Implementation of Text Recommendation 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 recommendation program using word frequency and cosine similarity in Python. Text recommendation is the process that provides suggestions to the user. 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. The basic steps of text recommendation are explained: preprocessing text, creating list of words, creating bag of words, creating word frequency, calculating cosine similarity, creating similarity score, sorting similarity score, and printing recommendations. The developed program was tested on an experimental text from Wikipedia. The program successfully performed the basic steps of text recommendation and provided the required results.

Keywords
Artificial Intelligence, Machine Learning, Natural Language Processing, Text Mining, Text Recommendation, 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 is becoming 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) which is focused on the study of computer algorithms to improve the performance of computer programs.

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

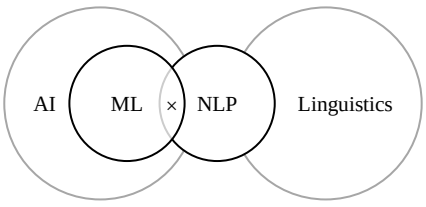

Fig 1: Field of Text Recommendation

2. LIREATURE REVIEW
The review of literature revealed the major contributions in the field of text recommendation [1-15]. The research started in the late seventies. In 1979, Elaine Rich [16] developed the first recommendation system "Grundy" to help users in finding their favorite books.

Over time, many recommendation systems were developed for different purposes. For example, "Tapestry" was developed in 1992 to help users in filtering mail system [17], and "GroupLens" was developed in 1994 at the University of Minnesota to help users in filtering news articles [18].

In the late 1990s, Amazon [19] developed a recommendation system based on collaborative filtering [20]. The success story of Amazon encouraged other companies to implement their own recommendation systems [21]. For example, in 2006 Netflix [22] launched "NetFlix Prize" to improve the performance of its recommendation system, and in 2010 Youtube [23] implemented its recommendation system.

Recommendation systems are mainly used in filtering items, for example: products, articles, books, news, TV programs, movies, songs, etc.

The research in recommendation systems is booming by the implementation of the new developed methods in machine learning.

The word frequency was initially proposed by Hans Luhn [24] to measure the importance of words in the text.

The cosine similarity was introduced by Gerard Salton [25-29] to measure the similarity between texts. He also developed the Vector Space Model (VSM) to represent text as a vector of numbers or weights.

The fundamental concepts of recommendation systems are explained in the following section:

Recommendation System:
Recommendation system is a program that provides suggestions to the user. The function of recommendation system is to find the similarity between users and items based on their features.

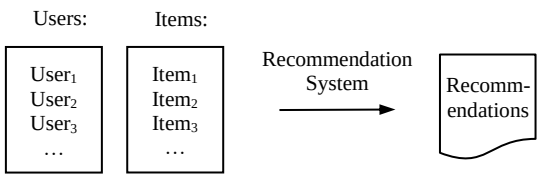

Fig 2: Concept of Recommendation System

Types of Recommendation Systems:
Recommendation systems are divided into two main groups: Content-based and Collaborative.

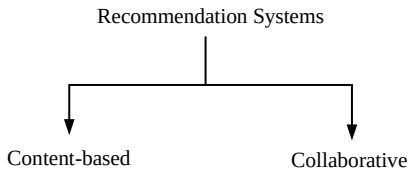

Fig 3: Types of Recommendation Systems

In the content-based type: the work is focused on finding similarity between "items". For example: the items that are similar to the items preferred by the user are selected and added to the recommendations.

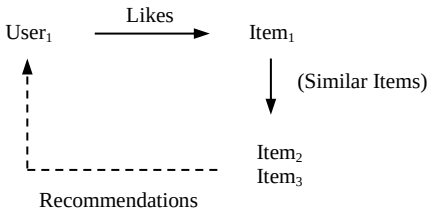

Fig 4: Content-based Recommendation System

In the collaborative type: the work is focused on finding similarity between "users". For example: the items that are preferred by similar users are selected and added to the recommendations.

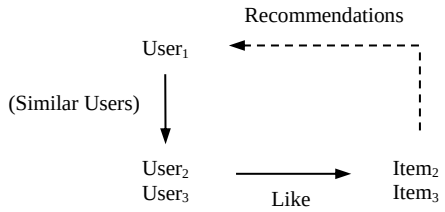

Fig 5: Collaborative Recommendation System

In this research, the content-based type is applied.

