Webpage Topic Analyzer

Author: Lingqiong Tan
Date: Aug 27, 2017

Introduction

This tool is a Webpage Topic Analyzer which takes an url as input and generates a list of possible topic with one word or two word.

User Manual

1. How to execute

- Install all package needed in all python files.
- Get to the directory stores files in command line.
- Run: python main.py "http://www.cnn.com/2013/06/10/politics/edward-snowden-profile/" (or any other url you want to analyze)
- You will get possible topics for this url like this: *[(u'NSA', u'controversy'), (u'risks', u'NSA'), (u'Whistleblower', u'NSA'), u'Snowden', (u'Patriot', u'Act')]*

2. Customize parameters

This tool allows user to customize delimiter, lemmatization, stemming, lowercase and stop words, gram weight by simple customize the corresponding parameters when calling functions.

Structure

1. Main program: Main program takes url from standard input and does parsing separately for all texts in html, text in title and text in headings like h1, h2 and so on. Then for each part of texts, do unigram and bigram analysis. After getting the results for all unigrams and bigrams, do uni-bi-gram analysis which will rank the top topics candidates among all of them.
2. PageParser: PageParser parses the HTML page by BeautifulSoup and allows users to generate visible text of whole page, title and headings.
3. PageTopicAnalyzer: PageTopicAnalyzer is a powerful word-count-like tool which allows user to customize their needs on delimiters, lemmatization, stemming, lowercase and stop words. It provides functions for unigram, bigram, uni-bi-gram and rank top grams.

Functions

PageParser.py

- ***getPage(url)***: Use urllib2 to open url, if fails, give an exception. And then use BeautifulSoup to parse html page.

- ***isVisible(text)***: Since there are lots noise in html page, this allows user to get cleaner visible text. This function is refer to https://stackoverflow.com/questions/1936466/beautifulsoup-grab-visible-webpage-text.
- ***getAllText()***: This function gets visible text for the whole html page, regardless of different tags.
- ***getTitle()***: This function get visible text for only <title> … </title>. The design idea is considering the importance of title in revealing page topic.
- ***getHeadings(number=4)***: Similar to getTitle(). The default headings taken considered are h1, h2, h3, h4.

PageTopicAnalyzer.py

- ***__init__(text="", deli=None, lem=False, stem=False, low=True, stopWord=None)*:** Initialization for delimiter, lemmatization, stemming, lowercase and stop words before doing unigram and bigram. It allows user to add/remove stop words from default delimiter lists and stop word lists.
- ***seoSW(path)***: This take a file path as input and add the delimiters specified in that file to nltk English stop words list. Here an online SEO stop word list is used.
- ***__add__(other)***: This is a implementation of add function for this object. It return a new PageTopicAnalyzer.
- ***rank(top)***: This return top topics list according to the given number.
- ***unigram()***: Unigram function assumes each topic appears with one word and does word count for all meaningful single words.
- ***bigram()***: Bigram function assumes each topic appears with two consecutive meaningful words.
- ***uni_bi_gram(unigrams, bigrams)***: This function generate top grams based on unigrams and bigrams. It first assumes that if an unigram appears in a bigram, the bigram is a confident topic. It then consider the remaining unigrams and bigrams.
- ***weighted(weight)***: This allows user to custom a weight after doing unigram and bigram. Since texts in title or headings should reveal page topic better, user may want to give them a higher weight.