1 Workflow

1. The first day has been almost spent to find and study every information about the best libraries for HTMLParsing, Web Scraping, Web Crawling and how to classify a text, since my poor background about the topic.

2. The first step was extracting the text from the web page using an HTMLParser, BeautifulSoup has been chosen for the good documentation both from the official website and from the community.

3. The text is then analyzed to find tags that could be useful for the correct classification of the website in the most general manner. In this step also the Firebug Firefox extension has been used.

4. After the study of different pages it was clear that there were no common tags useful to classify the web page (for example sometimes in the meta-data the keywords are present, but it does not happen every time).

5. Because of that it has been decided to analyze all the text present and find the recurring words in order to obtain with a good probability the keywords useful to classify the web site. To do that the NLTK library has been used.

6. Anyway if the tag *keywords* is found it will be printed together with the words found in the text with the highest occurrence frequency

7. Moreover both relative and absolute links present in the web page have been extracted because they could be used in the future to obtain more information about the content of the web site.

8. The recurring words are found creating a dictionary with the words and the relative occurrences, the keywords are then the first ten (or more depending by the user) are printed.

2 The code

init Given the url of a web page the domain is extracted and the BeatifulSoup Objects is instantiated

getTitle The title has been extracted because it has been considered the first step to classify the content of a web page.

getText Useless sections of the web page as script and style are removed, then the text is extracted

getWords To manage effectively the text in Python it should be cleaned and pre-processed:

- the punctuation has been removed,
- the words are extracted,
- the words are transformed from unicode to string,
- the words transformed in lower case,
- the words are separated with the *TreebankWordTokenizer().tokenize()* method because it preserves the contractions.

findKeywords Once the words are extracted and saved a dictionary is created with the words and the relative occurrences. To avoid duplicates, synonymous and plurals the method *WordNetLemmatizer().lemmatize* has been used. Moreover taking again advantage of NLTK library only the nouns are considered as keywords.

Results Example of code output:

```
Input url : http://www.cnn.com/2013/06/10/politics/edward-snowden-profile/
Title of the webpage: Man behind NSA leaks says he did it to safeguard privacy, liberty - CNNPolitics.com
The keywords are  : politics, Man behind NSA leaks says he did it to safeguard privacy, liberty - CNNPoliti
cs.com
The following keywords are found in the text and are ordered by relevance:
        1 : snowden
        2 : nsa
        3 : u s
        4 : government
        5 : video
        6 : surveillance
        7 : intelligence
        8 : program
        9 : contractor
        10 : world
```

Figure 1: Example of code output using the CNN web page

3 Assumption

- The NLTK packages should be downloaded if are not present yet
- Only the English web pages are considered
- The method used is not the best but it is fast enough and could be applied to a lot of web pages. A better approach could be using a Naive Bayesian Classifier, but it requires a training set.

4 Library versions

Python 2.7

NLTK 3.04

urllib2 2.7

pandas 0.16.2

bs4 4.3.2