**CSCI 2580 - Web Search Engines - Final Project**
**Spring 2016**
**Purnima Padmanabhan**
**Erdem Sahin**

The search engine web interface is accessible via the following link:
http://cims.nyu.edu/~es2697/csci2580_final.html

Our code is organized in the following directory structure:

| | |
|---|---|
| python: | The Shingles algorithm implementation |
| indexer: | The Indexer. Runs the Shingles algorithm and UnionFind and combines it with PageRank |
| retriever: | The Retriever. Runs queries against the index |
| evaluator: | The evaluation module (to calculate precision, recall, F-score). |
| util: | Common utilities used across multiple modules. |
| htmlparser: | Utility to parse the HTML DOM tree (from Programming Assignment 1). |
| www: | The HTML file and the CGI script used to display the search results |

The following folders contain various jars needed to run the code:

| | |
|---|---|
| cli: | The Apache commons command line options library[1] |
| JTidy: | The JTidy HTML parser library.[2] |
| jwi: | A library to access the WordNet dictionary.[3] |
| Lucene: | Apache Lucene library (for search engine indexing and querying).[4] |

The following folders contain various input files needed by our code:

| | |
|---|---|
| dict: | The WordNet English dictionary[5] |
| input-sample: | A subset of the Science Articles used for evaluation purposes.[6] |
| wiki-input: | The expanded Wikipedia Articles dataset used for indexing (as well as evaluation). |

The other directories are for outputs from various stages of the algorithm.

We tested our implementation on the “energon” lab machines (e.g. energon2). This environment gave us sufficient RAM and CPU resources to successfully run our code.

The code should be compiled via executing the build.sh

```
$ ./build.sh
```

[1] https://commons.apache.org/proper/commons-cli/
[2] http://jtidy.sourceforge.net/
[3] http://projects.csail.mit.edu/jwi/
[4] https://lucene.apache.org/core/
[5] http://wordnet.princeton.edu/wordnet/
[6] http://textfiles.com/science/

First run the evaluator to generate duplicate documents for the Wikipedia dataset and evaluate the results:

```
$ ./run-evaluator-html.sh
```

Then run the indexer to index the Wikipedia dataset + duplicates generated in the above step. This will run shingles, PageRank and UnionFind.[7]

```
$ ./run-indexer.sh
```

The index is already queried via the web interface. To query the index via the command line, please execute:

```
$ ./run-retriever.sh index "san juan"
```

`index` is the directory containing the index and “san juan” is the search query.

To run the evaluation on the text dataset, please execute:

```
$ ./run-evaluator-text.sh
```

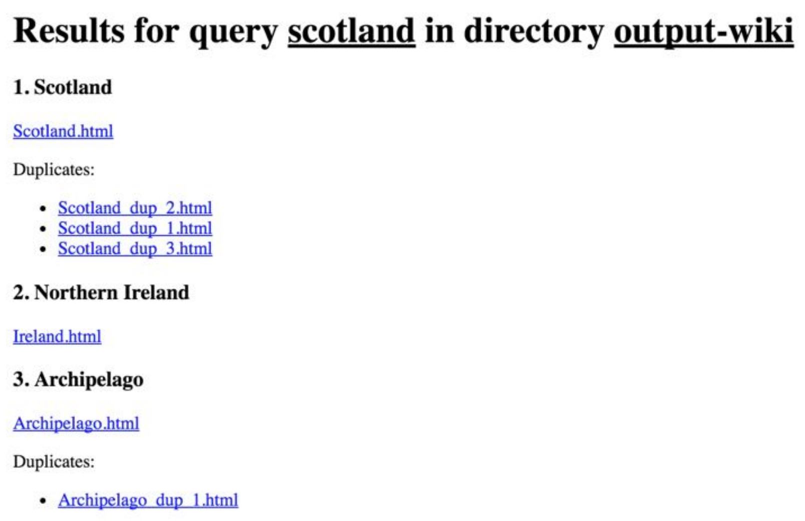
This is for evaluation purposes only. It takes ~15 minutes to run.

Finally, we have a script called “deploy-web.sh” that deploys the index, the required java .class files, and the HTML/cgi files to the public_html directory for serving and sets appropriate permissions. Please do not execute this script as it will only work for user “es2697”. It has already been executed.

[7] http://algs4.cs.princeton.edu/15uf/UF.java.html

**Sample queries to demonstrate near-duplicate detection and authoritative document selection:**

Query: “scotland”



**Results for query scotland in directory output-wiki**

**1. Scotland**

Scotland.html

Duplicates:

- Scotland_dup_2.html
- Scotland_dup_1.html
- Scotland_dup_3.html

**2. Northern Ireland**

Ireland.html

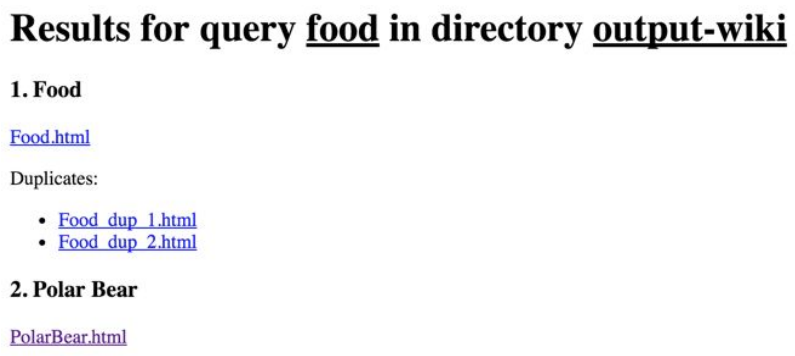
**3. Archipelago**

Archipelago.html

Duplicates:

