

Raport z laboratorium 4

Filip Nikolow

23 kwietnia 2021

1 Cel laboratorium

Celem laboratorium było zapoznanie się z algorytmem obliczania odległości edycyjnej wraz z odwzorzeniem sekwencji edycji przekształcających jeden z ciągów w drugi oraz z algorytmem znajdowania najdłuższego wspólnego podciągu.

2 Realizacja poszczególnych poleceń

Kod do każdego z poleceń załączam na końcu sprawozdania.

2.1 Polecenie 1 - odległość edycyjna

Zaimplementowałem algorytm obliczający odległość edycyjną dwóch łańcuchów. Wagi wszystkich operacji (usunięcia, dodania, zamiany) przyjąłem równe 1.

2.2 Polecenia 2,3 - wizualizacja

Poniżej załączam wynik wizualizacji działania algorytmu na załączonych parach łańcuchów:

- los - kloc
- Łódź - Lodz
- kwintesencja - quintessence
- ATGAATCTTACCGCCTCG - ATGAGGCTCTGGCCCCTG

Po liście operacji jest wypisywana także ich ilość.

```
los
(+k)los
klo(s->c)
kloc
2
Łódź
(Ł->L)ódź
L(ó->o)dź
Lod(ż->z)
Lodz
3
kwintesencja
(k->q)wintesencja
```

```

q(w->u)intesencja
quinte(+s)sencja
quintessenc(j->e)a
quintessence(-a)
quintessence
5
ATGAATCTTACCGCCTCG
ATGA(A->G)TCTTACCGCCTCG
ATGAG(T->G)CTTACCGCCTCG
ATGAGGCT(+C)TACCGCCTCG
ATGAGGCTCT(A->G)CCGCCTCG
ATGAGGCTCTG(+G)CCGCCTCG
ATGAGGCTCTGGCC(-G)CCTCG
ATGAGGCTCTGGCCCCT(-C)G
ATGAGGCTCTGGCCCCTG
7
ab
(a->c)b
c(b->d)
cd
2

```

2.3 Polecenie 4 - LCS

Zaimplementowałem algorytm obliczający LCS dwóch ciągów. Poniżej załączam rezultat działania algorytmu dla tych samych par co w sekcji wyżej.

```

2 lo
1 d
8 intesenc
13 ATGATCTCCCCTG

```

2.4 Polecenia 5,6 - tokenizacja

Za pomocą biblioteki spaCy dokonałem podziału tekstu na tokeny, następnie utworzyłem dwie kopie składające się z 97% oryginalnie występujących tokenów.

2.5 Polecenie 7 - długość LCS podziałów po usunięciu

Bez zapisywania, wczytywania i retokenizowania: len1: 2619, len2: 2619, LCS: 2544
 Z zapisaniem, wczytaniem i retokenizacją: len1: 2603, len2: 2602, LCS: 2481

2.6 Polecenie 8,9 - narzędzie na wzór diff'a

Poniżej załączam wynik działania tego narzędzia dla tekstów utworzonych w poleceniach 5,6. Konkretniej, załączam dwa podziały - pierwszy powstał poprzez tokenizację, usunięcie 3% tokenów i następnie działanie programu, natomiast drugi dodatkowo dokonał zapisu tekstu do pliku a następnie jego wczytania (i tym samym reinterpretacji tokenów przez spaCy):

```

=====Diff without retokenizing=====

```

len1: 2619, len2: 2619, lcs: 2544

```
> [6] '978'
> [10] '\n '
< [11] 'rodu, '
> [11] 'w '
> [13] ', '
< [18] '| '
> [18] 'Kapulet'
> [29] 'PANI '
> [30] 'małżonka '
> [33] ', '
> [38] 'Rzecz '
> [46] 'Dwa jednakó '
< [47] '\n'
> [47] 'Tam'
> [50] 'dwo bowiem '
> [61] 'usuniem '
< [90] 'musiał '
> [90] '\n\n'
< [93] 'SAMSON'
< [135] 'Pobiwszy , : \n\n\n'
> [137] 'o '
> [144] 'kolei'
> [149] 'że '
< [150] 'dwóch '
< [167] 'Nie '
< [172] 'bać '
> [176] 'Miejmy zacząć'
> [196] 'Nie '
> [206] 'Grzegorza '
> [208] 'sobą'
> [213] '\n\n\n'
> [218] 'się '
< [236] '\n\n'
< [238] 'lepszy'
< [245] '/'
< [252] ': jeden '
> [252] 'mego '
< [257] '.'
< [272] 'schowajcie , '
> [287] '\n'
< [297] '/'
> [303] '! '
< [305] 'Wchodzą Kapulet '
> [312] 'miecz'
< [316] 'z '
> [317] 'ci '
< [321] 'mówię'
> [323] '.'
> [328] 'MONTEKI'
< [336] 'PANI \n\n'
> [337] 'MONTEKI'
```

