Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Dr. J. Kunkel, H. Hatzel

Exercise Sheet 2
Big Data Analytics
WiSe 2016/2017
Due Date: 2016-11-04, 430 P

Please document your code, without sufficent documentation you won't recive any points.

1 Exploration of the IRIS Data Set (R) (100 P)

The Iris data set is a dataset describing key characteristics of Iris flowers. **In the following exercise sheets for exploration of data sets, we expect that you will try various approaches to visualize and analyze the data. Will will only provide a few guiding questions.**
Use the methods of descriptive statistics as described in the lecture to attain useful or interesting information on the data set. For example:

- Summary
- Histograms
- Density plots
- Time series to visualize time dependent data

Visualize the results in suitable diagrams, document your process in a PDF.

As this is the first task of this nature, you may search for analysis of this data set available on the Internet (search for two different analysis). Apply the same analysis yourself and learn the process along the way.

1.1 Guiding questions

- Is there any correlation between sepal and petal measurements?
- Do width and length of sepal and petal correlate?
- How are the sepal lengths distributed?

1.2 Notes

There is no need to apply any machine learning or related techniques that were not introduced in the lecture yet.

Submission:

`1-exploration.pdf`	Results of your exploration with visualizations. List several sources for the analysis of this data set.
`1-exploration.R`	Your R code snippets used for the exploration

2 Statistical Analysis of Word Counts (R) (150 P)

The following task is based on the analysis in *Excercise 1, Task 4 („Word Frequencies in Moby-Dick")*. If your solution is not suitable for further analysis, use the file `/home/bigdata/2/moby-dick.csv` on the cluster.

2.1 Descriptive Statistics

Proceed as follows:

- Use your results from *Excercise 1, Task 4 („Word Frequencies in Moby-Dick“)*
- Import the data in R
- Perform statistical analysis as described below
- Document your results

Include at least the following statistical observations:

- Average occurrences of any word
- Create a histogram showing frequency distributions of any give word across all chapters
- Create a histogram showing frequency distributions of all words across chapters
- Plot the density of the number of times a word occurs

Try to find a strategy to filter out words that are of little indication to a chapter's subject matter. That means that they do not carry relevant information. Use your strategy to create a list of words that should be excluded from data analysis.

2.2 Inductive Statistics

We will now conduct a short analysis using inductive statistics:

- Calculate the correlation of word occurrences with the chapter length.
  Which words show a strong correlation, for what reason does this work? Save a list of the most strongly correlated words as `strongly-correlated.csv`.
- Analyze correlation of word frequency and word length.
- Create a simple linear model of words which occurrence correlates strongly with the chapter length. Identify which chapters deviate most from the model, thereby, finding outliers. Analyze these outliers using descriptive statistics to gain an understanding of why they might deviate.

Submission:

`2-statistics.R`	Your R script used for the analysis.
`2-statistics.pdf`	Your analysis with the required plats and a short analysis.
`2-exclude-words.csv`	List of words that carry no information and should be excluded.

3 Cleaning and Extraction of Wikipedia Text (Python) (180 P)

The Wikipedia dataset contains a lot of interesting machine readable data. Unfortunately, most data is not trivially processed as there are often multiple ways the same data might be recorded. This task aims to remove Wikimarkup syntax from Wikipedia articles to allow better analysis of the data and extract interesting data to enable further analysis.
Since there are a lot of syntax elements to remove/analyze, we split them among the working groups. Combining all groups' code should result in improved data quality useful for further analysis.
Clean the syntax elements you were assigned. Take care not to remove syntax elements that are in a context where the syntax is supposed to be part of the article. Specifically this means that you need to detect when syntax elements are inside a `pre`- or `nowiki`-block. For further details see: `https://en.wikipedia.org/wiki/Help:Wiki_markup#Limiting_formatting_.2F_escaping_wiki_markup`

To make a combination of all solutions possible please conform to the following structure:

```
def clean_extract_<syntaxelement>(text):
    """
    This function removes one specific syntax element, name it accordingly.
    """
    # Remove the syntax element here (unless in pre- or nowiki-contexts as discussed above)
    return (cleaned_text, list of removed content)
```

If useful, extract the content of the semi-structured tags and return it, i.e., the cleaning function for links returns a list of links.
Each group is assigned two of the following tasks (the group ID is provided) and the additionally returned data:

G1 Text formatting (italic, bold, ...); return a list for one type of text formatting

G1 Links; return the list of links

G2 Section formatting (Headings, lists, indents, mixture, pre-formatting); return the section headers

G2 HTML-Tags; returns the list of colored text elements

G3 Images; return links to images

G3 Tables (remove all tables); returns list of columns in the table header

G4 Lists; returns nothing

G4 Redirects; returns the list of redirects

G5 Symbols (replace all Mediawiki specific symbols with their Unicode counterpart); returns nothing

G5 Categories; returns the list of categories

G6 Coordinates; returns the list of coordinates referenced in the document

Test your implementation and make use of it in a program that, given the Wikipedia CSV file, creates a CSV file containing lines of article name, the cleaned data, and an array of extracted data.
Please Note: This exercise's goal is not to perfectly handle all syntax elements, but instead improve text quality for subsequent analysis.

3.1 Hints

Details for the extraction of categories and coordinates are provided in the following, please take them into consideration when processing your syntax elements.

3.1.1 Coordinates

Many articles on specific countries or places have an infobox that provides semi-structured information. The relevant information for this task is structured as follows:

```
{{Infobox
... # Some more information that is irrelevant for this task
 | iso region          = DE-HH
 | PLZ                 = 20001-21149, 22001-22769
 | coordinates_display = title
 | latd                = 53
 | latm                = 33
 | lats                = 55
 | latNS               = N
 | longd               = 10
```

```
 | longm               = 00
 | longs               = 05
 | longEW              = E
 | date                = August 2010
}}
```

Some articles don't use the infobox but instead use the `coord template`. To ensure we only pick up coordinates actually describing the whole article, we will only parse coord templates which come with the `display=title` tag. A detailed explanation of coord templates can be found at: `https://en.wikipedia.org/wiki/Template:Coord`.
When no infobox is available, look for a coord template instead.

Hint: Not all articles follow the same conventions for whitespace and newlines!
Many of the coordinates specified using minutes and second, please convert them to a consistent decimal form. Encode the cardinal direction as the sign of your coordinates. Points south of the equator or west of the prime meridian take negative values. Return the decimal values for longitude and latitude.
Your program should be able to save its output to a CSV file.

3.1.2 Categories

An article can be a part of many categories. For our purposes, we would like to know which category is the most relevant to the article. A metric that works fairly well for this is to just order the importance of categories by the order the appear in the articles markup. The category "Hamburg" for instance appears in the following articles: `["Hamburg", "City-States", ..., "States of the Holy Roman Empire", ...]`,
Membership in categories is denoted as follows in an article:

```
[[Category:City-states]]
```

In each article, categories are listed in the section *external links*. Article authors have to option to specify a number for explicit sorting of the categories.

```
[[Category:Hamburg| ]]

[[Category:Germany| 01]]
```

Your program should parse the external links section, then output the article name together with a list of categories into a CSV file. If the authors provide an explicit sorting order, use it to reorder the articles.

Submission:

`3-wiki-markup-cleaner.py` Your source code for removing your assigned syntax elements.