

Illinois Institute of Technology

CS429: Introduction to Information Retrieval Fall 2005

Project Part 1: Parsing & Stemming

Date: September 6th

Due Date: September 22nd by no later than 3:15pm on blackboard digital drop box. Live section also must hand in a hard copy of all deliverables but the code.

Grading: This assignment is 10 points out of total points allocated for assignments in the semester. It will be graded on the scale of 100.

NOTE: *You can use IRIS software, or use any programming language to implement this assignment.*

Objective & Requirements

As discussed in the class, one of the components of a search engine is the Parser that identifies each token of the text. For this part of the project you need to modify the existing parser provided to you as a simple search engine prototype called IRIS (you have the option to write the parser from the scratch and not to use IRIS), to be able to parse the SGML tagged documents provided to the class for this project.

In this part of the project you will also experiment the effect of Stemmer in the size of lexicon compare to not using any Stemmer. Plug in the Stemmer provided to you by the class (Porter or Lovins) or any other alternative Stemmer and record statistics (if using any alternative Stemmer, you need to describe that Stemmer as part of you deliverables).

Your parser should be able to do the following:

- 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 (like ^,*,#,@ ...). However, consider special cases such as:
 - Identify and store special tokens- such as
 - a. Monetary terms, keep the entire term (such as: \$45.29). \$45,000 and \$45000 both should be handled as \$45000.00.
 - b. Hyphenated terms: keep terms that have a single letter (a-z) and then a number such as "F-16", "I-20" as one term without hyphen: "f16" and "I20".
 - c. **UPDATED:** Hyphenated terms: keep terms that have a 1 to 2 digit number (in range of 00/0 to 24) and "hour" as a term (such as "1-Hour" or 01-hour) as one term "1hour;" and also keep the term after hyphen as a new term "hour". The same for "week": a 1 to 2 digit number (in range of 00/0 to 52) and "week" as a term (such as "1-week" or 01-week) as one term "1week;" and also keep the term after hyphen as a new term "week". But separate all other cases into two terms such as 52-hour should be kept only as "hour"; 2-man should be kept only as "man", etc.

- d. Hyphenated terms: keep terms that have a prefix such as “pre” and “post” before term with the term: “pre-processing” as “preprocessing” and also keep as “processing”.
- e. **UPDATED:** Change dates such as listed below (a-f) to one of the formats among the same list (see the constraints listed below). Report which one you used. (Your rule should rule out invalid dates (such as “242/11/2004”, “05/40/2000”). Take care of these invalid dates and do not store them. Make sure that if changing yy to yyyy that 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 1995 nor to 1895). If changing yyyy to yy also consider last 100 years; for example 1995 can be changed to 95 but 1895 can not be changed to 95.
- MM/DD/YYYY
 - MM-DD-YYYY
 - MM/DD/YY
 - MM-DD-YY
 - Month Name DD, YYYY (e.g., January 11, 1995)
 - MMM-DD-YYYY
- f. Change Comma separated digits such as 1,000 to 1000.
- g. Keep the tokens including dash, such as 100-99-8978 or 99-56, intact.
- h. **UPDATED:** Change tokens with a period “.” at the beginning such as “.pdf” to “pdf”. These file extensions should be validated to be of length three. Exceptions are MPEG animation alias MPE, MPG; HTML alias HTM; and SGML. If you know of any one else consider that.

- Only parse text between <TEXT> and </TEXT> Don’t parse the code inside the comment (<!-- --> comment).
- Do not include *stop words*.
- Identify pair-wise phrases- Phrases are to be identified as two 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”.

Runs & reports

Use the parser you created/modified for following runs. Note that your parser includes all cases specified above. The only parameter you might change for your experiments is “phrases”, as explained below:

1. Run 1) Use the small collection. Parse the collection and note the execution time. Output first 100 tokens to the file named log_terms_specialcase_yes_phrase_no. Your output should have a count for the total number of terms. Your parser should identify the special tokens and no phrase. (Set the value of PHRASES=NO in your configuration file.)
2. Run 2) Use the small collection. Parse the collection and note the execution time. Output first 100 tokens to the file named log_terms_specialcase_yes_phrase_yes. Your output should have a count for the total number of terms. Your parser should identify the special tokens and phrases. (Set the value of PHRASES= YES in your configuration file.)

3. Run 3) Use the big collection. Parse the collection and note the execution time. Output first 100 tokens to the file named log_terms_big_specialcase_yes_phrase_yes. Your output should have a count for the total number of terms. Your parser should identify the special tokens and phrases (Set the value of PHRASES= YES in your configuration file). (Note: There are several special character in the big collection. You might need to change your parser so that you can process this file).

Based on runs 1,2, and 3 fill out the information in this table:

Run (Execution time in milliseconds	Number of terms
Run 1 (no phrasing)		
Run 2 (with phrasing)		
Run 3) same as Run 2 on big collection		

4. **UPDATED:** Plug/add the **Porter stemmer** to your program. Make sure that the special terms and phrases identified earlier are not stemmed. Use the *large collection* for getting the following statistics and set PHRASES=NO in configuration file. Answer the following items (a-j) in the exact order. :
 - a. Top 70 stems sorted in decreasing order of term frequency in the collection, using Porter stemmer.
 - b. Top 70 stems sorted alphabetically using Porter stemmer.
 - c. How many times each rule has been fired in Porter stemmer. Give the rule name and the number of times the rule is fired. Sort rule names based on the frequency (# of times rule fired).
 - d. Time consumed for parsing, using Porter.
 - e. Identify top 5 rules fired. Keep those 5 rules and remove rest of the rules from Porter. Run it again and print top 70 stems as in (a), Record time for execution.
 - f. Record time consumed using no stemmer.
 - g. Fill up the following table:

Run	Execution time in milliseconds
Porter Stemmer with all rules	
Porter Stemmer with top 5 rules	
No Stemmer	

Deliverables

All students (LIVE, IITV, INTERNET sections) must submit on the blackboard.

The copy submitted on Digital Drop Box should be a single zipped file that includes Summary; Design Document; Results; Readme file; the Source Code ready to compile and run.

1. Summary (10%) include the following:
 - a. Status of the project, complete/incomplete. If incomplete state what is incomplete.
 - b. Time spent on the project. - number of hours.

- c. Problems encountered - List at least 3 to 4 biggest problems that you encountered while you were working on the project and how you solved them.
 - d. Things you wish you had been told prior to being given the assignment.
 - e. Observations- Your observations on the runs and the generated results.
2. Design Document (20 %). The design document should be written prior to coding. There should **not** be any code in your design document. The goal of your design document should be for any programmer to be able to implement the project in any language using only your design document as a reference.
3. Code & Readme file (10 %) submit the portion of your code that relates to this part of the project, and you wrote or modified. You should also submit a Readme file explaining how to compile and run your tests.
4. Results (60%). Submit the statistics specified above under *runs & report*. Enclose the first 100 tokens for each run.