Aufgabe 1: Störung

Team-ID: 00246

Team: http://localhost:80/

Bearbeiter/-innen dieser Aufgabe:
Alessio Caputo

20. November 2022

Inhaltsverzeichnis

Lösungsidee

Um alle möglichen Sätze, oder Wortabfolgen zu bestimmen wird die zusuchende Wortabfolge an jede mögliche Position im Text angelegt. Sollte eine übereinstimmung vorhanden sein, so wird die Wortabfolge ausgegeben.

Umsetzung

Die Lösungsidee wird in Python implementiert. Zunächst wird die Datei, die den Text enthällt eingelesen und die enzelnen Wörter in eine Liste geschrieben. Vorher werden alle Satzzeichen und Zeilenumbrüche aus dem Text eliminiert, da diese für die Berechnungen keine, oder nur eine hindernde Rolle spielen. Anschließend wird die Datei mit der Störung eingelesen und mit dem selben Verfahren in eine zweite Liste geschrieben. Nun wird mit zwei ineinandergeschachtelten For-Schleifen jede Position in der Wörtterliste mit der Störung verglichen. Die zweite for-Schleife überprüft von der zu überprüfenden Position aus alle weiteren Wörter. Dabei werden die Wörter „_“ einfach ignoriert. Sollten alle Wörter eine übereinstimmung aufweisen, so wird der ermittelte Textabschnitt ausgegeben. Sollte keine übereinstimmung festgestellt werden, so wird die zweite For-Schleife frühzeitig abgebrochen, um die nächste Position zu überprüfen.

Beispiele

stoerung0.txt	das kommt mir gar nicht richtig vor
stoerung1.txt	ich muß in clara verwandelt ich muß doch clara sein
stoerung2.txt	fressen katzen gern spatzen fressen katzen gern spatzen fressen spatzen gern katzen
stoerung3.txt	das spiel fing an das publikum fing an
stoerung4.txt	ein sehr schöner tag
stoerung5.txt	wollen sie so gut sein

Auf die erste Lösung von stoerung1.txt schauend kann gesehen werden, dass auch nur alleinstehende Satzausschnitte ausgegeben werden können. Wenn länger darüber nachgedacht wird kann man zu dem schluss kommen, dass nur von den Satzanfängen aus überprüft werden muss. Damit würde die Effizienz des Programmes deutlich gesteigert werden. Auch ist in der Aufgabenstellung die Rede von Sätzen. Dennoch ist es wichtig auch diese Lösungen anzuzeigen, da der Absender auch einen „halben Satz“ versenden könnte. Eine Kombination der beiden Ansätze wäre eventuell auch von vorteil, da hier zuerst die „wahrscheinlicheren“ Positionen ermittelt werden und somit eine schnellere außgabe dieser erfolt. Andererseits ist das Script bereits sehr performant (kein delay nach ein- und ausgabe bemerkbar), was dieses Feature letzlich drosseln würde.

Quellcode

```
from sys import argv
punctuation = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~«»\n'


def parseStringListToSentence(l:list) -> str:
    s = ""
    for i in l:
        s = s+i+" "
    return s


with open(argv[1], "r") as textfile:
    f = textfile.read()
    text = f..translate(str.maketrans('', '', punctuation)).lower().split(" ")
    with open(argv[2], "r") as searchsentensefile:
        searchwords = searchsentensefile.read().replace("\n", "").split(" ")
        for i in range(len(text)-len(searchwords)):
            for j in range(len(searchwords)):
                if searchwords[j] == "_":
                    if j == len(searchwords)-1:
                        print(parseStringListToSentence(text[i:i+len(searchwords)]))
                    else:
                        continue
                elif text[i+j] == searchwords[j]:
                    if j == len(searchwords)-1:
                        print(parseStringListToSentence(text[i:i+len(searchwords)]))
                else:
                    break
```