**ABSTRACT**

The use of TF-IDF (Term Frequency and Inverse Document Frequency) is conferred in examining the relevance of key-words to the corpus. The study is about – *how the algorithm can be applied in number of documents to find the relevance sentences or passage?* *Can changing Conventional Term-Frequency formula can cause any effect in results*? To be specific: -What are the difference in results for the

*TF = (number of word in doc) / (length of the doc)* and *TF = (number of words)*?

First : The working principle of TF-IDF are discussed. Second: The algorithm is used for both approaches for accounting TF. After that both approaches are compared. After that strength and weakness of the algorithm is compared. How the weaknesses can be handled.

**INTRODUCTION**

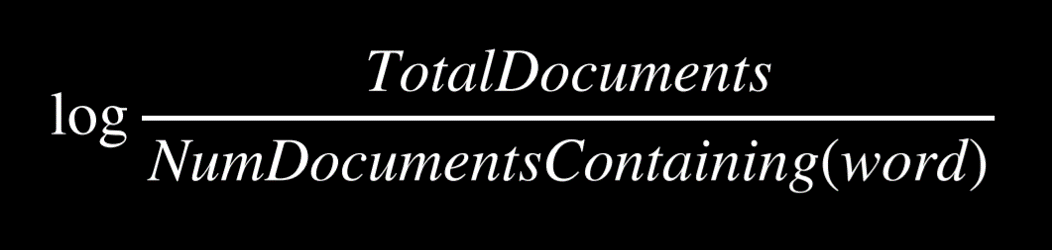
The processing of text data is always an interesting and challenging job in an organization. There are many techniques/methods/algorithms are discovered till now, but this study is focused in particularly TF-IDF methods. TF-IDF is a statistical measure that evaluates how relevant word is to a document 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).

**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.

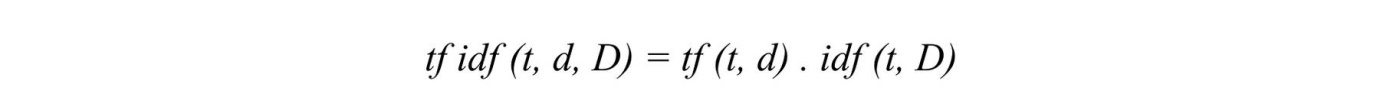


So if the word is very common and appears in many documents, this number will approach 0, otherwise it will approach 1.

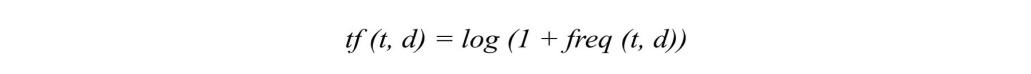
**TF-IDF**

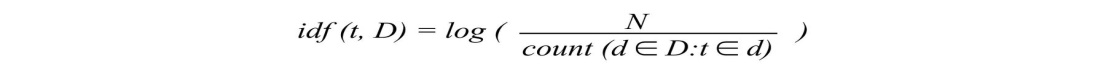
Multiplying these two values results in the TF-IDFs score. The higher the score, the more relevant that word is in that particular document.

The TF-IDF score for the word ‘t’ in the document ‘d’ from the document set ‘D’ is calculated as follows:



Where:





