**ABSTRACT**

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. The weight is a measure of the importance of the word in the document. It is conferred in examining the relevance of key-words to the corpus. Generally score for each word is computed to signify its importance in the corpus using tf –idf formula. In this paper we analyze the result of applying two variations of tf-idf formula to determine which gives more relevant answer(sentence or passage) on performing a set of same queries to these algorithms. In the first one term frequency of file is normalized based on length of document and in the latter term frequency is used as raw in calculating the tf-idf of the file. We also provide evidence through comparisons that the normalized term frequency algorithm performs better than the unnormalized one.

**INTRODUCTION**

The amount of information has increased tremendously over past years due to rapid growth of internet and expansion of data. Due to which it is difficult to fetch relevant information.However, the information retrieval systems are being used in order to get the desired information. Each information retrieval system mainly composed of two parts the indexer and the ranking algorithm. The index is helpful in finding the set of documents which best fullfills the information need.The purpose of the ranking algorithm is to retrieve from the collection of documents the most relevant ones based on the rank. Ranking directly affects retrieval quality.

The processing of text data is always an interesting and challenging job in an organization. There are many techniques/methods/algorithms are discovered till now, but this study is focused particularly on TF-IDF methods.

TF-IDF is a statistical measure that evaluates how relevant word is to a document in a collection of documents.Term frequency(tf) indicates how important a specific term in a document.Whereas inverse document frequency(idf) is the weight of the term.It aims to reduce the weight if the the term’s occurrences is scattered.Ranking of the document is tf-idf score which is the product of tf and idf.

The goal of this project is the implement the tf-idf algorithm in two variations normalised tf and unormalised tf .We perform set of query to find the relevant documents using the above variations and compare the results between them.

TF It has many use cases, most importantly in automated text analysis and scoring words in machine learning algorithms for Natural Language Processing (NLP). The processing of text data is always an interesting and challenging job in an organization. There are many techniques/methods/algorithms are discovered till now, but this study is focused particularly on TF-IDF methods. TF-IDF is a statistical measure that evaluates how relevant word is to a document in a collection of documents. It has many use cases, most importantly in automated text analysis and scoring words in machine learning algorithms for Natural Language Processing (NLP).

**TECHNOLGY STACK USED AND DATA SET:**

**Python:**

• Python is a high-level, general-purpose, open source, strictly typed programming Language. The language provides constructs intended to enable clear programs on both a small and large scale.

• Python was created By Guido van Rossum.

• The Python Software Foundation (PSF) is the organization behind Python.

NLTK Library:

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](https://www.nltk.org/nltk_data/) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active [discussion forum](https://groups.google.com/group/nltk-users).

DATASET:

The dataset used here was taken from website <https://cs50.harvard.edu/ai/2020/projects> wich has data about different somain of Computer Science.

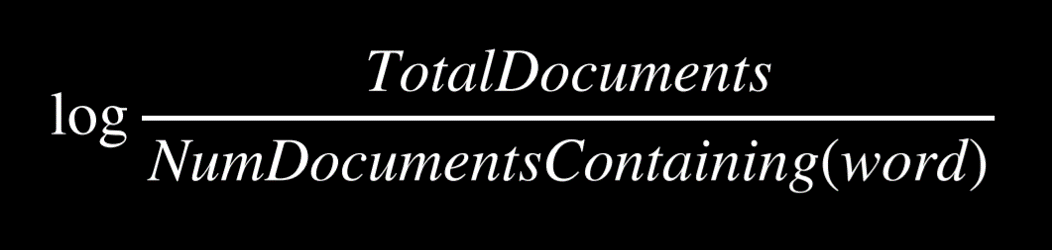
**BACKGROUND**

**TF (Term Frequency):**

The TF is a frequency count of a term in a document. There are several ways of calculating this frequency, with simple raw count of instance of a word appears in a document. Then there are ways to adjust the frequency of the most frequent word in a document. One of the way is to divide the frequency count with the length of the document (the number of set of different instances).

