Informatik-BW: Assignment 2 (SS 2023)

H. Leitner

29.03.2023

Zusammenfassung

Die Aufgabe im *Assignment 2* besteht darin, eine Textdatei einzulesen, ein Wort-Histogramm der gelesenen Wörter zu erzeugen und die am häufigsten verwendeten Wörter auszugeben.

1 Ziel der Hausübung

In diesem Hausübungsbeispiel sollen Sie den Umgang mit grundlegenden Python Operatoren, Python Datentypen und Python Kontrollanweisungen (bedingte Anweisung, Schleife) und Python Funktionen vertiefen. Die Verwendung von *sequentiellen Datentypen (Listen, Dictionaries)* wird hierbei eine zentrale Rolle in diesem Hausübungsbeispiel spielen. Weiters sollen Sie Kommandozeilen-Parameter einlesen können und grundlegende Funktionen zum Einlesen von Inhalten aus Dateien erlernen.

2 Aufgabenstellung

Die Aufgabe im *Assignment 2* besteht darin, eine Textdatei einzulesen, ein Wort-Histogramm der gelesenen Wörter zu erzeugen und die am häufigsten verwendeten Wörter auszugeben. Zusätzlich soll eine grafische Darstellung des Wort-Histogramms angezeigt werden. Es folgt eine detaillierte Beschreibung der Funktionsweise ihres Programms:

- Auswerten der Kommandozeilen-Parameter
- Zeilenweises Einlesen der angegebenen Textdatei
- Pro Zeile ist folgendes zu tun:
  - Zerlegen in einzelne Wörter
  - Satzzeichen bzw. Sonderzeichen am Anfang und Ende des Wortes entfernen
  - Alle Buchstaben des Wortes in Kleinbuchstaben umwandeln

  - Wort in einem *python dictionary* ablegen

- Ausgabe der folgenden Informationen (Beispiele, wie die Ausgabe genau aussehen kann, finden sie weiter unten):
  - Anzahl der Worte in der Textdatei
  - Anzahl unterschiedlicher Worte in der Textdatei
  - Die n-häufigsten Worte der Textdatei (die Anzahl "n" kann über Kommandozeilen-Parameter definiert werden; default Wert für "n" ist 10)
  - Grafische Darstellung des Wort-Histogramms

2.1 Einlesen von Kommandozeilen-Parameter

Ihr Programm muss die folgenden Kommandozeilen-Parameter verarbeiten können (Beispiel):

```
python assignment2.py -file word_hist0.txt -num 5
```

Die Reihenfolge der Parameter ist nicht fix vorgegeben; der folgende Aufruf entspricht also genau dem obigen Beispiel:

```
python assignment2.py -num 5 -file word_hist0.txt
```

Bedeutung der Kommandozeilen-Parameter:

- *-file <input-file-name>*: Angabe einer Textdatei, die verarbeitet werden soll; die folgenden fünf Testdateien (word_hist0.txt, Oliver_Twist-Charles_Dickens.txt, Pride_and_Prejudice-Jane_Austen.txt, The_Jungle_Book-Rudyard_Kipling.txt, Treasure_Island-Robert_Louis_Stevenson.txt) sind über das TU Graz TeachCenter verfügbar.

- *-num <anzahl-wörter>*: gibt an, wieviele der "häufigsten" Wörter ausgegeben werden sollen. Dieser Kommandozeilen-Parameter ist *optional*, muss also nicht zwingend verwendet werden; fehlt der Parameter, werden die *10 häufigsten* Wörter ausgegeben. Die zwei folgenden Programmaufrufe haben also die gleiche Wirkung:

  ```
  python assignment2.py -file word_hist0.txt -num 10
  python assignment2.py -file word_hist0.txt
  ```

Fehlen die notwendigen Parameter (Angabe der Textdatei), soll der folgende *Hilfetext* ausgegeben und das Programm beendet werden:

```
Usage: python assignment2.py -file <input-file-name> [-num <num_words>]
```

Wird auch die *Bonus Aufgabe* (siehe 2.4) implementiert, müssen zusätzliche Kommandozeilen-Parameter unterstützt werden: *-stop* und *-out*. Ein Aufruf mit allen Kommandozeilen-Parametern würde demnach so aussehen (in einer Zeile):

```
python assignment2.py -file word_hist0.txt -num 10 -stop stopwords-en.txt
       -out words.csv
```

Der vollständige Hilfetext sieht in diesem Fall so aus:

```
Usage: python assignment2.py -file <input-file-name> [-num <num_words>]
               [-stop <stopwords-filename>] [-out <csv-out-file>]
```

Bedeutung der zusätzlichen Kommandozeilen-Parameter finden sie unter Kapitel 2.4.