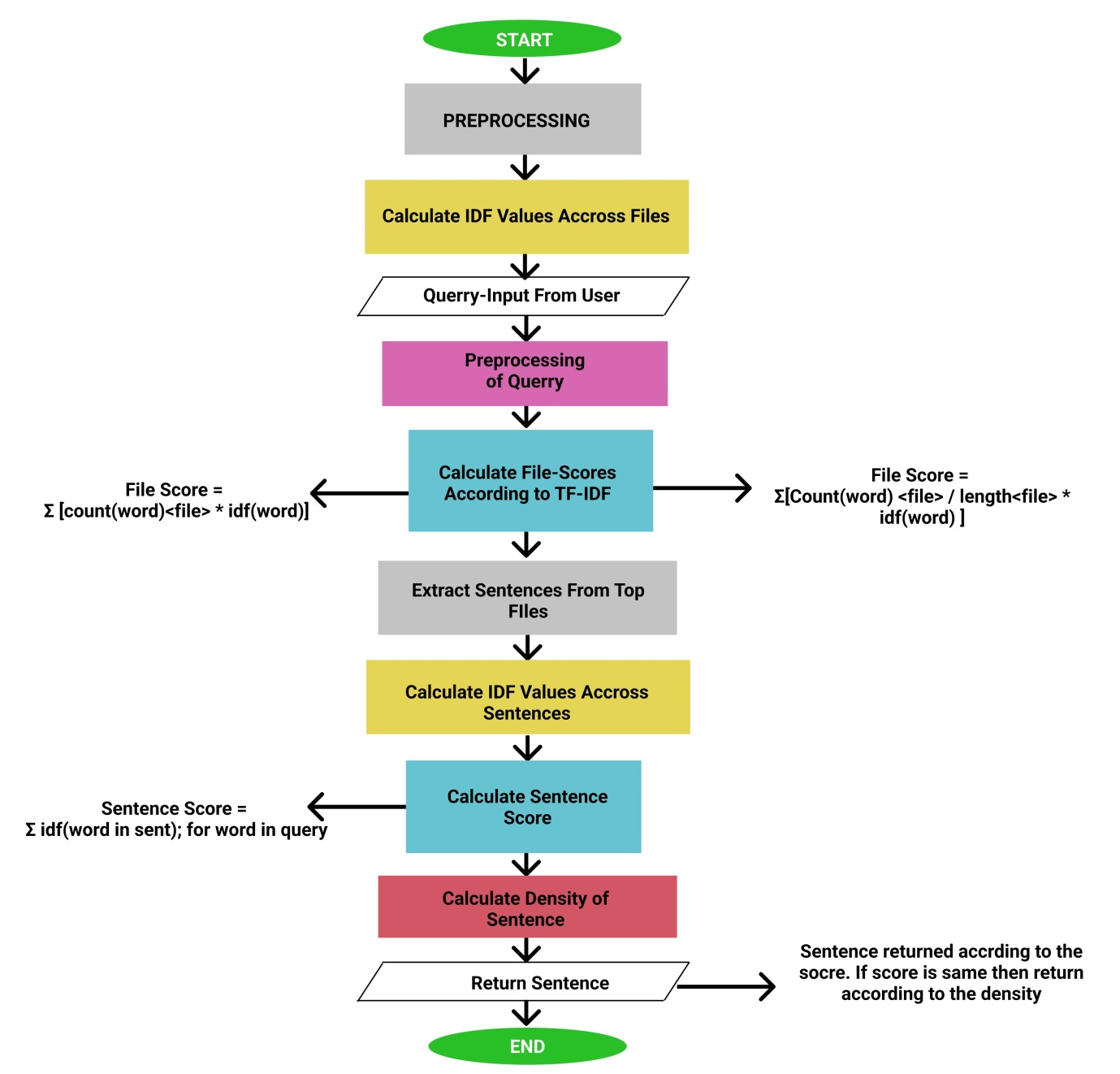
**What is the Flow of Algorithm?**

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. The user will asks some queries that the answer might be present in those text documents. The algorithm first find out the most relevance document among of them. And again it will look for the most relevance information based on the query the used has asked.

