Preprocessing

전처리

지난 시간 요약

자연어처리

자연어처리란 전처리 -> 임베딩 -> 분석의 단계로 이루어진다

전처리 단계

토크나이저

문장이 있을 때 단어/어절 등으로 나누는 것을 토큰화라고 한다!

저는 배가 고픕니다.

['저', '는', '배', '가', '고프', 'ㅂ니다']

전처리 단계

토크나이저

영어를 기준으로, n't 이나 ph.d. 처럼 나누기 애매한 것들이 있다
(. , ' 와 같은 것들을 구두점이라고 한다)

No, I don't.　　**I'm Ph.D. student**

????

전처리 단계

토크나이저

우리가 직접 정규표현식으로 할 수도 있겠지만....

from nltk.tokenize
import word_tokenize
혹은 WordPunctTokenizer
혹은 TreebankWordTokenizer

이미 다 만들어져있다!

전처리 단계

토크나이저

일단 실험을 위해서 corpus 를 다운받자!

```
import nltk
nltk.download('book')
from nltk.book import *

nltk.corpus.gutenberg.fileids()
alice = nltk.corpus.gutenberg.raw("carroll-alice.txt")[:1000]
```

전처리 단계

토크나이저

가장 기본적인 형태 (구두점있는 부분은 합쳐서 토큰화해줌!)

```
import word_tokenize

text = ''
word_tokenize(text)
```

전처리 단계

토크나이저

n't 같은 부분은 n, ', t 각각 따로 두는 모듈!

```
import WordPunctTokenizer

text = ''
WordPuctTokenizer().tokenize(text)
```

전처리 단계

토크나이저

가장 표준화 된 모델은 아래! (하이픈 유지, ' 분리)

```
import TreebankWordTokenizer

text = ''
TreebankWordTokenizer().tokenize(text)
```

전처리 단계

토크나이저

정규표현식을 사용할 수 있는 토크나이저도 있다!

```
import RegexpTokenizer

text = ''
RegexpTokenizer(정규표현식).tokenize(text)
```

토크나이저

실습1.

RegexpTokenizer를 이용해서,
문자들을 토큰화해보세요.
(구두점있어도 다 나누기)

전처리 단계

토크나이저

문장들도 토큰화할 수 있다! (문장별로 나눠서 리스트화)

```
import sent_tokenize

text = ''
sent_tokenize(text)
```

전처리 단계

토크나이저

한국어를 위한 단어 토크나이저는?

!pip install konlpy 로 일단 설치!

from konlpy.tag import Okt 혹은 Kkma
Kkma 는 어간까지 분리한다!

전처리 단계

토크나이저

한국어를 위한 문장 토크나이저는?

!pip install kss 로 일단 설치!

```
import kss
text = ''
kss.split_sentences(text)
```

전처리 단계

어간 추출

-ing 나 복수형에 대해서 원래 단어를 찾아야 의미를 정확히 파악 가능

plays -> play

loving -> love

from nltk.stem import PorterStemmer

혹은 LancasterStemmer

어간 추출

stem 을 통해서 해당하는 어간만 추출해낼 수 있다!
(어간 - 단어의 의미를 담는 부분)

```
from nltk.stem import PorterStemmer

word = ''
PorterStemmer.stem(word)
```

전처리 단계

불용어 제거

I, is, not, an 등은 문장에서 의미를 구성하는데 큰 역할을 하지 않음!
(이런 단어를 stopword = 불용어라고 한다)

nltk.download('stopwords')
from nltk.corpus import stopwords

하면 안에 stopwords들이 쭉 있다!

불용어 제거

실습2.

stopwords.words('english')
안에는 모든 불용어들이 있습니다.

for 문과 not in 을 사용하여 불용어를
모두 제거해보세요.

인코딩

컴퓨터는 영어로 된 텍스트를 알아들을 수 없다..!

임베딩을 위해서는

단어 -> 숫자로 매치되어야!
(book - 1, computer - 2, math - 3)

인코딩

가장 기본적인 것이 빈도수대로 인덱스를 부여하는 것!
(counter는 빈도수가 가장 높은 것이 가장 앞에 온다!)

```
from collections import Counter

text = ''
count_list = Counter(text)
print(count_list['해당단어'])
```

인코딩

빈도수가 1인 것들은 크게 의미가 없을 가능성이 있으므로...

count_list.most_common(n)
를 통해서
상위 n개 까지만 저장을 해야한다!
(리스트 형태로 반환)

인코딩

실습3.

빈도수가 높은 단어에 대해서 낮은 정수 인덱스를 부여한 딕셔너리를 만드세요.

인코딩

원-핫 인코딩?...

인덱스에 해당하는 인덱스만 1로 부여하는 것

the의 빈도수가 가장 낮고, 인덱스가 7이면
이를 원-핫벡터로 만들면 [0, 0, 0, 0, 0, 0, 0, 1]

인코딩

실습4.

원-핫 인코딩을 구현하시오.

인덱스의 총 길이 = 텍스트의 갯수
인덱스에 해당하는 인덱스만 1, 나머지는 0

인코딩

실습4 - 예시.

['the', 'of', 'king', 'return']

{'the': 0, 'of':1, 'king':2, 'return':3}

[[1, 0, 0, 0], [0, 1, 0, 0],
[0, 0, 1, 0], [0, 0, 0, 1]]

THANK YOU

kim jun tae