# Implementation of Keyword Extraction using Term Frequency-Inverse Document Frequency (TF-IDF) 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 keyword extraction program using Term Frequency-Inverse Document Frequency (TF-IDF) in Python. The purpose of keyword extraction is to identify the set of words (keywords) that describe the content of the text. The TF-IDF method is used to measure the importance of words in the text. The basic steps of keyword extraction are explained: preprocessing text, creating list of words, creating bag of words, creating word frequency (TF), creating inverse document frequency (IDF), creating word frequency-inverse document frequency (TF-IDF), creating keywords, and sorting keywords. The developed program was tested on an experimental text from Wikipedia. The program successfully performed the basic steps of keyword extraction and provided the required results.

## Keywords
Artificial Intelligence, Machine Learning, Natural Language Processing, Text Mining, Keyword Extraction, Term Frequency-Inverse Document Frequency, TF-IDF, 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) that is focused on the study of computer algorithms to improve the performance of computer programs.

Keyword extraction 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.

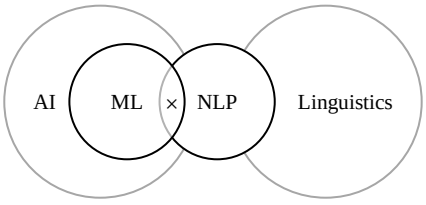


**Fig 1: Field of Keyword Extraction**

## 2. LIREATURE REVIEW
The review of literature revealed the major contributions in the field of keyword extraction [1-16]. The research started in the late fifties. In 1958, Hans Luhn [17] proposed a statistical method based on "Term Frequency" (TF) to measure the word importance in the text. He found that the word importance is simply proportional to the word frequency in the text. However, TF has a drawback, for example: common words (for example: the, it, is) occur more frequently in the text and therefore, will get high TF value.

Over time, researchers continued to develop new methods to overcome the limitations of TF. In 1972, Spark Jones [18, 19] suggested the "Inverse Document Frequency" (IDF) to measure the word importance in the whole set of documents. She found that the word importance is "inversely" proportional to the word frequency in the whole set of documents. For example: If the word occurs more frequently in the documents, then it has low significance.

Gerard Salton developed the Term Frequency-Inverse Document Frequency (TF-IDF) which is simply the product of TF and IDF [20-22, 23-28]. It is used to measure the word importance in the document with respect to the whole set of documents. TF-IDF is the most widely used weighting method in text mining because it is powerful in extracting keywords from the text. In 2015, a study showed that TF-IDF is implemented in (83%) of digital libraries [8].

Salton also introduced the Vector Space Model (VSM) to easily represent text as a vector of numbers or weights [23].

The fundamental concepts of keyword extraction are explained in the following section:

## Keyword Extraction:
Keyword extraction is the process that identifies the set of words (keywords) that describe the content of the text. Keywords are very important in text mining. They provide a short representation for the basic characteristics of the text.

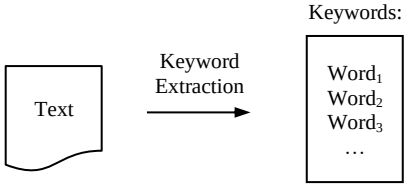


**Fig 2: Concept of Keyword Extraction**

Therefore, the text is represented as a vector of words as shown in the following view:

$$Text = (word_1, word_2, word_3, \dots)$$

## Applications of Keyword Extraction:

Keyword extraction is an essential process in information retrieval, text mining, and machine learning. It is used in various applications such as: indexing, searching, ranking, summarization, similarity, recommendation, clustering, classification, etc.

## Approaches of Keyword Extraction:

There are different approaches in keyword extraction, such as: frequency-based, graph-based, and neural networks.

In this research, the frequency-based approach is applied.

## Keyword Extraction System:

In the keyword extraction system; the input is the text given by the user. Then, the system will apply the TF-IDF method to give specific weights to the words based on their frequency in the text. Finally, the output is the resulting keywords.

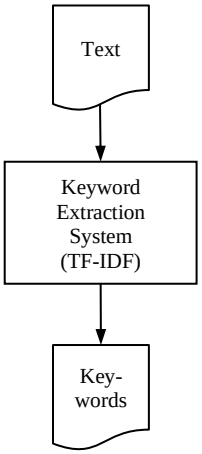


**Fig 3: Diagram of Text Extraction System**

## Preprocessing Text:

The 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:

$$List\ of\ Words = [\ word_1, word_2, word_3, \dots\ ]$$

## Bag of Words:

Bag of Words (BoW) is the set of words without repetition, as shown in the following view:

$$Bag\ of\ Words = (\ word_1, word_2, word_3, \dots\ )$$

## Term Frequency:

Term Frequency (TF) is the number of times a word occurs in the document divided by the total number of words in the document. It measures the importance of a word in the document. It is calculated by the following formula:

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

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

## Inverse Document Frequency:

Inverse Document Frequency (IDF) is the log value of the number of documents (corpus) divided by the number of documents in which a word occurs. It measures the word importance in the whole set of documents. It is calculated by the following formula:

$$IDF(w_i) = log(\frac{Nd}{Nd|w_i}) \qquad (2)$$

Where: $Nd$ is the total number of documents, and $Nd|w_i$ is the number of documents in which the word ($w_i$) occurs.

