CIS 678 Machine Learning
Programming Project #1

Exploratory Data Analysis

Overview

This is intended to be a "warm-up" exercise. The main purpose of the assignment is to begin thinking about and analyzing data and manipulating it via program code.

Background

Authors and editors are often concerned with the "readability" of their articles and manuscripts: how difficult is a sentence to understand? is this document written at a level appropriate for its target audience? As a student, you might wish to determine the cognitive complexity of your term paper or thesis.

One way to approach this problem is by computing the Flesch Index (FI), a numerical measure of the readability of English text. Originally invented for evaluating the difficulty of reading U.S. Army training manuals, it has since become ubiquitous.

Specifications

Download sample writings in the form of documents, analyze the documents using the Flesch Index, visualize the results, and then compare/discuss the results. That's basically the assignment.

1. Pre-processing: read in and clean the data

The datafiles (see course info page) consist of text documents downloaded from Project Gutenberg, a free repository of digital documents. Since the Flesch Index uses syllables, words, and sentences, you will need to handle punctuation, non-alphabetic characters, and whitespace as you parse the document.

2. Analysis: compute the Flesch Index (FI) of the data

The first step in the computation is to break the document down into syllables, words, and sentences. The basic idea is that polysyllabic words are more complex than simple words, and sentences with a large number of words (requiring a higher cognitive load to keep everything in memory) are more complex than short, simple sentences.

These two intuitions are captured in the succinct equation below:

$$\text{Flesch\_Index} = \; 206.835 - 84.6\left(\frac{numSyllables}{numWords}\right) - 1.015\left(\frac{numWords}{numSentences}\right)$$

where *numSyllables* is the total number of syllables in the document, *numWords* is the total number of words, and *numSentences* is the total number of sentences.

There are some simple rules to help in counting these values; you can experiment with variations on them; all that's important is that you apply your rules consistently.

Sentence:
Consider a sentence to have been encountered whenever you find a word that ends in a specific punctuation symbol: a period, question mark, or exclamation point.

Word:
A word is a contiguous sequence of alphabetic characters. Whitespace defines word boundaries.

Syllable:
A syllable is considered to have been encountered whenever you detect:

- Rule 1: a vowel at the start of a word *or*
- Rule 2: a vowel following a consonant in a word

One exception to Rule 2: a lone 'e' at the end of a word does not count as a syllable.

Note that we use the usual definitions of vowels and consonants.

3. Visualization: display the data

Put some thought into visually understanding your data. For example, create a histogram showing the distribution of polysyllabic words in a document (i.e. what percentage of words contain *k* syllables?). Be creative as you explore the data.

It would be nice if you could do this within your program, using a graphics library (e.g. matplotlib, R, gnuplot). But you can default to Excel if you do not already know a graphing API. The idea is to improve your skills at visually exploring/presenting results.

4. Discussion: what does the data tell you

Find your own data; explore and analyze. Provide a description of your experiments, an explanation of your results, anomalies detected, and any conclusions you were able to reach.

Notes:

- You must use Python for this project. All computations should be performed by your program, *not* by a built-in library routine (but you can use library routines for reading in files and creating visualizations).
- I recommend using jupyter notebooks (see "Deliverables" on next page).
- Be sure to demonstrate good programming style and practices.
- You may work together on this assignment.

Deliverables

- Submit a hard-copy of your source-code, sample output, and a full report.
- Your report should be more than just the visualizations your code produces; it should include the visualizations along with the discussion of your results. If you are using straight jupyter notebooks, I am okay with your report being intertwined with your code and results with appropriately formatted Markdown cells (including things like headings, explanations of what sections of code do, etc.). In this case, you can simply download your notebook as a PDF and print out that PDF.
- Be prepared to present and discuss your solution in class. E.g. what data structures did you employ? What graphing package/API did you use? What interesting problems (and solutions) did you encounter?