

# bigdata\_assignment01

## [과제 1] 한글 문장의 유사도 계산 (1)

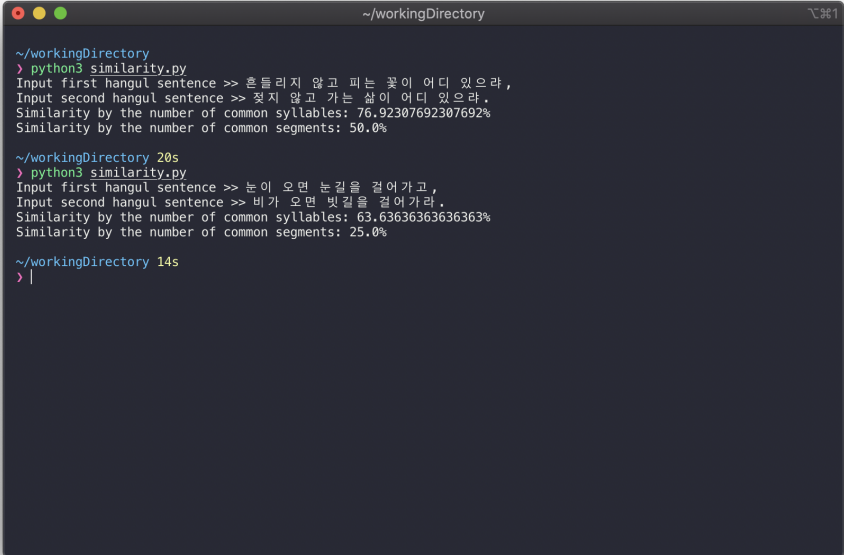
### 1. 한글 문장의 유사도 계산 (1)

- 입력: 한글 문장 2개 (매우 유사하거나 또는 약간 유사한 문장)
- 출력: 유사도 %
- 방법: 공통 음절 개수, 공통 어절 개수 등에 의한 유사도 계산

<참고 1> 위 과제 수행에 사용하는 언어는 C/C++, 자바, 파이썬 등 각자 사용하기 편한 언어를 사용하면 됩니다.

<참고 2> 과제 제출 내용: 소스코드, 보고서 (PDF 파일: 구현 방법 미 실행 화면 스샷 등의 설명 포함) → zip 파일 1개로 업로드

### 실행 결과



```
~/workingDirectory
> python3 similarity.py
Input first hangul sentence >> 흔들리지 않고 피는 꽃이 어디 있으랴,
Input second hangul sentence >> 져지 않고 가는 삶이 어디 있으랴.
Similarity by the number of common syllables: 76.92307692307692%
Similarity by the number of common segments: 50.0%

~/workingDirectory 20s
> python3 similarity.py
Input first hangul sentence >> 눈이 오면 눈길을 걸어가고,
Input second hangul sentence >> 비가 오면 빗길을 걸어가라.
Similarity by the number of common syllables: 63.63636363636363%
Similarity by the number of common segments: 25.0%

~/workingDirectory 14s
> |
```

## RUN

```
$ make
```

## 구현 방법

- 공통 음절 개수에 의한 유사도 측정

아래와 같은 순서로 공통 음절 유사도 측정 알고리즘을 구현하였다.

1. 입력받은 문장의 문장부호와 공백을 제거한다.
2. collections.Counter를 이용하여 각 text에 대한 음절 빈도를 측정한다.
3. 음절 갯수가 짧은 문장을 기준으로 공통 음절 갯수를 측정한다.
4. 짧은 문장의 음절 갯수로 공통 음절 갯수를 나누고 100을 곱하여 공통 음절 유사도를 계산하여 반환한다.

- 1을 구현한 clean\_text

```
def clean_text(text: str) -> str:  
    return re.sub('[.,!?~·:/\\"\' ]', '', text.strip())
```

- 2를 구현한 코드

```
from collections import Counter  
  
syllable_count1 = dict(Counter(text1))  
  
syllable_count2 = dict(Counter(text2))
```

- 3을 구현한 코드

```
for syllable in syllable_count1:  
    if syllable in syllable_count2:  
        commonness += min(syllable_count1[syllable], syllable_count2[syllable])
```

- 4를 구현한 코드

```
return 100 * commonness / len(text1)
```

-

- 공통 어절 개수에 의한 유사도 측정

아래와 같은 순서로 공통 어절 유사도 측정 알고리즘을 구현하였다.

1. 입력받은 문장의 문장부호와 공백을 제거한다.

2. `collections.Counter`를 이용하여 각 `text`에 대한 어절 빈도를 측정한다.

3. 어절 갯수가 짧은 문장을 기준으로 공통 어절 갯수를 측정한다.

4. 짧은 문장의 어절 갯수로 공통 어절 갯수를 나누고 100을 곱하여 공통 어절 유사도를 계산하여 반환한다.

- 1을 구현한 `clean_text`

```
def clean_text(text: str) -> str:
    return re.sub('[.,!?~·:/\\"\' ]', '', text.strip())
```

- 2를 구현한 코드

```
from collections import Counter

segment_count1 = dict(Counter(segments1))
segment_count2 = dict(Counter(segments2))
```

- 3을 구현한 코드

```
for segment in segment_count1:
    if segment in segment_count2:
        commonness += min(segment_count1[segment], segment_count2[segment])
```

- 4를 구현한 코드

```
return 100 * commonness / len(segments1)
```