Text Recommendation System:

In the text recommendation system; the input is the query text. Then, the system will process the text, calculate the cosine similarity between the query text and the given texts, and sort them by the similarity score. The output is the recommendations.

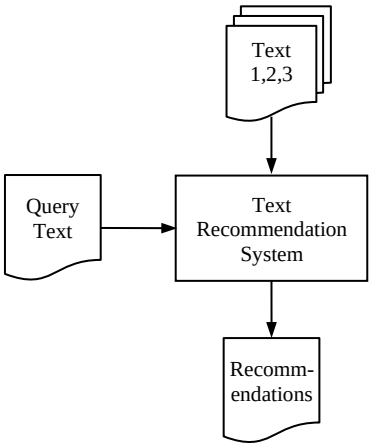

Fig 6: Diagram of Text Recommendation System

Preprocessing Text:

The raw text should be cleaned from the unwanted characters and words, for example: 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 as shown in the following view:

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

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 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 was originally derived from the calculus of vectors in mathematics.

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

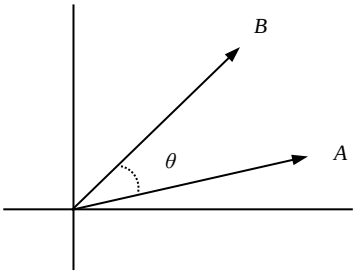

Fig 7: Representation of Vectors A and B

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:

$$A = (a_1, a_2, \ldots, a_n)$$

$$B = (b_1, b_2, \ldots, b_n)$$

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 cosine can take values between (0) and (1). For example: 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.

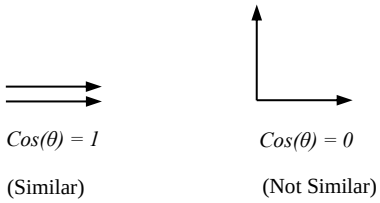

Fig 8: Value of Cosine Similarity

Python:

Python [30] is a high-level general purpose 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 [31], Pandas [32], Matplotlib [33], NLTK [34], and SKLearn [35].

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

3. RESEARCH METHODOLOGY

The basic steps of text recommendation are: (1) preprocessing text, (2) creating list of words, (3) creating bag of words, (4) creating word frequency, (5) calculating cosine similarity, (6) creating similarity score, (7) sorting similarity score, and (8) printing recommendations.

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

Fig 9: Steps of Text Recommendation

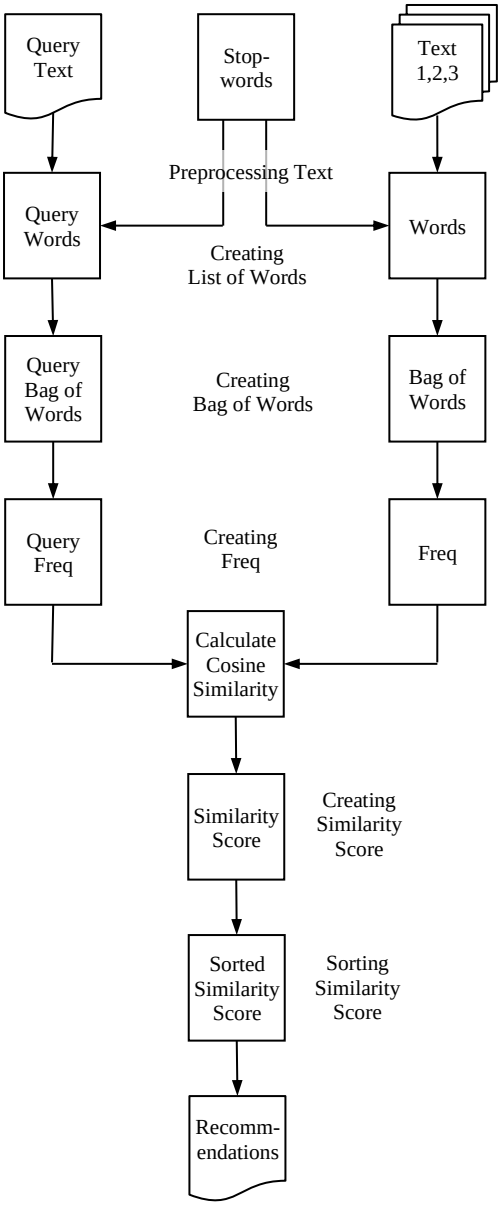

Fig 10: Flowchart of Text Recommendation

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

1. Preprocessing Text:

The text is cleaned from 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 = 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",
             "this", "that", "they", ... ]
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 11: 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 by formula (4). It is done by the following code:

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

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

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

6. Creating Similarity Score:

The similarity score holds the similarity scores of the texts with the query text.