**IDF (Inverse Document Frequency):**

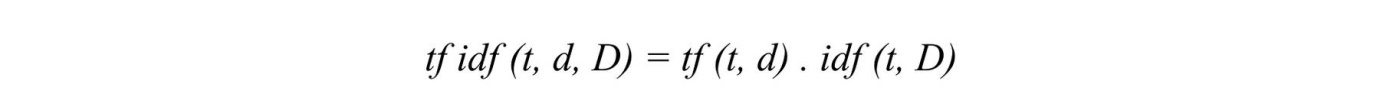
The Inverse Document Frequency is the frequency of a word across set of documents. These means how common or rare a word is in the entire document set. The closer it is 0, the more common, a word is. The metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the algorithm upon it.

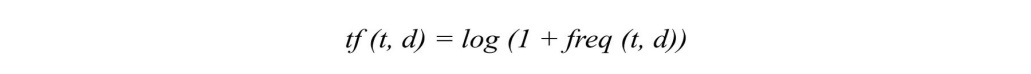


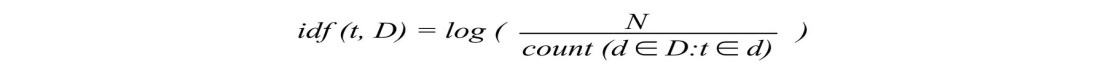
So if the word is very common and appears in many documents, this number will approach 0, otherwise it will approach 1.

**TF-IDF**

Multiplying these two values results in the TF-IDFs score. The higher the score, the more relevant that word is in that particular document.The TF-IDF score for the word ‘t’ in the document ‘d’ from the document set ‘D’ is calculated as follows:

Where:





**PROCEDURE**

The system has access to bunch of documents (corpus) of text. Using TF-IDF try to find out the most relevance sentences or passages from these documents.

**What we have implemented?**

We have a corpus of text containing information about different domain like Pyhton, Machine Learning, Artificial Intelligence etc. The user will asks some queries that the answer might be present in those text documents. The algorithm will first find out the most relevant document among them. And again it will look for the most relevant information based on the query the user has asked and return the answer.

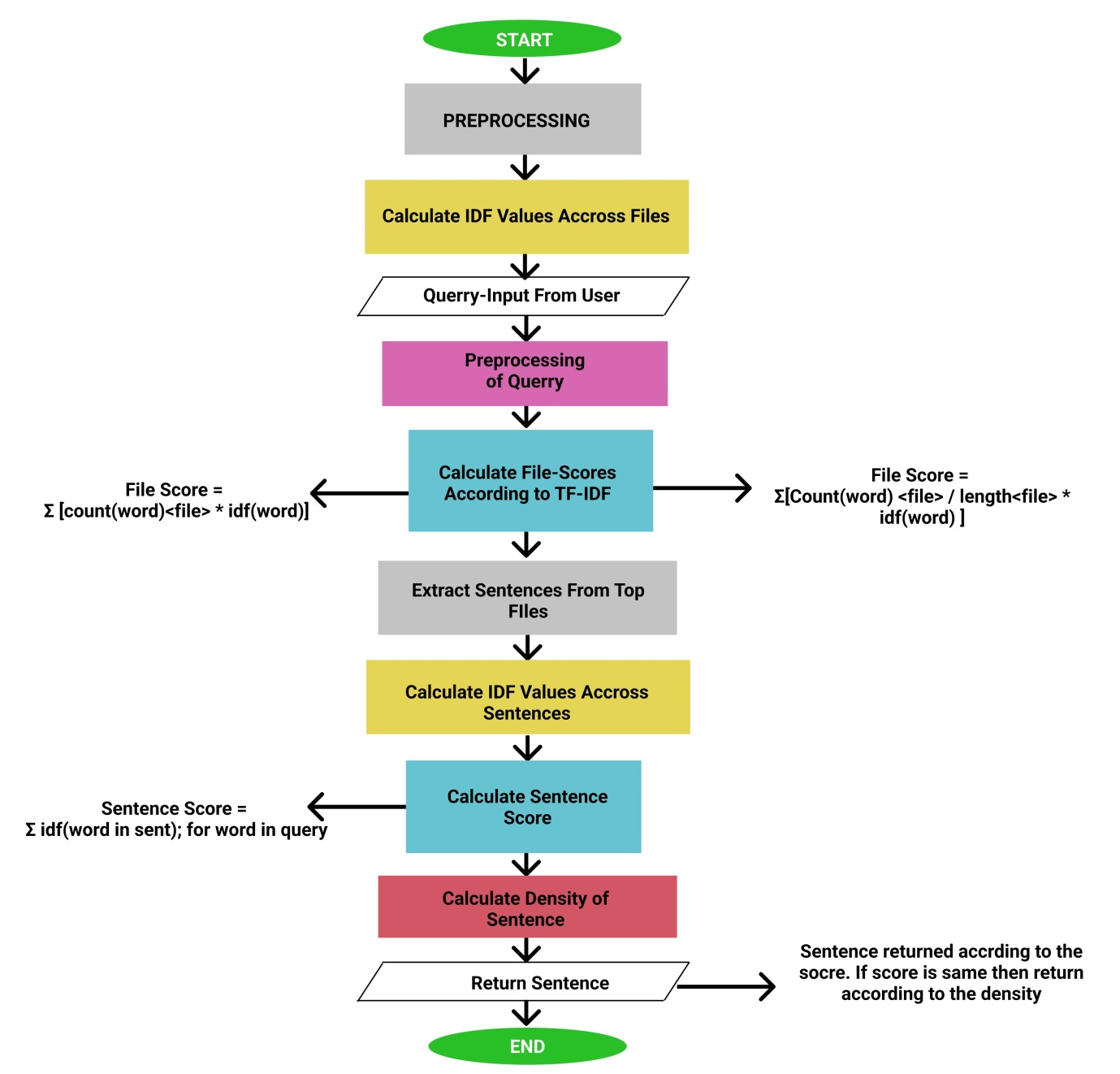
The flow starts with the preprocessing of the corpus data which involves process like ex. tokenization, removing stop words and etc. After that IDF value of very set of pre-processed unique words are calculated across the documents. As mentioned earlier idf value of the word is logarithm of ratio of total document in the corpus and the no. document in which word is present.

