Aliraza Punjani (amp2280) Kevin Li (kl2855)

PROJECT 2 GROUP 12

List of files:

1. README (this file)
2. qProber.py (python source file)
3. getWordsLynx.java (java file from course website to generate tokens)

Steps to run:

1. Compile getWordsLynx.java

   javac getWordsLynx.java

2. Execute qProber.py

   python qProber.py <BING_API_KEY> <t_es> <t_ec> <host>

   Eg: python qProber.py mz6wWvhFVxhgbqlz+aDPIa/V1uaygzWZreeE3L3+7CA 0.6 100 tomshardware.com

BING_SEARCH_KEY: mz6wWvhFVxhgbqlz+aDPIa/V1uaygzWZreeE3L3+7CA

BING_SEARCH_KEY (BACKUP): LJDM/w+jed2agcJ1wQCI0grtrAFcVHVz7IBkENCqyP8

P.S. files will be generated in the same directory

Implementation:

QUERY is Dictionary{Categories : their list of queries}

PARENT is Dictionary{Categories : their parent}

CHILD is Dictionary{Categories : their list of children}

URLs is Dictionary{Categories : their set of URLs}

CS is Dictionary{Categories : their content summaries}

Part 1

classify(C, D, tc, ts, r)

Recursive routine implements the algorithm from the QProber paper

getSpec(C, D)

Recursive routine calculates the specificity of a Category C for a Database D using the formula:

$$\text{ESpecificity(D, C)} = \frac{\text{ESpecificity(D,Parent(C))} \cdot \text{ECoverage(D,C)}}{\sum_{Cj \text{ is the child of } Parent(C)} (\text{ECoverage(D,Cj)})}$$

getCoverage(C, D)

Calculates the coverage of a Category C for a Database D

getMatches(query, site)

Fetches the number of matches for a query on a database website

Checks whether the match results for a query on a website have already been cached

If so, just returns the cached results

If not, executes the Bing API call to issue the query on the website, caches and returns the number of matches

Part 2

getUrls(category, site)

Generates the URL set for the document sample associated with a category node **C** and a database **D** (**sample-C-D**)

summary(site)

For every category, augment to its URL set the URL sets of its children categories (effectively doing a set Union, so no duplicate URLs)

Generate document **sample-C-D** for every category node **C** and the database **D**

Finally, parses **sample-C-D** to get the token list and generate "**topic content summary**" for each sample

We're not including multiple-word information in the content summaries

Calling chain