Word Density Analysis

How to Execute :

```
python main.py [url]
python main.py http://www.cnn.com/2013/06/10/politics/edward-snowden-profile/
```

or run this shell script, please see top 3 lines.

```
sh test.py
```

Example :

1. http://www.amazon.com/Cuisinart-CPT-122-Compact-2-Slice-Toaster/dp/B009GQ034C/ref=sr_1_1?s=kitchen&ie=UTF8&qid=1431620315&sr=1-1&keywords=toaster
   keywords : toaster, cuisinart, bread, slice, toast,

2. http://blog.rei.com/camp/how-to-introduce-your-indoorsy-friend-to-the-outdoors/
   keywords : friend, outdoors, introduce, indoorsy, keep,

3. http://www.cnn.com/2013/06/10/politics/edward-snowden-profile/
   keywords : nsa, man, safeguard, leaks, liberty,

Descriptions :

This program can extract keywords from any website. It takes url as an input, and produce 5 or less keyword as the output. The approach that I used is as follow:

1. Download the given webpage
2. Extract each part of the webpage using beautifulsoup library
   a. Title
   b. Meta keywords
   c. Header
   d. Content, for content we recursively search deep down in HTML tree and check the words vs tag density. i.e. <u>w</u> will have less word density than words (1/32 : 5/12)
3. Normalize the word
   a. make every words lowercase.
   b. remove non-alphanumeric character, except space bar. e.g. !,@,#,$,\r,\t,\n
4. Translate each of 4 part above to Bag of Word representation.
5. Remove stopwords
6. Merge all the bag, gives 3 times more weight to words from title,header, meta keywords
7. Report the maximum word counting.

Architecture

There are 3 major parts :

1. Html Module : Responsible for web crawling.
2. NLP Module : Responsible for word countings, and analysis.
3. Main Program : Responsible for taking input and perform program execution.

1. **Html Class:**

    a. **Constructor Html(url)**

    Return an instance of Html Class, the constructor crawl the given url webpage. In this part, we do some preprocessing i.e. remove <script>, <link>, <style> tag, also <b>, <a>, , <u>, , but for these tags we keep the content inside them.

    Output: Instance of Html class.

    b. **getTitle():**

    Produce a string of page title, space-bar separated between word, alpha-numeric, lowercase.

    Output: String of title.

    c. **getKeyword():**

    Produce a string of keywords, space-bar separated between word, alpha-numeric, lowercase.

    Output: String of keywords.

    d. **getHx():**

    Produce an array of string of header (h1-h4), space-bar separated between word, alpha-numeric, lowercase.

    Output: Array of String of header.

    e. **getContent():**

    Produce a string of content, space-bar separated between word, alpha-numeric, lowercase.

    Output: String of content.

    f. **getContentRes(node,ans):**

    This method recursively go through given html subtree. If the density of word is higher than tag, then we stop and save the text of the tree to the ans.

    Input : bs4 html subtree, answer array.

    Output : None

    g. **parseNode(node):**

    Input : bs4 Html subtree. e.g. " ab<a>cd "

    Output : text of a given html. e.g. "abcd"

2. **NLP Module**

    In this module, we have our own bag of word representation (array of tuples) i.e. [('word1':30), ('word2':35), ('word3':40),]

    a. **def parseText(txt):**

    This method will be called by parseNode and others functions. It makes the string to lowercase, only alphanumeric characters, ensure 1 space between words.

    Input : String -- e.g. "a! b@ C# D$"

Output : String -- e.g. "a b c d"

b. def oneGram(txt):

Create a bag of word representation of the given text txt.
Input: String txt.
Output: Bag of word.

c. def getStopword():

Output: List of stopwords.

d. def removeStopword(bag):

Given a bag of word, remove stop word.
Input: Bag of word.
Output: Bag of word.

e. def sortBag(bag):

Sort a given bag of word increasingly.
Input: Bag of word.
Output: Bag of word.

f. def mergeBag(bag1,bag2,weight=1):

Merge 2 bag of words, giving multiples times of weight to the first bag1.
Input: Bag2 of word, bag1, bag2.
Output: Merged Bag of word.

3. Main Program

In this part we do 7 step above in the Description part to produce the result.

Future Improvement

1. Stemming and lemmatization, to make the words more matchable, ie. car and cars, manage and managing.

2. From beautifulsoup library, When it extract the word from html subtree. It does not care about the meaning and spacebar. For example, cat dog , it translates to catdog, which we know that it is in the different paragraph, so the text should be cat dog. In this case, there can effect the counting of the word that is near the beginning and the end of each tag.

3. For correctness, we might translate abbreviations to it full form. i.e. You'll = you will. This is just for correctness of algorithm but have not much impact since, from my observations, most of this cases happen with You'll I'll he's we're case, most of them are in stopwords list and will be removed eventually.

4. We might try to use other topic model to capture the keyword from a text files, however we might need more dataset. For examples, LDA algorithm, it can automatically assign category for each webpage.