Then it asks a query from the user and again preprocessing is done in the user input. For calculating file score two techniques for term frequency are used both are shown in the diagram below.

After that it find out top the files form the bunch of documents based on the score of file. This is done by counting the frequency(tf-term frequency) of words in query in the each document.And multiplying with their corresponding idf values.Finally adding all the tf-idf values of all words in query to find out the scores of the document.Ranking of document is done accordingly based on the score.In the second method also same procedure is followed except the term frequency is normalized i.e.,it is divided with the length of document.

After that the computation limits to the particular documents which is in top list.The top files to be considered may be just a single file or more than that depending upon the user.It is basically taken one.

Once the top files have been found.Then the top sentences within that document is found.The IDF values across sentences present in the document is calculated followed by the score of the sentence is calculated and top matching sentences are found and returned.



|  |  |  |
| --- | --- | --- |
| SNO. | Words | IDF Values |
| 1 | Visual | 0.40546 |
| 2 | Require | 0.69314 |
| 3 | Learning | 0.0 |
| 4 | Design | 0.40546 |
| 5 | Model | 0.0 |
| 6 | Estimated | 0.69314 |
| 7 | One | 0.0 |
| 8 | Many | 0.0 |
| 9 | History | 0.0 |

Table:Words with their corresponding idf values

…..

