[Day-01-Lecture-05]

관련 논문

> **권기성, 최운호, 김동건 (2022), "문학 작품의 거리 측정을 활용한 야담의 이본 연구", 「한국고전연구 57집」 87~120쪽, 한국고전연구학회.**

- 18900개의 비교 쌍을 대상으로 거리를 측정하겠습니다.
- progress bar가 필요하니 tqdm을 먼저 설치하겠습니다.

```
pip install tqdm
```

- Pairwise comparison을 수행하는 파이썬 프로그램입니다.

```
#! okss_004_004_dist.py
#
#!-*-coding: utf-8-*-
#

from difflib import ndiff
from tqdm import tqdm

def lv_align(str1, str2):
    result = ""
    pos, removed = 0, 0

    for x in ndiff(str1, str2):
        if pos < len(str1) and str1[pos] == x[2]:
            pos += 1
            result += x[2]
            if x[0] == "-":
                removed += 1
            continue
        else:
            if removed > 0:
                removed -= 1
            else:
                result += "-"
    #print(result)
    return result

def lv_align_max(str1, str2):
    res01 = lv_align(str1, str2)
    res02 = lv_align(str2, str1)

    len01 = len(res01)
    len02 = len(res02)

    if (len01 > len02):
        return res01
```

```
    return res02


def lv_dist(s, t):

    if len(s) == 0:
        return len(t)

    if len(t) == 0:
        return len(s)

    dist = iterative_levenshtein(s, t)

    return dist

def iterative_levenshtein(s, t):

    rows = len(s) + 1
    cols = len(t) + 1
    dist = [ [0 for x in range(cols)] for x in range(rows)]

    # source prefixed can be transformed into empty strings
    # by deletions:

    for i in range(1, rows):
        dist[i][0] = i

    # target prefixes can be created from an empty source string
    # by inserting the characters

    for i in range(1, cols):
        dist[0][i] = i

    for col in range(1, cols):
        for row in range(1, rows):
            if s[row-1] == t[col-1]:
                cost = 0
            else:
                cost = 1
            dist[row][col] = min(dist[row-1][col] + 1,     # deletion
                    dist[row][col-1] + 1,          # insertion
                    dist[row-1][col-1] + cost)   # substitution

    #for r in range(rows):
    #   print(dist[r])

    return dist[row][col]


if __name__ == "__main__":

    f_out = open("okss_004_004_dist.txt", "w", encoding="utf-8")

```

```
92
93      with open("okss_004_003.txt", "r", encoding="utf-8") as f_in:
94          m_lines = f_in.readlines()
95          for bline in tqdm(m_lines):
96              bline = bline.strip("\n")
97              cur_data = bline.split("\t")
98
99              str_src = cur_data[1]
100             str_target = cur_data[3]
101             bsrc = cur_data[0]
102             btarget = cur_data[2]
103             res_lv = lv_align_max(str_src, str_target)
104             lv_distval = lv_dist(str_src, str_target)
105
106
107             str_out = "\t".join([bsrc, btarget, str(len(res_lv)),
    str(lv_distval) ])
108             f_out.write(str_out)
109             f_out.write("\n")
110
111     f_out.close()
```

- 18900개의 데이터가 다음과 같이 생성되었습니다.

```
BCGY  CCHN 67  63
BCGY  CCNG 12  7
BCGY  CDNG 177 175
BCGY  CKEY  11  7
BCGY  CKIM  12  6
```

