**INFORMATION RETRIEVAL (CS F469)**

**Assignment 1 – Design Document**

|  |  |  |
| --- | --- | --- |
|  | **NAME** | **ID No** |
| **1.** | S.Devendra Dheeraj Gupta | 2017B5A70670H |
| **2.** | K.Srinivas | 2017B3A70746H |
| **3.** | Abhirath singh | 2018A7PS0521H |
| **4.** | Ritvik C | 2018A7PS0180H |

**Group Details:**

**Abstract:**

The problem of retrieving relevant documents from a large document collection can be approached using many different algorithms. The three classic models used in information retrieval are Boolean, vector and probabilistic. This paper describes a specific implementation of an information retrieval system that is based upon the vector space model. The task is to build a domain specific information retrieval system. The domain we chose for the same is Research Articles**.**

**Pre-processing:**

1. **Acquiring Corpus:**

We have chosen research articlesas the domain for the purpose. The read\_files function takes the path of the folder containing corpus and stores them in a dictionary data structure – docs.

1. **Tokenization:**

The invertedIndex function does three important tasks that are essential for any information retrieval system. It does tokenization of documents, removing punctuations from the documents and performs stemming for the tokens before storing them in the final index. PorterStemmer from the python nltk package has been used for the same. Tokenization is achieved using word\_tokenize from nltk package. Stopwords are acquired from nltk.corpus. The stemmed tokens are stored in inverted index using a list data structure - vocab.

1. **Tf and idf:**
   1. **Tf:**

Term frequency is the count of how often the specific term appears within a document and is calculated so as to be independent of the length of the document.

**TF = 1 + log(t)**

Where t is the count of the term in a document. The termFrequency function does exactly this by taking the vocab and docs as parameters.

* 1. **Idf:**

df measures the frequency of the term across the whole document collection. Idf is the inverse of df such that rare terms are given importance and more frequent terms are given less importance. The same is calculated using the following formula

**IDF = log(N/df)**

The docFrequency and inverseDocfreq does the exact same function. The final idf weights are stored in a dictionary as key-value pairs.

* 1. **Tf-idf:**

The tf-idf score ofa term is the product of its tf-weight and idf-weight. The scores are calculated as

**w = log(1+tf) \* log(N/df)**

This is achieved using tfidf function. This function takes vocab, tf scores, idf scores and corpus as parameters and returns tf-idf scores in a dictionary data structure.

1. **Vector Space Model:**

The vectorspacemodel function finds the top 10 relevant documents related to the query passed to the function as an argument. The calculated scores are then sorted in descending order and the top 10 results are returned among all documents.

1. **Preprocessing and Query Retrieval times:**

Preprocessing time – Around 20 seconds for 20k unique terms in the inverted index

Query retireval time – 13 seconds.(Depends on size of query)

1. **Precision and Recall Calculations:**

A test file from 10 documents is created by selecting some portions from each document. 6 out of 10 relevant documents are retrieved. Therefore recall = 0.6.

Out of 10 retrieved documents 6 are relevant. Therefore precision = 0.6. Total corpus contained a total of 60 documents.

The whole process can be represented as follows.

