

# 빅데이터 최신기술

20152900 홍성표

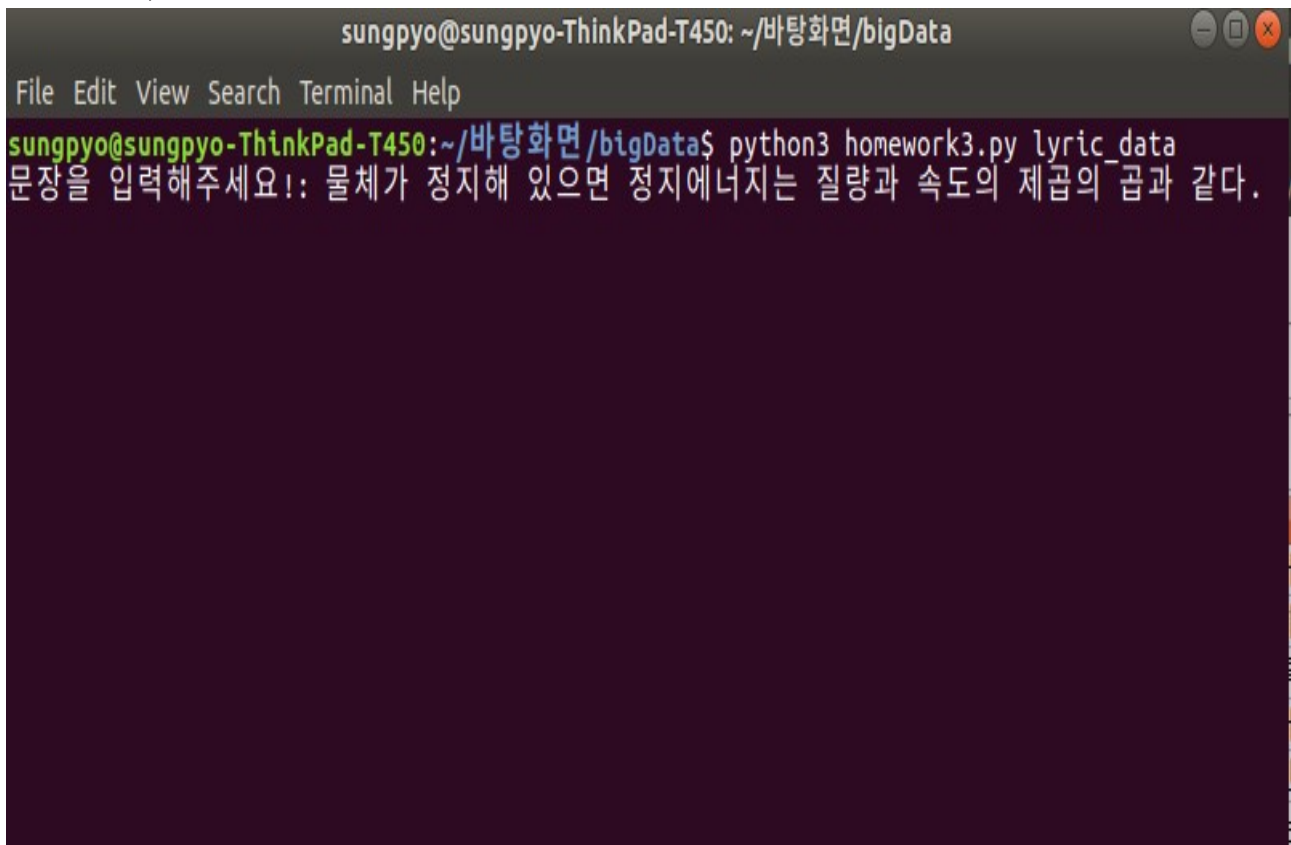
먼저 보고서를 쓰기전 대규모 말뭉치 파일을 다운 받았으나 100만개의 말뭉치가 작동하기에는 컴퓨터 성능이 안좋아 자주 컴퓨터가 멈추는 현상이 발견되어 10분의 1 규모인 10만개의 데이터 말뭉치 규모를 가지고 테스트를 하였습니다.

