1. **Project Finisher**: YaCai UNI: yc2788 ; Keqiu Hu UNI: kh2567
2. **Files included**:

source:test.entry.java

test.getWordLynx.java

test.outPutxml.java

test.xmlReader.java

test.intString.java

test.readClassIf.java

bin: test.entry.class

test.getWordLynx.class

test.outPutxml.class

test.xmlReader.class

test.intString.class

test.readClassIf.class

lib: commons-codec-1.7.jar

Makefile, Root, Computers, Sports, Health

1. **How to run**:

Input command below in your terminal:

make run ARG1=”accountkey” ARG2= tesARG3=tec ARG4=target

Note:

Account key：the accountkey for Bing API，which is showed in f）

tesin ARG2 is the Specificity(0-1) threshold

tecin ARG3 is the coverage threshold

target in ARG4 is the URL of the database to be classified, omitting ‘http’

1. **Internal Design of our project**:
2. Receive as input a target, which is simply a url, and two values, one is between 0 and 1, meaning the specificity threshold for the target to be classified. The other is coverage, meaning the coverage threshold for the target to be classified.
3. Begin from the root to level 1 and level 2

We read the file ‘root’ to get each query and connect bing, then we output the result to xml and parse the xml to get match numbers of each query and save it to Tree Map. Then we use these numbers to computer the coverage and specificity of computers, health and sports.

level 0 root

level 1 computers health sports

level 2 hardware programming Fitness Diseases Basketball Soccer

Output all the nodes until the specificity and coverage reach the threshold.

During the process, when we parse the xml produced by each query, we also save the urls it produced to TreeMap which using the categories as key and urls as values. In order to combine the searching results from children node and eliminate duplicate element, TreeSet is used to both eliminate duplicates and sort the words in order ( The characteristics of TreeSet).

After obtaining the URLs for all categories, we use the URL TreeSet to get content summay, for each node we retrieve in the first part, we extract the text of a page from the urls of each query using the java code supplied by teacher. The getWordsLynx function returns a set and we later cast that into TreeSet since it is a reference to a TreeSet when initialized. To calculate the document frequency of each terms, we traverse each distinguished items in the allTerms TreeSet (which store all different terms) and check if the Words List corresponding to the each URL has the item. If true, the document frequency is incremented by one.

Then the document frequency of each item is calculated and we can output the content summay to a text file, containing all the words in the sample -in dictionary order- together with their respective document frequencies.

1. **Classification Algorithm**

**Classify**(**Category** C, **Database** D, Tec, Tes, Specificity(D,C))

Result = 0

**if** C is a leaf node

**then retur**n {C}

Probe database D with the query probes for the subcategories of C

Calculate Category from the number of matches for the probes

**for each** subcategory Ci of C

Calculate Specificity(D, Ci) using Coverage and Specificity(D, C)

**if** Specificity(D, Ci)) Ci)

then Result = Result

**if** Result ==

**then return** {C} // D was not “pushed” down

**else return** Result

**f)** AccountKey : rOmf0BVgre5o9F/6xOU7wYIsPkTwRSPqVMv83o3Atgo=

Reference:

PANAGIOTIS G. IPEIROTIS, and LUIS GRAVANO. Classiﬁcation-Aware Hidden-Web Text