- Archipelago_dup_1.html

Query: “food”



**Results for query food in directory output-wiki**

**1. Food**

Food.html

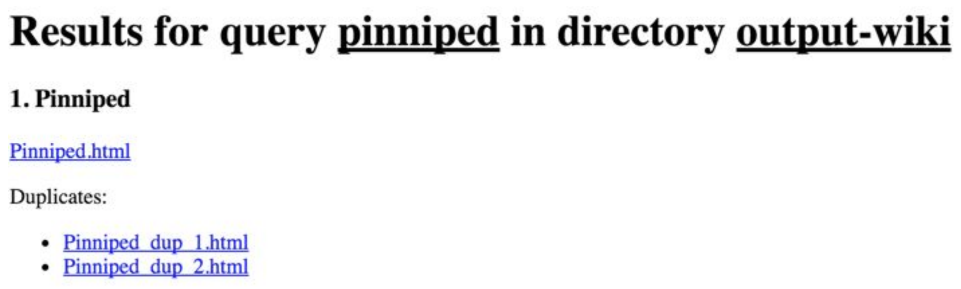
Duplicates:

- Food_dup_1.html
- Food_dup_2.html

**2. Polar Bear**

PolarBear.html

Query: “pinniped”



**Results for query pinniped in directory output-wiki**

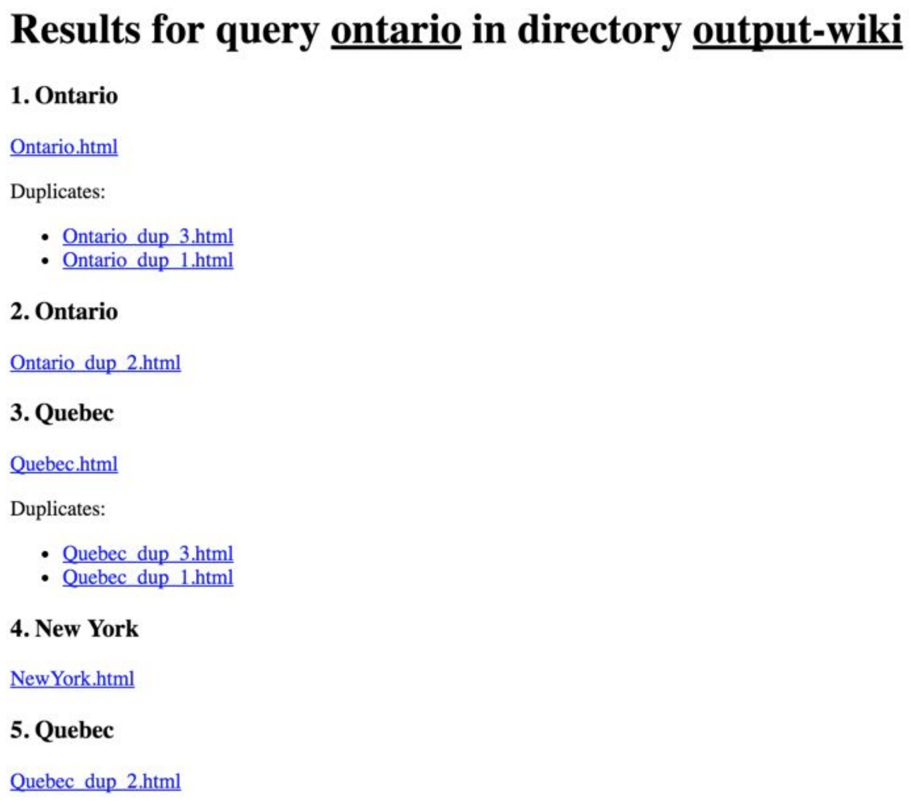
**1. Pinniped**

Pinniped.html

Duplicates:

- Pinniped_dup_1.html
- Pinniped_dup_2.html

Query: “ontario”


# Results for query ontario in directory output-wiki

**1. Ontario**

Ontario.html

Duplicates:

- Ontario_dup_3.html
- Ontario_dup_1.html

**2. Ontario**

Ontario_dup_2.html

**3. Quebec**

Quebec.html

Duplicates:

- Quebec_dup_3.html
- Quebec_dup_1.html

**4. New York**

NewYork.html

**5. Quebec**

Quebec_dup_2.html

In this case Ontario_dup_2.html is not clustered as a duplicate of Ontario.html, Ontario_dup_1.html and Ontario_dup_3.html, which are clustered together. But when we inspect the docs we observe that they are fairly different, which is a desired outcome. A similar situation is in play for Quebec_dup_2.html