< [338] '\n\n'
 < [345] 'niesforni '
 > [348] 'tę '
 > [349] 'tego'
 < [350] 'natychmiast'
 < [354] 'starcia'
 < [363] 'kiedys '
 > [366] 'wola będzie'
 < [368] 'dokładnie '
 > [368] 'aby '
 < [369] 'oznajmiona '
 < [378] 'zwadę'
 > [379] 'BENWOLIO'
 > [382] ', '
 > [386] '\n'
 < [391] 'Cięcia zbiegł '
 > [391] 'rozdzielił '
 < [392] ', '
 < [393] 'książę '
 > [396] 'Lecz '
 < [404] 'pierwej'
 > [407] 'już '
 < [409] '\n'
 < [412] 'się '
 > [413] 'Nie mu '
 < [415] '))'
 > [422] 'oblicza '
 < [426] 'na '
 > [431] 'to '
 < [439] 'znasz'
 > [457] 'słońcem'
 > [458] ', '
 > [466] '. '
 < [476] 'nas.'
 > [485] '{ '
 < [499] 'chwile{'
 < [500] 'spiesznie w '
 < [510] 'co '
 < [515] 'Miłość '
 > [522] 'brak '
 > [525] '\n\n'
 < [528] 'tamwzajemności'
 > [533] 'tak '
 < [541] 'dziś '
 > [541] '. '
 < [542] '? '
 < [544] '!'
 > [548] 'ruchu!'
 < [552] 'śmiejesz'
 > [559] '\n\n'
 > [566] 'uciskiem'
 < [576] ', '
 < [579] '\n'

```

> [581] ', '
< [594] 'ROMEO'
< [596] 'nie '
> [597] 'nie co '
> [602] 'to '
< [607] '\n'
> [613] 'dać '
< [620] 'Każ '
< [629] 'mi '
< [632] '\n\n'
< [634] 'którą '
> [640] '\n\n'
> [643] 'ona '
< [649] 'zjedna'
> [649] '\n'
< [656] '{z '
> [661] 'Tak , '
> [662] 'srogość '
> [663] 'potomność .'
< [665] 'piękna'
< [667] 'Temu '
> [681] 'BENWOLIO'
< [683] 'oczom '

```

=====Diff with retokenizing=====

```

len1: 2603, len2: 2602, lcs: 2481
< [6] '-83'
> [6] '97883-'
< [10] ':* '
> [10] ':\n '
< [11] 'rodu, '
> [11] '* w '
< [12] 'MONTEKIKAPULET '
> [13] 'MONTEKI, KAPULET '
< [18] '| '
> [18] 'Kapulet'
> [29] 'PANI '
> [30] 'małżonka '
< [32] 'domówmaski'
> [33] 'domówmaski, '
> [38] 'Rzecz '
> [46] 'Dwa jednako '
< [47] '\n'
> [47] 'Tam'
> [50] 'dwu bowiem '
> [61] 'usuniem '
< [90] 'GRZEGORZKto musiał \n\n\n'
> [90] 'GRZEGORZ\n\n'
> [92] 'Kto \n\n\n\n\n'
< [93] 'SAMSON\n\n'

