CS221 PROJECT 3 - M1 INDEX

Student Name	Student ID
Jun Ma	29846938
Chang Liu	47927446
Lu Chen	35888463

1.INTRODUCTION

Search engine indexing collects, parses, and stores data to facilitate fast and accurate information retrieval. Indexing is the first milestone for building a search engine. In this part, we use the pages that we stored by crawling the ics.uci.edu domain in project 2 and construct an inverted index of webpages.

2.DESIGN

In our project, we separate our document processing and indexing design into following steps.

- Read in bookkeeping.tsv under WEBPAGES_RAW directory, eliminate invalid URLs such as as .exe, .dmg and etc.. Reform the url with scheme and save as a list of (ID, URL) entries in validUrl.json. ID refers to the path of html file.
- Read from validUrl.json. For each url, parse its page source and get all the anchor text. Save the anchor text as a map of (URL, Anchor Text) entries.
- Parse each html source page, get the title and the body content separately.
- Design a data structure called Document to map document ID to the corresponding URL string, title content, body content and anchor text. For each part of the content, tags indicating the source of the entity is set. It will help on assignment of different weights to title, anchor and body in further scoring step.
- For each document, get the text belonging to tag TITLE/BODY/ANCHOR and parse them into tokens. Apply stop-word filter and stemmer to process raw tokens, get the useful token list in the stemming format. Build up an Inverted Index table for the tokens, the Array List tagged “pos” inside the baseEntities help to indicate the position and term frequency information.
- Stored the three part of index in a TreeMap, and merge them into one inverted index for a single document. Similarly, merge all documents into one complete index table, saving the document ID in the front of the index entry as reference.
- For each index entry in the complete index table, we calculate the term frequency in each document, as well as the term document frequency. According to the formula

$tf-idf=(1+\log_{10} tf)*\log_{10} idf$, calculate the tf-idf score for each document and save it after the document ID for further querying process.

For now, our index table for the webpages are finalized.

3.INDEX TABLE and ANALYSIS

The number of valid documents in our project is **16394**.The number of unique tokens we get is **157412**. We save the index table in index.txt file, the total size of which is 254MB.

Below is an entry example from our index table.

For the keyword “brent”, the documents containing it includes 2215, 2814,…,17851. For document 2215, we can tell that its tfIdf score is 8.95. In its body part, position 2,14,18 is the word “brent”; in its title part, position 2 is the keyword “brent”; in its anchor text part, position 4 is the keyword “brent”. Another example is document 6748, with tfIdf score 5.21. And it contains “brent” BODY” in position 354,592,1170 in body, but its anchor and title don’t exist “brent”.

INDEX EXAMPLE:

brent:[{“id":2215,"tfIdf":8.952112104593791,"baseEntries":[{"tag":"BODY","pos":[2,14,18]},
{"tag":"TITLE","pos":[2]},{"tag":"ANCHOR","pos":[4]}]},{"id":2841,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[12]}]},{"id":3042,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[19]}]},{"id":3620,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[321]}]},{"id":4467,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[10]}]},{"id":6011,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[23]}]},{"id":6748,"tfIdf":
5.207296303448435,"baseEntries":[{"tag":"BODY","pos":[354,592,1170]}]},{"id":6904,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[7576]}]},{"id":7179,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[862]}]},{"id":9835,"tfIdf":
6.454741032611656,"baseEntries":[{"tag":"BODY","pos":
[9067,9070,9079,9092,9097,9105,35418,46717]}]},{"id":10169,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[12947]}]},{"id":11196,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[24]}]},{"id":11557,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[105]}]},{"id":11719,"tfIdf":
3.81005197709412,"baseEntries":[{"tag":"BODY","pos":[206]}]},{"id":12137,"tfIdf":
9.729082522637206,"baseEntries":[{"tag":"BODY","pos":[1]},{"tag":"ANCHOR","pos":[1,3]},
{"tag":"TITLE","pos":[1]}]},{"id":12695,"tfIdf":3.81005197709412,"baseEntries":
[{"tag":"BODY","pos":[2269]}]},{"id":13946,"tfIdf":3.81005197709412,"baseEntries":
[{"tag":"BODY","pos":[299]}]},{"id":14442,"tfIdf":3.81005197709412,"baseEntries":
[{"tag":"BODY","pos":[375]}]},{"id":15543,"tfIdf":4.6916149955999655,"baseEntries":
[{"tag":"BODY","pos":[530,4571]}]},{"id":16271,"tfIdf":3.81005197709412,"baseEntries":
[{"tag":"BODY","pos":[26]}]},{"id":16341,"tfIdf":3.81005197709412,"baseEntries":
[{"tag":"BODY","pos":[5301]}]},{"id":17851,"tfIdf":3.81005197709412,"baseEntries":
[{"tag":"BODY","pos":[1723]}]}]

4.Further Improvement

Index is a key part for the precision of search engine. Generally, the more comprehensive indexing information we get, the better ranking results we will get for specific search query. So far we put our focus on the title, anchor text and body text of the page source. However, there are other details that can improve for better indexing performance.

1).Web pages may have different header levels, such resulting in different weights in ranking and scoring. We may separate H1, H2, H3 in parsing and relatively give different weight to stress the importance of higher headers.

2)The current index system is based on our WEBPAGES_RAW files, which is obviously a small portion of ics.uci.edu websites. The insufficient web pages may cause incorrect indexing and ranking results. We may expand the url pool with crawler and get more comprehensive indexing information.

3)Intuitively, the url of pages contains keywords that are direct reflection of web themes. We may further take the usage of keywords in url itself into account as well to improve querying accuracy.