The flow starts with the preprocessing of the data ex. tokenization, removing stop words and etc. After that IDF value of very set of pre-processed words are calculated across the documents. 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 files form the bunch of documents. After that the computation limits to the particular documents which is in top list. 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 |

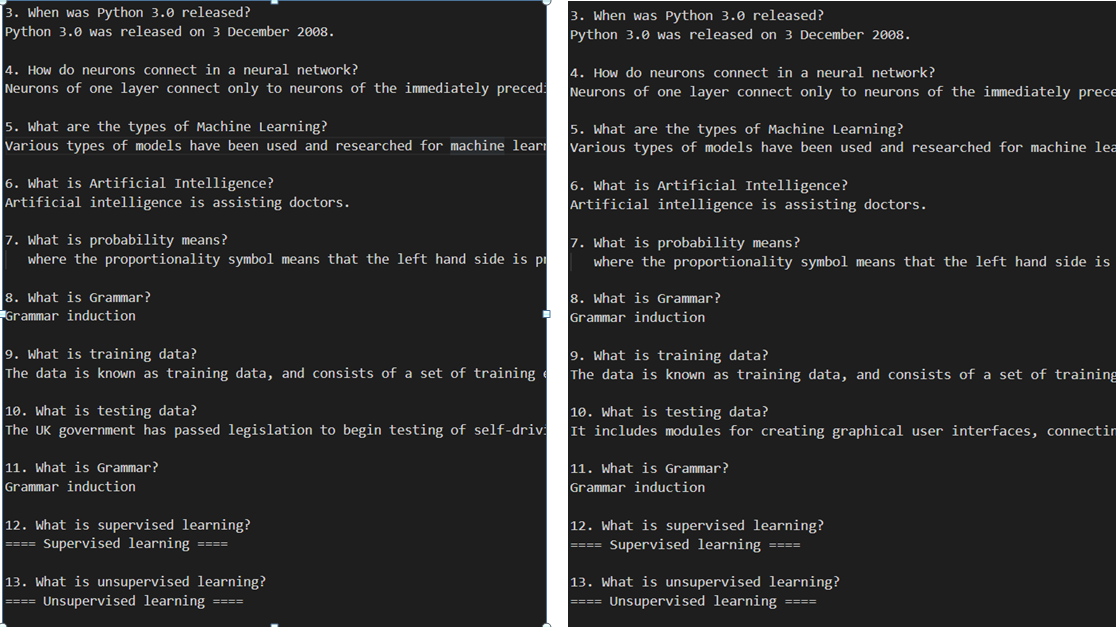
…..

**Comparison between both Computation methods** :

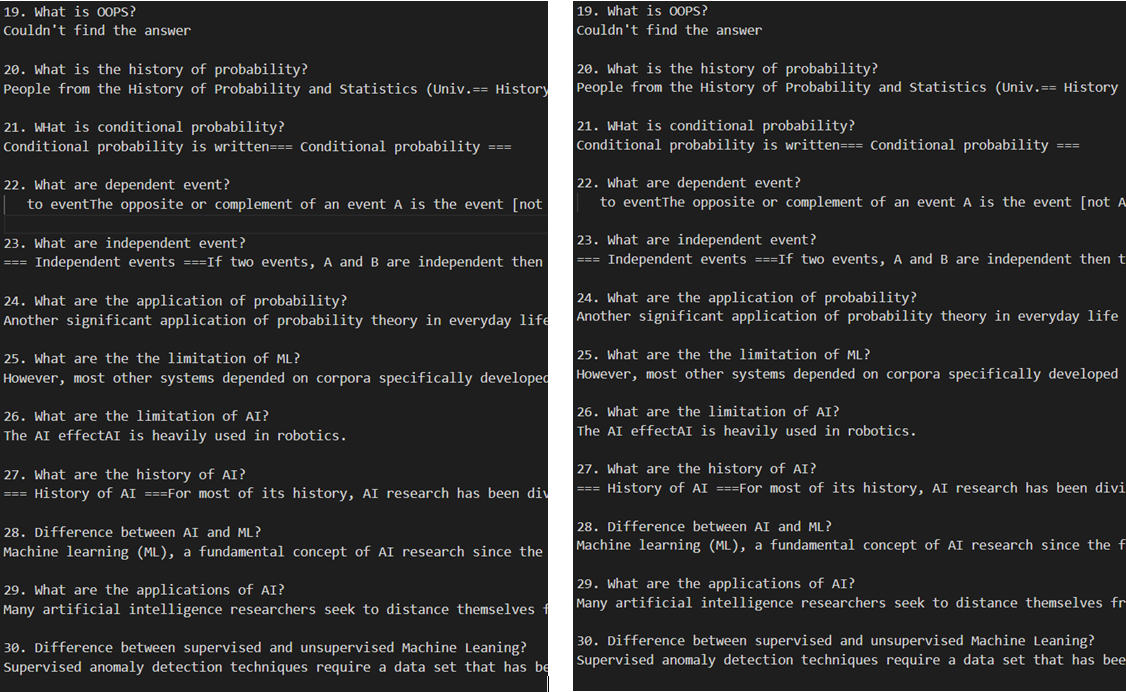
File Score = Σ [count(word)<file> \* idf(word)]

