**Project Report**

On

**Text Summarization and Document Similarity**

**By**

**ANUPREET SINGH 2017A7PS0955G**

**RITURAJ ROY 2017A7PS0957G**

**At**

**IDS INFOTECH, IT PARK, CHANDIGARH**

**A Practice School-I Station Of**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**(July,2019)**

#### Acknowledgement

We would like to extend our sincere gratitude towards Mr Pratap K Aggarwal, Managing Director of IDS Infotech Pvt Ltd for giving us the opportunity of interning at his prestigious company as a part of the Birla Institute of Science and Technology, Pilani, PS program.

We are extremely grateful for the support of our Project Mentor Anirudh Munj, without whose guidance and support we could not have achieved the progress we have in data analysis on titanic disaster data set project. He familiarized us with python libraries and methods of data analysis.

We wish to acknowledge the help from our PS Faculty Incharge, Dr. Rajeev Taliyan for guiding us through the Practice School Course.

**Abstract**

Document Similarity can be calculated using a number of techniques such Jaccard Similarity, Cosine Similarity, Cosine Similarity with TF-IDF etc. But due to the way cosine similarity works, larger the document is, larger are the chances of errors. So, we take an unconventional approach towards finding similar documents by using Cosine Similarity on the summary of the documents generated using TF-IDF. This approach will have lesser chances of error as it only considers the broader and more important details of the documents. To reduce the summary size, we score all the sentences accordingly by their importance and use the average score to generate a threshold for the summary. After generating the summary, we use cosine similarity on these to get similarity values. The final output will be a list of files in sorted order based on amount of similarity between these and the input file.

**THE MATHS**

**TF-IDF**

TF-IDF stands for Term Frequency-Inverse Document Frequency, and the TF-IDF weight is often used in data analytics and text mining. The TF-IDF weight for a term is generally calculated by multiplying the TF and IDF of that term. Variations in this calculation of weight can depend on the type of model being used. Individually the terms are calculated as follows:

Term Frequency (TF): **Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones.

Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

Inverse Document Frequency (IDF): **Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

*IDF(t)* =

Finally, the TF-IDF of a term is calculated as follows:

The higher the TF-IDF value, the rarer the term is and smaller the value, the more common the term is.

Benefits of TF-IDF include:

1. Removing stop words.
2. Find words with higher search volumes
3. Make sure the terms used are unique and relevant to the user.

**COSINE SIMILARITY**

Cosine Similarity is a metric used to find how similar or closely related two documents are. Mathematically, it is the cosine of the angle between two vectors in Three-Dimensional Space. The lower the angle, more the cosine similarity.

Cosine Similarity is extremely advantageous as even if the Euclidian Distance between two documents is large i.e. the size of the documents differs by large amounts, Cosine similarity will give precise and accurate results.

Mathematically,

where x, y are two vectors in Euclidian Form.

**DOCUMENT REPRESENTATION**

Several ways exist to model a text document. The most common one being the “Bag of Words” implementation. The Bag of Words model is widely used in text mining and information retrieval. In this implementation, the frequency of each word is its weight, meaning that the terms that appear more, will have a higher importance in describing the document.

Although more frequent words are more important in Bag of Words implementation, this is not necessarily true in practical purposes. For example, words like ‘a’, ‘the’, ’in’ are neither descriptive or important for the Document’s subject. Thus, a more complicated strategy of tokenizing the words is used before applying any algorithms.

Terms are basically words in context of Natural Language Processing. Before vectorizing a term, we apply several standard transformations. First, we remove the stop words. These are words that are non-descriptive for the topic of the document, such as ‘a’, ‘and’, ‘are’ and ‘do’. Several Python libraries provide us with several stop word removal techniques. Second, we use stemming such that words with different ending will map to the same root word. For example, ‘production’, ‘produce’, ‘produces’, ‘product’ will be mapped to the term ‘produc’ after stemming. Finally, we break each text document sentence to produce its summary.

**SUMMARY GENERATION**

There are two types of summary generation algorithms namely extractive and abstractive. While extractive algorithms work on obtaining the most important and content defining sentences from the document, abstractive summarization is based more on extracting important words and forming sentences on its own. To maintain document consistency, we will be doing extractive summarization.

Firstly, we will be finding the TF (Term Frequency) of each word in the preprocessed document using the formulas mentioned above. Same way, we will be finding the IDF (Inverse Document Frequency) using the same method as mentioned above. Multiplying these two would give us the TF-IDF values for each word. This will be the weight of each word in the document. More common the word is, lower the TF-IDF. Now, we will assign a score to each sentence based on the summation of TF-IDF values of each word in that sentence. This will play a major role in text summarization. Doing this for each sentence, we will get an average sentence score for the document and that score will be used to calculate the threshold to be considered for generating text summary.

The sentences having a score greater than the threshold score will be included in the summary. The output summary is comprised of meaningful sentences instead of random phrases. Thus, this process can be used for summary generation as well. The length of the summary can be modified by multiplying the generated threshold value by any integer ‘x’ that will be user generated. In our case, x=1.

**DOCUMENT SIMILARITY**

Now that we have generated the summary of all the documents in our dataset and the input document, we will use cosine similarity to find documents that are similar to the input.

Firstly, summary of all the files in the dataset and the input file will be generated using TF-IDF and stored. This will reduce the length of the text files for better cosine similarity functioning.

Now each summary is preprocessed again by removal of stop words and vectorized. Next, cosine similarity will be calculated by using the formulae provided above for each document with respect to the input document. The values will be sorted and a list consisting of names of the documents will be returned to the user. This list will be sorted with the most similar document being on top.

A pictorial representation for these summary values for different inputs is given below.

**Bibliography**

1. Numpy : https://www.numpy.org/devdocs/
2. Pandas : http://pandas.pydata.org/pandas-docs/stable/
3. Jupyter Notebook
4. Seaborn : https://seaborn.pydata.org/
5. Matplotlib: https://matplotlib.org/users/index.html