2.2 Anmerkungen und Hinweise

Ihr Programm soll für die Durchführung der verschiedenen Aufgaben eigene *Funktionen* verwenden. Ihr Programm könnte demnach die folgende Struktur haben (Vorschlag):

```
import string
import sys

def print_usage():
    print('Usage: python assignment2.py -file <input-file-name> [-num <num_words>]')

# Einlesen der Kommandozeilenparamter:
# -file <input-file-name>: Name einer Textdatei
# -num <num_words>: Anzahl der meisvorkommenden Wörter
# return: filename (Text), num (int)
def get_commandline_params(argv):
    print('in get_commandline_params:', argv)
    num = 10
    filename = None
    return filename, num

# Einlesen der Datei "filename"
# pro Zeile:
#   - in einzelne Worte aufspalten
#   - Satzzeichen (Sonderzeichen) am Anfang und Ende des Wortes entfernen
#   - Wort in Kleinbuchstaben umwandeln
#   - dict für Worte (key=<wort>, value=<#Auftreten des Wortes>)
def process_file(filename):
    print('in process_file:', filename)
    hist = dict()
    return hist
```

```
# Bestimmt eine sortierte Liste der Wörter nach Häufigkeiten
def most_common(hist):
    print('in most_common:', hist)
    most = []
    return most

# in der Funktion main stehen die Anweisungen des "Hauptprogramms"
def main():
    # print_usage(), wenn etwas mit den Kommandozeilenparametern nicht stimmt
    print_usage()

    # Einlesen der Kommandozeilenparameter
    filename, num = get_commandline_params(sys.argv[1:])

    # Histogramm dict für Wörter aus Datei aufbauen
    hist = process_file(filename)

    # Liste der Wörter nach Häufigkeit sortiert
    most = most_common(hist)

    print(f'\nWord histogramm of file: {filename}\n')
    print('Total number of words:', '...')
    print('Number of different words:', '...')
    print(f'The most common {num} words are:')
    print('...')

    # grafische Ausgabe des Wort-Histogramms:
    # ...

    # Ausgabe des Wort-Histogramms als csv-Datei (falls
    # Kommandozeilenparameter "-out" verwendet wurde...
    # ...

if __name__ == "__main__":
    main()
```

2.3 Programmausführung

So soll bzw. kann die Ausgabe Ihres Programms aussehen:

- bei Aufruf des Programms mit fehlenden oder falschen Parametern:
  python assignment2.py

  ```
  Usage: python assignment2.py -file <input-file-name> [-num <num_words>]
  ```

- bei korrektem Aufruf des Programms zum Beispiel mit:
  python assignment2.py -file Oliver_Twist-Charles_Dickens.txt -num 7 (Beispiel für die grafische Ausgabe des Ergebnisses: siehe Abb. 1)

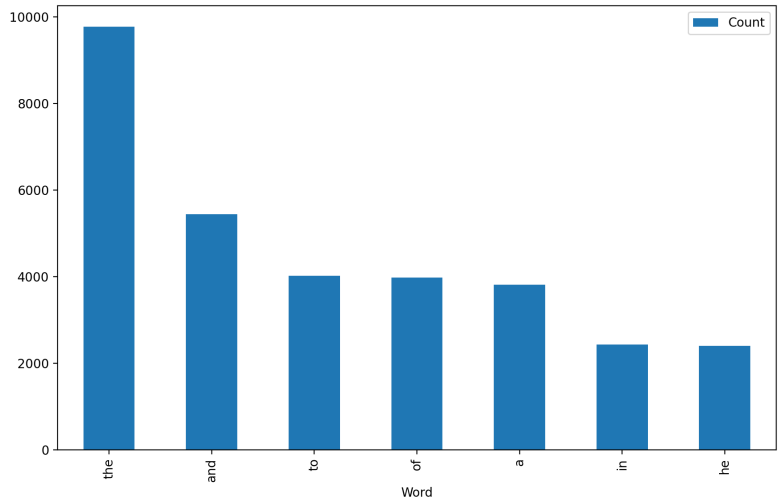

Abbildung 1: Word count: Oliver Twist

```
Word histogramm of file: Oliver_Twist-Charles_Dickens.txt

Total number of words: 162090
Number of different words: 11335
The most common 7 words are:
the                    9772
and                    5448
to                     4023
of                     3984
a                      3815
in                     2434
he                     2397
```

- Zuletzt noch das Ergebnis für die Testdatei *word_hist0.txt*; Aufruf mit: *python assignment2.py -file word_hist0.txt*

```
Word histogramm of file: word_hist0.txt

Total number of words: 91
Number of different words: 13
The most common 10 words are:
thirteen               13
twelve                 12
eleven                 11
ten                    10
nine                    9
eight                   8
seven                   7
```

```
six                     6
five                    5
four                    4
```

