UNISTUDY

파이썬

26강. 텍스트 처리

1. 텍스트 처리

- 메타문자를 이용하여 패턴을 만들고, re 모듈의 관련함수를 이용하여 텍스트를 처리하는 방법에 대해 서 알아본다. 특히 자연어를 분석하기 위해서 한글 문서나 영문 문서를 대상으로 문장부호나 특수문자 등을 제거하는 텍스트 전처리 과정을 알아본다.

2. 올바른 문장 선택

- 비정형 자료를 수집한 경우 올바르지 않은 문장이 포함되는 경우가 자주 발생한다. 이러한 경우 올바른 문장만 선택하는 텍스트 전처리 과정을 살펴보자. 예를 들면 다음과 같은 multi_line 변수의 문자열을 줄 단위로 분리한 다음 'http://'로 시작되지 않은 웹 주소(www.hongkildong.com)는 비정상 자료 라고 가정하여 수집 자료에서 제외시키는 사례를 예문으로 알아본다.

- multi_line= """http://www.naver.com http://www.daum.net www.hongkildong.com"""

2. 올바른 문장 선택

chapter07.lecture.step04_split_compile.py

```
# re 모듈관련 함수 import
from re import split, match, compile

# 텍스트 자료
multi_line= """http://www.naver.com
http://www.daum.net
www.hongkildong.com"""
```

Python Console

2. 올바른 문장 선택

```
# (1) 구분자를 이용하여 문자열 분리
web_site = split("\n", multi_line) # split
('pattern', string)
print(web_site)

# (2) 패턴 객체 만들기
pat = compile("http://")

# (3) 패턴 객체를 이용하여 정상 웹 주소 선택하기
sel_site = [site for site in web_site if ma
tch(pat, site)]
print(sel_site)
```

```
['http://www.naver.com', 'h
ttp://www.daum.net', 'www.h
ongkildong.com']
```

```
['http://www.naver.com', 'h
ttp://www.daum.net']
```

2. 올바른 문장 선택

- 문자열 검사 예
- # (1) 구분자를 이용하여 문자열 분리
- re 모듈의 split('pattern', string) 함수는 pattern을 구분자로 string을 분리하는 역할을 한다. "₩n"을 패턴으로 지정했기 때문에 multi_line을 대상으로 줄 단위로 분리되어 3개의 문자열 원소를 갖는 리스트를 반환한다.
- # (2) 패턴 객체 만들기
- re 모듈의 compile('pattern') 함수는 pattern으로 지정한 문자열을 패턴 객체로 반환한다. 이렇게 만들어진 패턴 객체는 re 모듈의 함수에서 'pattern' 파라미터 자리에 넣을 수 있다.

2. 올바른 문장 선택

- # (3) 패턴 객체를 이용하여 정상 웹 주소 선택하기
- re 모듈의 match('pattern', string) 함수는 string에서 pattern과 일치하는 문자열이 있으면 match 객체를 반환하고, 일치하는 문자열이 없으면 None으로 반환한다. 따라서 compile에서 만든 패턴 객체를 'pattern' 파라미터로 사용하고, 줄 단위로 분리한 web 주소를 string으로 사용해서 패턴과 일치하는 문자열만 sel_site에 추가한다. 최종 결과는 정상적인 웹 주소만 리스트의 원소로 선택된 것을 확인할 수 있다.

3. 자연어 전처리

- 자연어를 대상으로 토픽분석이나 감성분석 등을 수행하기 위해서는 먼저 해당 문서를 전처리해야 한 다. 예를 들면 문장에서 명사만 추출하여 단어의 출현빈도수를 분석하기 위해서는 문서에 포함된 문장 부호, 특수문자, 숫자 등을 제거해야 한다. 다음 예문은 re 모듈에서 제공하는 findall()과 sub() 함수를 이용하여 텍스트를 전처리하는 과정이다.

3. 자연어 전처리

chapter07.lecture.step05_text_preprocessing.py

```
# re 모듈관련 함수 import
from re import findall, sub


# 텍스트
texts = [' 우리나라 대한민국, 우리나라$$ 만세', '비아그&라 500GRAM 정력 최고!', '나는 대한민국 사람', '보험료 15000원에 평생 보장 마감 임박', '나는 홍길동']


# 단계1: 소문자 변경
texts_re1 = [t.lower() for t in texts]
print('texts_re1 :', texts_re1)


# 단계2: 숫자 제거
texts_re2 = [sub("[0-9]", '', text) for text in texts_re1]
print('texts_re2 :', texts_re2)


# 단계3: 문장부호 제거
texts_re3 = [sub('[,.?!:;]', '', text) for text in texts_re2]
print('texts re3 :', texts re3)
```

Python Console

```
texts_re1 : [' 우리나라 대한민국, 우리나라$$ 만세', '비아그&라 500gram 정력 최고!', '나는 대한민국 사람', '보험료 15000원에 평생 보장 마감 임박', '나는 홍길동']

texts_re2 : [' 우리나라 대한민국, 우리나라$$ 만세', '비아그&라 gram 정력 최고!', '나는 대한민국 사람', '보험료 원에 평생 보장 마감 임박', '나는 홍길동']

texts_re3 : [' 우리나라 대한민국 우리나라$$ 만세', '비아그&라 gram 정력 최고', '나는 대한민국 사람', '보험료 원에 평생 보장 마감 임박', '나는 홍길동']
```

