Aufgabe 1: Störung

Team-ID: 00382

Team-Name: Sehr gutes BWINF Team

Bearbeiter/-innen dieser Aufgabe:
Paul Franosch, Joshua Benning, Lenny Schürer, Marinus Lentile

20. November 2022

Inhaltsverzeichnis

1 Lösungsidee

Das gesuchte Satzfragment wird eingelesen und in eine Regular Expression überführt. Außerdem wird das Buch von seinen Satz- und Sonderzeichen sowie Einrückungen bereinigt. Anschließend wird mittels der Regular Expression alle in Betracht kommenden Textstellen gesucht und letztendlich ausgegeben.

2 Umsetzung

Die Implementierung erfolgt in Python 3.9.
Es wird das Python `re` module verwendet. Dieses ermöglicht es, Reguläre Ausdrücke (Regular Expressions, Regex) in Python zu verwenden.
Zunächst wird das Buch eingelesen und bereinigt, um das spätere Suchen von passenden Satzfragmenten zu vereinfachen. Dazu werden verschiedene Regular Expressions sowie Python String Manipulation verwendet. Danach wird das jeweilige Textfragment eingelesen, und daraus ein entsprechender Regex geformt, welcher alle passenden Textstellen sucht.

2.1 Buch bereinigen

Im ersten Schritt wird alles, das kein Wort oder Leerzeichen ist, entfernt.
Dies passiert mittels folgendem Regex: `[^\w\s]`
Die `re\#sub` Funktion findet und ersetzt alle Textstellen, welche vom angegeben Regex gefunden werden. Anschließend werden alle Zeilenendezeichen durch Leerzeichen ersetzt und alle Einrückungen mittels Tabulator-Taste, sowie Unterstriche, entfernt. Dafür wird Pythons `String\#replace` Methode verwendet. Abschließend werden alle überflüssigen Leerzeichen mittels diesem Regex `\s{2,}` gefunden und durch ein einzelnes Leerzeichen ersetzt.

2.2 Textfragment finden

Um alle passenden Textstellen im nun bereinigten Buchtext zu finden, werden abermals Regular Expressions benutzt. Ziel ist es, eine Regular Expression anhand des Textfragments zu generieren, welche auf alle Textstellen zutrifft, die für das jeweilige Fragment "passend" sind. Dafür wird das eingelesene Textfragment an den Leerzeichen getrennt. Über die dadurch entstandenen Satzfragmentteile wird im nächsten Schritt iteriert. Handelt es sich bei einem Teil um einen Unterstrich, soll also ein beliebiges Wort an dieser Stelle des Fragmentes passen, so wird dem entstehenden Regex dieser Regex `([^\s]+)\W` angehängt. Handelt es sich bei einem Teil allerdings um ein tatsächliches Wort, wird der Prefix `(`, dann das Wort, und dann der Suffix `)\W` angehängt.
Anschließend wird der so generierte Regex auf den Buchtext angewandt, um alle passenden Textstellen zu finden. Hierbei wird die Großkleinschreibung der Wörter ignoriert. Alle gefundenen Textstellen werden abschließend ausgegeben, wobei mehrfach auftretende Textstellen nur einmal ausgegeben werden.

3 Beispiele

Es folgen die Programmausgaben für die Beispiele der BWINF-Website.

3.1 stoerung0

Generierter Regexausdruck: `(das)\W([^\s]+)\W(mir)\W([^\s]+)\W([^\s]+)\W([^\s]+)\W(vor)\W`
Passende Textstelle: Das kommt mir gar nicht richtig vor

3.2 stoerung1

Generierter Regexausdruck: `(ich)\W(muß)\W([^\s]+)\W(clara)\W([^\s]+)\W`
Passende Textstelle: Ich muß in Clara verwandelt
Passende Textstelle: Ich muß doch Clara sein

3.3 stoerung2

Generierter Regexausdruck: `(fressen)\W([^\s]+)\W(gern)\W([^\s]+)\W`
Passende Textstelle: Fressen Katzen gern Spatzen
Passende Textstelle: Fressen Spatzen gern Katzen

3.4 stoerung3

Generierter Regexausdruck: `(das)\W([^\s]+)\W(fing)\W([^\s]+)\W`
Passende Textstelle: Das Publikum fing an
Passende Textstelle: das Spiel fing an

3.5 stoerung4

Generierter Regexausdruck: `(ein)\W([^\s]+)\W([^\s]+)\W(tag)\W`
Passende Textstelle: ein sehr schöner Tag

3.6 stoerung5

Generierter Regexausdruck: `(wollen)\W([^\s]+)\W(so)\W([^\s]+)\W(sein)\W`
Passende Textstelle: Wollen Sie so gut sein

4 Quellcode

```python
# coding=utf-8
import re


def create_regex(string):
    out = ""

    word_prefix = "("
    word_suffix = ")\\W"
    wildcard = "([^\\s]+)\\W"

    for element in string.split(' '):
        if element == "_":
            out += wildcard
        else:
            out += word_prefix + element + word_suffix

    return out


def clean(string):
    everything_but_words_and_spaces = re.compile(r"[^\w\s]")
    redundant_spaces = re.compile(r"\s{2,}")
    string = re.sub(everything_but_words_and_spaces, '', string)
    string = string.replace('\n', ' ')
    string = string.replace('\t', '')
    string = string.replace('_', '')
    string = re.sub(redundant_spaces, ' ', string)
    return string


if __name__ == '__main__':
    nr = input("Bitte gib an, welche Beispielnummer bearbeitet werden soll: ")
    with open("resources/Alice_im_Wunderland.txt", encoding="utf-8") as file:
        data = file.read()
        data = clean(data)
        # print(data)
        with open(f'resources/stoerung{nr}.txt', encoding="utf-8") as riddle:
            riddle_data = riddle.read()
            regex = create_regex(riddle_data)
            print("Generierter Regexausdruck: " + regex)
            matches = set(re.findall(regex, data, flags=re.IGNORECASE))
            for match in matches:
                print(f'Passende Textstelle: {" ".join(match)}')
```