Aufgabe 1: Störung

Team-ID: 00464

Team: Enrique Lopez

Bearbeiter/-innen dieser Aufgabe:
Enrique Lopez

1. November 2022

Inhaltsverzeichnis

Lösungsidee

Zunächst muss der Inhalt der Textdatei, welche das Buch enthält, in das Programm eingelesen und alle Sonderzeichen aus ihr entfernt werden, worauf die einzelnen Wörter einer Liste hinzugefügt werden. Da alle Beispiele ausschließlich kleingeschrieben sind, müssen die Wörter auch in Kleinschrift umgewandelt werden.

Nun liest man auch den Lückensatz ein und ermittelt die Wortanzahl und die Position des ersten Wortes, welches keine Lücke ist. Vorkommnisse dieses Wortes können nun in der Wortliste des Buches gesucht werden und alle Sätze im Buch werden herausgefiltert, welche die richtige Länge und das Suchwort an der richtigen Position besitzen. Daraufhin wird zusätzlich eine Kopie dieser Sätze erstellt.

Jetzt werden die Positionen der Lücken im Lückensatz sowohl in diesem selbst als auch in den ausgewählten Sätzen entfernt, sodass darin nur noch die Wörter übrig bleiben, welche sich an einer Position im Lückensatz befinden, an welcher keine Lücke ist. Nun kann jeder Ausgewählte Satz mit dem veränderten Vorlagesatz verglichen werden:

1. Sind sie identisch, ist der Satz eine Lösung und die Kopie wird ausgegeben

2. Sind sie verschieden, so ist der Satz keine Lösung und wird nicht ausgegeben

Umsetzung

Eine Funktion wird zur Ermittlung einer passenden Zeile definiert. Dabei werden ein Textdokument mit der enthaltenen Zeile und ein Textdokument mit dem Lieblingsbuch übergeben.

Die Lückenzeile wird eingelesen und die einzelnen Worte und Lücken werden als Objekte einer Liste hinzugefügt.

Die Zeilen des Lieblingsbuches werden eingelesen und zu einem großen String zusammengefügt und daraufhin wird über die Zeichen des Strings iteriert, wobei nur diejenigen Zeichen in einen neuen String übernommen werden, welche keine Sonderzeichen sind. Der neue String wird an den Whitespaces gespalten und es entsteht eine Liste mit allen Wörtern des Buches in Reihenfolge. Die Wörter dieser Liste werden durch Iteration mit .lower() zu Kleinbuchstaben konvertiert.

Die Länge des Lückensatzes und die Position des ersten Wortes im Lückensatz werden ermittelt.

Nun wird über die Wörter des Buches mit einer for-Schleife iteriert. Wenn ein Wort im Buch mit dem ersten regulären Wort im Lückensatz übereinstimmt, so wird mithilfe von Slicing ein Satz mit genauer Länge und Position des Wortes in die neue Liste “selection” übernommen. Für später wird eine Kopie dieser Liste erstellt, wobei hierbei copy.deepcopy() verwendet werden muss, damit auch wirklich eine neue Nested List erstellt wird und nicht nur eine neue Referenz auf den alten Inhalt.

Die Indexe der Lücken im Vorlagesatz werden einer Liste hinzugefügt und die Lücken werden aus dem Satz entfernt. Nun wird in jedem Satz aus der “selection” Liste die Worte an den Positionen der Lückenindexen zunächst in ein Spezielles Symbol (‘*’) verändert, damit sie anschließend entfernt werden können.

Nun können Vorlage und Auswahlsätze verglichen und einer Liste namens “Solution” übergeben werden. Diese wird schließlich zum Set gemacht, um Duplikate zu vermeiden, und ausgegeben. Wenn das Set leer ist, so gibt es keine passenden Buchstellen und ein Text wird ausgegeben.

Schließlich wird die fertige Funktion ausgerufen. Über Text-Input kann man nun den Dateinamen mit dem Lückensatz angeben und bekommt die passenden Buchseiten in einem Set ausgegeben. Das Programm ist überraschend schnell.

Beispiele

1. stoerung0.txt das _ mir _ _ _ vor

```
Out[5]: {'das kommt mir gar nicht richtig vor'}
```

2. stoerung1.txt ich muß _ clara _

```
Out[6]: {'ich muß doch clara sein', 'ich muß in clara verwandelt'}
```

3. stoerung2.txt fressen _ gern _

```
Out[7]: {'fressen katzen gern spatzen', 'fressen spatzen gern katzen'}
```

4. stoerung3.txt das _ fing _

```
Out[8]: {'das spiel fing an'}
```

5. stoerung4.txt ein _ _ tag

```
Out[9]: {'ein sehr schöner tag'}
```

6. stoerung5.txt wollen _ so _ sein

```
Out[10]: {'wollen sie so gut sein'}
```

7. lücke_erste_stelle.txt _ kannst _ _ weilchen _

```
Out[11]: {'du kannst ihn ein weilchen warten'}
```

8. viele_sätze.txt alice _ _ _ _ _ _

Set mit 379 mit “alice” beginnenden Sätzen wird ausgegeben

Quellcode

```
def find_lines(line,book = "Alice_im_Wunderland.txt"):
    # line - String mit Namen der Text Datei mit Zeile und Lücken. Beispiel: "line1.txt"

    import copy

    #Lückenzeile auslesen
    with open(line) as l:
        line = l.readline()
        words = line.split()

    #Buch auslesen, alle Sonderzeichen entfernen und Liste mit Wörtern machen
    with open(book, encoding="utf-8") as b:
        full = ''.join(b.readlines())
        fuller = ""
        for char in full:
            if not char in [",",".","!","?","(",")","[","]","»","«","-","_","*",";"]:
                fuller += char
        fuller = fuller.split()
        for i in range(len(fuller)):
            fuller[i] = fuller[i].lower()

        #Zeilenlänge
        length = len(words)

        #Erstes verfügbares Wort in Lückenzeile ermitteln
        for i in range(len(words)):
            if words[i] != "_":
                first = i
                break

        #Gleiche Länge + Wort -> Auswahlliste
        selection = []
        for i in range(len(fuller)):
            if fuller[i] == words[first]:
                selection.append(fuller[i-first:i+(length-first)])

        #Kopie mit Originalsätzen
        satze = copy.deepcopy(selection)


        #Positionen der Lücken im Satz ermitteln
        indexes = []
        for i in range(len(words)):
                if words[i] == '_':
```

```
                indexes.append(i)

        #Alle Lücken im Lückensatz entfernen
        for i in range(words.count("_")):
            words.remove("_")

        #Lückenstellen bei Sätze in Auswahl entfernen
        solution = []
        for n in range(len(selection)):
            for i in range(length):

                if i in indexes:
                    selection[n][i] = "*"

            for i in range(selection[n].count("*")):
                selection[n].remove('*')


            #Lösungen: Gleiche Worte wie Vorlage a gleichen Positionen
            if selection[n] == words:
                solution.append(" ".join(satze[n]))
        solution = set(solution)
        if len(solution) == 0:
            return print('Keine passenden Buchstellen!')
        print(len(solution))
        return solution


find_lines(line = input('Dateiname der Lückenzeile: '))
```