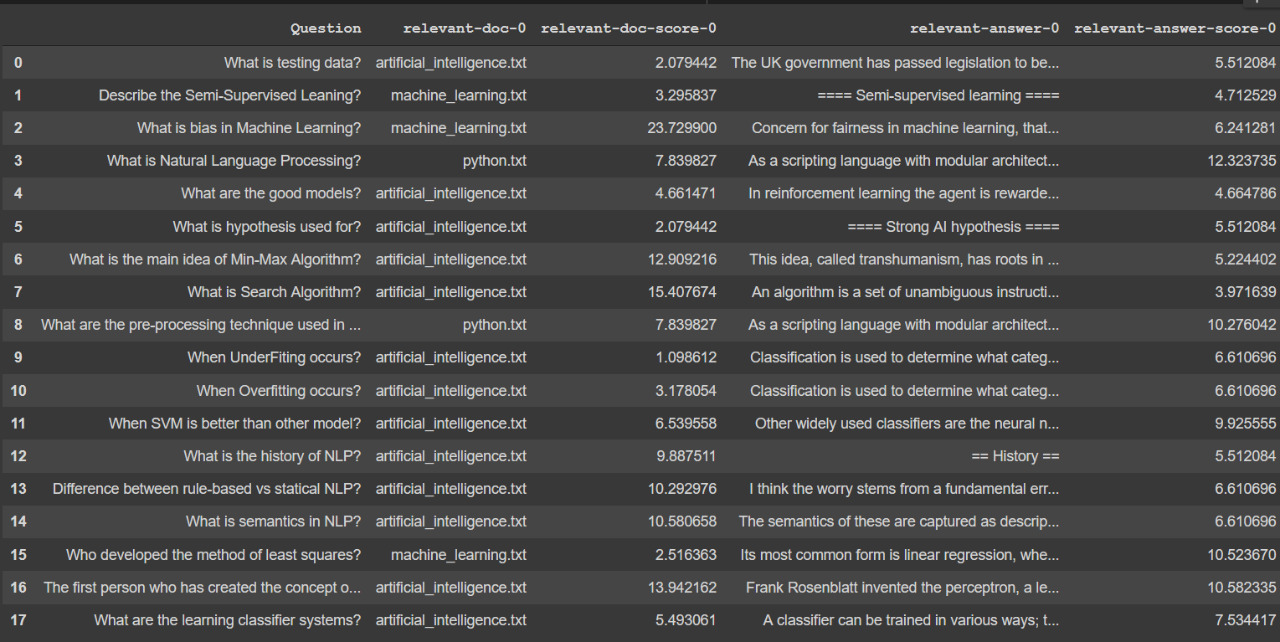
**Comparison between both Computation methods** :

File Score = Σ [count(word)<file> \* idf(word)]

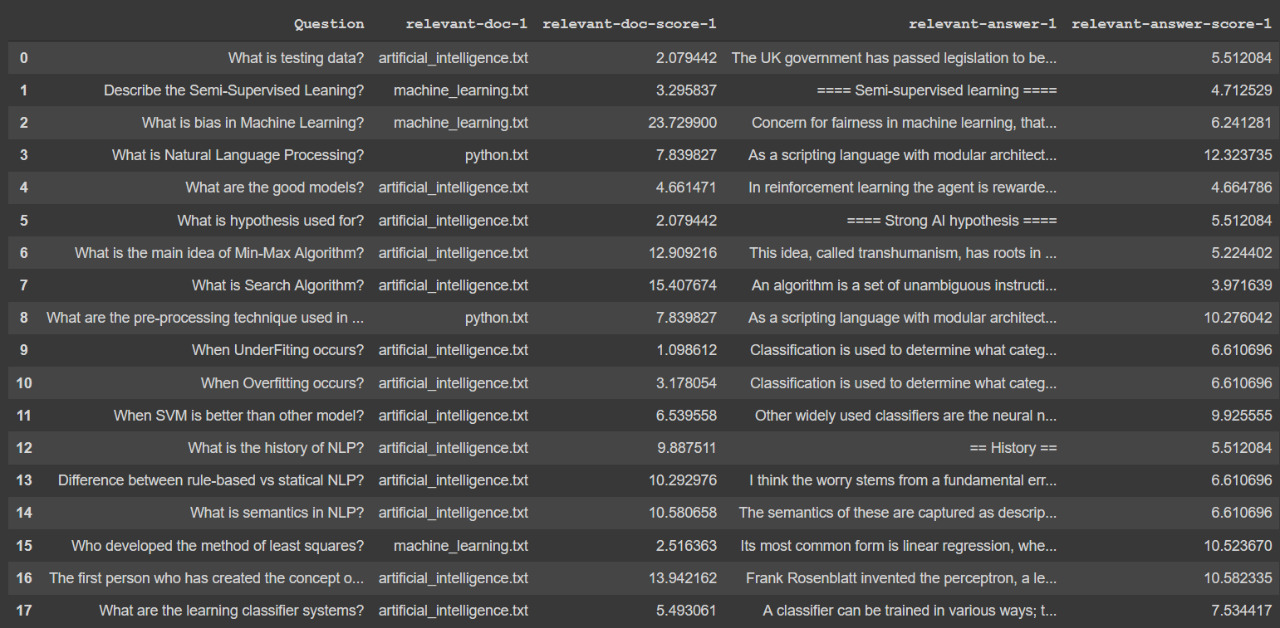
File Score = Σ[Count(word) <file> / length<file> \* idf(word) ]

