빅데이터최신기술 과제 3 - 문장생성확률 계산(Unigram)

20171701 정지현

전체 레포지토리 링크: https://github.com/ghyeon0/BigData_Homework/tree/master/HW3

혹시 보고서 형식이 많이 깨진다면 https://github.com/ghyeon0/BigData_Homework/blob/master/HW3/hw3.md 를 참조 부탁드립니다.

1. 알고리즘

```
f = open("sample.txt", "r")
sample_data = f.read()
f.close()
```

- 파일에서 텍스트를 불러옴.

```
sample_data = re.sub('[^가-힣0-9a-zA-Z\\s]', ' ', sample_data)
sample_data = sample_data.split()
```

- 정규표현식을 이용해서 문장 부호 및 특수문자 등을 띄어쓰기로 치환한 다음, 띄어쓰기를 기준으로 단어를 분리한다.

```
parsed_data = []

for each in sample_data:
    if each != " ":
        parsed_data.append(each)
```

- 위에서 치환하면서 띄어쓰기 한 칸이 단어로 인식되는 경우가 발생하여 그 경우를 제외해주었다.

```
word_count = {}
total_word_count = 0

for each in sample_data:
    if each not in word_count:
        word_count[each] = 1
    else:
        word_count[each] += 1
    total_word_count += 1
```

- 각 단어의 개수와 전체 단어의 개수를 세어주었다.
- 각 단어는 dictionary를 이용하여 갯수를 세었다.

```
probability_dic = {}
for key in word_count.keys():
    probability_dic[key] = word_count[key] / total_word_count
```

- 위에서 저장한 dictionary의 각 key별로 어절 출현 확률을 계산해서 저장했다.

```
if len(sys.argv) == 2:
    input_string = sys.argv[1]
else:
    input_string = input("Input Text: ")

input_string = re.sub('[^가-힣0-9a-zA-Z\\s]', ' ', input_string)

input_data = input_string.split()
```

- 생성 확률을 계산할 문장에 대해서도 문장 부호 제거 후 띄어쓰기를 기준으로 분할하는 작업을 진행한다.

```
sentence_probability = 1

for each in input_data:
    try:
        sentence_probability *= probability_dic[each]
    except KeyError:
        print("Key Not Found", "Key:", each)
        sys.exit()

print("문장 생성 확률:", sentence_probability)
```

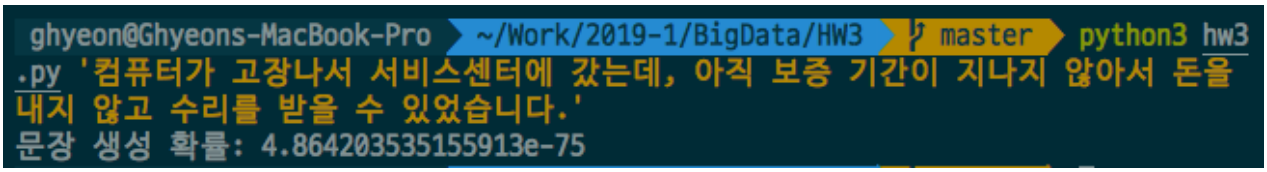
- 각 단어에 대한 확률을 모두 곱하면서 최종 확률을 구한다.
- 단, 키가 존재하지 않는다면 Key Not Found를 출력하고 종료한다.

2. 사용법

```
python3 hw3.py {확률 계산할 문장}
```

3. 실행결과

```
python3 hw3.py '컴퓨터가 고장나서 서비스센터에 갔는데, 아직 보증 기간이 지나지 않아서 돈을 내지 않고 수리를 받을 수 있었습니다.'
```



```
python3 hw3.py '그는 루머에 대해 사실이 아니라고 밝혔다.' # 확률이 높은 몇 가지 단어를 합친 문장
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW3 master python3 hw3
.py '그는 루머에 대해 사실이 아니라고 밝혔다.'
문장 생성 확률: 1.4329623621243793e-22
```

```
python3 hw3.py '어제 저녁에는 친구와 만나서 치킨을 먹었다.'
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW3 master python3 hw3
.py '어제 저녁에는 친구와 만나서 치킨을 먹었다.'
문장 생성 확률: 1.5780994120149207e-30
```

```
python3 hw3.py '나는 밥을 먹고 학교에 간다'
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW3 master ● python3 h
w3.py '나는 밥을 먹고 학교에 간다'
문장 생성 확률: 1.1141575916874963e-22
```

```
python3 hw3.py '나는 법을 먹고 학교에 간다'
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW3 master ● python3 h
w3.py '나는 법을 먹고 학교에 간다'
문장 생성 확률: 6.093759889229571e-22
```

```
python3 hw3.py '나는 밥을 묵고 학교에 간다'
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW3 master ● python3 h
w3.py '나는 밥을 묵고 학교에 간다'
문장 생성 확률: 1.4855434555833284e-24
```

```
python3 hw3.py '너는 밥을 먹고 학교에 간다'
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW3 master ● python3 h
w3.py '너는 밥을 먹고 학교에 간다'
문장 생성 확률: 3.6877834027770 69e-24
```

```
python3 hw3.py '너는 법을 묵고 학교에 산다'
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW3 master ● python3 h
w3.py '너는 법을 묵고 학교에 산다'
문장 생성 확률: 7.103870274531927e-26
```

- Unigram 분석의 한계로 '밥을 먹고' 보다 '법을 먹고'의 확률이 더 높게 나오는 문제가 발생하였다.

4. 아쉬웠던 점

문장부호만 제거하는것이 아닌 조사같은 불용어 부분도 제거했으면 좀 더 좋은 결과를 얻을 수 있었을 것 같다.

5. 전체 소스코드

```
import re
import sys

f = open("sample.txt", "r")
sample_data = f.read()
f.close()

sample_data = re.sub('[^가-힣0-9a-zA-Z\\s]', ' ', sample_data)
sample_data = sample_data.split()

parsed_data = []

for each in sample_data:
    if each != " ":
        parsed_data.append(each)

word_count = {}
total_word_count = 0

for each in sample_data:
    if each not in word_count:
        word_count[each] = 1
    else:
        word_count[each] += 1
    total_word_count += 1

probability_dic = {}
for key in word_count.keys():
    probability_dic[key] = word_count[key] / total_word_count

input_string = sys.argv[1]
input_string = re.sub('[^가-힣0-9a-zA-Z\\s]', ' ', input_string)

input_data = input_string.split()

sentence_probability = 1

for each in input_data:
    try:
        sentence_probability *= probability_dic[each]
    except KeyError:
        print("Key Not Found", "Key:", each)
        sys.exit()

print("문장 생성 확률:", sentence_probability)
```