```

< [135] 'Pobiwszy ludzi, wyrę kobietach: rzeź \n\n'n'
 > [137] 'o ludziwyrę kobietachrzeż '
 > [144] 'kolei'
 > [149] 'że '
 < [150] 'dwóch '
 < [167] 'Nie '
 < [172] 'bać '
 > [176] 'Miejmy zacząć'
 > [196] 'Nie '
 > [206] 'Grzegorza '
 > [208] 'sobą'
 > [213] '\n\n\n'
 > [218] 'się '
 < [236] 'ABRAHAM\n\n'
 < [238] 'Nie lepszy'
 > [238] 'ABRAHAMNie '
 < [245] '/'
 < [252] 'Powiedz: lepszycien '
 > [252] 'Powiedzlepszyciego '
 < [257] '.'
 < [272] 'schowajcie wiecie, co '
 > [272] 'wiecieco '
 < [287] 'nazadAlbo '
 > [287] 'nazad\n'
 > [288] 'Albo '
 < [297] '/'
 < [302] 'pałekDalej '
 > [303] 'pałek! Dalej '
 < [305] 'Wchodzą Kapulet '
 > [312] 'miecz'
 < [316] 'z '
 > [317] 'ci '
 < [321] 'mówię'
 > [323] '.'
 < [324] '\n\n\n\n\n'
 > [325] '\n\n\n\n'
 > [328] 'MONTEKI\n\n'
 < [336] 'PANI \n\n'
 > [337] 'MONTEKINie tobą./ '
 < [338] 'Nie tobą.\n\n'
 < [340] '/'
 < [345] 'niesforni '
 > [348] 'tę '
 > [349] 'tego'
 < [350] 'natychmiast'
 < [354] 'starcia'
 < [363] 'kiedyś '
 > [366] 'wola będzie'
 < [368] 'dokładnie '
 > [368] 'aby '
 < [369] 'oznajmiona '
 > [376] '\n\n\n\n'
 < [378] 'zwadę'

< [379] '\n\n\n\n\n'
 > [379] 'BENWOLIO\n\n'
 > [382] 'bili, kiedym '
 < [385] 'bilikiedym '
 > [386] '\n'
 < [391] 'Cięcica zbiegł '
 > [391] 'rozdzielił '
 < [392] ', '
 < [393] 'książę '
 > [396] 'Lecz '
 < [404] 'pierwej'
 > [407] 'już '
 < [409] '\n'
 < [412] 'się '
 > [413] 'Nie mu '
 < [415] ') '
 > [422] 'oblicza '
 < [426] 'na '
 > [431] 'to '
 < [439] 'znasz'
 > [457] 'słońcem'
 > [458] ', '
 > [466] 'nadchodzi. Odstąpcie '
 < [469] 'nadchodziOdstąpcie '
 < [476] 'nas.'
 > [485] 'Jeszcze{ż '
 < [488] 'Jeszczeż '
 > [496] 'Moiż '
 < [499] 'chwileMoi{ż '
 < [500] 'spiesznie w '
 < [510] 'co '
 < [515] 'Miłość '
 > [522] 'brak '
 > [525] 'ROMEO\n\n'
 > [527] 'Brak '
 < [528] 'ROMEOBrak tamwzajemności'
 > [533] 'tak '
 < [541] 'dziś '
 > [541] 'spórNie .'
 < [542] 'spór? Nie '
 < [544] '!' '
 > [548] 'ruchu!'
 < [552] 'śmiejesz'
 > [559] 'ROMEO\n\n'
 < [560] 'ROMEONad '
 > [561] 'Nad '
 > [566] 'uciskiem'
 < [576] 'ulgą, ale '
 > [577] 'ulgąale '
 < [579] 'płonie;\n'
 < [580] 'Morze '
 > [580] 'płonie;Morze '
 > [581] ', '

```

< [591] '\n\n\n'
> [591] '\n\n\n\n\n'
< [594] 'ROMEO\n\n'
< [596] 'nie '
> [597] 'nie co '
> [602] 'to '
< [607] '\n'
> [613] 'dać '
< [620] 'Każ '
< [629] 'mi '
> [631] 'ROMEOBiegle '
< [632] 'ROMEO\n\n'
< [634] 'Biegle którą '
> [640] 'ROMEO\n\n'
> [642] 'A '
< [643] 'ROMEOA '
> [643] 'ona '
< [649] 'zjedna'
> [649] '\n'
< [656] 'Wiecznie{ż z '
> [656] 'Wiecznież '
> [661] 'Tak ,'
> [662] 'srogość '
> [663] 'potomność .'
< [665] 'piękna'
< [667] 'Temu '
< [678] '\n\n\n\n\n'
> [678] '\n\n\n'
> [681] 'BENWOLIO\n\n'
< [683] 'oczom '