## Term Frequency-Inverse Document Frequency:

Term Frequency-Inverse Document Frequency (TF-IDF) is the product of term frequency (TF) and inverse document frequency (IDF). It measures the word importance in the document with respect to the whole set of documents. It is calculated by the following formula:

$$TF\text{-}IDF(w_i) = TF(w_i) \times IDF(w_i) \qquad (3)$$

## Python:

Python [29] 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 [30], Pandas [31], Matplotlib [32], NLTK [33], and SK Learn [34].

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

# 3. RESEARCH METHODOLOGY

The basic steps of keyword extraction are: (1) preprocessing text, (2) creating list of words, (3) creating bag of words, (4) creating term frequency (TF), (5) creating inverse document frequency (IDF), (6) creating term frequency-inverse document frequency (TF-IDF), (7) creating keywords, and (8) sorting keywords.

1. Preprocessing Text
2. Creating List of Words
3. Creating Bag of Words
4. Creating Term Frequency (TF)
5. Creating Inverse Document Frequency (IDF)
6. Creating Term Frequency- Inverse Document Frequency (TF-IDF)
7. Creating Keywords
8. Sorting Keywords

**Fig 4: Steps of Keyword Extraction**

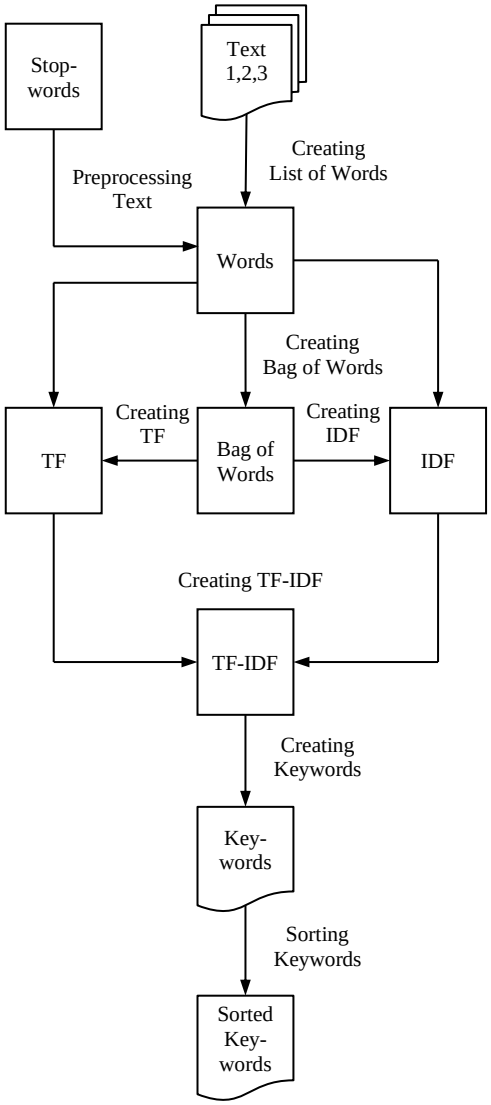


**Fig 5: Flowchart of Keyword Extraction**

The steps of keyword extraction are explained in details in the following section:

## 1. Preprocessing Text:

The text is preprocessed to clean it 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",
             "it", "he", "she", "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 Term Frequency:

The term frequency (TF) holds the TF values of words. It is shown in the following view:

| Word | TF |
|---|---|
| $w_1$ | $tf(w_1)$ |
| $w_2$ | $tf(w_2)$ |
| $w_3$ | $tf(w_3)$ |
| … | … |
| $w_n$ | $tf(w_n)$ |

**Fig 6: Structure of Word Frequency**

Where: $tf(w_i)$ is the TF value of word ($w_i$). It is done by the following code:

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

## 5. Creating Inverse Document Frequency:

The inverse document frequency (IDF) holds the IDF values of words. It is shown in the following view:

| Word | IDF |
|---|---|
| $w_1$ | $idf(w_1)$ |
| $w_2$ | $idf(w_2)$ |
| $w_3$ | $idf(w_3)$ |
| … | … |
| $w_n$ | $idf(w_n)$ |