<과제 목적> 대규모 말뭉치(빅 텍스트 데이터) 분석의 문제점, 해결방안 실습

- 입력: 한글 문장 1개
- 출력: 입력문장과 가장 유사한 n개의 문장 추출 및 유사도, 소요시간 출력
- 방법: 위 2번의 형태소분석(또는 WPM) 방식의 유사도 계산 모듈을 대규모 말뭉치에 적용, 말뭉치의 각 문장들과 순서대로 비교하여 가장 유사도가 높은 상위 n개 문장 및 유사도, 소요시간을 출력. n값은 실행할 때 command line 인자(또는 사용자 입력)으로 받음.

실행장면 입니다.

1. 사용자에게 문장을 입력 받습니다.(물체가 정지해 있으면 정지에너지는 질량과 속도의 제곱의 곱과 같다.)

A terminal window titled 'sungpyo@sungpyo-ThinkPad-T450: ~/바탕화면/bigData'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The prompt is 'sungpyo@sungpyo-ThinkPad-T450:~/바탕화면/bigData\$'. The command entered is 'python3 homework3.py lyric\_data'. The output shows the first line of the script: '문장을 입력해주세요!: 물체가 정지해 있으면 정지에너지는 질량과 속도의 제곱의 곱과 같다.'

```
sungpyo@sungpyo-ThinkPad-T450: ~/바탕화면/bigData
File Edit View Search Terminal Help
sungpyo@sungpyo-ThinkPad-T450:~/바탕화면/bigData$ python3 homework3.py lyric_data
문장을 입력해주세요!: 물체가 정지해 있으면 정지에너지는 질량과 속도의 제곱의 곱과 같다.
```

2. 문장을 입력하게되면 분석겨로가 가장 유사한 문장과 유사한 문장을 보여줍니다.

가장 유사한 문장: 여기서 물체가 정지해 있는 상태라면, 정지에너지는 정지질량과 빛의 속도의 제곱의 곱과 같다는 공식을 얻어낼 수 있다.

나머지는 가장 유사한 순서대로 나타내었습니다.

```
sungpyo@sungpyo-ThinkPad-T450:~/바탕화면/bigData$ python3 homework3.py lyric_data
```

문장을 입력해주세요!: 물체가 정지해 있으면 정지에너지는 질량과 속도의 제곱의 곱과 같다

분석 결과 가장 유사한 문장: 여기서 물체가 정지해 있는 상태라면 , 정지에너지 는 정지질량과 빛의 속도의 제곱의 곱과 같다는 공식을 얻어낼 수 있다 .

유사한 문장 순서입니다: 여섯 가지 타입의 2차식은 다음과 같다 : \* 제곱은 근과 같다 (  $ax^2 = bx$  ) \* 제곱은 숫자와 같다 (  $ax^2 = c$  ) \* 근은 숫자와 같다 (  $bx = c$  ) \* 제곱과 근의 합은 숫자와 같다 (  $ax^2 + bx = c$  ) \* 제곱과 숫자의 합은 근과 같다 (  $ax^2 + c = bx$  ) \* 근과 숫자의 합은 제곱과 같다 (  $bx + c = ax^2$  ) ( 현대수학에서는 제곱이  $x^2$  , 근은  $x$  , 숫자는 일상적으로 사용되는 숫자를 의미한다 . ) 이슬람 수학자들은 인도 수학자들과는 다르게 음수에 대해서는 전혀 논하지 않았다 .

유사한 문장 순서입니다: == 특수 상대성 이론과 에너지보존의 법칙 == 고전적인 뉴턴역학에서 정지해있는 물체의 에너지는 정지해있으므로 운동에너지는 0이고 , 내부에너지로 화학 결합에너지와 열 에너지 , 그리고 적당한 장 에 의한 위치 에너지를 가진다 .

