Data Mining

Homework 1

Due: 20/3/2013, 15:30.

You must hand in the homeworks electronically and before the due date and time. Check the web page for instructions about collaboration, about being late, and about handing in the homework.

Problem 1. The following "paradox" is an example of how some concepts can be counter intuitive when dealing with probabilities, even in our everyday life.

Data-miningitis is a very rare disease that each person has probability 1 in a 100,000 to have it. Luckily, there exists a very good test for the disease, which has accuracy 99.9%.

Your instructor, who is afraid of having it, went to take the test and he was found positive. Given the result of the test, what is the probability that he indeed suffers from data-miningitis? First define an appropriate sample space, then the events required so that you can compute the probability.

Problem 2. Assume that a monkey sits in front of a keyboard and hits randomly the 26 letters, each with the same probability. Assume that it types 100,000,000,000 letters; what is the expected number of times that the word "mining" appears?

Problem 3. Quite often you can analyze your data just by using simple unix tools. Some usefull commands are the `grep`, `sort`, `uniq`, `cut`, `sed`, `awk`, `join`, `head`, `tail`, `wget`, `curl`. You can find more information using the `man` command or by checking the web. Shell scripting can help you even more.

As a simple exercise do a simple analysis of the reviews in
`http://snap.stanford.edu/data/web-RateBeer.html`.
After you download and unzip the file, use some of the commands above to find the 10 beers with the highest number of reviews. (**Hint:** You can do it with a single command line, by chaining commands through pipes!)

Problem 4. We will now go one step further and start practicing with Python. Write a Python program to find the top-10 beers with the highest average overall score among the beers that have had at least 100 reviews.

Problem 5. We continue getting familiar with Python by downloading and parsing some web pages. For downloading the web pages you may use the package `Requests` or the package `urllib2`. To parse the page you can either use regular expressions through the package `re` (you should anyway become familiar with regular expressions), or, probably better, use an HTML/XML parser. The `Beautiful Soup` package is a good one but it loads the whole file in memory. This is fine for this problem, since the pages to parse are small, but be careful if you want to use it on large XML files; for those ones check the `lxml` library and the tutorial at

`http://www.ibm.com/developerworks/xml/library/x-hiperfparse/`

Write a program that will download from `http://www.kijiji.it` and parse all the job positions in Rome about *Informatica/Grafica/Web* with a *Contratto*. Save in a tab-separated value (TSV) file, for every job (one line per job), the *title*, *short description* (from the job summary page), *the location*, the *publication date* of the job announcement, and the *URL link* to its web page. Make sure that you have a delay (use: `sys.sleep()`) between different downloads of kijiji pages.