Aufgabe 1: Störung

Team-ID: 00811

Team: Lorian

Bearbeiter/-innen dieser Aufgabe:
Lorian Linnertz

15. November 2022

Inhaltsverzeichnis

Lösungsidee

Meine Lösungsidee, ist eine relativ simple. Sie besteht aus drei Etappen. Als erstes vereinfacht man den Text, in dem man alle Sonderzeichen entfernt und alle Wörter in Kleinbuchstaben in eine große Liste fügt.

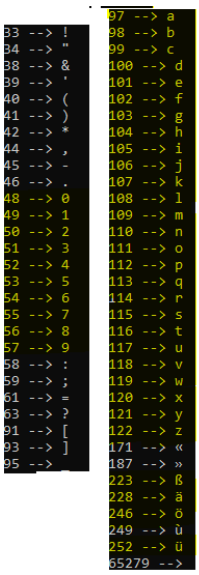
Als nächste wurde man das erste Wort aus der Störung nehmen und solange durch die Liste mit den Wörtern iterieren bis man einen Treffer hat. Bedeutet dass das Wort aus der Liste das gleiche wie das erste Wort der Störung.

Wenn man nun die Textstelle gefunden hat wo der Anfangsverdacht besteht, dass sich die Störung dort befinden könnte überprüft man nun die restlichen Wörter aus der Störung. Falls diese auch übereinstimmen, so hat man eine mögliche Lösung für die Störung gefunden.

Man iteriert so lange durch den Text bis man am Schluss angekommen ist.

Umsetzung

Bei der Umsetzung des 1.Schrittes war der komplizierteste Teil die Entfernung aller Sonderzeichen. Um diese Problem zulösen, habe ich ein sparates Program geschrieben, welches durch alle Zeichen iteriert und eine Liste erstellt, welche alle ORDs enthält welche im Text vorkommen. Aus dieser Liste habe ich alle Ords welche ein Sonderzeichen repräsentiern mir rausgeschrieben um diese Werte im Hauptprogram zu entfernen. In diesem Fall will ich alle ORDs welche nicht gelb sind entfernen. Um dies Umzusetzten,



habe ich die Build-in Funktion translate() verwendet: Diese Funktion benötigt ein Dic in welchem man die ords als Keys benutzt und dann festlegt durch was diese Ord ersetzt werden soll. Indem ich für alle Ords die ich entfernen will None eingesetzt habe, wurden alle Sonderzeichen entfernt.

In der Funktion find_stoerung interiert das Programm durch alle Wörter aus dem Text und überprüft ob dieses Wort das gleiche ist wie das 1.Wort aus der Störung. Sollte dies Zutreffen, so ruft diese if_Schleife die Funktion Check_Stoerung auf.

Die Funktion Check_Stoerung überprüft nun die Restlichen Wörter aus der Störung, bedeutet falls wie im Beispiel das erste Wort aus der Stoerung mit dem Wort übereinstimmt, so überprüft es auch noch ob das 3.Wort mit dem Textübereinstimmt, indem es 2 Wörter weiter nachvorne geht im Text und so weiter. Für jedes Wort fügt die Funktion ein True/False Wert in die Liste ein jenach dem ob das Wort übereinstimmt oder nicht. Und gibt diese Liste zurück.

Falls nun alle Werte in dieser Liste True sind, so hat man einen Treffer gefunden, welcher in der Liste mit den möglichen Resultaten gespeichert wird.

Man iteriert einmal durch den ganzen Text und anschließend gibt man die Resultate aus.

PS: Mein Program ist komplexer aufgebaut, als es für nötig erscheint, da ich ein Spezialfall abdecken wollt, welche zwar nicht in den Beispielen vorkommt, theoretisch aber möglich ist. Dieser Spezialfall ist, wenn das erste Wort ein unbekanntes Wort ist, somit kann man nicht mehr nach dem ersten Wort in der Störung suchen, sondern muss nach dem erst möglichen Wort suchen, dazu habe ich die Funktion find_words erstellt, welche alle Bekannten Wörter mit ihren Indexen in Form eines Tupels in einer Liste speichert. Wenn man nun das erstmögliche Wort betrachten möchte, nimmt man einfach das [0] Element aus der Liste. Wichtig ist ebenfalls, dass man überall im Program, wo man mit den Indexen rechnet den Index von Element[0] abzieht, da die Anderen Indexe ebenfalls um diesen Wert zuhoch sind.

Beispiele

----------stoerung0.txt----------

Störung: ['das', '_', 'mir', '_', '_', '_', 'vor']

1.Stelle: das kommt mir gar nicht richtig vor

----------stoerung1.txt----------

Störung: ['ich', 'muß', '_', 'clara', '_']

1.Stelle: ich muß in clara verwandelt

2.Stelle: ich muß doch clara sein

----------stoerung2.txt----------

Störung: ['fressen', '_', 'gern', '_']

1.Stelle: fressen katzen gern spatzen

2.Stelle: fressen spatzen gern katzen

----------stoerung3.txt----------

Störung: ['das', '_', 'fing', '_']

1.Stelle: das spiel fing an

2.Stelle: das publikum fing an

----------stoerung4.txt----------

Störung: ['ein', '_', '_', 'tag']