유사한 문장 순서입니다: 질량이 정지질량이라면 에너지는 정지에너지라 불리고 , 질량이 상대론적 질량이라면 에너지는 전체에너지이다 .

유사한 문장 순서입니다: 그런데 만일 두 우주선의 창이 모두 가려져 있어서 우주선 내에서는 자신의 우주선이 지상에 정지해 있는지 아니면 우주 공간에서 일정한 가속도  $g$ 로 등가속도 운동을 계속하고 있는지 모른다고 가정할 때 , 우주선 내에서 실험하는 사람은 물체가 가속도  $g$ 로 낙하하는 것을 보고서 우주선이 지구에 정지해 있는 것인지 아니면 우주선이 우주 공간을가속도  $g$ 로 등속도 운동을 하고 있는 것인지 구별할 방법이 없다 .

유사한 문장 순서입니다: 아인슈타인 이 특수 상대성 이론을 발견하면서 질량도 결국 에너지의 한 종류라는 것을 증명해내고 , 그 크기는 질량과 빛의 속도의 제곱에 비례하는 어마어마한 크기라는 것을 유도해냈다 .

유사한 문장 순서입니다: 그리고 어떤 물체의 정지질량은 그 물체가 정지해 있다는 특정한 경우의 상대론적 질량 값이라고 볼 수 있다 .

유사한 문장 순서입니다: 라이프니츠는 질량과 속도의 제곱을 곱한 양으로 측정되는 활력 은 역학적 과정에서 보존된다고 주장했다 .

유사한 문장 순서입니다: 뉴턴 역학 에서는 무게 있는 물체는 정지해 있을 때 운동에너지를 가지지 않고 , 경우에 따라 ( 상대적으로 적은 양의 ) 화학 에너지 , 열에너지 , 등 내부적으로 저장된 에너지 또는 역장 에서의 위치에 따른 위치에너지 를 가진다 .



가장 유사한 문장 순서입니다.

유사한 문장 순서입니다: 물리학에서 질량 - 에너지 동등성 ( ) 은 모든 질량 은 그에 상당하는 에너지를 가지고 그 역 또한 성립한다 ( 모든 에너지 는 그에 상당하는 질량을 가진다 ) 는 개념이다 .

유사한 문장 순서입니다: 질량과 에너지 보존 질량 - 에너지 등가 개념은 질량 보존의 법칙과 에너지 보존 을 하나로 묶는 것이다 .

유사한 문장 순서입니다: 왜냐하면 에너지는 생성되거나 사라질 수 없고 안에 갇힌 에너지는 어떠한 형태를 띠건 관계없이 질량을 갖기 때문이다 .

유사한 문장 순서입니다: 이러한 경우 , 해방 또는 잃어버린 에너지는 결손 질량과  $c^2$ 의 곱인 것이다 .

유사한 문장 순서입니다: 계의 정지질량과 부분의 정지질량의 합은 시스템을 형성할 때 복사된 결합에너지 만큼 차이가 난다 .

유사한 문장 순서입니다: 그러나 계의 정지질량은 , 그 계 전체가 정지상태인 좌표계 안에서 항상 그 부분의 상대론적 질량의 합이다 .

유사한 문장 순서입니다: 만약 높은 우선순위를 가지는 두 치환기가 같은 면에 있으면 , 그 배열은 Z이고 , 만약 다른 면에 있으면 , 그 배열은 E이다 .

유사한 문장 순서입니다:  $sp^5$ 의 충돌 실험에서부터 W 보존의 존재를 밝히는 신호가 나왔다. W 보존은 쿼크 쌍이나 렙톤 - 중성미자 쌍으로 붕괴하는데 , 1983년 UA1 그룹에서 전자나 뮤온이 W 보존에서 붕괴되어 만들어짐을 뜻하는 빔의 수직 방향의 큰 에너지 성분이 관측되었고 , 검출된 전자나 뮤온의 에너지와 손실된 에너지의 합이 W 보존의 예상 질량과 비슷함을 보여주는 6개의 신호를 발견한 것이다 .