```

3 Kody modułów

3.1 Odległość edycyjna

```

1  from enum import Enum
2
3
4  class Step(Enum):
5      NO_CHANGE = 0
6      DEL = 1
7      INS = 2
8      REPL = 3
9
10
11 def levenshtein(s1, s2):
12     n = len(s1)
13     m = len(s2)
14     distance = [[0 for i in range(m + 1)] for j in range(n + 1)]
15     path = [[0 for i in range(m + 1)] for j in range(n + 1)]
16     for i in range(n + 1):

```



```

17     distance[i][0] = i
18     path[i][0] = Step.DEL
19 for i in range(m + 1):
20     distance[0][i] = i
21     path[0][i] = Step.INS
22 for i in range(1, n + 1):
23     for j in range(1, m + 1):
24         if s1[i - 1] == s2[j - 1]:
25             distance[i][j] = distance[i - 1][j - 1]
26             path[i][j] = Step.NO_CHANGE
27         else:
28             distance[i][j] = distance[i - 1][j]
29             path[i][j] = Step.DEL
30             if distance[i][j] > distance[i][j - 1]:
31                 distance[i][j] = distance[i][j - 1]
32                 path[i][j] = Step.INS
33             if distance[i][j] > distance[i - 1][j - 1]:
34                 distance[i][j] = distance[i - 1][j - 1]
35                 path[i][j] = Step.REPL
36             distance[i][j] += 1
37 return distance[n][m], path
38
39
40 def visualize(s1, s2, path):
41     i = len(s1)
42     j = len(s2)
43     steps = []
44     while i != 0 or j != 0:
45         c = path[i][j]
46         steps.append(c)
47         if c == Step.DEL:
48             i -= 1
49         elif c == Step.INS:
50             j -= 1
51         else:
52             i -= 1
53             j -= 1
54     steps.reverse()
55     text = s1
56     print(text)
57     i, j = 0, 0
58     for c in steps:
59         if c == Step.INS:
60             print(text[:i] + '(' + s2[j] + ')' + text[i:])
61             text = text[:i] + s2[j] + text[i:]
62             i += 1
63             j += 1
64         elif c == Step.DEL:
65             print(text[:i] + '-' + text[i] + ')' + text[i + 1:])
66             text = text[:i] + text[i + 1:]
67         elif c == Step.NO_CHANGE:
68             i += 1
69             j += 1

```

```

70         else:
71             print(text[:i] + '(' + text[i] + '->' + s2[j] + ')') + text[i + 1:])
72             text = text[:i] + s2[j] + text[i + 1:]
73             i += 1
74             j += 1
75         print(text)
76
77
78 if __name__ == '__main__':
79     t1, t2 = "los", "kloc"
80     dist, path = levenshtein(t1, t2)
81     visualize(t1, t2, path)
82     print(dist)
83     t1, t2 = "Łódź", "Lodz"
84     dist, path = levenshtein(t1, t2)
85     visualize(t1, t2, path)
86     print(dist)
87     t1, t2 = "kwintesencja", "quintessence"
88     dist, path = levenshtein(t1, t2)
89     visualize(t1, t2, path)
90     print(dist)
91     t1, t2 = "ATGAATCTTACCGCCTCG", "ATGAGGCTCTGGCCCCTG"
92     dist, path = levenshtein(t1, t2)
93     visualize(t1, t2, path)
94     print(dist)
95
96     t1, t2 = "ab", "cd"
97     dist, path = levenshtein(t1, t2)
98     visualize(t1, t2, path)
99     print(dist)

```

3.2 Najdłuższy wspólny podciąg

```

1  from enum import Enum
2
3
4  class Step(Enum):
5      MATCH = 0
6      LEFT = 1
7      UP = 2
8
9
10 def lcs(s1, s2):
11     n = len(s1)
12     m = len(s2)
13     length = [[0 for i in range(m + 1)] for j in range(n + 1)]
14     path = [[0 for i in range(m + 1)] for j in range(n + 1)]
15     for i in range(n + 1):
16         length[i][0] = 0
17         path[i][0] = Step.UP
18     for i in range(m + 1):