The set 90 questions query was performed on these two algorithm.The questions was read from the file and then the relevant document file among the corpus was selected and within that file also top sentences were selected for answer.The answer were categorized as 1,2 and 3 sentence.

The answer with the coresponsing questions were written on the files which were later compared.



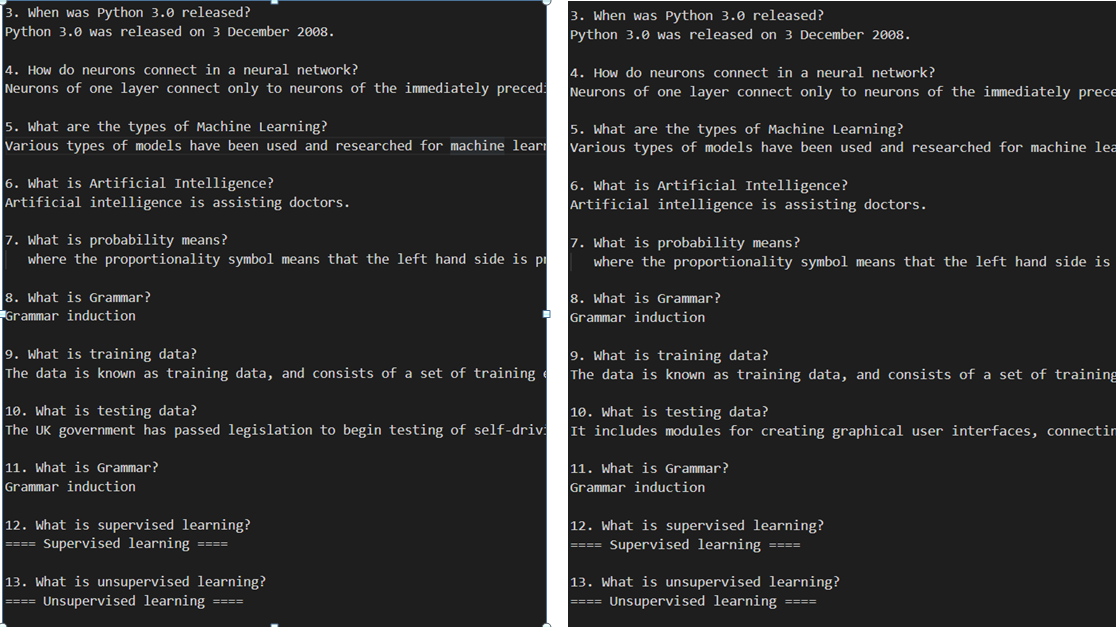
Questions asked on query with the relevant docs and answer with their respective score ( normalized formula).



