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

Übungsblatt 2 zur Vorlesung
Big Data Analytics
im WiSe 2015/2016
Abgabe: 30.10.2015, 465 P

Bitte dokumentieren Sie die benötigte Bearbeitungszeit für die einzelnen Aufgaben in `bearbeitungszeit.txt` (Format der Zeilen: `AufgabenNr Zeit`). Bitte denken Sie auch daran uns Feedback zur Veranstaltung zu geben:

`http://goo.gl/forms/B01IIDHvW2`

Bitte denken Sie daran ihren Source Code zu kommentieren.

1 Bereinigung des Wikipedia-Textes: `wiki-clean-frequency.csv` und `wiki-clean.csv` (90 P)

In dieser Aufgabe erweitern wir Ihre Lösung von *Blatt 1, Aufgabe 4 („Wortanalyse der Wikipedia")* um die Analyse zu verbessern. Ziel der Aufgabe ist wieder die Häufigkeiten der Wörter zu erfassen, dieses mal wollen wir jedoch zuerst die Texte bereinigen und die Worte vor der Aggregation auf ihren Wortstamm zurückführen.
Als Ausgabeformat für die Datei `wiki-clean-frequency.csv` wählen wir wieder:

```
1 Artikelname, Artikellänge, Häufigkeit Wort1, Wort2, ... WortN
```

Wobei $Wort_i$ hierbei auf den Wortstamm zurückgeführt wurde (siehe unten).
Ebenfalls wollen die Datei `wiki-clean.csv` mit bereinigtem Markup gleichzeitig mit abspeichern. Diese sollte das gleiche Format wie die Eingabedatei `wikipedia-text.csv` aufweisen, nur dass die Spalte mit dem Text um das Markup bereinigt wurde.
Hinweis: Da die Wortanalysen hier noch etwas rechenaufwändiger sind testen sie mit dem Datensatz `/home/hatzel/bigdata/wikipedia-text-tiny.csv`. Sie können sich natürlich immer mit dem Programm `head` einen kürzeren Datensatz zum Testen erstellen.

1.1 Wiki-Markup

Ihnen ist wahrscheinlich aufgefallen, dass viele der Worte mehrfach in den Daten auftauchen, jedoch zum Teil von z.B. Syntax zur Verlinkung umgeben sind und somit trotzdem als unterschieldiche Worte in den Datensatz aufgenommen werden. Entfernen Sie aus dem Text des Datensatzes Teile des Wiki-Markups (`https://www.mediawiki.org/wiki/Help:Formatting`) und exportieren Sie den gefilterten Text mit den bereits bestehenden Informationen in eine neue CSV Datei. Speichern Sie diese Datei, wir werden diesen Datensatz in Zukunft einfach mit `wiki-clean.csv` bezeichnen.
Ein Beispiel:

```
The official name reflects [[History of Hamburg|its history]]
```

Hier wird ein Link auf „History of Hamburg" gesetzt Ihre neue CSV Datei soll dann den folgenden Text enthalten:

```
The official name reflects its history
```

Entfernen Sie mindestens das Markup für:

- Bold text

- Italic text
- Internal Links[1]
- Piped Links

Hinweis: Eine Korrekte Behandlung der gesamten Wiki-Markup Syntax ist **nicht** gefordert.

1.2 Wortstamm

Viele Wörter kommen im Datensatz in abgewandelter Form mehrfach vor, um detailliert Analyse zu ermöglichen bietet es sich an jedes Wort auf seinen Wortstamm zurückzuführen. Für diese Aufgabe verwenden wir das Natural Language Toolkit[2].
Im Modul `stem` befindet sich eine Auswahl sog. Stemmer, die einzelne Wörter auf ihren Wortstamm zurückführen können. Wir verwenden den snowball Stemmer. Erweitern Sie Ihre Lösung dahingehend das alle Wörter auf ihren Stamm zurückgeführt werden, speichern Sie Ihre Ergebnisse.
Die Verwendung des Stemmers geschieht folgendermaßen:

```
1 from nltk.stem.snowball import EnglishStemmer
2
3 my_words = ["tests", "performance"] # Hier die gefundenen Worte speichern
4
5 stemmer = EnglishStemmer()
6 for word in my_words:
7     new_word = stemmer.stem(word)
```

Abgabe:

`1-wiki-markup-cleaner.py` Ihre Source-Code für das Entfernen von Wiki-Markup und das zurückführen auf Wortstämme inklusive Kommentaren. Bewahren Sie `wiki-clean.csv` für die folgenden Aufgaben auf.

