Credit Hours System
CMPN306 - Advanced Programming Techniques

Cairo University
Faculty of Engineering

Search Engine Report
Team (17)

Submitted to: Dr. Michael Nawar

Submitted by: Mariam Mohammed Ameen – 1170258
Lina Ayman Hamza – 1170113
Passant Ahmed Maher – 1170030
Maria Ossama Fathy – 4180292

Web Crawler Algorithms

We use a Linked Hashset because when we want to add URLs in it, they are added in order (1-2-3….5000).

1) Robot Handler

It takes the URL as an input parameter then performs these checks on it:

1- Adds "/robots.txt" to the URL and opens the file if exists.
2- Reads the lines until it finds the line that has "User-agent: *" written.
3- After this line it keeps reading lines until it finds the line with "Disallow:" written.
4- Under this line there will be the URLs or paths that aren't allowed for access.
5- Get these URLs and add them to the setOfRobotLinks if it doesn't exist already.
6- This set is used to check if a link is restricted or not while crawling.

2) No two URLs having the same content

To handle repeated content, we do the following:

1- For each URL, we get a compact string containing the first 4 letters of each paragraph.
2- Then each time we fetch a new URL during crawling, we check if this URL has the same content of another URL that was already added to the setOfLinks.
3- If yes, then we don't add this URL to the setOfLinks and unvisitedLinks.
4- Else, we add the compact string to the setOfContent.

3) Crawl

1- Get first non-visited link and add it to saved links and set it as visited.
2- Put its content in a compact string to check on it later if it is found or not.
3- If found, we won't crawl on it. Else, crawl.
4- Check if the URL is not restricted and get its hyperlinks.
5- Add the hyperlinks in the set of links and unvisited links.
6- After getting all the hyper links remove the URL from the saved links.
7- After each add we output them in files to check on them later if interruption occurred.
8- If we get 5000 pages we stop crawling and break from the loop.

4) Interruption

To handle interruption, we do the following in the main:

1- We read the Saved_Links.txt to get the savedLinks and check if they contain URLs then we add them to the top of the unvisitedLinks to crawl them first.
2- We read the Unvisited_Links.txt to get the rest of the unvisitedLinks.
3- We read the Seeds.txt to get the setOfLinks and check if the unvisitedLinks is empty then no interruption occurred then we put in it the setOfLinks, else we neglect it.
4- We read the Visited_Links.txt to get the visitedLinks.
5- We read the SeedsOutput.txt to get the setOfLinks and their hyperlinks and check if it is not empty then add it back to setOfLinks to output them later
6- Finally, if we are crawling for the first time all these files should be empty so we would start crawling on the seeds, else we would start from the link that we were crawling on when the interruption occurred.

Indexer Algorithms

1) Web Parsing:

we parse the data of each link from all the tags of the html page. {div, section, h1, h2, h3, h4, h5, h6, li... etc.}

2) Removing Stop Words

we remove the stop words from these words then pass these words to the stemmer.

3) The porter stemming Algorithm:

goal: return each word pared from the URL to its root form.
the function name is stem () this function returns the stemmed word after 6 steps of the stemming process.
step 1: remove the plural {s, es} and the verbs conjugations {ed, d, ing...}
step 2: turns y into i if there is a vowel in the word
step 3: turn the double suffices into 2 separate parts ex ization-->ize,ation
step 4: transform parts of the word to its original part {icate-->ic,ative-->at,alize-->al,} , also remove uncessary parts {ful,ness...etc.}
step 5: remove {ant,al,ance,ence,ible,ment...etc.}
step 6: remove if there's an e at the end of the word

4) Preprocessing done and words are ready in a map

we put these words after this in a map where the key is the word, and the value is the frequency of this word in the same URL.

5) Storing the words in the Database with the URL, TF and IDF

we store each word and its URL and the TF and IDF calculated.
In this step we check if this word already exists in the database or not, if it exists, we only add the URL and the TF and update the IDF in the same document, but if this is a new word then we store in a new document with its URL, TF and IDF

Query Processor Algorithms

1) We take the search word from the interface.
2) We call the preprocessing function to remove stop words and pass it to the stemmer.
3) Then we search with this word with a query in the database and retrieve that matched result (URLs).
4) We return it in a basicDBList then pass it to the interface to display it.

Web Interface Algorithms

We are using servlets in java and Apache to make the GUI. We have an index html which is the homepage of the search engine interface which contains the search form.

1) Retrieve results (processReq)

1- It gets the input of the user.
2- Then we make a new instance of the query processing class.
3- We send the input text to the function search query after pre-processing it.
4- It returns an array of all resulted links.
5- We iterate on this link to show them in another page (max URLs in a page is 10).

2) Pagination

1- We calculate the maximum pages from the size of the resulted URLs.
2- We update this number in a class variable and each time we loop on this number to print hyperlinks with the same numbers as the pages count, we also need this count to know where should I start in the loop that shows the links and content.
3- We added the routing of this pages (their number is their route for easy code handling) in the web.xml file.

3) Snippets

1- This function takes the URL and the search word as inputs.
2- It fetches the URL and gets its paragraphs.
3- It loops on the paragraphs and searches for a paragraph that contains the search word.
4- If not found, look for any other element in the URL that contains the search.
5- If still not found, get the content of the body.
6- Now it takes a substring of this paragraph depending on the position of the word in the paragraph/element.
7- The substring is 200 words max.