diagram:

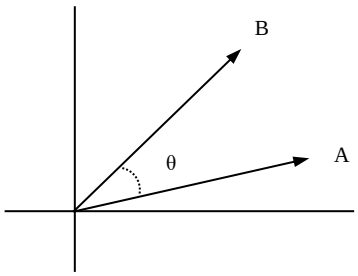

Fig 9: Representation of Vectors A and B

The dot product of the two vectors (*A.B*) is calculated by the following formula:

$$A \cdot B = \|A\| \, \|B\| \, Cos(\theta) \qquad (2)$$

Where: ||*A*|| and ||*B*|| are the norms of vectors *A* and *B* respectively, and θ is the angle between the two vectors.

Then, the cosine of the angle is calculated by the following formula:

$$Cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} \qquad (3)$$

In general, for any two vectors *A* and *B* in the space, where:

$$A = (a_1, a_2, \ldots, a_n)$$
$$B = (b_1, b_2, \ldots, b_n)$$

Then, the cosine of the angle is calculated by the following formula:

$$Cos(\theta) = \frac{\sum (a_i \, b_i)}{\sqrt{\sum a_i^2 \, \sum b_i^2}} \qquad (4)$$

Where: a_i and b_i are the values of vectors *A* and *B* respectively.

The cosine similarity shows the "percentage" of similarity between the two vectors. The resulting cosine can take values between (0) and (1). If the cosine value is (1) then the two vectors are similar, and if the cosine value is (0) then the two vectors are not similar.

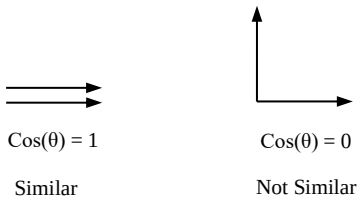

Fig 10: Values of Cosine Similarity

Python:

Python [27] is a general high-level programming language. It is simple, easy to learn, and powerful. It is the most preferred programming language by the developers of machine learning applications.

Python provides additional libraries such as: Numpy [28], Pandas [29], Matplotlib [30], NLTK [31], and SK Learn [32].

In this research, the standard functions of Python are applied without using any additional library.

3. RESEARCH METHODOLOGY

The basic steps of text similarity are: preprocessing text, creating list of words, creating bag of words, creating word frequency, calculating cosine similarity, and printing similarity score.

1. Preprocessing Text
2. Creating List of Words
3. Creating Bag of Words
4. Creating Word Frequency
5. Calculating Cosine Similarity
6. Printing Similarity Score

Fig 11: Steps of Text Similarity

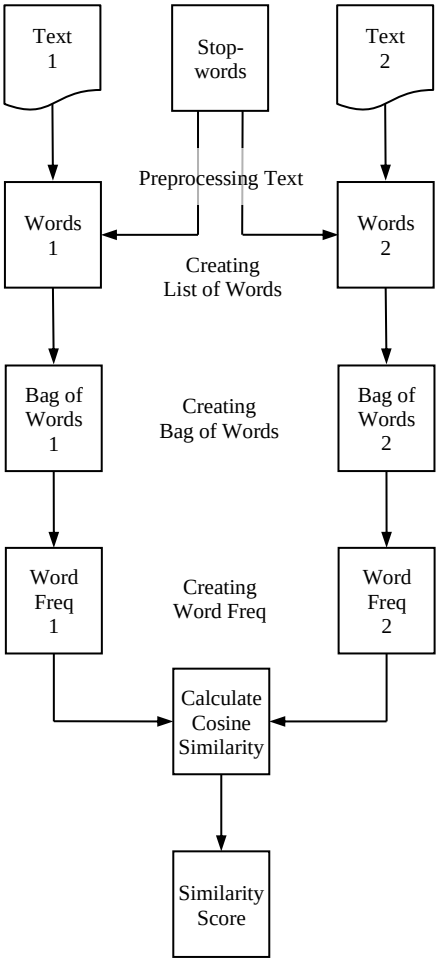

Fig 12: Flowchart of Text Similarity

The basic steps of text similarity are explained in details in the following section:

