Information Retrieval Assignment-1
Design Document

1. Introduction

The following document is the design document for the Information Retrieval (CS F469) Assignment #1.

Contributors:

- G V Sandeep (2014A7PS106H)
- Kushagra Agrawal (2014AAPS334H)
- Snehal Wadhwani (2014A7PS430H)

1.1 Aim

To implement a retrieval system based on vector space model on the given dataset containing weblogs of Apple products and services .

1.2 Scope

This document describes the implementation details of the information retrieval system and its functionality. This implementation uses **vector space model** for normal document search and **boolean retrieval model** for phrase search.

1.3 Definitions, Acronyms and Abbreviations

- tf : Term Frequency
- idf : Inverse Document Frequency
- Category : Each blog post in the corpus belongs to zero or more categories.
- Outlinks : Number of references or links in the blogpost to the external content.
- Inlinks : Number of links citing a particular blogpost
- Phrase Query : A query is classified as a phrase query if it starts and ends with ‘ “ ‘

2.Design Overview:

2.1 Description Of Problem:

To list out the most relevant documents from the corpus based on the inputs from the user.

The query processor takes the following inputs:

- A Query string
- Date Range (Default : The entire time range)
- Category (Default : All categories)

2.2 Technologies Used:

- Programming Language used : Python v2.7.12
- Data set in the form of CSV

2.3 System Architecture:

- User Interface:
  - Tkinter module of python has been used to design the GUI for the application.
  - The GUI consists of a text input where the user can submit his query.
  - It has a drop down menu to select Date Range.
  - It has a group of radio buttons for selecting the category.

- Data Description :

Data extracted from the CSV file is stored as a list of list of lists in the variable called megaList.

Every list in the megaList represents a document which contains the following components:

- Title : A list of normalized words in the document's title. The words are tokenized using python's nltk package and normalized using the Porter's Stemming algorithm.
- Date : The date string of the document is converted into an UNIX friendly timestamp using mktime function of python
- Categories : It is list of categories to which the document belongs to.
- Post : It contains the tokenized and normalized words of the post.
- Outlinks, Inlinks, Comments : These numbers have been normalized using the formula 1+log(num) where num is their value.
- Permalink : Permanent link to the blogppost.
- Megalist also contains the title as it is.

- Data Storage Model:
  - The inverted indexes built are in the form of a dictionary where each unique word forms the key and has a value as another dictionary. This dictionary again has the document number in which the word exists as key and a list of positional indexes of the word as the value.
  - The idf of each word is stored as a dictionary with every word as a key and its idf as value.
  - The tf of each word is stored as a dictionary with every word as a key and another dictionary as its value. This dictionary has every document that it occurs in as a key and it's tf as the value. These tf values which form the document vector are length normalized.

2.4 Application Operation:

Phrase Query :

The function takes in two words in the phrase query at a time, normalizes and tokenizes it, and then searches the positional index for the docs in which both the words occur. Post the execution of the function, it returns a list containing all the docs in which the phrase occurs.

Free Text Query :

The string entered as a normal query (i.e. without the double quotes) is tokenized and normalized in a similar way to the text of the document. The term frequency of every word in the query is calculated then multiplied with the idf of the word to get the total weight of the word in the query. These values are length normalized to get a unit query vector (much like the document vector). The cosine similarity to every document is then calculated giving separate values for similarity with the document's title, blogger and post which are added up to give the total document score w.r.t the query. The top 10 documents based on the score (if they exist) are extracted. If two or more documents are found to have the same score, a further score is calculated based on the document's inlinks, outlinks and comments with inlinks carrying more weight than the other two to resolve this score conflict.

2.5 Dependencies:

- Nltk : A python package used for tokenizing words
- Stemming.py : A python module for normalizing the words using the Porter Stemming's Algorithm.

- Tkinter: A python module for implementing GUI.
- pyDoc : Used for auto generating documentation.
- Math : A python module used for calculating logarithms, squares and square roots.
- Datetime : A python module used for converting date strings to UNIX time stamps.

3. References

- For Corpus :
  - R. Zafarani and H. Liu, (2009). Social Computing Data Repository at ASU [http://socialcomputing.asu.edu]. Tempe, AZ: Arizona State University, School of Computing, Informatics and Decision Systems Engineering.
- For Porter's Stemming Algorithm Implementation :
  - http://tartarus.org/~martin/PorterStemmer/python.txt
- For the project link :
  - https://github.com/greetsandeep/IR_Assign1
- For Pydoc:
  - https://hg.python.org/cpython/file/2.7/Lib/pydoc.py