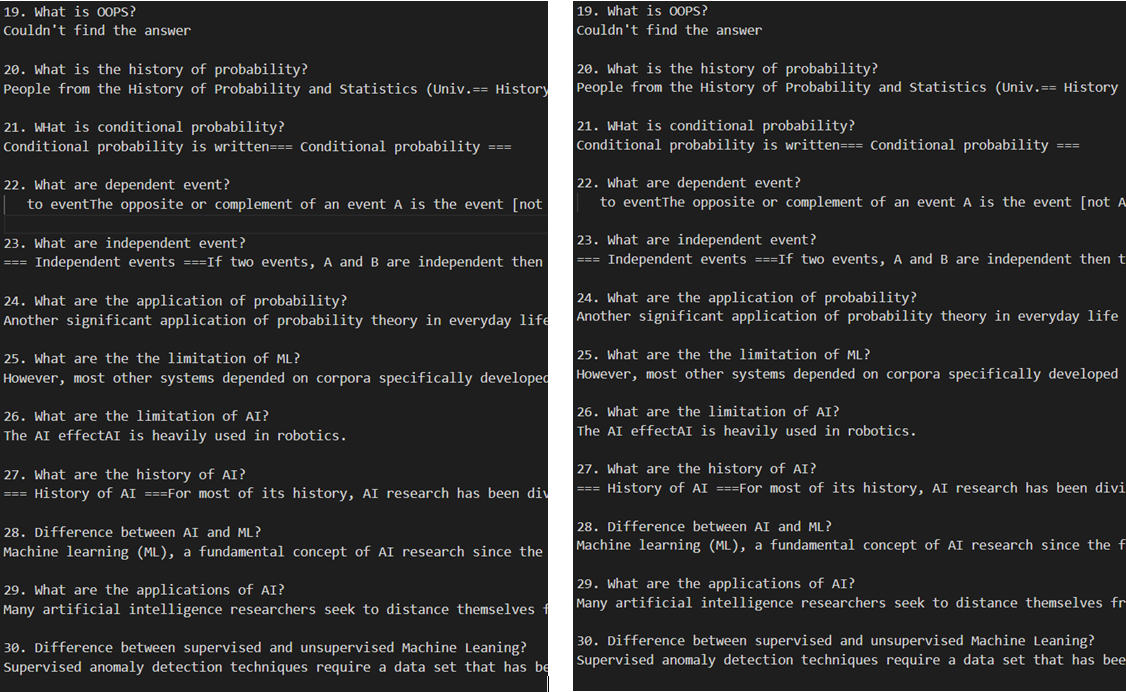
Questions asked on query with the relevant doc and answer with their respectiv score(unormalized formula)

The below shown figure are the comparisons of the result returned using different methods:

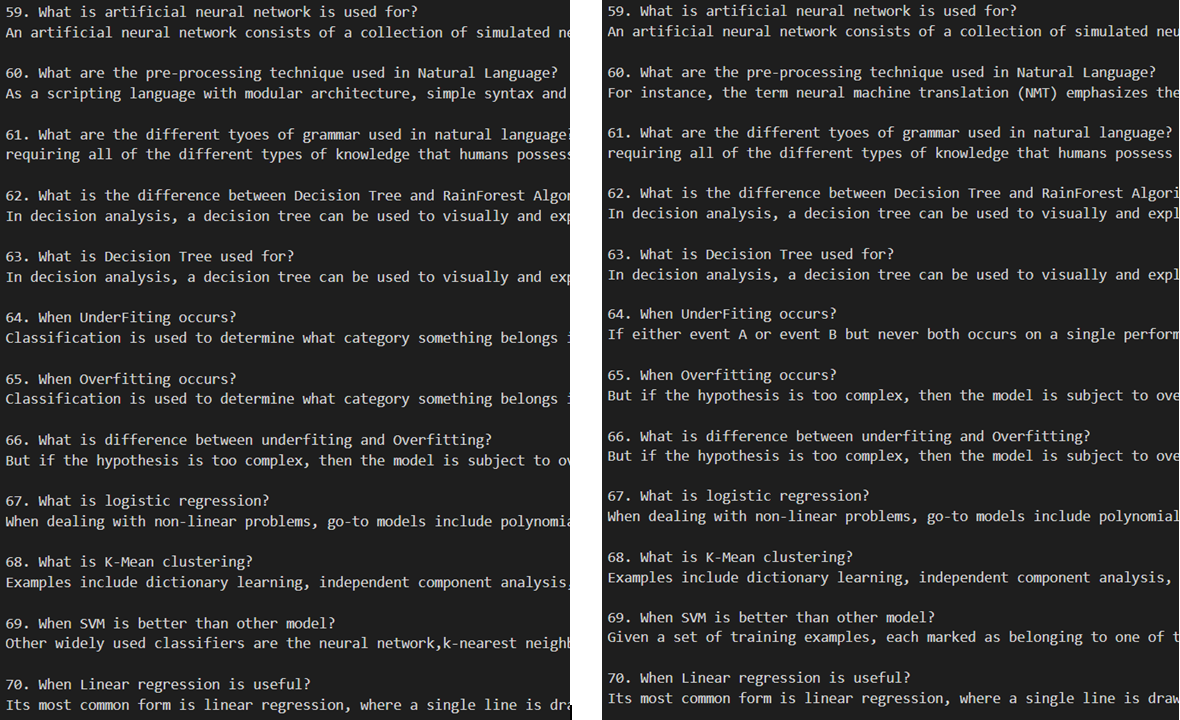
With number of matching sentences = 1



With number of matching sentences = 2



With number of sentences = 3



CONCLUSION

The set of 90 questions query was performed and the result of both the methods were compared by matching the characters present in them.