```

```

19         length[0][i] = 0
20         path[0][i] = Step.LEFT
21     for i in range(1, n + 1):
22         for j in range(1, m + 1):
23             if s1[i - 1] == s2[j - 1]:
24                 length[i][j] = length[i - 1][j - 1] + 1
25                 path[i][j] = Step.MATCH
26             else:
27                 length[i][j] = length[i - 1][j]
28                 path[i][j] = Step.UP
29                 if length[i][j] < length[i][j - 1]:
30                     length[i][j] = length[i][j - 1]
31                     path[i][j] = Step.LEFT
32     return length[n][m], visualize(s1, s2, path)
33
34
35 def visualize(s1, s2, path):
36     i = len(s1)
37     j = len(s2)
38     subseq = []
39     while i != 0 or j != 0:
40         c = path[i][j]
41         if c == Step.UP:
42             i -= 1
43         elif c == Step.LEFT:
44             j -= 1
45         else:
46             i -= 1
47             j -= 1
48         subseq.append(s1[i])
49     subseq.reverse()
50     return subseq
51
52
53 if __name__ == '__main__':
54     t1, t2 = "los", "kloc"
55     count, ss = lcs(t1, t2)
56     print(count, ''.join(ss))
57     t1, t2 = "Łódź", "Lodz"
58     count, ss = lcs(t1, t2)
59     print(count, ''.join(ss))
60     t1, t2 = "kwintesencja", "quintessence"
61     count, ss = lcs(t1, t2)
62     print(count, ''.join(ss))
63     t1, t2 = "ATGAATCTTACCGCCTCG", "ATGAGGCTCTGGCCCTG"
64     count, ss = lcs(t1, t2)
65     print(count, ''.join(ss))

```

3.3 Diff

```
1 import random
2 from spacy.lang.pl import Polish
3 from lcs import lcs
4 from collections import defaultdict
5
6
7 def tokenize(text):
8     nlp = Polish()
9     return list(nlp.tokenizer(text))
10
11
12 def remove_random_elements_from_list(lis, p=0.03):
13     to_remove = set(random.sample(lis, int(len(lis) * p)))
14     return [x for x in lis if x not in to_remove]
15
16
17 def create_altered_file(origin_name, dest_name):
18     with open(origin_name, "r") as f:
19         text = f.read()
20     with open(dest_name, "w") as f:
21         tokens = remove_random_elements_from_list(tokenize(text))
22         for token in tokens:
23             f.write(token.text_with_ws)
24     return tokens
25
26
27 def diff_files(filename1, filename2):
28     with open(filename1, "r") as f:
29         text1 = f.read()
30     with open(filename2, "r") as f:
31         text2 = f.read()
32     tokens1 = [t.text_with_ws for t in tokenize(text1)]
33     tokens2 = [t.text_with_ws for t in tokenize(text2)]
34     diff(tokens1, tokens2)
35
36
37 def diff(tokens1, tokens2):
38
39     def diff_from_common(tokens, common, rep):
40         t, c = 0, 0
41         line = 1
42         diffs = defaultdict(list)
43         while c < len(common):
44             if tokens[t] != common[c]:
45                 diffs[line].append(tokens[t])
46             else:
47                 c += 1
48             line += tokens[t].count('\n')
49             t += 1
50         while t < len(tokens):
```

```

51         diffs[line].append(tokens[t])
52         line += tokens[t].count('\n')
53         t += 1
54     result = []
55     for key, val in diffs.items():
56         result.append((key, f"{rep} [{key}] " + repr(''.join(val))))
57     return result
58
59     common = lcs(tokens1, tokens2)[1]
60     print(f"len1: {len(tokens1)}, len2: {len(tokens2)}, lcs: {len(common)}")
61     diff1 = diff_from_common(tokens1, common, '<')
62     diff2 = diff_from_common(tokens2, common, '>')
63     difflist = sorted(diff1 + diff2)
64     for line in difflist:
65         print(line[1])
66
67
68 if __name__ == '__main__':
69     origin_name = "../sources/romeo-i-julia-700.txt"
70     f1 = "../sources/tokenized1.txt"
71     f2 = "../sources/tokenized2.txt"
72     tokens1 = create_altered_file(origin_name, f1)
73     tokens2 = create_altered_file(origin_name, f2)
74     print("=====Diff without retokenizing=====\\n\\n")
75     diff([t.text_with_ws for t in tokens1], [t.text_with_ws for t in tokens2])
76     print('\\n\\n=====Diff with retokenizing=====\\n\\n')
77     diff_files(f1, f2)

```