유사한 문장 순서입니다: 두 광자는 각각 한 입자의 정지질량에너지를 지녀야 하고 , 하나의 광자와 무거운 핵의 경우 광자는 입자 쌍 전체의 정지질량에너지를 지녀야한다 .

유사한 문장 순서입니다: 열역학의 제1법칙은 이 에너지가 보존된다는 것이다 : 내부에너지의 차이는 받은 열에너지에서 계가 한 일을 뺀 값과 같다 .

유사한 문장 순서입니다: 프랑슘 - 223은 베타 붕괴하여 라듐 - 223으로 되며 붕괴에너지는 1149 keV 정도이고 , 적은 비율 ( 0.006 % ) 로는 알파 붕괴되어 아스타틴 - 219로 되고 , 붕괴 에너지는 5.4 MeV정도이다 .

유사한 문장 순서입니다: 공보관하고 ... 들어 있으면 빼고 ... 흥 : 안 들어 있으면 거기서 하 는 대로 ... 1138 흥 : 그리고 저 고흥길이 한 번 소동이 있었어요 .

### 3. 마지막으로 실행 시간을 출력하여 확인합니다.

총 실행시간은 41초 정도이며 10만개 데이터로 과제를 하였음에도 불구하고 굉장한 시간이 걸림을 알 수 있습니다.

유사한 문장 순서입니다: 상대론적 질량과 정지질량 사이의 차는 상대론적 운동에너지 ( 나누기  $c^2$  ) 이다 .

유사한 문장 순서입니다: 마찬가지로 어떤 종류의 에너지가 정지한 물체에 갇들면 증가된 질량도 그 갇든 에너지 ( 나누기  $c^2$  ) 만큼이 된다 .

총실행시간은 다음과 같습니다(단위 초): 41.55381636799984

sungpyo@sungpyo-ThinkPad-T450:~/바탕화면/bigData\$

다음은 소스 코드입니다.

```
from openpyxl import load_workbook
import argparse, sys, operator
import timeit
```

1. 코드를 작성하기 전에 필요한 모듈을 import 하는 작업입니다.

```
start = timeit.default_timer()
```

2. 시간을 측정하기 위한 함수입니다.

```
data = args.data
base = input('문장을 입력해주세요!: ')
```

3. 문장을 입력받고 그 문장을 base에 저장해줍니다.

```
wb = load_workbook('%s.xlsx'%data)
sentence_sheet = wb.worksheets[0]
max_row = sentence_sheet.max_row
```

4. 데이터를 엑셀 형태로 저장하였기때문에 엑셀 파일을 열기위한 모듈 함수를 사용하였습니다.  
worksheets는 0번 워크시트입니다.

```
sentence_list = []
```

```
for i in range(1, max_row+1):
    sentence_list.append(sentence_sheet['B%d'%i].value)
```

5. 워크시트에 있는 값들을 list에 넣어줍니다.

```
#n-gram 유사도 비교
def ngram(s, num):
    res = []
    slen = len(s) - num + 1
    for i in range(slen):
        ss = s[i:i+num]
        res.append(ss)
    return res
def diff_ngram(sa, sb, num):
    a = ngram(sa, num)
    b = ngram(sb, num)
```

```

r = []
cnt = 0
for i in a:
    for j in b:
        if i == j:
            cnt += 1
            r.append(i)
return cnt / len(a), r

```

```

three_gram_score_list = []
three_gram_word_list = []

```

```

for s in sentence_list:
    # 3-gram
    r3, word3 = diff_ngram(base, s, 3)
    three_gram_score_list.append(r3)
    three_gram_word_list.append(word3)

```

6. 지난 과제에서 했던 ngram 유사도 측정방식 입니다.

```

# 3-gram
three_max_index =
three_gram_score_list.index(max(three_gram_score_list))

