Raport z laboratorium 4

Filip Nikolow

9 maja 2021

1 Cel laboratorium

Celem laboratorium było zapoznanie się z algorytmem obliczania odległości edycyjnej wraz z odwtorzeniem sekwencji edycji przekształcających jeden z ciągów w drugi oraz z algorytmem znajdywania 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(ź->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

Zaimlementowałem 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 - narzedzie 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 jednako '
< [47] '\n'
> [47] 'Tam'
> [50] 'dwu 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 zaczną'
> [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] 'kiedyś '
> [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, wywrę kobietach: rzeź \n\n\n'
> [137] 'o ludziwywrę kobietachrzeź '
> [144] 'kolei'
> [149] 'że '
< [150] 'dwóch '
< [167] 'Nie '
< [172] 'bać '
> [176] 'Miejmy zaczną'
> [196] 'Nie '
> [206] 'Grzegorza '
> [208] 'sobą'
> [213] '\n\n\n'
> [218] 'się '
< [236] 'ABRAHAM\n\n'
< [238] 'Nie lepszy'
> [238] 'ABRAHAMNie '
< [245] '/'
< [252] 'Powiedz: lepszyjeden '
> [252] 'Powiedzlepszymego '
< [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'
> [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'
< [378] 'zwadę'
```

```
< [379] '\n\n\n\n\n'
> [379] 'BENWOLIO\n\n'
> [382] 'bili, kiedym '
< [385] 'bilikiedym '
> [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] '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):
```

```
        distance[i][0] = i
        path[i][0] = Step.DEL
    for i in range(m + 1):
        distance[0][i] = i
        path[0][i] = Step.INS
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if s1[i - 1] == s2[j - 1]:
                distance[i][j] = distance[i - 1][j - 1]
                path[i][j] = Step.NO_CHANGE
            else:
                distance[i][j] = distance[i - 1][j]
                path[i][j] = Step.DEL
                if distance[i][j] > distance[i][j - 1]:
                    distance[i][j] = distance[i][j - 1]
                    path[i][j] = Step.INS
                if distance[i][j] > distance[i - 1][j - 1]:
                    distance[i][j] = distance[i - 1][j - 1]
                    path[i][j] = Step.REPL
                distance[i][j] += 1
    return distance[n][m], path


def visualize(s1, s2, path):
    i = len(s1)
    j = len(s2)
    steps = []
    while i != 0 or j != 0:
        c = path[i][j]
        steps.append(c)
        if c == Step.DEL:
            i -= 1
        elif c == Step.INS:
            j -= 1
        else:
            i -= 1
            j -= 1
    steps.reverse()
    text = s1
    print(text)
    i, j = 0, 0
    for c in steps:
        if c == Step.INS:
            print(text[:i] + '(+' + s2[j] + ')' + text[i:])
            text = text[:i] + s2[j] + text[i:]
            i += 1
            j += 1
        elif c == Step.DEL:
            print(text[:i] + '(-' + text[i] + ')' + text[i + 1:])
            text = text[:i] + text[i + 1:]
        elif c == Step.NO_CHANGE:
            i += 1
            j += 1
```

```
        else:
            print(text[:i] + '(' + text[i] + '->' + s2[j] + ')' + text[i + 1:])
            text = text[:i] + s2[j] + text[i + 1:]
            i += 1
            j += 1
    print(text)


if __name__ == '__main__':
    t1, t2 = "los", "kloc"
    dist, path = levenshtein(t1, t2)
    visualize(t1, t2, path)
    print(dist)
    t1, t2 = "Łódź", "Lodz"
    dist, path = levenshtein(t1, t2)
    visualize(t1, t2, path)
    print(dist)
    t1, t2 = "kwintesencja", "quintessence"
    dist, path = levenshtein(t1, t2)
    visualize(t1, t2, path)
    print(dist)
    t1, t2 = "ATGAATCTTACCGCCTCG", "ATGAGGCTCTGGCCCCTG"
    dist, path = levenshtein(t1, t2)
    visualize(t1, t2, path)
    print(dist)

    t1, t2 = "ab", "cd"
    dist, path = levenshtein(t1, t2)
    visualize(t1, t2, path)
    print(dist)
```

3.2 Najdłuższy wspólny podciąg