2.4 Bonus Aufgabe

Zusätzlich zu den obigen Anforderungen, können auch noch die folgenden Anforderungen implementiert (=programmiert) werden. Bei korrekter Ausführung gibt es dafür bis zu *5 Bonuspunkte*.
Um die Ausgabe des Wort-Histogramms noch nützlicher zu gestalten, sollen (sehr) häufig vorkommende Wörter wie *Bindewörter* (und, als, denn, ...), *Artikel* (der, die, das) etc. nicht berücksichtigt werden. Dazu verwendet man eine *Stopwort-Liste* (eine deutsche und englische Stopwort-Liste ist über das TU Graz TeachCenter verfügbar (siehe unten). Vorgehensweise:

- Stopwort-Liste muss eingelesen werden; dazu braucht es einen zusätzlichen (optionalen) Kommandozeilen-Parameter: *-stop <stopwords-filename>*
- Stopworte werden in einer python Liste gespeichert
- Beim Aufbau des Wort-Histogramms werden Wörter, die aus der Textdatei (z.Bsp. Oliver_Twist-Charles_Dickens.txt) eingelesen werden, gegen die Stopwort-Liste geprüft: nur wenn das Wort nicht in der Stopwort-Liste enthalten ist, wird es in das Wort-Histogramm aufgenommen, andernfalls wird es einfach ignoriert.

Zusätzlich kann der Benutzer über Angabe des Kommandozeilen-Parameters *-out <csv-out-file>* die Ausgabe des Wort-Histogramms in einer *.csv*-Datei speichern.
Anmerkung: Eine *.csv*-Datei speichert Tabellendaten in Textform ab, wobei als Trennzeichen der Felder einer Tabellenzeile meist ein „Komma (,)“ verwendet wird.
Beispiel: bei folgendem Aufruf
`python assignment2.py -file word_hist0.txt -out words.csv`
soll eine Datei mit Namen *words.csv* erzeugt werden, die den folgenden Inhalt hat:

```
Word,Count
thirteen,13
twelve,12
eleven,11
ten,10
nine,9
eight,8
seven,7
six,6
five,5
four,4
```

2.5 Hilfs- und Testdateien

Es folgt eine Liste der Dateien, die zum Testen ihres Programms verwendet werden sollen (inkl. Beschreibung). Diese Dateien sind über das TU Graz TeachCenter unter `https://tc.tugraz.at/main/course/view.php?id=1187` abrufbar. Die

Textdateien “Oliver Twist” von *Charles Dickens*, “Pride and Prejudice” von *Jane Austen* “The Jungle Book” von *Rudyard Kipling* und “Treasure Island” von *Robert Louis Stevenson* werden vom *Gutenberg Projekt* (siehe `https://www.gutenberg.org`) zur Verfügung gestellt. Von dieser Web-Site können natürlich noch zusätzliche Textdateien für weitere Test heruntergeladen werden.

InfBW_SS2023_assignment2.pdf: Anforderungdefinition bzw. Aufgabenbeschreibung

word_hist0.txt: einfache Testdatei bestehend aus den 13 unterschiedlichen Wörtern “one”, “two” bis “thirteen”

Oliver_Twist-Charles_Dickens.txt: “Oliver Twist” von *Charles Dickens*

Pride_and_Prejudice-Jane_Austen.txt: “Pride and Prejudice” von *Jane Austen*

The_Jungle_Book-Rudyard_Kipling.txt: “The Jungle Book” von *Rudyard Kipling*

Treasure_Island-Robert_Louis_Stevenson.txt: “Treasure Island” von *Robert Louis Stevenson*

stopwords-de.txt: deutsche Stopwort-Liste (nur für Bonusaufgabe notwendig)

stopwords-en.txt: englische Stopwort-Liste (nur für Bonusaufgabe notwendig)

3 Abgabe

Eine Datei mit Namen: *assignment2.py* bzw. *assignment2.ipynb* (für “jupyter notebook” Nutzer)

- *assignment2.py:* (bzw. *assignment2.ipynb*) enthält ihr Python Programm

Achtung: sollten ihre Dateinamen nicht den Vorgaben entsprechen, kommt es zu einem Punketabzug!

Erreichbare Punktezahl: *35*
Bonuspunkte: *5*
Max. erreichbare Punktezahl inkl. Bonuspunkte: *40*

Abgabetermin: bis zum *05.05.2023 22:00 Uhr*

Ich wünsche Ihnen viel Erfolg!!