Text	Score
t_1	$score_1$
t_2	$score_2$
t_3	$score_3$
…	…
t_n	$score_n$

Fig 12: Structure of Similarity Score

Where: $score_i$ is the similarity score of text (t_i) with the query text. It is done by the following code:

```
score = {}
for text, freq in freqs:
    score[text] = cosine(qfreq,freq)
```

7. Sorting Similarity Score:

The similarity score is sorted in reverse order by the score value. In Python, sorting a list is done using the (sorted) function as shown in the following code:

```
sorted_list = sorted(list, reverse=True)
```

However, sorting a dictionary is more complicated that a list because the structure of dictionary is composed of paired (key, value) items.

8. Printing Recommendations:

The recommendations consist of the texts that have similarity scores above the average score. It is done by the following code:

```
for text, value in sorted_score.items():
    if (value >= average):
        print(text)
```

4. RESULTS AND DISCUSSION

The developed program was tested on an experimental text from Wikipedia [36]. The program performed the basic steps of text recommendation and provided the required results. The program output is shown in the following section:

List of Words:

The list of words is shown in the following view:

```
List of Words:
       filtering
       information
       items
       particular
       pertinent
       ...
```

Bag of Words:

The bag of words is shown in the following view:

```
Bag of Words:
        across
        additional
        algorithms
        alternative
        approach
        ...
```

Word Frequency:

The word frequency is shown in the following view:

```
Word Frequency:
        Across                  0.0054644809
        Additional              0.0054644809
        algorithms              0.0054644809
        alternative             0.0054644809
        approach                0.0054644809
        ...
```

Similarity Score:

The similarity score is shown in the following view:

```
Similarity Score:
        Text 1                  0.3913118961
        Text 2                  0.0
        Text 3                  0.3407546685
        Text 4                  0.3465516400
        Text 5                  0.0
        Text 6                  0.4001633653
        Text 7                  0.4168439339
        Text 8                  0.3474041669
        Text 9                  0.0
        Text 10                 0.0
        Text 11                 0.0
        Text 12                 0.3391511082
```

The following chart shows a visual representation of the similarity score:

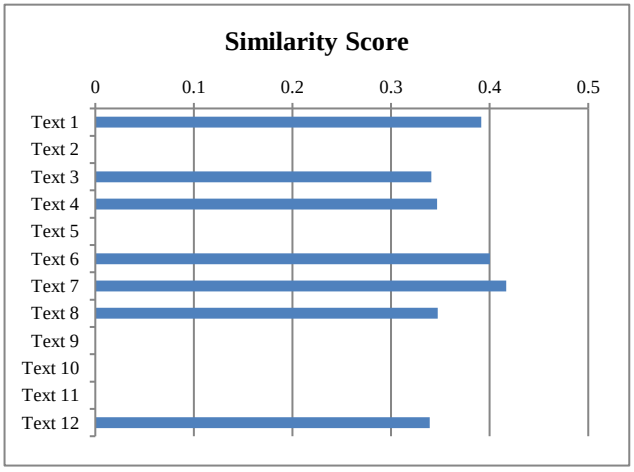

Fig 13: Chart of Similarity Score

Sorted Similarity Score:

The sorted similarity score is shown in the following view:

```
Sorted Similarity Score:
        Text 7                  0.4168439339
        Text 6                  0.4001633653
        Text 1                  0.3913118961
        Text 8                  0.3474041669
        Text 4                  0.3465516400
        Text 3                  0.3407546685
        Text 12                 0.3391511082
```

The following chart shows a visual representation of the sorted similarity score:

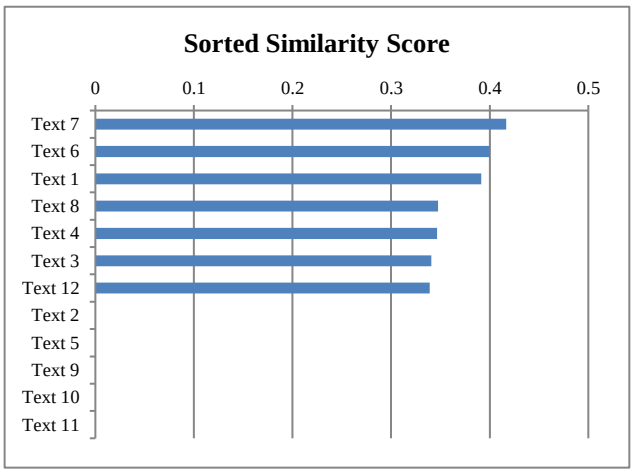

Fig 14: Chart of Sorted Similarity Score

Average Score:

The average score is shown in the following view:

```
Average Score = 0.2151817316
```

Recommendations:

The recommendations are shown in the following view:

```
Query:
Text recommendation

Recommendations:
Recommendation systems have also been de ...
There are also popular recommendation sy ...
A recommendation system is a subclass of ...
Recommendation systems usually make use  ...
Recommendation systems are used in a var ...
Recommendation systems are particularly  ...
Recommendation systems are a useful alte ...
```

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

5. CONCLUSION

Text recommendation is one of the important applications of machine learning. The purpose of text recommendation is to provide suggestions to the user. 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 text recommendation program using word frequency and cosine similarity in Python. The developed program performed the basic steps of text recommendation: preprocessing text, creating list of words, creating bag of words, creating word frequency, calculating cosine similarity, creating similarity score, sorting similarity score, and printing recommendations.

The program was tested on an experimental text from Wikipedia and provided the required results: list of words, bag

of words, word frequency, similarity score, sorted similarity score, and recommendations.
In future work, more research is certainly needed to improve and develop the current methods of text recommendation. In addition, they should be more investigated in 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. (2016). "Recommender Systems: The Textbook". Springer.

[4] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). "Recommender Systems: An Introduction". Cambridge University Press.

[5] Burke, R., Felfernig, A., & Göker, M. H. (2011). "Recommender Systems: An Overview". AI Magazine, 32(3), 13-18.

[6] Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). "A Literature Review and Classification of Recommender Systems Research". Expert Systems with Applications, 39(11), 10059-10072.

[7] Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). "Recommender Systems Survey". Knowledge-based Systems, 46, 109-132.

[8] Ricci, F., Rokach, L., & Shapira, B. (2015). "Recommender Systems: Introduction and Challenges". Recommender Systems Handbook, 1-34.

[9] Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). "Recommender System Application Developments: A Survey". Decision Support Systems, 74, 12-32.

[10] Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). "Paper Recommender Systems: A Literature Survey". International Journal on Digital Libraries, 17, 305-338.

[11] Fayyaz, Z., Ebrahimian, M., Nawara, D., Ibrahim, A., & Kashef, R. (2020). "Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities". Applied Sciences, 10(21), 7748

[12] Jannach, D., Pu, P., Ricci, F., & Zanker, M. (2021). "Recommender Systems: Past, Present, Future". AI Magazine, 42(3), 3-6.

[13] Kanwal, S., Nawaz, S., Malik, M. K., & Nawaz, Z. (2021). "A Review of Text-based Recommendation Systems". IEEE Access, 9, 31638-31661.

[14] Roy, D., & Dutta, M. (2022). "A Systematic Review and Research Perspective on Recommender Systems". Journal of Big Data, 9(1), 59.

[15] Ko, H., Lee S., Park Y., & Choi A. (2022). "A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields". Electronics, 11(1), 141.

[16] Rich, E. (1979). "User Modeling via Stereotypes". Cognitive science, 3(4), 329-354.

[17] Goldberg, D., Nichols, D., Oki, B., & Terry, D. (1992). "Using Collaborative Filtering to Weave an Information Tapestry". Communications of the ACM, 35(12), 61-70.

[18] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). "Grouplens: An Open Architecture for Collaborative Filtering of NetNews". In ACM Conference on Computer Supported Cooperative Work, 175-186.

[19] Amazon: https://www.amazon.com

[20] Linden, G., Smith, B., & York, J. (2003). "Amazon.com Recommendations: Item-to-Item Collaborative Filtering". IEEE Internet Computing, 7(1), 76–80.

[21] Schafer, J. B., Konstan, J., & Riedl, J. (1999). "Recommender Systems in E-Commerce". In Proceedings of the 1st ACM Conference on Electronic Commerce, 158-166.

[22] NetFlix: https://www.netflix.com

[23] Youtube: https://www.youtube.com

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

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

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

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

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

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

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

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

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

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

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

[35] SK Learnt: https://scikit-learn.org

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