After comparing the both formulas over 90 questions by using the parameter number of matching sentences to be returned = 1, 2, 3

We found that out of 90, 73 answers were same in all the parameters for number of matching sentences (1, 2 ,3) that had taken. That means 81.1111% were same answer.

Although it is found that in the different answers cases. Some answers are appeared to most resemble accordance with the query. i.e. Normalizing the term frequency by dividing it with the length of document.

We can conclude that normalization of term frequency leads to the better information retrieval.This is because as the total length of documents can vary from very small to large, so it is a possibility that any term may occur more frequently in large documents in comparison to small documents. So, to solve this issue, the occurrence of any term in a document is divided by the total terms present in that document, to find the term frequency.

But the main point to be noted is that both techniques are mostly same.

PROS AND CONS:

Pros:

It is easy to compute as it does not involve complex calculation thus making it simpler than the othe algorithm.It has some basic metric to extract the most descriptive terms in a documentIt is easy to compute the similarity between 2 documents using it.Measure the uniqueness and relevance of content effectively.

Cons:

The limitations with the TF-IDF algorithms that need to be noted that the algorithm cannot identify the words even a slight change for example change in tense. It will consider ‘go’ and ‘goes’ a different entity, ‘play’ and ‘playing’ as different entity. Due to this limitations when TF-IDF algorithm is applied, sometimes it gives some unexpected results.

TF-IDF cannot check the semantics of text in the documents and due this it only useful until lexical level.

There are many techniques that can be used to improve the performance and accuracy such as Decision Tree, Pattern or rule based classifiers, SVM classifiers, Neural Network classifiers and Bayesian classifiers etc.

SOLUTIONS TO OVERCOME LIMITATIONS:

Preprocessing of the documents can effect the results we are getting.Stemming process can be used to overcome the issues of TF-IDF not being able to identify that ‘go’ and ‘goes’are basically the same words. Secondly the Stop Words can be added as much as possible so that the words that are not of any values as ‘the’ , ‘for’ are filtered. These will ensure to some extent that you are getting useful words as output.

Many improved version of tf-idf algorithm has been proposed.Those algorithm incorporated hill climbing for boosting performance.A variant of tf-idf algorithm can be applied in cross language by using statistical translation.Gentic algorithm can be also used to improved the tf-idf.

REFRENCES:

[1] Shazad Qaiser,Ramsha Ali (2018):Text Mining:Use of TF-IDF to examine the relevance of words to documents. July 2018 [International Journal of Computer Applications](https://www.researchgate.net/journal/International-Journal-of-Computer-Applications-0975-8887) 181(1) DOI:[10.5120/ijca2018917395](http://dx.doi.org/10.5120/ijca2018917395" \t "_blank)

# [2]Arfiani Nur Khusna,Indri Augistina Informatics Engineering Universities Ahmad Dahlan Yogyakarta,Indonessia. Implementation of Information Retrieval Using Tf-Idf Weighting Method On Detik.Com’s Website October 2018

DOI:[10.1109/TSSA.2018.8708744](http://dx.doi.org/10.1109/TSSA.2018.8708744) Conference: 2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)

# [3]S. Jabri, A. Dahbi, T. Gadi and A. Bassir, Ranking of text documents using TF-IDF weighting and association rules miningApril 2018 DOI:[10.1109/ICOA.2018.8370597](http://dx.doi.org/10.1109/ICOA.2018.8370597) Conference: 2018 4th International Conference on Optimization and Applications (ICOA) Project: [Mining of association rules](https://www.researchgate.net/project/Mining-of-association-rules)

# [4] <https://www.nltk.org/>

# [5] <https://cs50.harvard.edu/ai/2020/>