Projekt : Textstatistik

In diesem Projekt sollen Funktionen geschrieben werden, welche Woerter in einem Text identifizieren und zaehlen. Diese Funktionen werden benutzt, um zu zeigen, dass die Haeufigkeiten der Woerter in einem Text sehr heterogen sind und ungefaehr einer Zipf-Verteilung folgen.

Als Ergebnis der Projektarbeit sollen sowohl die Python Programme und die Ergebnisse der Datenanalysen abgegeben werden, als auch ein begleitender Text, welcher Ihre Beschaeftigung mit diesem Projekt dokumentiert, also Vorueberlegungen, Erlaeuterungen und Erkenntnisse Ihrer Arbeit zusammenfasst. Achten Sie auch auf eine gute Dokumentation Ihrer Python Programme mit Hilfe von Kommentaren im Programmcode.

- Fuer die statistische Analyse der Woerter in einem Text wird eine Funktion `L=splitstring(s)` benoetigt, die als Argument einen String `s` hat, und eine Liste `L` der Woerter zurueckliefert, die in diesem String vorkommen. Schreiben Sie eine solche Funktion.

Hinweis: Sie koennen hierfuer regulaere Ausdruecke verwenden.

- Es soll nun eine Python Funktion `n,W = TextStats(file,ReturnDict=False)` erstellt werden, welche eine Textdatei `file` einliest, und die verschiedenen Woerter im Text identifiziert und auszaehlt. Als Ergebnis soll die Funktion die Gesamtzahl der Woerter `n` , sowie ein Woerterbuch `W` vom Typ `dict` zurueckliefern. Wenn das Schluesselwortargument `ReturnDict` den boolean Wert `True` hat, so soll das Woerterbuch durch die unterschiedlichen Woerter des Textes indiziert werden und als Werte die Anzahl der jeweiligen Woerter im Text besitzen. Zum Beispiel `W['die']==913` bedeutet, dass das Wort 'die' 913 mal im Text vorkommt. Wenn das Schluesselwortargument `ReturnDict` nicht angegeben wird, oder den boolean Wert `False` hat, so soll ein leeres Woerterbuch `W=dict([])` zurueckgegeben werden. Benutzen Sie Ihre Funktion `splitstring(s)` zur Identifizierung der Woerter in jeder Textzeile.
- Laden Sie die beiden Teile des Buches Die Leiden des jungen Werther von Johann Wolfgang von Goethe vom Project Gutenberg :

  http://www.gutenberg.org/cache/epub/2407/pg2407.txt (http://www.gutenberg.org/cache/epub/2407/pg2407.txt)
  http://www.gutenberg.org/cache/epub/2408/pg2408.txt (http://www.gutenberg.org/cache/epub/2408/pg2408.txt),

  und fuegen Sie die Texte, ohne die englischen Einleitungen und Lizenzbelehlungen in einer Datei zusammen.
- Wenden Sie die Funktion `textStats(file,ReturnDict=True)` auf diesen Text an. Wieviele Woerter stehen im Text?
- Geben Sie eine Tabelle mit den Top-100 der haeufigsten Woerter und deren Anzahl im Text aus. Welches sind die haeufigsten Substantive bzw. Eigennamen?
- Plotten Sie die Anzahl der verschieden Woerter im Text ueber deren Rang in doppelt-logarithmischen Skalen.
- Legen Sie eine Gerade durch die doppelt-logarithmischen Daten des Rang-Plots und geben Sie den Anstieg an.
- Die Anzahl der Vorkommen eines Wortes ist durch eine Zipf-Verteilung gegeben. Welchen Exponenten hat die Verteilung im Fall dieses Textes?

Von Affen und Schreibmaschinen :

- Schreiben Sie eine Funktion, welche eine zufaellige Folge von Buchstaben, Satzzeichen und Leerzeichen in eine Textdatei schreibt. Dabei soll in jedem Schritt ein Leer- oder beliebiges Satzzeichen mit der Wahrscheinlichkeit `p` ausgegeben werden, oder mit der Wahrscheinlichkeit `(1 - p)` ein beliebiger zufaelliger Buchstabe. Fuegen Sie gegebenenfalls auch regelmaeßig Zeilenumbrueche `'\n'` hinzu. Nachdem Sie eine hinreichend große Textdatei erzeugt haben, fuehren Sie nocheinmal eine Statistik ueber die zufaelligen Woerter in diesem Text durch.

Regulaere Ausdruecke

das Python `re` Modul : https://docs.python.org/3/library/re.html (https://docs.python.org/3/library/re.html)

Python re cheat-sheet : https://github.com/tartley/python-regex-cheatsheet/releases/download/v0.3.3/cheatsheet.pdf (https://github.com/tartley/python-regex-cheatsheet/releases/download/v0.3.3/cheatsheet.pdf)

Zipf Verteilung

Wikipedia : Zipf-Law (https://en.wikipedia.org/wiki/Zipf's_law)