CAB431 Week 3 Workshop

Pre-Processing: Parsing, Tokenizing and Stopping words removal

TASK 1: Design a parsing function, parse_doc(input, stops), to read an xml file and represent the file as the following data structure:

```
(word_count, {docid:curr_doc})
```

where

- *word_count* is the number of words in <text> … <\text>
- *docid* is simply assigned by the 'itemid' in <newsitem>
- *curr_doc* is a dictionary of term_frequency pairs.

You only need to tokenize the '<text>…</text>' part of the document into words, and exclude all tags, and discard punctuations and numbers. Then please remove stopping words and at last get all terms used in the '<text>…</text>' part.

- o You can download the stopping words list from the Canvas.

You can initialize dictionary *curr_doc* ={}, *then* add terms into *curr_doc* ={} by go through terms in lines of the file. You may need to check if the new term (key) exists in *curr_doc* and then update its value (the frequency).

The following is an example of the return value of **parse_doc()** for file "6146.xml" (see the Canvas):

```
(133, {'6146': {'argentine': 1, 'bonds': 1, 'slightly': 1,
'higher': 1, 'small': 1, 'technical': 2, 'bounce': 2, … ,
'newsroom': 1}})
```

TASK 2: Design the main function to read an xml file and common-english-words.txt (the list of stopping words), call function parse_doc(), and print the itemid (docid), the number of words (word_count) and the number of terms (len(curr_doc)).

The following is an example of the outputs:

Document itemid: 6146 contains: 133 words and 75 terms

TASK 3: Display the document's terms and their frequencies, and sort alphabetically by terms in ascending order.

Please note it is impossible to sort a dictionary, only to get a representation of a dictionary that is sorted. **Try the following commands:**

```
>>> x = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0}
>>> {k: v for k, v in sorted(x.items(), key=lambda item: item[1])}
>>> {k: v for k, v in sorted(x.items(), reverse=False)}
>>> {k: v for k, v in sorted(x.items(), reverse=True)}
```

<u>Outputs Example of Task 3:</u>

```
addition : 1
aires : 1
amid : 1
argentina : 2
argentine : 1
awaiting : 1
axel : 1
bank : 1
between : 1
bocon : 1
```

```
bonds : 1
bounce : 2
buenos : 1
bugge : 1
change : 1
congress : 1
deficit : 1
delegation : 1
denominated : 1
dollar : 1
due : 2
during : 1
early : 1
economic : 1
economy : 1
events : 1
expect : 1
expected : 2
fernandez : 1
fiscal : 1
foreign : 1
frb : 1
friday : 1
fund : 1
general : 1
government : 1
higher : 1
including : 1
international : 1
large : 1
low : 1
```