```

```

tmp_sentence_list = []
tmp_track_list = []

```

```

print('\n분석 결과 가장 유사한 문장: %s\n '%sentence_list[three_max_index])

```

```

tmp_sentence_list.append(sentence_list[three_max_index])

```

```

total = []

```

7. 정확도가 높은 3-gram을 사용하였습니다. 여기서 가장 유사한 문장을 출력해줍니다.

```

try:
    for i in range(0, len(three_gram_score_list)):
        if three_gram_score_list[i] > 0.15:
            if(sentence_list[i] not in tmp_sentence_list ):
                tmp_sentence_list.append(sentence_list[i])
            total.append([three_gram_score_list[i], sentence_list[i]])

```

```

total.sort(key=lambda x:x[0], reverse=True)

```

```
if len(total) > 0:
    for t in total:
        print('유사한 문장 순서입니다: %s\n' % (t[1]))
```

```
except KeyError:
    sys.exit(1)
```

8. 이제 base문장과 비교하여 유사한 문장을 출력합니다.

```
stop = timeit.default_timer()
print('총실행시간은 다음과 같습니다(단위 초): ', stop - start)
```

9 마지막으로 시간을 출력하고 함수를 종료합니다.

<전체 소스 코드입니다.>

```
from openpyxl import load_workbook
import argparse, sys, operator
import timeit
```

```
start = timeit.default_timer()
```

```
parser = argparse.ArgumentParser()
parser.add_argument('data', help='lyric data you wanna compare with specific sentence you will provide')
args = parser.parse_args()
```

```
data = args.data
base = input('문장을 입력해주세요!: ')
```

```
wb = load_workbook('%s.xlsx'%data)
sentence_sheet = wb.worksheets[0]
max_row = sentence_sheet.max_row
```

```
sentence_list = []
```

```
for i in range(1, max_row+1):
    sentence_list.append(sentence_sheet['B%d'%i].value)
```

```
# for i in sentence_list:
#     print(i)
```

```
track_song_info_dict = {}
track_artist_info_dict = {}
```

```
# for i in range (1,5):
#     print(len(sentence_list[i]))
```

```
#n-gram 유사도 비교
```

```
def ngram(s, num):
    res = []
    slen = len(s) - num + 1
    for i in range(slen):
        ss = s[i:i+num]
        res.append(ss)
    return res

def diff_ngram(sa, sb, num):
    a = ngram(sa, num)
    b = ngram(sb, num)
    r = []
    cnt = 0
    for i in a:
        for j in b:
            if i == j:
                cnt += 1
                r.append(i)
    return cnt / len(a), r
```

```
three_gram_score_list = []
three_gram_word_list = []
```

```
for s in sentence_list:
    # 3-gram
    r3, word3 = diff_ngram(base, s, 3)
    three_gram_score_list.append(r3)
    three_gram_word_list.append(word3)
```

```
# 3-gram
three_max_index = three_gram_score_list.index(max(three_gram_score_list))
```

```
tmp_sentence_list = []
tmp_track_list = []
```

```
print("\n분석 결과 가장 유사한 문장: %s\n '%s'%sentence_list[three_max_index])
```

```
tmp_sentence_list.append(sentence_list[three_max_index])

total = []

try:
    for i in range(0, len(three_gram_score_list)):
        if three_gram_score_list[i] > 0.15:
            if(sentence_list[i] not in tmp_sentence_list ):
                tmp_sentence_list.append(sentence_list[i])
                total.append([three_gram_score_list[i], sentence_list[i]])

    total.sort(key=lambda x:x[0], reverse=True)

    if len(total) >0:
        for t in total:
            print('유사한 문장 순서입니다: %s\n' % (t[1]))

except KeyError:
    sys.exit(1)

stop = timeit.default_timer()
print ('총실행시간은 다음과 같습니다(단위 초): ',stop - start)
```