Georgetown University

COSC 488: Introduction to Information Retrieval
Spring 2018

Nazli Goharian (nazli@ir.cs.georgetown.edu)

Project Part 1: Pre-Processing Documents & Building Inverted Index
Date: Jan 23
Due Date: Feb 13 by 11:59 PM

Grading: This assignment is 12 points out of total points (40 points) allocated for projects in the semester. It will be graded on the scale of 100.

Objective:

In this project you identify the tokens and build an efficient and scalable IR System. To build the index, you need to identify the tokens that need to be stored. Your search engine will have several indexes, namely, single term index, positional index of single terms, phrase index, and stem index. Use the sort-based algorithm for building the inverted index. You will compare the timing with the case that is based on the assumption of unlimited memory. Data are from a TREC benchmark dataset.

Parser/Tokenizer Requirements:

1. Identify each token – this is each single term that can be a word, a number, etc. Each token is identified as the one separated from the other token by a space, period, symbols (^,*,#,@, $ …). These symbols should not be stored. However, consider special cases such as those listed under the special tokens below.
2. Perform case folding and change all to lower case.
3. Identify and store special tokens- such as:
   a) Terms such as the following examples may be written in different ways and thus, should be normalized and stored and searched in a common way: Ph.D., Ph.D, Phd, PhD, and phd are the same and should be stored as one common way, for example as phd. Similarly, other cases such as , U.S.A, USA, usa should be stored as a common way such as "usa"; B.S., BS should be stored as "bs"; M.S., MS should be "ms"; etc. (see the class notes for various cases listed on the slides).
   b) Do not remove the monetary values along with their symbols.
   c) Alphabet-digit: Example: "F-16", "I-20". Keep them as one term without hyphen: "f16" and "i20". The alphabets are stored as a separate term if there are three or more letters. Examples: CDC-50 should be stored both as "cdc50", and as "cdc".
   d) Digit-alphabet: Same rule as in (c) but vise versa. Keep the combination as one term and keep also the alphabet(s) after the hyphen as a separate term, if it is more than 3 letters. Example: "1-hour" is stored both as "1hour" and as "hour".
   e) Hyphenated terms: keep terms that have a prefix such as "pre" and "post" before term with the term. Example: "pre-processing" is stored both as "preprocessing" and as "processing". If the pre-fix is not "pre", "post", "re", etc (come up with some common pre-fixes), then also store separately both terms before and after hyphen, plus the combination as one term. Example: "black-tie" should be stored as "black", "tie", and "blacktie". Also consider the case that the term has up to three (1,2, or 3) hypens, such as "part-of-speech". This should be stored as "part", "speech", and "partofspeech" in the single term index.

f) Change dates such as listed below to one of the formats among the same list (see the constraints listed below). Your rule should rule out invalid dates such as "242/11/2004" or "05/40/2000") and do not store them. When changing yy to yyyy note that it is only from the last 100 years (95 can be changed to 1995 but not to 1895 nor to 2095; or 05 can be changed to 2005 but not to 1905 nor to 1805). Example of the date formats are:
MM/DD/YYYY
MM-DD-YYYY
Month Name DD, YYYY (e.g., January 11, 1995)
MMM-DD-YYYY

g) Chang digit formats such as 1000.00, 1,000.00 and 1,000 to 1000.
h) Store the file extensions such as .pdf, .html, etc. without the period.
i) Store Email addresses as one term.
j) Store IP addresses.
k) Store URLs.

4. Identify two-term and three-term phrases - Phrases are to be identified as term sequences that do not cross stop-words, punctuation marks, special symbols like ('.', ':', '@', '#' etc) or special terms (mentioned above). For example in a sentence like "New York is the city that never sleeps", the phrases would be "New York" and "never sleeps". Keep a count of their frequency to filter out non-useful phrases. You also have the option to use a *Part-Of-Speech Tagger* and *Name Entity Taggers* to identify phrases.

Note: Only tokenize text between <TEXT> and </TEXT>, excluding comments <!-- -->.

5. Plug/add the **Porter stemmer** to stem the terms for the lexicon of the stem index.

Index-builder Requirements:

Create several indexes:

a. Single term index -- do not include stop terms
b. Single term (including stop terms) positional index
c. Stem index
d. Phrase index

Read the memory constraint parameter. This parameter specifies the memory requirements in term of number of triples, i.e., amount of data can be kept in memory. Put this constraint as 1000, 10,000, and 100,000 triples. Use sort-based algorithm to create inverted index. Capture system time needed to make the inverted index. This is from the time your IR engine starts to read the documents from the disk to tokenize/parse till the whole collection is indexed, i.e, the inverted index is built and resides on the disk. If the size of the triple lists after processing each document is greater than the size of memory constraint, then you should make memory available before processing the next document by writing the triples onto the disk. Assumptions are:

- Distinct term map for each document that is being processed fits into memory.
- Lexicon fits into memory.

Reports: Provide the statistics for reports 1 & 2 and provide your analysis:

Report 1: Index statistics for each index :

	Lexicon (# of terms)	Index size Lexicon+PL(byte)	Max df	Min df	Mean df	Median df
Single term index						
Stem index						
Phrase index						
Single term positional index --						

Report 2: Running time based on each given memory constraints and each index type using sort-based inversion method

Triple list size / **Statistics**	**1000**	**10,000**	**100,000**	**Unlimited Memory**
Time taken to create temporary files (up to merging)				
Time taken to merge temp files				
Time taken to build Inverted Index in milliseconds (the whole process from reading documents to building inverted index)				

Deliverables – Upload onto the blackboard:

Cover page (1 pt): should contain the following in the exact order as specified:

a. Status of this assignment: Complete or Incomplete. If incomplete, state clearly what is incomplete.
b. Time spent on this assignment. Number of hours.
c. Things you wish you had been told prior to being given the assignment.

Design Document (10 pts): The design document should be written prior to coding. There should **not** be any code in your design document. No specific template is provided to you for your design. You may draw a diagram to show the architecture and the flow of the software components, and provide the write-up of your design decisions.

A working IR Engine, Reports & Analysis of Results (89 pts): A working system, satisfying the requirements. Results should be used to provide a good analysis of your engine. Thus, you are expected to provide a good analysis along with your results. You may be asked to give a demo of your IR engine, demonstrating that all requirements are implemented and are functional, and answering the questions.