```
from enum import Enum


class Step(Enum):
    MATCH = 0
    LEFT = 1
    UP = 2


def lcs(s1, s2):
    n = len(s1)
    m = len(s2)
    length = [[0 for i in range(m + 1)] for j in range(n + 1)]
    path = [[0 for i in range(m + 1)] for j in range(n + 1)]
    for i in range(n + 1):
        length[i][0] = 0
        path[i][0] = Step.UP
    for i in range(m + 1):
```

```
        length[0][i] = 0
        path[0][i] = Step.LEFT
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if s1[i - 1] == s2[j - 1]:
                length[i][j] = length[i - 1][j - 1] + 1
                path[i][j] = Step.MATCH
            else:
                length[i][j] = length[i - 1][j]
                path[i][j] = Step.UP
                if length[i][j] < length[i][j - 1]:
                    length[i][j] = length[i][j - 1]
                    path[i][j] = Step.LEFT
    return length[n][m], visualize(s1, s2, path)


def visualize(s1, s2, path):
    i = len(s1)
    j = len(s2)
    subseq = []
    while i != 0 or j != 0:
        c = path[i][j]
        if c == Step.UP:
            i -= 1
        elif c == Step.LEFT:
            j -= 1
        else:
            i -= 1
            j -= 1
            subseq.append(s1[i])
    subseq.reverse()
    return subseq


if __name__ == '__main__':
    t1, t2 = "los", "kloc"
    count, ss = lcs(t1, t2)
    print(count, ''.join(ss))
    t1, t2 = "Łódź", "Lodz"
    count, ss = lcs(t1, t2)
    print(count, ''.join(ss))
    t1, t2 = "kwintesencja", "quintessence"
    count, ss = lcs(t1, t2)
    print(count, ''.join(ss))
    t1, t2 = "ATGAATCTTACCGCCTCG", "ATGAGGCTCTGGCCCCTG"
    count, ss = lcs(t1, t2)
    print(count, ''.join(ss))
```

3.3 Diff

```
import random
from spacy.lang.pl import Polish
from lcs import lcs
from collections import defaultdict


def tokenize(text):
    nlp = Polish()
    return list(nlp.tokenizer(text))


def remove_random_elements_from_list(lis, p=0.03):
    to_remove = set(random.sample(lis, int(len(lis) * p)))
    return [x for x in lis if x not in to_remove]


def create_altered_file(origin_name, dest_name):
    with open(origin_name, "r") as f:
        text = f.read()
    with open(dest_name, "w") as f:
        tokens = remove_random_elements_from_list(tokenize(text))
        for token in tokens:
            f.write(token.text_with_ws)
    return tokens


def diff_files(filename1, filename2):
    with open(filename1, "r") as f:
        text1 = f.read()
    with open(filename2, "r") as f:
        text2 = f.read()
    tokens1 = [t.text_with_ws for t in tokenize(text1)]
    tokens2 = [t.text_with_ws for t in tokenize(text2)]
    diff(tokens1, tokens2)


def diff(tokens1, tokens2):

    def diff_from_common(tokens, common, rep):
        t, c = 0, 0
        line = 1
        diffs = defaultdict(list)
        while c < len(common):
            if tokens[t] != common[c]:
                diffs[line].append(tokens[t])
            else:
                c += 1
            line += tokens[t].count('\n')
            t += 1
        while t < len(tokens):
```

```
            diffs[line].append(tokens[t])
            line += tokens[t].count('\n')
            t += 1
        result = []
        for key, val in diffs.items():
            result.append((key, f"{rep} [{key}] " + repr(''.join(val))))
        return result

    common = lcs(tokens1, tokens2)[1]
    print(f"len1: {len(tokens1)}, len2: {len(tokens2)}, lcs: {len(common)}")
    diff1 = diff_from_common(tokens1, common, '<')
    diff2 = diff_from_common(tokens2, common, '>')
    difflist = sorted(diff1 + diff2)
    for line in difflist:
        print(line[1])


if __name__ == '__main__':
    origin_name = "../sources/romeo-i-julia-700.txt"
    f1 = "../sources/tokenized1.txt"
    f2 = "../sources/tokenized2.txt"
    tokens1 = create_altered_file(origin_name, f1)
    tokens2 = create_altered_file(origin_name, f2)
    print("===============Diff without retokenizing===============\n\n")
    diff([t.text_with_ws for t in tokens1], [t.text_with_ws for t in tokens2])
    print('\n\n===============Diff with retokenizing===============\n\n')
    diff_files(f1, f2)
```