Aufgabe 1: Alice im Wunderland

Team-ID: 01093

Team-Name: Kimi and his Droogs

Bearbeiter/-innen dieser Aufgabe:
Kimi Müller

20. November 2022

Inhaltsverzeichnis

Anleitung: Trage oben in den Zeilen 8 bis 11 die Aufgabennummer, die Team-ID, den Team-Namen und alle Bearbeiter/-innen dieser Aufgabe mit Vor- und Nachnamen ein. Vergiss nicht, auch den Aufgabennamen anzupassen (statt „LaTeX-Dokument“)!

Dann kannst du dieses Dokument mit deiner LaTeX-Umgebung übersetzen.

Die Texte, die hier bereits stehen, geben ein paar Hinweise zur Einsendung. Du solltest sie aber in deiner Einsendung wieder entfernen!

1 Lösungsidee

Abstrahiert ist das Lückenwort eine Buchstabenfolge beliebiger Länge >= 1. Daher suchen wir einfach nach dem Kontext des Lückenwortes, die Lücke wird da mit dieser Buchstabenfolge ersetzt.

2 Umsetzung

Mithilfe des regulären Ausdrucks "w+"für das Lückenwort suchen wir nach Übereinstimmungen. "wïst eine Abkürzung für alle Wortbestandteile, so muss sich keine Sorge um Umlaute o.ä. gemacht werden. "+ßteht für "1 oder mehrere Instanzen des vorherigen Worts"

3 Beispiele

Die inkludierten Beispiele:

```
['das kommt mir gar nicht richtig vor']
['ich mu doch clara sein', 'ich mu in clara verwandelt']
['fressen spatzen gern katzen', 'fressen katzen gern spatzen']
['das spiel fing an', 'das publikum fing an']
['ein sehr schöner tag']
['wollen sie so gut sein']
```

Zeile 3 sticht als besonders hervor, wo zum ersten Mal unerwartetes Verhalten auftrat. Wird das Buch unabhängig von Zeilenabsätzen durchsucht, kommt 'fressen katzen gern spatzen' doppelt vor. Dem wird vorgebeugt, indem identische Treffer gelöscht.

```
['alice in späterer zeit']
['aufschrift eingemachte']
```

Beispiel (1) ist relevant, da im Buch ein Teil des gesuchten innerhalb einer neuen Zeile ist. Daraus ergibt sich sowohl das Problem, dass nicht nach Leerzeichen gematcht werden muss, sondern alle "leeren Zeichen"(neben Leerzeichen auch tabs, neue Zeile/carriage return etc.) gematcht werden können, als auch dass die neue Zeile im Ergebnis übernommen werden würde. Dies wurde gelöst, indem alle "leeren Zeichenßu Leerzeichen konvertiert wurden.

Beispiel (2) wurde erstellt, da es Sonderzeichen enthielt.

4 Quellcode

Der Vollständigkeit halber möchte ich hier den Proof-of-concept in bash zeigen, der mit den genannten Sonderfällen nicht funktioniert:

```
gaptext=`cat $1`;
cat $2 | grep -iEo "${gaptext//_/"\w+"}"
```

Zur eigentlichen Lösung in Python:

```
def convert_gap_text_to_regex(gap_text):
    """
Converts a gap text to a precompiled regular expression
    """
    re_gap_text = gap_text.strip()
    re_gap_text = re_gap_text.replace(" ","\ ")
    re_gap_text = re_gap_text.replace("_", "\w+")
    return re.compile(re_gap_text)

def tidy_up_source_book(source_book):
    """
Eliminates special Characters and converts any whitespace to a space char
    """
    clean_source = re.sub("[,.:!?;źń_]*", "", source_book)
    clean_source = re.sub("\s+", " ", clean_source)
    clean_source = clean_source.lower()
    return clean_source


def find_all_results(interference, favourite_book):
    """
This function mends the other two into a function that outputs all matches.
    """
    regex = convert_gap_text_to_regex(interference)
    book = tidy_up_source_book(favourite_book)
    return list(set(regex.findall( book)))
```