IR ASSIGNMENT 1

Team Members:

Abhirath NB	(2020A7PS0260H)
Shreya Koditala	(2020A7PS0176H)
Neha Gunta	(2020A7PS0073H)
Udandra Rohith Siddhartha	(2019A_PS0683H)

Problem Statement:

Retrieving a passage/paragraph from the sections like exclusions/inclusions for the documents (Any kind of documents like word/PDF/Image). Each policy documents have multiple sections like inclusions, exclusions, conditions, definitions, extensions, covered sections, and so on. Each document must be extracted with its text information. From the policy documents, the section's (Already mentioned above) entire passage/paragraph needs to be retrieved depending on the query.

Application Architecture

We have chosen to complete this assignment using python, owing to the wide range of libraries it provides that makes the task of preprocessing quite easy and efficient.

1. Pre-Processing

Extraction of text from documents: We have used **PyMuPDF library** to extract text from the documents of the corpus. In order to extract the paragraphs, we have figured out the most used font styles of the document and identified the paragraphs based on that.

Stop Words: To increase the computation and space efficiency, we have removed the stop words, which are the most commonly occurring words that don't give any additional value to the document vector. We have used the **NLTK library** to get the list of stop words.

Tokenization: We have used the **PunktSentenceTokenizer** class of NLTK package to tokenize the documents in corpus.

Stemming: We have used Porter's Stemming Algorithm to perform stemming on the terms of the corpus. We have used the **PorterStemmer class** of the NLTK package to achieve the same.

2. Indexing

We have identified and differentiated the paragraphs of the corpus by comparing font sizes, and using the most used font-size statements as paragraphs. We then indexed the terms in paragraphs and created tf,df matrices.

3. Query Processing

The query processing is done using a basic tf-idf scoring system and then the best 3 results are retrieved and shown in the result.

4. Retrieval and Ranking

We have used the TF-IDF technique to find how related the terms and documents are. TF-IDF assigns a weight to each term in a document by considering the term's frequency within the document (TF) and the inverse document frequency (IDF) of the term across a corpus of documents. The idea is that terms that appear frequently within a particular document but are rare across other documents are likely to be more important for that document's meaning. In the TF-IDF scoring, we have used normal ‘nt’ scoring technique, which uses term frequency as it is and uses Inverse document frequency, which is the log of Number of paragraphs divided by the document frequency if the document frequency is not zero, else just zero.

Data Structures Used:

- DataFrames (Matrices): DataFrames are a great way to store large quantities of data while also can be used like a dictionary, in python, to store equivalent data. The data stored in this structure is very optimized and hence, retrieval is faster in such scenarios.
- Set: Sets are great data structures when it comes to filtering redundant text from the entire data, and hence can be used to identify the unique words in the corpus.
- Dictionaries: Even though DataFrames can be used in place of a Dictionary, when it comes to smaller amounts of data, retrieval in a dictionary is much faster compared to a DataFrame, which works well with large amounts of data