Birla Institute of Technology & Science – Pilani
Second Semester 2014-15

Date: 20.02.2015

LAB-2 ASSIGNMENT 1

Course Name : Information Retrieval
Course No : CS F469
Submission Deadline : 10:00 Hrs. On 21.02.2015
Max Marks : 5M
Type : Individual Assignment

NOTE 1: You are free to implement in any language of your choice. You should be comfortable in explaining your approach.

NOTE 2: Assignments of only those students will be evaluated who have attended lab on 20.02.2015 (Friday). It is also important to note that the deadline is a hard deadline.

NOTE 3: Solutions have to be mailed to (h2009399@pilani.bits-pilani.ac.in CC: mehala@pilani.bits-pilani.ac.in) with subject "Information Retrieval Take Home Assignment 1". In Mail Content: Add your ID No. and the timing of your Evaluation slot during 23-25th Febraury 2015.

AIM: The purpose of this task is to design a Permuterm index system for effective document retrieval of wild queries.

BASIC:
A Permuterm index is one of the efficient ways of handling wild card queries. A general wild card query can have any number of * at any position. A Permuterm index uses a special symbol **$** to mark the end of the term. It contains various rotations of each term augmented with **$** all linked to the original vocabulary term. It can also be referred to as term mapping or Permuterm tree. To create Permuterm index we look at every term that goes into the standard inverted index. We then need to create rotations of that term.

Rotation Description:
These are all the possible permutations of the term. We mark the end of the term by a **$**sign. This helps us in identifying where the original term ended. For example, Rotations of term **hello** are:
hello$ ->ello$h -> llo$he -> lo$hel -> o$hell -> $hello
The Posting List of all rotations points to the original term (**hello**). All the rotations and their posting list are stored in a Permuterm Dictionary.

Query Look-Up
When a user enters a wild card query say **X*Y**, We need to first perform a look-up of **Y$X*** in Permuterm dictionary i.e. we want to rotate first so as to push wild card character towards the end and then perform look-up. For example for query = **hel*o**
We will look up in Permuterm dictionary **o$hel***which will lead us to different terms in posting list some of which can be **hello, helio, helmo**. Once these terms are available we lookup these in the inverted index dictionary to bring up the required documents.

It is important to note that Permuterm index usually quadruples the size of the dictionary. So it is required to consider efficient data structure like B-Tree for avoiding problems (heap space, large time complexity).

PROBLEM:

Part 1:

Given a set of documents you need to create an inverted index of the set of terms in documents (excluding stop words) for each term in the inverted index you need to create a permute term index and its posting list as explained above. The terms to generate the inverted index should be space tokenized terms from the documents.

Part 2:

Given a user query your system should be able to effectively retrieve set of documents associated with the query. The Query type can be any of the following X*, X*Y, *X, X*Y*Z where X, Y, Z may compose of one or more characters of the query. For example s*ng will use X=s, Y=ng, similarly s*ng*ng will have X=s, Y=ng, Z=ng.

INPUT

>>Part1: To generate Permuterm Index

Dataset.txt: Line-separated set of documents. Each line denotes one document.
StopWords.txt: File Containing list of stop words

OUTPUT

>>Part1

InvertedIndex.txt: A File Containing Inverted Index
PermutermIndex.txt: A File Containing Permuterm Index

INPUT

>>Part2: Effective Query Retrieval

Dataset.txt: Line-separated set of documents. Each line denotes one document.
InvertedIndex.txt: This is the InvertedIndex.txt file generated as the output of Part1.
PermutermIndex.txt: This contains Permuterm Index of all the terms present in InvertedIndex.txt. This is PermutermIndex.txt generated as the output of Part1
Query: A WildCardQuery entered by users to retrieve documents

OUTPUT

>>Part2

RetrievedDocuments.txt: Contains Set of Documents Retrieved for a query.

SAMPLE

Part1:

INPUT

Dataset.txt

>> big data is the future
>> data is the new oil
>> linkedin predicts data mining and statistics bubble burst

Stopwords.txt

is, the, a, an, have, of for, was, will, were, often, this, that, it, can, who, whom, what, and, or, upon

OUTPUT

InvertedIndex.txt

>> big 1
>> bubble 3
>> burst 3
>> data 1,2,3
>> future 1
>> linkedin 3
>> mining 3
>> new 2
>> oil 2
>> predicts 3
>> statistics 3

PermuteTermIndex.txt

>> big$ big
>> ig$b big
>> g$bi big
>> $big big
……
……
>> data$ data
>> ata$d data
>> ta$da data
>> a$dat data
>> $data data
……

Part2:

Input

DataSet.txt
InvertedIndex.txt
PermuteTermIndex.txt
Query: For Example say (f*e)

Output

RetrievedDocuments.txt

>> big data is the future

We are retrieving these documents using future Look up string e$futur -> future

Deliverables:

1. Working solution of the problem in any language of your choice.
2. Help file briefing your setup, i.e. input requirements, class descriptions, function descriptions, name of the output file generated
3. Approach file: A small document describing the approach you used to develop your system. Keep it short and simple, verbose documents will not yield extra marks. You may include a system diagram (not mandatory, but for your own understanding) to explain your system flow.

----------**BEST OF LUCK**----------