2 Verbesserte Analyse der Wikipedia-Artikel (45 P)

Nun soll die grafische Darstellung der Wikipedia-Artikel aus *Blatt 1, Aufgabe 4.2.3 („Worthäufigkeiten in Python“)* den bereinigten Text (`wiki-clean.csv`) aus *Aufgabe 1* zu verarbeiten. Zusätzlich soll die Häufigkeit eines gegeben Begriffs (übergeben als Kommandozeilenparameter) und aller seiner Synonyme in dem von uns gegebenen Ausschnitts der Wikipedia gefunden werden. Diese werden dann erneut als Bild ausgegeben, wobei Synonyme zusätzlich als weiterer Farbkanal codiert werden: Codieren Sie die Worthäufigkeiten der Synonyme in den roten Farbkanal und das Auftreten des Suchworts bzw. Wortstamms in Grün, d.h. ein Artikel mit gelben Pixel enthält sehr häufig sowohl Wortstamm als auch Synonyme des Eingabewortes. Testen Sie ob es geschickt ist die Worthäufigkeiten zusätzlich auf die Wortanzahl eines Artikels zu normieren oder ob es genügt die relative Worthäufigkeit eines Wortes über alle Artikel zu erfassen.
Suchen Sie nun ebenfalls im Wortsatz nach Synonymen für den angegebenen Begriff (spezifischer seinem Wortstamm). Als Beispiel: Das übergebene Suchwort ist „performant” daraus macht der Stemmer „perform”. Synonyme dafür sind: achieve, act, behave, complete, do, execute, finish, function, implement, meet, observe, operate, take und work.
Web scraping beschreibt Techniken um Informationen von Webseiten zu sammeln. Um Synonyme für Wörter zu finden fordern wir nun Inhalte (Webseiten) von: `http://www.thesaurus.com/` an. *Anmerkung:*

[1] `https://www.mediawiki.org/wiki/Help:Links`
[2] `http://www.nltk.org/`

Sie brauchen nur einen Abruf Website pro Programmaufruf zu machen da wir nur nach Synonymen zum Startwort suchen, so entsteht auch keine übermäßige Serverlast für den Anbieter.
Wie bei unserem Beispiel oben sollen auch bei Ihnen nur die Synonyme mit der höchsten Relevanz, also jene mit dem orangfarbigen Hintergrund (bzw. den entsprechenden CSS Klassen im HTML) berücksichtigt werden.

2.1 Hinweise

In Python ist das Modul `urllib` für HTTP Anfragen zu verwenden. In diesem einfachen Fall können Sie reguläre Ausdrücke verwenden um die relevanten Informationen aus dem HTML-Code zu extrahieren.
Da das Parsen der HTML Struktur mit regulären Ausdrücken nicht akkurat möglich ist und an mehreren Stellen im Dokument ähnliche Markup-Sequenzen auftauchen, empfiehlt es sich das Dokument einfach mit `str.split()` geschickt aufzuspalten und dann nur mit dem relevanten Teil des Dokuments zu arbeiten. Wenn Sie eher weniger Erfahrung mit regulären Ausdrücken haben, empfiehlt es sich diese interaktiv mit der Python-Shell oder z.B. auf `http://www.pythex.org` zu testen.

2.1.1 Python Code-Gerüst

```
import png
import math
import sys
from operator import itemgetter
import re
import urllib.request
from nltk.stem.snowball import EnglishStemmer

def get_synonyms(term):
    response = urllib.request.urlopen("http://www.thesaurus.com/browse/" + term)
    text = response.read()
    ...
    return synonym_list


# Zähle die Worthäufigkeiten im Text NACHDEM Wiki-Markup entfernt wurde.
# Gebe eine Liste mit Tripeln (Wortstamm, Synonyme, 0) zurück.
# Diese Information könnte ebenfalls direkt aus der CSV Datei von wiki-markup-cleaner.py ausgelesen werden.
def wordFrequency(cleanCSV, articles, word, synonyms):
    # Wort Stemmer verwenden um das Wort in ein Stammwort überzuführen
    new_word = stemmer.stem(word)

    csvData = ... readCSV(cleanCSV)...
    # Wort Stemmer verwenden um Text in Articles zu einem Stammwort überzuführen

    # TODO: korrekte Ausgabe
    count = [ (x, x, 0) for x in range(1,8500) ]
    return count

# Normalisiere die Anzahl der Worthäufigkeiten zu einer relativen Frequenz basierend auf dem Maximum
def normalizeFrequency(list_):
    mx0 = float(max(list_, key=itemgetter(0))[0])/float(255)
    mx1 = float(max(list_, key=itemgetter(1))[1])/float(255)
    mx2 = float(max(list_, key=itemgetter(2))[2])/float(255)
    if mx0 == 0: mx0 = 1
    if mx1 == 0: mx1 = 1
    if mx2 == 0: mx2 = 1
    return [ f(x) for x in list_ for f in (lambda y: int(y[0]/mx0), lambda y: int(y[1]/mx1), lambda y: int(y[2]/mx2)) ]


if __name__ == "__main__":
    # Eine Liste mit Tupeln der Worthäufigkeiten für jeden Artikel (hier 900 Artikel)
    # (Wortstamm, Synonyme, 0), wir fügen ein dritten Kanal für die spätere Verwendung mit 0 ein.


    word = sys.argv[1]
    count_data = wordFrequency("wiki-clean.csv", word, get_synonyms(word))
    word_freq = normalizeFrequency(count_data)

    # Berechne die Grö ß e
    width = int(math.ceil(math.sqrt(len(count_data))))

    # Extend the list to be squared
    word_freq.extend([0,0,0] * ( width**2 - len(count_data)))

    with open("mypngfile" + ".png", "wb") as picture_file:
        picture = png.Writer(width, width, compression=-1, bitdepth=8)
        picture.write_array(picture_file, word_freq)
```