1.Preprocessing Text:

The text is preprocessed to remove the unwanted characters and words. It is done by the following steps:

1.1 Converting Text into Lower Case:

The text is converted into lower case. It is done by the following code:

```
text = raw_text.lower()
```

1.2 Removing Punctuation:

The punctuation symbols (like: !@#$ …) are removed from the text. It is done by the following code:

```
letters = "abcdefghijklmnopqrstuvwxyz"
for c in text:
    if (c not in letters):
        text = text.replace(c," ")
```

1.3 Removing Stopwords:

The Stopwords (like: I, am, is, are, …) are removed from the text. It is done by the following code:

```
stopwords = ["i", "am", "is", "are", "we",
             "he", "she", "it", "the",
             "they", "that", "this", ... ]
for word in text:
    if (word in stopwords):
        text = text.replace(word,"")
```

2. Creating List of Words:

The text is split into words. It is done by the following code:

```
words = text.split()
```

3. Creating Bag of Words:

The bag of words is the set of words. It is done by the following code:

```
bag_of_words = set(words)
```

4. Creating Word Frequency:

The word frequency holds the frequencies of words.

Word	Frequency
w_1	*freq*(w_1)
w_2	*freq* (w_2)
w_3	*freq* (w_3)
…	…
w_n	*freq* (w_n)

Fig 13: Structure of Word Frequency

Where: *freq*(w_i) is the frequency of word (w_i). It is done by the following code:

```
Nw = len(words)
freq = {}
for word in bag_of_words:
    freq[word] = words.count(word) / Nw
```

5. Calculating Cosine Similarity:

The cosine similarity is calculated using formula (4). It is done by the following code:

```
# calculate the dot product of two vectors
def dot(vector1, vector2):
    sum = 0
    for key in vector1:
        if key in vector2:
            sum += vecor1[key] * vector2[key]
    return sum

# calculate the norm of a vector
def norm(vector):
    sum = 0
    for key in vector:
        sum += vector[key]**2
    return math.sqrt(sum)

# calculate the cosine similarity
value1 = dot(freq1, freq2)
value2 = norm(freq1) * norm(freq2)
cosine = value1 / value2
```

6. Printing Similarity Score:

The similarity score is printed. It is done by the following code:

```
print("Similarity Score = ", cosine)
```

4. RESULTS AND DISCUSSION

The developed program was tested on an experimental text from Wikipedia [33]. The program performed the basic steps of text similarity and provided the required results. The resulting output is shown in the following section:

List of Words:

The list of words is shown in the following view:

```
List of Words:
Text 1:
        abstract
        analogous
        available
        based
        chooses
        clinical
        collectively
        comprise
        content
        content
        ...
Text 2:
        abstraction
        abstractive
        abstractive
        abstractive
        applied
        apply
        based
        build
        called
        cases
        ...
```

Bag of Words:

The bag of words is shown in the following view:

```
Bag of Words
Text 1:
        abstract
        analogous
        available
        based
        chooses
        clinical
        collectively
        comprise
        content
```

```
        data
        ...
Text 2:
        abstraction
        abstractive
        applied
        apply
        based
        build
        called
        cases
        challenging
        closer
        ...
```

Word Frequency:

The word frequency is shown in the following view:

```
Word Frequency:
Text 1:
        abstract                0.0135135135
        analogous               0.0135135135
        available               0.0135135135
        based                   0.0135135135
        chooses                 0.0135135135
        clinical                0.0135135135
        collectively            0.0135135135
        comprise                0.0135135135
        content                 0.0405405405
        data                    0.0135135135
        ...
Text 2:
        abstraction             0.0135135135
        abstractive             0.0405405405
        applied                 0.0135135135
        apply                   0.0135135135
        based                   0.0135135135
        build                   0.0135135135
        called                  0.0135135135
        cases                   0.0135135135
        challenging             0.0135135135
        closer                  0.0135135135
        ...
```