- 1번, 2번 컬럼은 약호입니다.
- 3번 컬럼은 대응되는 두 단락의 정렬된 길이(aligned length)입니다. Source를 Target으로 변환할 때 얼마나 많은 연산이 적용됐는지 거리를 표현합니다.
- 이제 다음 단계는 3번과 4번을 이용해서 거리를 [0-1] 사이의 값으로 변환하는 정규화 과정을 거치는 것입니다.

```
1   #!python
2   #
3   #!-*-coding=utf-8-*-
4   #
5   # okss_004_005.py
6
7   from tqdm import tqdm
8
9   if __name__ == "__main__":
10
11      with open("okss_004_004_dist.txt", "r", encoding="utf-8") as f_in:
12          m_lines = [l.strip() for l in f_in.readlines()]
13
14          with open("okss_004_005.txt", "w", encoding="utf-8") as f_out:
15
16              for line in tqdm(m_lines):
17                  cur_line = line.split('\t')
```

```
                d_n = int(cur_line[3])  # numerator(분자)
                d_d = int(cur_line[2])  # denominator(분모)
                d_val = 1 if d_n == 0 or d_d == 0 else (d_n / d_d)
                print(cur_line[0], cur_line[1], d_d, d_n, d_val,
                      file=f_out, sep="\t")

```

- 거리까지 측정한 결과는 다음과 같습니다.

```
BCGY CCHN 67 63 0.9402985074626866
BCGY CCNG 12 7 0.5833333333333334
BCGY CDNG 177 175 0.9887005649717514
BCGY CKEY 11 7 0.6363636363636364
BCGY CKIM 12 6 0.5
BCGY DCGY 9 0 1
BCGY DDNG 177 175 0.9887005649717514
BCGY DDON 13 8 0.6153846153846154
BCGY DKIM 12 6 0.5
BCGY EKIM 12 6 0.5
```

- 이 파일 안에는 각 작품 별로 30개의 대응 단락이 있습니다.

```
BCGY    CCHN    57  53  0.929825
BCGY    CCHN    57  50  0.877193
BCGY    CCHN    58  47  0.810345
BCGY    CCHN    200 189 0.945
BCGY    CCHN    265 261 0.984906
BCGY    CCHN    75  68  0.906667
BCGY    CCHN    119 110 0.92437
BCGY    CCHN    156 147 0.942308
BCGY    CCHN    169 124 0.733728
BCGY    CCHN    253 220 0.869565
BCGY    CCHN    145 105 0.724138
BCGY    CCHN    51  43  0.843137
BCGY    CCHN    197 197 1
BCGY    CCHN    448 448 1
BCGY    CCHN    137 84  0.613139
BCGY    CCHN    165 139 0.842424
BCGY    CCHN    127 106 0.834646
BCGY    CCHN    113 113 1
BCGY    CCHN    211 160 0.758294
BCGY    CCHN    89  82  0.921348
BCGY    CCHN    171 109 0.637427
BCGY    CCHN    111 64  0.576577
BCGY    CCHN    208 160 0.769231
BCGY    CCHN    42  32  0.761905
BCGY    CCHN    98  89  0.908163
BCGY    CCHN    14  14  1
BCGY    CCHN    122 103 0.844262
BCGY    CCHN    55  55  1
BCGY    CCHN    21  21  1
BCGY    CCHN    50  50  1
```

