Aufgabe 1 - Störung

> **Team-ID** 00339
>
> **Team-Name** SilverBean
>
> **Bearbeiter der Aufgabe** Anton Batgauer
>
> 20. November 2022

Einleitung

Für die Lösung der Aufgabe "Störung" wird ein Suchalgorythmus verwendet, der jedes Wort des Buchs "Alice im Wunderland" mit den Elementen der Störung abgleicht. Sollte ein Abschnitt im Buch gefunden werden bei dem jedes Wort der eingelesenen Störung übereinstimmt, wird diese Stelle gespeichert und die Suche fortgeführt bis das Ende des Buchs erreicht wurde.

Lösungsidee

Zu aller erst wird das Buch in eine Liste umgewandelt, wobei jedes Element der Liste einer Zeile im Buch entspricht. Ähnliches wird mit der Störung gemacht, hier ist aber jedes Element ein Wort.

Für die Umsetztung wird nun eine Liste "arr" eingeführt, die wie ein temporärer Speicher ist und die Zeilen des Buchs behält, um sie mit der Stoerung abzugleichen. Man kann sich diese Liste wie die Augen eines Lesers vorstellen.

Der Suchalgorythmus läuft wie folgt ab:

1. Eine Zeile aus dem Buch wird Wort für Wort in die Liste "arr" hinzugefügt
2. Solange die Länge der Liste "arr" größer ist als die Länge der Störung, wird die Liste "arr" in einem for-loop mit der Störung abgegelichen:
    2. 1 Wird eine Stelle gefunden, so wird diese gespeichert und aus der Liste "arr" entfernt
    2. 2 Wird keine Stelle gefunden, so wird nur das erste Wort entfernt und der Algorythmus ab Punkt 2 wiederholt
3. Sollte der unter 2. beschriebene Fall nicht eintreten, wird der Suchalgorythmus von Punkt 1 gestartet, um Lösungen in Zeilenumbrüchen finden zu können

Dieser Ablauf wird für jede Zeile des Buchs in einer while-Schleife ausgeführt und am Ende die Lösungen ausgegeben. Je nach Menge an gefundenen Buchabschnitten lässt sich beurteilen, ob die Störung auf eine eindutige Stelle verweist, auf mehrer mögliche Stellen oder ob es keine gibt. Um die Suche im Buch zu vereinfachen gibt es auch eine Zeilenangabe.

Umsetzung

Die Lösung der Aufgabe wurde in Pyhton 3.10.5 implementiert. Dabei werden das Buch "Alice im Wunderland" als Textdatei und die Störungsbeispiele als Textdatei in den eingebauten Datentyp list umgewandelt. Auch der Zwischenspeicher des eingelesenen Buchabschnitts handelt es sich um den Datentyp list. Die Lösungen werden dabei in einer Liste als tupel gespeichert und am

Ende ausgegeben. Beim Start des Programms wird der Benutzer nach dem Pfad für die Störungsdatei gefragt. Nach Durchlauf werden dann entweder die gefundenen Abschnitte mit Zeilenangabe ausgegeben oder eine Meldung, dass nichts gefunden wurde.

Beispiele

Die folgenden Beispiele sind die Ausgaben für das gegebene "Alice_im_Wunderland.txt" und die einzelnen Stoerfälle stoerung0.txt bis stoerung5.txt und lassen sich auf der bwinf-Website finden.

Pfad: stoerung0.txt
line: 440
»Das kommt mir gar nicht richtig vor,«

Pfad: stoerung1.txt
line: 425
Ich muß in Clara verwandelt

line: 442
»Ich muß doch Clara sein,

Pfad: stoerung2.txt
line: 214
»Fressen Katzen
gern Spatzen?

line: 214
Fressen Katzen gern Spatzen?

line: 215
Fressen Spatzen gern Katzen?«

Pfad: stoerung3.txt
line: 2320
das Spiel fing an.

line: 3302
'Das Publikum fing an,

Pfad: stoerung4.txt
line: 2293
ein sehr schöner Tag!«

Pfad: stoerung5.txt
line: 2185
»Wollen Sie so gut sein,

Quellcode

```
# Buch einlesen
with open("beispiele/Alice_im_Wunderland.txt", encoding="utf-8") as file:
    buch = list(filter(None, file.readlines()))

# Gestörte Nachricht einlesen
with open(input("Pfad: "), encoding="utf-8") as file:
    stoerung = file.readline().split(" ")

line_number = 0
list_of_results = []
arr = []

while line_number < len(buch):
    arr += buch[line_number].split(" ")
    while len(arr) > len(stoerung):
        for i, word in enumerate(stoerung):
            if not (word in arr[i] or word.capitalize() in arr[i] or word ==
"_"):
                # Wenn die Lösung nicht gefunden wird (was die meiste Zeit
passiert) wird das erste Wort gelöscht
                arr.pop(0)
                break
            if i == len(stoerung) - 1:
                # hier ist die lösung gefunden worden
                list_of_results.append((" ".join(arr[: len(stoerung)]),
line_number))

                # Die gefundene Lösung wird aus arr entfernt
                for g in range(len(stoerung)):
                    arr.pop(0)
                break
    line_number += 1

# Lösungsausgabe
for line, line_number in list_of_results:
    print("line: " + str(line_number + 1))
    print(line)
    print("")
if not list_of_results:
    print("nichts gefunden")
```