The existence of common words proves the similarity between the two texts. They are shown in the following view:

```
Common Words:
        1       based
        2       content
        3       document
        4       extracted
        5       extraction
        6       original
        7       summarization
        8       summary
        9       text
        10      video
```

The frequencies of the common words are shown in the following table:

Table 1: Frequencies of Common Words

Word	Text 1	Text 2
1	0.0135135135	0.0135135135
2	0.0405405405	0.0270270270
3	0.0270270270	0.0270270270
4	0.0405405405	0.0135135135
5	0.0405405405	0.0270270270
6	0.0135135135	0.0540540541
7	0.0135135135	0.0405405405
8	0.0135135135	0.0135135135
9	0.0405405405	0.0675675676
10	0.0135135135	0.0135135135

The following chart shows a visual representation of the common words:

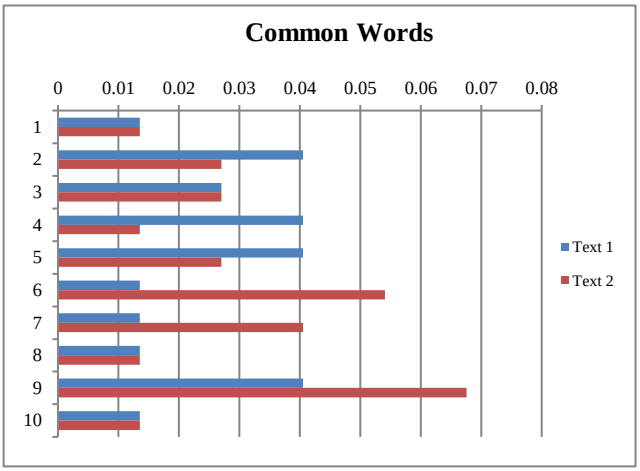

Fig 14: Chart of Common Words

Similarity Score:

The similarity score is shown in the following view:

```
Similarity Score = 0.3467255099
```

The resulting similarity score shows that the percentage of similarity between the two texts is about (35%).

Further, the cosine similarity matrix (2×2) is shown in the following view:

	Text 1	Text 2
Text 1	1	0.3467255099
Text 2	0.3467255099	1

Fig 15: Cosine Similarity Matrix

In summary, the program output clearly demonstrates that the developed program successfully performed the basic steps of text similarity and provided the required results.

5. CONCLUSION

Text similarity is one of the important applications of machine learning. The purpose of text similarity is to measure the similarity between texts and show if they are similar or not. The word frequency is used to measure the importance of words in the text, and cosine similarity is used to measure the similarity between texts.

In this research, the author developed a program to measure the similarity between texts using word frequency and cosine similarity in Python. The developed program performed the basic steps of text similarity: preprocessing text, creating list of words, creating bag of words, creating word frequency, calculating cosine similarity, and printing similarity score.
The program was tested on an experimental text from Wikipedia and provided the required results: list of words, bag of words, word frequency, and similarity score.

In future work, more research is certainly required to improve and develop the current methods of text similarity. In addition, they should be more investigated on different domains and languages such as Arabic.

6. REFERENCES

[1] Sammut, C., & Webb, G. I. (2011). "Encyclopedia of Machine Learning". Springer.

[2] Aggarwal, C. (2015). "Data Mining: The Textbook". New York: Springer.

[3] Aggarwal, C. (2018). "Machine Learning for Text". New York: Springer.

[4] Hotho, A., Nürnberger, A., & Paass, G. (2005). "A Brief Survey of Text Mining". LDV Forum - GLDV Journal for Computational Linguistics and Language Technology. 20, 19-62.

