COMP 4321 - Project

Leung Ka Wa, 20770807, kwleungau@connect.ust.hk

Program code Structure

java source code

- **Crawler.java** - Crawler class
- **StopStem.java** - StopStem class
- **URLIndex.java** - URLIndex class, use to manipulate the URL.db
- **WordIndex.java** - WordIndex class, use to manipulate the WordDB.db
- **Tester.java** - Tester class, use to test and run the program
- **SearchEngine.java** - SearchEngine class, use to perform IR.

Library

No extra library from lab is used in this project.

Design of the jdbm database scheme

URL.db

It contain of 4 objects. Each of them is a HTree.

- **PageID** - Store the URL and its pageID.

  (**String**)URL = (**UUID**)pageID. Example:

  http://library.hkust.edu.hk/events/staff-workshops/ = b9275b04-58a1-3f4f-ab38-606e30a198a8

  Design decision: A ID mapping.

- **ParentToChilden** - Store the parent to children relationship.

  (**UUID**)parentID = Vector⟨**UUID**⟩(childID). Example:

  5ed456f8-36f3-3ca2-8451-62696e13f7fc = [b9275b04-58a1-3f4f-ab38-606e30a198a8, 9e4a5a31-dbb7-39d0-82ab-8d8b37595564, bd93a542-88ec-3b29-b739-9faf1ffc3bdc, . . .]

  Design decision: Easly can get the out link of a page for later use, e.g. PageRank, hub weight, authority weight and HITS.

- **ChildToParents** - Store the child to parents relationship.

  (**UUID**)childID = Vector⟨**UUID**⟩(parentID). Example:

  5ed456f8-36f3-3ca2-8451-62696e13f7fc = [b9275b04-58a1-3f4f-ab38-606e30a198a8, 9e4a5a31-dbb7-39d0-82ab-8d8b37595564, bd93a542-88ec-3b29-b739-9faf1ffc3bdc, . . .]

  Design decision: Easly can get the in link of a page for later use, e.g. PageRank, hub weight, authority weight and HITS.

- **PageToTitle** - Store the pageID and its originial title.

  (**UUID**)pageID = (**String**)title. Example:

  bd93a542-88ec-3b29-b739-9faf1ffc3bdc = This is the Title

  Design decision: Store the whole title for display use.

- **PageMeta** - Store the pageID and its metadata. My self defined class-**PageMeta** have 4 attributes.

  Date lastModified; **int** pageSize; **int** pageSizeAfterStopStem; **int** pageSizeUnique

  (**UUID**)pageID = (**PageMeta**)data. Example:

  bd93a542-88ec-3b29-b739-9faf1ffc3bdc = (**PageMeta**){lastModified = 2018-11-30 00:00:00.0, pageSize = 100, pageSizeAfterStopStem = 50, pageSizeUnique = 30}

  Design decision: Store the metadata of the page for later algorithm, e.g. tfxidf, also the class can be easily extended to store more metadata.

WordDB.db

It contain of 4 objects. Each of them is a HTree.

- **WordID** - Store the word and its ID.

  (**String**)word = (**UUID**)wordID. Example:

  intellig = b9275b04-58a1-3f4f-ab38-606e30a198a8

  Design decision: A ID mapping.

- **Inverted** - Store the wordID and posting list.

  (**UUID**)wordID = Map⟨(**UUID**)pageID, Vector⟨**Integer**⟩(position)⟩. Example:

  b9275b04-58a1-3f4f-ab38-606e30a198a8 = {9bfc960c-53e4-3faf-8623-b44c251584c1=[1, 5, 10], 114471e0-e3dd-39d8-aa8a-11f77c85a7fa=[50, 60], 8019de9c-bcf5-3600-814b-53ed90ab33bb=[10], . . . }

  Design decision: Store the posting list of the word for tfxidf and phase search. Also, finding the document with highest word frequency is easy.

- **Forward** - Store the pageID and its forward word list.