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

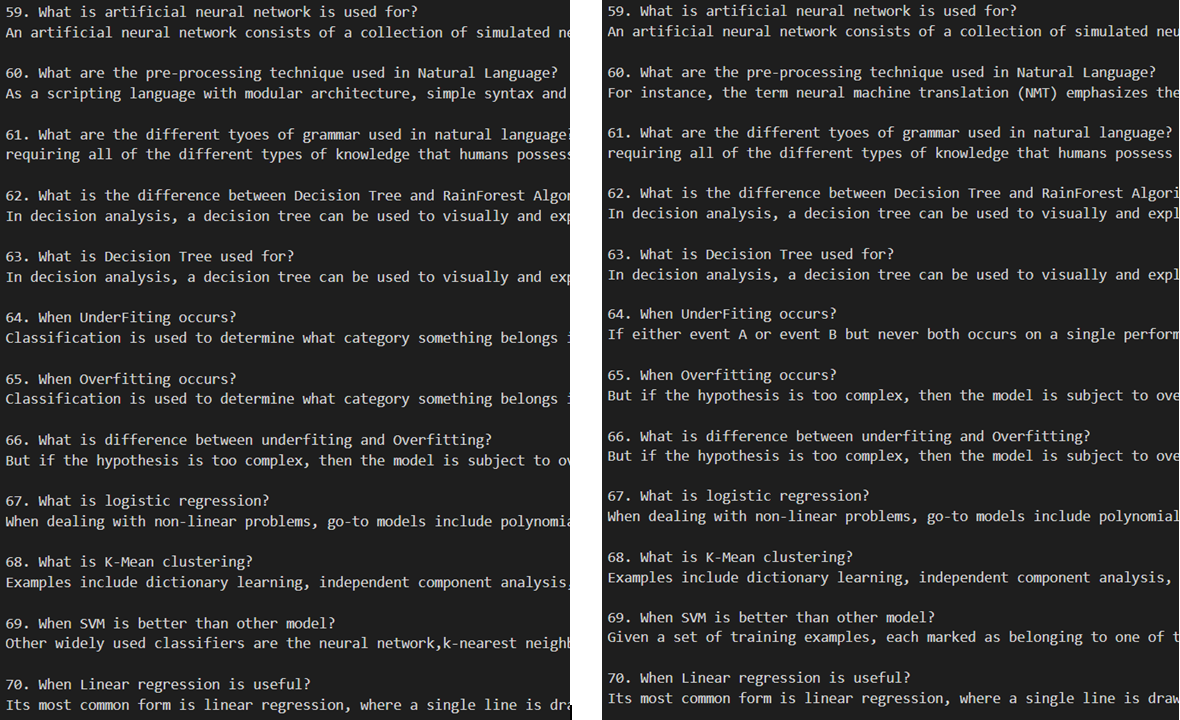
with number of matching sentences = 1



With number of matching sentences = 2



With number of sentences = 3



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. Ie. Normalizing the term frequency by dividing it with the length of document.

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

**Limitations:**

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:**

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.