[5] Gomaa, W. H., & Fahmy, A. A. (2013). "A Survey of Text Similarity Approaches". International Journal of Computer Applications, 68(13), 13-18.

[6] Breitinger, C., Gipp, B., Langer, S. (2015). "Research-Paper Recommender Systems: A Literature Survey". International Journal on Digital Libraries, 17(4), 305-338.

[7] Vijaymeena, M. K., & Kavitha, K. (2016). "A Survey on Similarity Measures in Text Mining". Machine Learning and Applications: An International Journal, 3(2), 19-28.

[8] Gunawan, D., Sembiring, C. A., & Budiman, M. A. (2018). "The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents". In Journal of Physics: Conference Series (Vol. 978, p. 012120). IOP Publishing.

[9] Prasetya, D. D., Wibawa, A. P., & Hirashima, T. (2018). "The Performance of Text Similarity Algorithms". International Journal of Advances in Intelligent Informatics, 4(1), 63-69.

[10] Shahmirzadi, O., Lugowski, A., & Younge, K. (2019). "Text Similarity in Vector Space Models: A Comparative Study". In 2019 18th IEEE international conference on machine learning and applications (ICMLA) (pp. 659-666). IEEE.

[11] Wang, J., & Dong, Y. (2020). "Measurement of Text Similarity: A Survey". Information, 11(9), 421.

[12] Luhn, H. (1958). "The Automatic Creation of Literature Abstracts". IBM Journal of Research and Development, 2(2), 159-165.

[13] Salton, G. & Lesk, M. E. (1965). "The SMART Automatic Document Retrieval Systems: An Illustration". Communications of the ACM. 8 (6): 391-398.

[14] Salton, G. (1971). "The SMART Retrieval System: Experiments in Automatic Document Retrieval". Englewood Cliffs, N.J.: Prentice Hall Inc.

[15] Salton, G., Wong, A., & Yang, C. S. (1975). "A Vector Space Model for Automatic Indexing". Communications of the ACM, 18(11), 613-620.

[16] Salton, G., Yang, C. S., & Yu, C. T. (1975). "A Theory of Term Importance in Automatic Text Analysis". Journal of the American Society for Information Science, 26(1), 33-44.

[17] Salton, G. & McGill, M. (1983). "Introduction to Modern Information Retrieval". McGraw Hill Book Co, New York.

[18] Salton, G., & Buckley, C. (1988). "Term-Weighting Approaches in Automatic Text Retrieval". *Information Processing and Management*, 24(5), 513-523.

[19] Salton, G. (1989). "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer". Addison- Wesley Publishing Company, USA.

[20] Salton, G., Allan, J., & Buckley, C. (1994). "Automatic Structuring and Retrieval of Large Text Files". Communications of the ACM, 37(2), 97-108.

[21] Salton, G., Singhal, A., Mitra, M., & Buckley, C. (1997). "Automatic Text Structuring and Summarization". Information Processing & Management, 33(2), 193-207.

[22] Sparck Jones, K. (1972). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". Journal of Documentation. 28(1), 11–21.

[23] Sparck Jones, K. (2004). "IDF Term Weighting and IR Research Lessons". Journal of Documentation, 60(5), 521-523.

[24] Robertson, S. (1972). "Term Specificity". Journal of Documentation, 28(1), 164-165.

[25] Robertson, S. (1974). "Documentation Note: Specificity and Weighted Retrieval". Journal of Documentation, 30(1), 41-46.

[26] Robertson, S. (2004). "Understanding Inverse Document Frequency: On Theoretical Arguments for IDF". Journal of Documentation, 60(5), 503-520.

[27] Python: https://www.python.org

[28] Numpy: https://www.numpy.org

[29] Pandas: https:// pandas.pydata.org

[30] Matplotlib: https://www. matplotlib.org

[31] NLTK: https://www.nltk.org

[32] SK Learn: https://scikit-learn.org

[33] Wikipedia: https://en.wikipedia.org