CAB431 Tutorial (Week 3): Pre-Processing: Parsing, Tokenizing and Stopping words removal

****

TASK 1: Parsing - read files from RCV1v2, find the documentID and record it to a collection of BowDocument Objects.

- The documentID is simply assigned by the 'itemid' in <newsitem>
- In this step, the created BowDocument can be initialed with found documentID and an empty Map (e.g., dictionary, or HashMap) of key-value pair of (*String* term: *int* frequency).
- Build up a collection of BowDocument for given dataset, this collection should be a map structure with documentID as key and BowDocument object as value.
- Create a method (or function) to print out all documentIDs by iterating above collection and calling BowDocument's method getDocId().

TASK 2: Tokenizing – update Task 1 program to fill term:freq map for every document.

- You only need to tokenize the '<text>…</text>' part of document, exclude all tags, and discard punctuations, numbers.
- Use addTerm() of BowDocument to add new term to term map or increase term frequency when the term occur again.
- Create a method displayDocInfo(int aDocId) to display term list with a given docuemntID, by searching collection of BowDocument in Task 1, and calling getTermFreqMap() of found document. The output should be like:
    - Doc *docId* has *termCount* different terms:

      Term1, 3

      Term2, 1

      Term3, 4

      ….

- *Please think about the terms with high frequency, which may be some useful words for describing the document in the future.*

TASK 3: Stopping words – use given stopping words list to ignore/remove all stopping words from the term list of documents.

- Download the stopping words list from QUT Blackboard, read through first, compare with your notes of high frequency terms.
- Update your program to read in given stopping words list and store to a list stopWordsList.
- Update your program, when adding term to term map, check the term if or not exist in stopping words list, ignore such term if it is in.
- Call the method displayDocInfo() again and compare the output with Task 2.

TASK 4: Sort and display document term:freq list *by term* or/and *by frequency*.

You may do this after you have finished and passed all the above 3 tasks in week 3 tutorial.

Examples of output

Document 741200 contains 50 terms and have total 100 words.
quot:4
lead:3
car:3
german:2
over:2
soper:2
victori:2
dalma:1
merced:1
steve:1
austrian:1
fifth:1
han:1
downpour:1
handl:1
…

Document 741000 contains 42 terms and have total 110 words.
under:7
over:4
tee:2
peter:2
trinidad:1
par:1
lehman:1
scotland:1
sunday:1
darren:1
mark:1
…

APPENDIX

Create a "Wrapper Class" of Bag-of-Words representation of a document.

The BowDocument class should have properties of documentID and a HashMap in which terms are keys and their frequencies are values. The BowDocument class should have the following methods (functions) besides a constructor of BowDocument class:

```
    private String docId;
    private HashMap<String, Integer> termFreqMap;
    /**
     * Constructor
     *  Set the ID of the document, and initiate an empty term:frequency map.
     * call addTerm to add terms to map
     * @param docId
     */
  public BowDocument(String docId){
     //your code here
  }
    /**
     * Add a term occurrence to the BOW representation
     * @param term
     */
  public void addTerm(String term){
     //your code here
  }
    /**
     *
     * @param term
```

```
    * @return the term occurrence count for the given term
    * return 0 if the term does not appear in the document
    */
public int getTermCount(String term){
    //your code here
  }
  /**
    *
    * @return sorted list of all terms occurring in the document
    */
public ArrayList<String> getSortedTermList(){
    //your code here
  }
  /**
    * @return map of term:freq pairs.
    */
public HashMap<String, Integer> getTermFreqMap(){
    //your code here
  }
  /**
    *
    * @return the ID of this document
    */
  public String getDocId(){
        //your code here
  }
```