  (**UUID**)pageID = Map⟨(**UUID**)wordID, Vector⟨**Integer**⟩(position)⟩. Example:

  b9275b04-58a1-3f4f-ab38-606e30a198a8 = {9bfc960c-53e4-3faf-8623-b44c251584c1=[1, 5, 10], 114471e0-e3dd-39d8-aa8a-11f77c85a7fa=[50, 60], 8019de9c-bcf5-3600-814b-53ed90ab33bb=[10], . . . }

  Design decision: Store the forward word list of the page for later algorithm, e.g. tfxidf. Finding the words and their position and frequency in a document is easy.

- **TitleInverted** - Store the wordID and posting list of title.

  (**UUID**)wordID = Map⟨(**UUID**)pageID, Vector⟨**Integer**⟩(position)⟩. Example:

  b9275b04-58a1-3f4f-ab38-606e30a198a8 = {9bfc960c-53e4-3faf-8623-b44c251584c1=[1, 5, 10], 114471e0-e3dd-39d8-aa8a-11f77c85a7fa=[50, 60], 8019de9c-bcf5-3600-814b-53ed90ab33bb=[10], . . . }

  Design decision: Store the posting list of the word in title to favor matches in title.

Running the program

How to run the program

The Tester class is the main class of this program. Pass command line argument to it to run the program.

As I am using VS code to develop this project, I was just simply using the java extension and run the program without mannually compile the project. For me the command line is:

```
/usr/bin/env /Users/boscoleung/opt/anaconda3/bin/java @/var/folders/f1
    /6mvnwxt109n9rswystbch0t40000gn/T/cp_dh97avm16bvprxybpew6los8v.
    argfile Tester <argument>
```

If you want to compile the project mannually, you can run the following command (work on mac):

```
javac -cp ":lib/*" -d bin $(find . -name "*.java")
javac -cp ":lib/*" -d bin $(find . -path ./apache-tomcat-10.1.6 -prune -o -name "*.java" -print)
```

A bin folder containing all the classes will be created.

Then run the program with the following command:

```
java -cp ".:bin:lib/*" Tester <argument>
```

- **-runCrawler** - Run the crawler, progress will be printed to the console. The starting URL and the number of pages to crawl can be modified in the Tester.runCrawler().
- **-printSpiderResult** - Output the result of the crawler to spider_result.txt. This may moment to produce the complete result.
- **-printAllURLdb** - Output all the data in the URL.db to AllURLdb.txt.
- **-printPageTitle** - Output all the data in the URL.db PageToTitle to PageTitle.txt.
- **-printURLPageID** - Output all the data in the URL.db PageID to URLPageID.txt.
- **-printPageMeta** - Output all the data in the URL.db PageMeta to PageMeta.txt.
- **-printParentToChilden** - Output all the data in the URL.db ParentToChildren to ParentToChildren.txt.
- **-printChildToParents** - Output all the data in the URL.db ChildToParents to ChildToParents.txt.
- **-printAllWordDB** - Output all the data in the WordDB.db to AllWordDB.txt.
- **-printWordID** - Output all the data in the WordDB.db WordID to WordID.txt.
- **-printInverted** - Output all the data in the WordDB.db Inverted to Inverted.txt.
- **-printTitleInverted** - Output all the data in the WordDB.db TitleInverted to TitleInverted.txt.
- **-printForward** - Output all the data in the WordDB.db Forward to Forward.txt.

Special Notice

Word Extraction

I have set

```
sb.setLinks(false)
```

in the word extraction part, which is different from the lab.

Since if it is set to true, it will create some keywords like *httpslibraryhkusteduhkaboutushoursservicepointshoursservic* and *httpslibraryhkusteduhkhelpforalumnialumni*, it doesn't seem to make sense.

Word Processing

Stop word removal and transformation into stems using the Porter's algorithm have implemented in this phase.

Crawler Strategy

I have implemented the BFS strategy in this phase. And the crawler will pick the next URL according the occurrence order in the webpage.

Page Last Modified Date and Page Size

Currently I am using the following method to get the last modified date of the page:

```
url.openConnection().getLastModified();
```

But it seems that the last modified date may missing. In this case, the last modified date will be set to

```
url.openConnection().getDate();
```

which is the date of accessing.
For the page size, I am using the following method to get the page size:

```
url.openConnection().getContentLength();
```

If the page size is missing, I will use the page size value method obtain in lab 2 instead.