3. 자연어 전처리

```
# 단계4: 특수문자 제거 : re.sub() 이용
spec str = '[@#$%^&*()]'
texts_re4 = [sub(spec_str, '', text) for text
in texts_re3]
print('texts re4 :', texts re4)

# 단계5: 영문자 제거
texts_re5 = [''.join(findall("[^a-z]", text))
for text in texts_re4]
print('texts_re5 :', texts_re5)

# 단계6: 공백제거
texts_re6 = [' '.join(text.split()) for text
in texts_re5] # 공백기준 split -> join 결합
print('texts_re6 :', texts_re6)
```

```
texts_re4 : [' 우리나라 대한민국 우리나라 만세', '비아그라 gram 정력 최고', '나는 대한민국 사람', '보험료 원에 평생 보장 마감 임박', '나는 홍길등']

texts_re5 : [' 우리나라 대한민국 우리나라 만세', '비아그라 정력 최고', '나는 대한민국 사람', '보험료 원에 평생 보장 마감 임박','나는 홍길동']

texts_re6 : ['우리나라 대한민국 우리나라 만세', '비아그라 정력 최고', '나는 대한민국 사람', '보험료 원에 평생 보장 마감 임박', '나는 홍길동']
```

3. 자연어 전처리

- 자연어 전처리 예

- # 단계1: 소문자 변경
- for문을 이용하여 texts 문자열 원소를 하나씩 변수 t로 넘겨받아서 문자열 객체에서 지원하는 lower()함수를 이용하여 모든 영문자를 소문자로 변경한다. 영문자를 처리하기 위해서는 일괄적으로 소문자 또는 대문자로 변경해야 한다.(대문자 변경 : t.upper())
- # 단계2: 숫자 제거
- 단계1의 결과 변수를 대상으로 sub()함수를 이용하여 숫자를 공백으로 치환하여 숫자를 제거한다.
- # 단계3: 문장부호 제거
- 단계2의 결과 변수를 대상으로 sub()함수를 이용하여 문장부호(,.?!:;)를 공백으로 치환하여 제거 한다.

3. 자연어 전처리

- # 단계4: 특수문자 제거
- 단계3의 결과 변수를 대상으로 sub()함수를 이용하여 특수문자(@#$%^&*())를 공백으로 치환하여 제거한다.
- # 단계5: 영문자 제거
- 단계4의 결과 변수를 대상으로 findall()함수를 이용하여 영문자를 제외시킨 후 join()함수를 이용하여 문자들을 결합하여 단어를 만든다.('우','리','나','라' → '우리나라')
- # 단계6: 공백 제거
- 단계5의 결과 변수를 대상으로 split()함수를 이용하여 한 칸 이상의 공백을 기준으로 문자열을 분리한 후 ' '.join()함수를 이용하여 공백을 기준으로 결합한다.
- ※ 주의 : join() 함수 앞에 오는 구분자는 한 칸의 공백이다.

4. 전처리 함수

- 텍스트를 대상으로 불용어를 제거하는 전처리 과정('7.3.2 자연어 전처리')을 보면 일련의 단계에 의해 서 진행된다. 만약 100개의 문장을 가지고 있는 텍스트 파일의 문장을 대상으로 전처리를 수행하기 위 해서는 일련의 단계를 100번 수행해야 한다. 따라서 전처리를 수행하는 일련의 단계를 함수로 정의해 놓고, 필요한 경우 호출을 통해서 재사용하는 것이 효과적이다.
- 다음 예문은 자연어 전처리'의 예문을 함수로 정의해 놓고, 함수 호출을 통해서 텍스트를 전처 리하는 예문이다.(Python Console 출력은 주석으로 대신한다.)

4. 전처리 함수

chapter07.lecture.step06_text_func.py

```
# re 모듈관련 함수 import
from re import findall, sub

# 텍스트 전처리
texts = [' 우리나라    대한민국, 우리나라$% 만세', '비아그&라 500GRAM 정력 최고!', '나는 대한민국 사람', '보험료 15000원에 평생 보장 마감 임박', '나는 홍길동']

# (1) 텍스트 전처리 함수
def crean_text(text) : # 문장 인수
    # 1~6단계
    texts_re = text.lower()  # 소문자 변경
    texts_re2 = sub('[0-9]', '', texts_re)  # 숫자 제거
    texts_re3 = sub('[,.?!;:]', '', texts_re2) # 문장부호 제거
    texts_re4 = sub('[@#$%^&*()]', '', texts_re3) #특수문자 제거
    texts_re5 = sub('[a-z]', '', texts_re4) # 영문자 제거
    texts_re6 = ' '.join(texts_re5.split()) # white space 제거
    return texts_re6 # 반환값

# (2) 함수 호출
texts_result = [clean_text(text) for text in texts]
print(texts_result)
# 출력 결과
'''
['우리나라 대한민국 우리나라 만세', '비아그라 정력 최고', '나는 대한민국 사람', '보험료 원에 평생보장 마감 임박', '나는 홍길동']
'''
```

4. 전처리 함수

- 전처리 함수 예
- # (1) 텍스트 전처리 함수
- 한 문장을 인수로 받아서 불용어를 단계별로 제거한 후 결과를 반환하는 함수이다. 여기서 문장은 texts 변수의 원소 하나를 의미한다. 현재 texts 변수는 5개의 문자열을 갖는 리스트 변수이다.(원소 를 문장으로 표현 함)
- # (2) 함수 호출
- texts의 원소를 하나씩 넘겨받아서 clean_text()함수에 인수로 넘겨서 1~6단계의 전처리 과정을 거쳐서 결과 값을 반환받는다. 처리 결과를 받은 texts_result 변수를 출력하면 전처리 결과를 확인할 수 있다.

THANK YOU