CSCE 561 Assignment 1, Fall 2017

Vijay V Raghavan

Assigned: Sept. 20, 2017

Due: Oct. 2, 2017

---

1. *All details of work for each question must be submitted.*
2. *Staple the question and answer sheet together.*
3. *Make a cover with Name, CLID.*
4. *Number all pages and give an index on the cover page to each question.*
5. *Any sort of cheating will not be tolerated. More information on cheating policy can be found on class webpage.*

Q1. (**20 points**) Using the following definitions for fuzzy-set theory for Λ and V known as Einstein's combination operator, that is

$$a \Lambda b = a * b/(1 + (1 - a) * (1 - b))$$

$$a V b = (a + b)/(1 + a * b),$$

where a and b membership function values of an element x belonging to fuzzy sets A and B.

Verify whether the following set-theoretic axioms hold:

a) Distributive Law: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

b) De Morgan's Law: $\neg (A \cap B) = \neg A \cup \neg B$

c) Absorption Law: $A \cup (A \cap B) = A$

d) Associative Law: $A \cup (B \cup C) = (A \cup B \cup C) = (A \cup B) \cup C$

Q2: Table 1 represents a matrix of documents and terms. Each document is represented by a row. Each element in the row represents the values of membership function of corresponding terms relative to the associated document. Compute the RSVs of the following queries for each of the documents d1 through d4

	t1	t2	t3	t4	t5
d1	0.3	0.5	0.0	0.6	0.7
d2	0.5	0.6	0.3	0.5	0.3
d3	0.6	0.0	0.8	0.1	0.7
d4	0.8	0.2	0.0	0.6	0.3

a) Assume Λ and are defined, respectively, as min and max **(20 Points)**
   1) ¬ (t1 Λ t2 Λ t5) V t4 V (t2 Λ ¬ t2)
   2) ¬ (t1 Λ t2 Λ t4) V (t3 Λ t5)
   3) (t5 Λ ¬ t2) V (t2 Λ ¬ t4 Λ t5)

b) Suppose Λ and V are defined as 'bold' combination operators

   a Λ b = max(0,a+b-1), and

   a V b = min(a+b,1),

   where a and b membership function values of an element x belonging to fuzzy sets A and B. For this part, use **0.5 level fuzzy sets.** **(10 Points)**

   1) t1 Λ ¬ t4
   2) t3 V (t2 Λ t4) V (¬ t1 Λ t5)

Q3: This is a programming assignment where you design and implement your own information retrieval system. This is project contains two parts. In part 1, you will build an indexing component, where a large collection of documents is indexed into a searchable, persistent data structure. In Part 2, you will add a searching component, using vector space model (This will be turned in as a part of assignment #2).

Part 1 **(20 Points)**

Read a set of files, parse them into documents and terms, and produce data structures associated with the inverted index. These data structures will be used in part 2. The program needs to save several data structures to disk, at minimum; these include the dictionary (table of words occurring in the text, appropriate metadata/statistics and pointers to inverted index), the document list (with links to original text) and inverted index itself.

Note: During query processing, the Dictionary part of the Inverted File structure will reside in the main memory.

In order to fully implement the vector space retrieval model you may need to retain several pieces of information about the documents in the dictionary and the postings lists of the inverted index data structures such as the total number of documents, the term weight values for each document, etc. Try to store values that you anticipate are needed to calculate the retrieval status values that can only be generated with high run-time computational cost, if not precomputed and stored. In other words, writing out the necessary intermediate results from Part 1 and reading back into Part 2 may be a better approach than re-computing everything from scratch in memory each time.

Resources

Here are the resources needed for Part 1.

Corpus: Corpus for this assignment is the Cranfield document collection, a set of 1398 abstracts from aerodynamics journal articles. This historically significant dataset was first used in the field of IR for accurate measurement of retrieval performance. A description of the corpus follows:

- cran.all: The Cranfield collection corpus file with many fields. The title (T) and the content of the abstract (W) are the two fields that you may use in this assignment.
- query.text: A set of 225 queries that can be used to test the retrieval performance of your system.
- qrels.text: Query relevance judgements for each query. Each query is accompanied with a set of documents that are relevant in separate lines.

Note: qrels.text is not needed now. It is used in future for system evaluation purposes.

- README.txt: Fine grained description of each column in the dataset files.

Download Link: http://ir.dcs.gla.ac.uk/resources/test_collections/cran/
Stopwords: http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop
Stemmer: http://tartarus.org/~martin/PorterStemmer/

Steps

1) Parse the files containing the documents into individual documents (if necessary) and words
2) Remove stop words and stem (using the Porter stemmer) the remaining words
3) Invert the collection in memory (Each postings entry should have a Doc-ID and the associated term frequency value)
4) Store the data structures associated with the inverted file

Deliverables

1) Corpus Statistics:
    - How many documents are in the collection?
    - How many words are in the collection and how many unique words are in the indexing vocabulary of the collection?
    - How many postings (inverted entries) are created for this collection, the length of the shortest and longest postings lists? Average length of the postings lists?
2) Inverted Index Statistics:
    - How much time did it take to index the collection?
    - How much disk space was consumed to store the inverted index of the collection and related data structures (including the dictionary, document list)?
    - What is the size of the inverted index relative to the size of dataset?

Part 2 (to be turned in along with Assignment #2) (30 Points)

A program that accepts any query from the query file and searches for documents based on the data structures from Part 1. Implement the search algorithm based on vector space model of retrieval. Your program should be able to return ranked results for a query. The program should display up to 20 results in descending order of their RSV values along with the total number of results (non-zero RSV values), snippet of text from document and time taken to retrieve those results.

Weighting Function

Use normalized frequency based on weight of term *i* in the document *j* is calculated as

$tf_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$, Where 1 ≤ k ≤ M, M is the total number of words in document j and f_{ij} is the frequency of i^{th} term in j^{th} document.

Steps

1) Implement Vector Space retrieval model and inverted index based search algorithm
2) User should be able to select a query interactively and obtain ranked results

Deliverables

Code and accompanying documentation: Your code needs to be well documented with comments. The comments should not literally verbalize what the code is doing; instead, it should provide a high-level summary of what the code is supposed to accomplish, listing aspects such as tacit assumptions and invariants (if you are using sophisticated data structures). You also need to include a README.txt file that describes how to compile your program and run it on various datasets.

Implementation

You may code your project in any programming language (such as C#, C++, Java, Python, Perl,VB).