Abgabe:

`2-synonyms.py` Ihren angepassten Sourcecode der Wort-Analyse mit Suche nach Synonymen, welches ein Bild basierend auf dem auf der Kommandozeilenzeilenparemeter übergebenen Suchwort erstellt.

3 Erstellung eines Datenmodells für Wikipedia (150 P)

Überlegen Sie sich ein Datenmodell zur Ablage von Wikipedia-Artikeln. Ein Artikel wird hierbei charakterisiert durch:

- Titel
- Volltext
- Kategorie (bzw. Unterkategorie)
- Verweise auf verwandte Artikel
- Externe Links
- Sektionen (und Untersektionen)

Dieses Datenmodell soll folgende Operationen effizient ermöglichen:

- Zugriff basierend auf dem Artikeltitel
- Navigation zwischen verwandten Artikeln
- Navigation zwischen Artikeln gleicher Kategorien bzw. Unterkategorien
- Identifikation von Artikeln, die eine bestimmte Sektion bzw. Untersektion aufweist

Zusätzlich wollen wir im Datenmodell Worthäufigkeiten für Artikel speichern.
Versuchen Sie hierfür jeweils ein Document Model[3], Columnar Model, Key-Value Model, Graph Model und Relational Model zu erstellen. Beschreiben Sie die wie die einzelnen Operationen umgesetzt werden und evtl. auftretende Probleme.

Abgabe:

`3-datenmodell.pdf` Dokument mit der Beschreibung Ihrer Datenmodelle (gerne auch Bilder zur Verdeutlichung).

4 Statistische Analyse der Worthäufigkeiten (R) (180 P)

Im folgenden wollen wir die erweiterte Wortanalyse der Wikipedia mit statistischen Methoden in R untersuchen und verbessern.
Hierfür gehen wir wie folgt vor:

- Verwenden Sie die Lösung aus *Aufgabe 1* und exportieren Sie die bereingten Textdaten als CSV. Verwenden Sie als Grundlage Ihren Datensatz `wiki-clean-frequency.csv` aus Aufgabe 1.1.
- Importieren Sie den Datensatz in R und aggregieren Sie die Worthäufigkeiten für alle Artikel.
- Führen Sie die unten aufgelisteten statistischen Analysen durch.

[3] Gehen Sie für das Document Model davon aus, dass Sie einzelne Dokumente über einen unique key zugreifen können.

- Dokumentieren Sie Ihre Ergebnisse.

Statistische Betrachtungen bei der Exploration der Daten:

- Durchschnittliche Häufigkeit eines beliebigen Wortes
- Erstellen Sie ein Histogramm welches die Häufigkeitsverteilung eines beliebigen Wortes über alle Artikel hinweg anzeigt.
- Erstellen Sie ein Histogramm welches die Häufigkeitsverteilung aller Wortvorkommnisse in Artikeln angibt.
- Plotten Sie die Dichte der Worthäufigkeiten.

Überlegen Sie welche Worte für die Inhalte der Artikel evtl. irrelevant sind und daher ignoriert werden sollten. Welche statistischen Betrachtungen kann man hierfür nutzen? Speichern Sie eine Liste mit Stoppworten als CSV-Datei unter `stoppwortlist.csv`.
Nun versuchen wir eine kurze Analyse mit Hilfe der induktiven Statistik vorzunehmen:

- Ermitteln Sie die Korrelation aller Worthäufigkeiten zu der Artikellänge. Welche Worte zeigen eine starke Korrelation und wie können Sie diese Information nutzen?
- Analysieren Sie ebenfalls die Korrelation der Wortfrequenz und Wortlänge.
- Erstellen Sie ein einfaches lineares Modell der stark korrelierten Wörter zur Artikellänge. Identifizieren Sie die Artikel, die am stärksten vom Modell abweichen (Outlier). Analysieren Sie diese mit Hilfe der deskriptiven Statistik.

Abgabe:

`4-statistics.R`	Ihre R Befehle mit kurzen Kommentaren.
`4-statistics.pdf`	Ihre Analyse mit Plots und einer kurzen Bewertung.
`4-stoppwortlist.csv`	Eine Liste mit Stoppwörtern (Ein Stoppwort pro Zeile).