- 이 파일에서 데이터를 모아서 동일 작품의 값을 다 합해야 합니다. SQL이나 Pandas Dataframe을 사용하면 해결될 수 있지만, 자료가 만들어지는 과정을 이해하기 위해서 단계별로 진행해 보겠습니다.
- 각 작품마다 30개의 누적된 단락 차이를 계산하겠습니다.
- 다음 코드를 이용해서 36개 작품에서 비교한 630개 distance value를 생성하겠습니다.

```python
#!python
#
#!-*-coding=utf-8-*-
#
#

from tqdm import tqdm
from decimal import Decimal

if __name__ == "__main__":

    with open("okss_004_005.txt", "r", encoding="utf-8") as f_in:
        m_lines = [l.strip() for l in f_in.readlines()]

    print("Aggregating data values...")

    # Constructing values dictionary
    dic_agg_dist = {}
    for line in tqdm(m_lines):
        cur_elt = line.split('\t')
        cur_key = cur_elt[0] + "_" + cur_elt[1]
        if cur_key in dic_agg_dist.keys():
            cur_val = dic_agg_dist[cur_key]
            cur_val = cur_val + float(cur_elt[4])
            dic_agg_dist[cur_key] = cur_val
        else:
            cur_val = float(cur_elt[4])
            dic_agg_dist[cur_key] = cur_val


    print("Writing Distance values...")
    with open("okss_004_006.txt", "w", encoding="utf-8") as f_out:
        for k, v in tqdm(dic_agg_dist.items()):
            cur_elt = k.split('_')
            if Decimal(v) == Decimal(0):
                v = 0.00000001
            print(cur_elt[0], cur_elt[1], v, sep="\t", file=f_out)
```

```
BCGY    CCHN    25.803456497691943
BCGY    CCNG    23.361412488879317
BCGY    CDNG    25.436944790563967
BCGY    CKEY    23.102383004576094
BCGY    CKIM    23.43629957300532
BCGY    DCGY    24.253929347552216
BCGY    DDNG    25.43887824738651
BCGY    DDON    23.075590171137577
...
(630개의 데이터 짝이 구성되었습니다.)
```

- 이제 630개의 데이터를 테이블로 바꾸도록 하겠습니다.
- 아래 코드는 주어진 파일에서 데이터를 읽어서 대각 행렬로 바꾸어 줍니다.

```python
#!python
#
#!-*-coding=utf-8-*-
#
# okss_004_007.py
#

from tqdm import tqdm

if __name__ == "__main__":

    with open("okss_004_006.txt", "r", encoding="utf-8") as f_in:
        m_lines = [l.strip() for l in f_in.readlines()]

        dic_book_code = {}
        # 작품 ID 구하기
        print("Preprocessing book codes.")
        for line in tqdm(m_lines):

            cur_elts = line.split("\t")
            if cur_elts[0] in dic_book_code.keys():
                dic_book_code[cur_elts[0]] += 1
            else:
                dic_book_code[cur_elts[0]] = 1

            if cur_elts[1] in dic_book_code.keys():
                dic_book_code[cur_elts[1]] += 1
            else:
                dic_book_code[cur_elts[1]] = 1

        tbl_size = len(dic_book_code.keys())
        tbl_dist = [ [0] * tbl_size for _ in range(tbl_size)]
        list_keys = list(dic_book_code.keys())

        for idx_i in range(len(list_keys)):
            dic_book_code[list_keys[idx_i]] = idx_i

```

```
38
39          # Table Index 안에 값 채워 넣기.
40          print("\nFilling the distance values into the matrix.")
41          for line in tqdm(m_lines):
42              cur_elts = line.split("\t")
43              elt_01 = cur_elts[0]
44              elt_02 = cur_elts[1]
45              elt_dist = float(cur_elts[2])
46              idx_01 = dic_book_code[elt_01]
47              idx_02 = dic_book_code[elt_02]
48              tbl_dist[idx_01][idx_02] = elt_dist
49              tbl_dist[idx_02][idx_01] = elt_dist
50
51          # Table 출력하기
52          print("\nPrinting the table to the output file.")
53
54          with open("okss_004_007.txt", "w", encoding="utf-8") as f_out:
55              # Header 출력
56              str_header = '\t'.join(list_keys)
57              print("", str_header, sep="\t", file=f_out)
58              for idx_i in tqdm(range(len(tbl_dist))):
59                  list_line = list(map(str, tbl_dist[idx_i]))
60                  str_key = list_keys[idx_i]
61                  str_line = "\t".join(list_line)
62                  print(str_key, str_line, sep="\t", file=f_out)
```

- 출력된 결과를 엑셀 파일로 같이 읽어 보도록 하겠습니다.
- 엑셀 파일에서 특정 컬럼들을 선택해서 정렬(오름차순)한 뒤 그 차이를 차트로 그려보도록 하겠습니다.