1.Stelle: ein sehr schöner tag

----------stoerung5.txt----------

Störung: ['wollen', '_', 'so', '_', 'sein']

1.Stelle: wollen sie so gut sein

----------Sonderfall.txt----------

Störung: ['_', '_', 'das', '_', 'mir', '_', '_', '_', 'vor']

1.Stelle: wohl genähret das kommt mir gar nicht richtig vor

*** Dies ist eine veränderte Variante von stoerung0.txt***

Quellcode

```
def create_dic():
    dic = {}
    for i in [33,38,39,40,41,42,44,45,46,48,49,50,51,52,53,54,55,56,57,58,59,61,63,91,93,95,171,187,65279]:#list(range(33,96)) + list(range(122,222)) + [65279]:
        dic[i] = None #Als ersatz setzen wir None ein, bedeutet dass wir diesen Ord durch nichts ersetzten und somit löschen
    return dic


dic = create_dic() #Erstellt ein Dic mit allen Ords, welche entfernt werden sollen
text = []
with open("Alice_im_Wunderland.txt",encoding="utf-8") as f: #Öffnet den Text mit encoding="utf-8"
    lines = f.readlines()
    for l in lines: #Diese for schleife liest alle Linien des Textes ein
        l = l.strip().lower() #Entfernt "\n" und macht alle Buchstaben zu lower Cases
        l = l.translate(dic) #Entfernt alle unnötige zeichen, welche keine Buchstaben oder Zahlen sind mithilfe der translate Funktion
        l = l.split() #Spaltet den String in eine Liste aus den einzelnen Wörtern auf
        text.extend(l) #Fügt die Wörter zum rest des Textes hinzu



def get_stoerung(name): #Diese Funktion soll aus der Datei name, die Störung rauslesen
    with open(name,encoding="utf-8") as f:#Dazu liest es das Dokument ein
```

```
    line = f.readlines()
    line = line[0].strip() #Und entnimmt nur die erste Zeile und entfernt das "\n"
    line = line.split() #am ende Spaltet es den String noch in seine einzelnen Wörter/Zeichen auf und gibt eine Liste zurück
    return line


def find_words(stoerung): #Diese Funktion soll alle bekannten Wörter in der Störung extrahieren
  list = [] #In dieser Liste sollen die Indexe gepseichert werden auf welchen sich ein Wort befindet
  for i,word in enumerate(stoerung): #Iteriert durch die Störung
    if word != "_": #Falls es sich bei dem Wort nicht um ein gesuchtes Wort handelt, dann
      list.append((i,word)) #fügt man das Wort mitsamt ihrer Position in der Störung zur Liste hinzu
  return list



def check_stoerung(stoerung_list,text,index):
  bool_list = [] #Hier soll ein True/False Wert für jedes bekannte Wort aus der Störung eingefügt werden. True bedeutet dass das Wort aus der Störung mit der Textstelle übereinstimmt, False bedeutet nicht
  for s in stoerung_list: #iteriert durch die bereits bekannten Wörter
    bool_list.append(text[index+s[0]-stoerung_list[0][0]] == s[1]) #index+s[0] Dieser Ausdruck überprüft ob das nächste Wort nach dem ersten Treffer auch übereinstimmt. "-stoerung_list[0][0]" Dieser Ausdruck dient dazu falls das erste Wort bei der Störung eine Unbekannte ist. PS: Im Normalfall ist dieser ausdruck = -0
  return bool_list


def find_stoerung(stoerung):
  stoerung_list = find_words(stoerung)
  possible_results = []
  for i,word in enumerate(text):
    if stoerung_list[0][1] == word and all(check_stoerung(stoerung_list,text,i)): #Überprüft ob das erste Wort der Störung mit der aktuellen Textstelle übereinstimmt. Falls dies Zutrifft, so kontrolliert die Funktion check_stoerung() ob ebenfalls der Rest der Störung mit der aktuellen Textstellen übereinstimmt
      #Falls alle bools der Funktion check_stoerung True sind, so stimmt die Störung mit der Aktuellen Textstelle überein
      possibility = text[i-stoerung_list[0][0]:i+len(stoerung)-stoerung_list[0][0]] #Dann wird die Textstelle als possibility gespeichert. Hier dient die Stelle "-stoerung_list[0][0]" wieder für den Sonderfall, dass die Störung mit einem Unbekannten Wort anfangen sollte
      if possibility not in possible_results: #überprüft ob es keine Duplikate gibt
        possible_results.append(possibility) #fügt die mögliche Lösung zur Liste hinzu
  output(possible_results)


def output(result): #Printet die Gefunde Stelle
```

```
    for i,r in enumerate(result):
        string = " ".join(r)
        print(f"{i+1}.Stelle: {string}")


if __name__ == '__main__':
    filenames = ["stoerung0.txt","stoerung1.txt","stoerung2.txt","stoerung3.txt","stoerung4.txt","stoerung5.txt"]#"stoerung0.txt","stoerung1.txt","stoerung2.txt","stoerung3.txt","stoerung4.txt","stoerung5.txt"
    for name in filenames:
        print(f"\n----------{name}----------\n")
        stoerung = get_stoerung(name)
        print(f"Störung: {stoerung}\n")
        find_stoerung(stoerung)
```