**Fig 7: Structure of Inverse Document Frequency**

Where: $idf(w_i)$ is the IDf value of word ($w_i$). It is done by the following code:

```
Nd = len(docs)
idf = {}
for word in bag_of_words:
    sum = 0
    for doc in docs:
        if (word in doc):
            sum +=1
    Ndw = sum
    idf[word] = math.log10(Nd / Ndw)
```

## 6. Creating Term Frequency-Inverse Document Frequency:

The term frequency-inverse document frequency (TF-IDF) holds the TF-IDF values of words. It is shown in the following view:

| Word | TF-IDF |
|---|---|
| $w_1$ | *tfidf*($w_1$) |
| $w_2$ | *tfidf* ($w_2$) |
| $w_3$ | *tfidf* ($w_3$) |
| … | … |
| $w_n$ | *tfidf* ($w_n$) |

**Fig 8: Structure of Term Frequency- Inverse Document Frequency**

Where: *tfidf* ($w_i$) is the TF-IDF value of word ($w_i$). It is done by the following code:

```
tfidf = {}
for word, value in tf.items():
    tfidf[word] = value * idf[word]
```

## 7. Creating Keywords:

The Keywords consist of the words that have TF-IDF values above the average value. It is done by the following code:

```
keywords = {}
for word, value in tfidf.items():
    if (value >= average):
    keywords[word] = value
```

## 8. Sorting Keywords:

The keywords are sorted by the TF-IDF values in reversed order. In Python, sorting a list is simply done by the (sorted) function as shown in the following code:

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

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

## 4. RESULTS AND DISCUSSION

The developed program was tested on an experimental text from Wikipedia [35]. The program performed the basic steps of keyword extraction 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:
        algorithms
        automation
        computation
        computation
        computer
        ...
Text 2:
        amount
        answering
        computation
        computations
        computed
        ...
Text 3:
        aims
        communicating
        compressing
        data
        data
        ...
...
```

## Bag of Words:

The bag of words is shown in the following view:

```
Bag of Words:
        adaptation
        aims
        algorithms
        amount
        analysis
        ...
```

## Term Frequency:

The term frequency (TF) is shown in the following view:

```
Term Frequency (TF):
Text 1:
        algorithms              0.0454545455
        automation              0.0454545455
        computation             0.0909090909
        computer                0.0454545455
        computing               0.0454545455
        ...
Text 2:
        amount                  0.0833333333
        answering               0.0833333333
        computation             0.0833333333
        computations            0.0833333333
        computed                0.0833333333
        ...
Text 3:
        aims                    0.0666666667
        communicating           0.0666666667
        compressing             0.0666666667
        data                    0.1333333333
        find                    0.0666666667
        ...
...
```

## Inverse Document Frequency:

The inverse document frequency (IDF) is shown in the following view:

```
Inverse Document Frequency (IDF):
        adaptation              1.0791812460
        aims                    0.4771212547
        algorithms              0.7781512504
        amount                  1.0791812460
        analysis                0.7781512504
        ...
```

## Term Frequency-Inverse Document Frequency:

The term frequency-inverse document frequency (TF-IDF) is shown in the following view:

```
Term Frequency-Inverse Document Frequency (TF-IDF):
Text 1:
        algorithms              0.0353705114
        automation              0.0490536930
        computation             0.0707410228
        computer                0.0172823292
        computing               0.0490536930
        ...
Text 2:
        amount                  0.0899317705
        answering               0.0899317705
        computation             0.0648459375
        computations            0.0899317705
        computed                0.0899317705
        ...
Text 3:
        aims                    0.0318080836
        communicating           0.0719454164
        compressing             0.0719454164
        data                    0.0636161673
        find                    0.0719454164
        ...
...
```

## Keywords:

The keywords are shown in the following view:

```
Keywords:
Text 1:
        automation              0.0490536930
        computation             0.0707410228
        computing               0.0490536930
        hardware                0.0490536930
        implementing            0.0490536930
        ...
Text 2:
        amount                  0.0899317705
        answering               0.0899317705
        computations            0.0899317705
        computed                0.0899317705
        focused                 0.0899317705
        ...
Text 3:
        communicating           0.0719454164
        compressing             0.0719454164
        data                    0.0636161673
        find                    0.0719454164
        operations              0.0719454164
        ...
...
```

## Sorted Keywords:

The sorted keywords are shown in the following view:

```
Sorted Keywords:
Text 1:
        computation             0.0707410228
        wide                    0.0490536930
        topics                  0.0490536930
        theoretical             0.0490536930
        systems                 0.0490536930
        ...
Text 2:
        resources               0.0899317705
        required                0.0899317705
        questions               0.0899317705
        perform                 0.0899317705
        focused                 0.0899317705
        ...
Text 3:
        storing                 0.0719454164
        signal                  0.0719454164
        reliably                0.0719454164
        processing              0.0719454164
        operations              0.0719454164
        ...
...
```

The following chart shows a visual representation for the number of keywords in the texts:

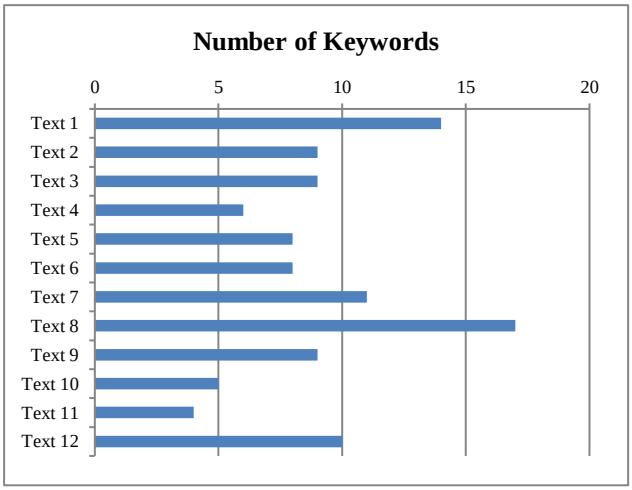


**Fig 9: Chart of Number of Keywords**

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

## 5. CONCLUSION

Keyword extraction is one of the important applications of machine learning. The purpose of keyword extraction is to identify the set of words (keywords) that describe the content of the text. The TF-IDF method is used to measure the importance of words in the text.

In this research, the author developed a keyword extraction program using the word frequency-inverse document frequency (TF-IDF) in Python. The program performed the basic steps of keyword extraction: preprocessing text, creating list of words, creating bag of words, creating word frequency (TF), creating inverse document frequency (IDF), creating word frequency-inverse document frequency (TF-IDF), creating keywords, and sorting keywords.

The program was tested on an experimental text from Wikipedia and provided the required results: list of words, bag of words, term frequency (TF), inverse document frequency (IDF), term frequency-inverse document frequency (TF-IDF), keywords, and sorted keywords.

In future work, more research is certainly needed to improve and develop the current methods of keyword extraction. 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] Lee, S., & Kim, H. J. (2008). "Automatic Keyword Extraction from News Articles using TF-IDF Model". Networked Computing and Advanced Information Management, 2.

[4] Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). "Automatic Keyword Extraction from Individual Documents". In Text Mining: Applications and Theory, 1-20.

[5] Kaur, J., & Gupta, V. (2010). "Effective Approaches for Extraction of Keywords". International Journal of Computer Science Issues, 7(6), 144-148.

[6] Hong, B., & Zhen, D. (2012). "An Extended Keyword Extraction Method". Physics Procedia, 24, 1120-1127.

[7] Beliga, S. (2014). "Keyword Extraction: A Review of Methods and Approaches". University of Rijeka, Department of Informatics, Rijeka, 1(9).

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

[9] Siddiqi, S., & Sharan, A. (2015). "Keyword and Keyphrase Extraction Techniques: A Literature Review". International Journal of Computer Applications, 109(2), 18-23.

[10] Gupta, T. (2017). "Keyword Extraction: A Review". International Journal of Engineering Applied Sciences and Technology, 2(4), 215-220.

[11] Bharti, S. K., & Babu, K. S. (2017). "Automatic Keyword Extraction for Text Summarization: A Survey". arXiv preprint arXiv:1704.03242.

[12] Qaiser, S., & Ali, R. (2018). "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents". International Journal of Computer Applications, 181(1), 25-29.

[13] Thushara, M. G., Mownika, T., & Mangamuru, R. (2019). "A Comparative Study on Different Keyword Extraction Algorithms". In 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC) (pp. 969-973). IEEE.

[14] Firoozeh, N., Nazarenko, A., Alizon, F., & Daille, B. (2020). "Keyword Extraction: Issues and Methods". Natural Language Engineering, 26(3), 259-291.

[15] Xu, Z., & Zhang, J. (2021). "Extracting Keywords from Texts based on Word Frequency and Association Features". Procedia Computer Science, 187, 77-82.

[16] Li, J. (2021). "A Comparative Study of Keyword Extraction Algorithms for English Texts". Journal of Intelligent Systems, 30(1), 808-815.

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

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

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

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

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

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

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

[24] 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.

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

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

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

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

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

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

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

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

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

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

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