형태소 분석

- 자연 언어의 문장을 "형태소"라는 의미를 갖는 최소 단위로 분할하고, 품사를 판별하는 작업이다.

- 활용 분야
  - 기계 번역
  - 텍스트 마이닝

- 한국어 형태소 분석 라이브러리
  - KoNLPy를 사용하면 한나눔, 꼬꼬마, Lomoran, MeCab, 트위터 등의 형태소 분석기를 사용할 수 있다.

- KoNLPy의 형태소 분석기
  - pip install konlpy
  - pip install jpype1 (jpyte와 관련된 오류 발생시)
  - 참조
    - http://konlpy.org/ko/latest/install/
    - http://konlpy.org/ko/latest/api/konlpy.tag

간단한 형태소 분석

- 다음과 같이 간단한 문장에 대하여 형태소 분석을 수행해 본다.
- 파일 이름 : konlpy_basic.py

```
from konlpy.tag import Twitter

# Twitter를 이용하여 twitter 객체를 생성한다.
twitter = Twitter()

text = '아버지 가방에 들어 가신다. 그래욕ㅋㅋ'

# pos() : 형태소를 분석해준다.
# norm 매개 변수 : 그래욕ㅋㅋ → 그래요
# stem 매개 변수 : 그래욕ㅋㅋ → 그렇다.
malist = twitter.pos( text, norm=True, stem = True )

print( malist )
```

```
# 출력 결과
[('아버지', 'Noun'), ('가방', 'Noun'), ('에', 'Josa'), ('들다',
'Verb'), ('가신', 'Noun'), ('다', 'Josa'), ('.', 'Punctuation'), ('그
렇다', 'Adjective'), ('욕', 'Noun'), ('ㅋㅋ', 'KoreanParticle')]
```

출현 빈도 분석

- 샘플 파일을 이용하여 간단한 예를 살펴 보기로 한다
- 파일 이름 : word_count.py

```
from bs4 import BeautifulSoup
from konlpy.tag import Twitter

fp = open("president_address.txt", "r", encoding="utf-8")

soup = BeautifulSoup(fp, "html.parser")

text = soup.string
print( text )
print(´-----------------------´)

# 텍스트를 한 줄씩 처리하기 --- (※2)
twitter = Twitter()
word_dic = {}

lines = text.split("\n")

for line in lines: # 한 줄 단위로 형태소 분석
    malist = twitter.pos(line)
    for word in malist:
        if word[1] == "Noun": #  명사 확인하기 --- (※3)
            if not (word[0] in word_dic):
                word_dic[word[0]] = 0
            word_dic[word[0]] += 1 # 카운트하기
```

```
# 많이 사용된 명사 출력하기 --- (※4)
keys = sorted(word_dic.items(), key=lambda x:x[1], reverse=True)

for word, count in keys[:50]:
    print("{0}({1}) ".format(word, count), end="")

print()
```

출력 결과

```
경제 재도약을 위해 국민 여러분께 드리는 말씀

저는 오늘, 우리 경제의 기초를 튼튼히 하고, 재도약을 위한
정부의 국정운영방안에 대해서 말씀드리고자 합니다.
그 계획과 추진은 국민 여러분의 동의가 있어야 하고
.....
-----------------------
개혁(33) 경제(32) 것(30) 국민(29) 수(25) 우리(25) 여러분(18) 금융
(17) 세계(17) 서비스(15) 교육(15) 정부(15) 산업(15) 청년(14) 일자리
(14) 문화(12) 재(12) 지금(11) 도약(11) 제(10) 임금(10) 고용(9) 과제
(9) 국가(9) 등(9) …
```

Word2vec

- Word2vec은 단어의 의미를 벡터 형식으로 표현해주는 알고리즘이다.
- 주요 기능
  - 연관된 단어들의 추출.
  - 단어와 단어들의 유사도 확인.
  - 선형적인 계산 가능.
    - 예시 : king – man + woman = queen

- Word2vec을 사용한 도구
  - gensim 서드-파티 라이브러리
  - 자연 언어를 벡터로 변환 해주는 라이브러리(Word2vec도 포함되어 있다.)
  - pip install gensim

gensim의 word2vec 사용하기

- 대통령 연설문을 word2vec을 사용해보도록 한다.
- KoNLPy의 Twitter 형태소 분석기로 형태소를 나누고, Word2vec로 읽어 들이는 예시이다.
- 파일 이름 : word2vec_txt.py

```
from bs4 import BeautifulSoup
from konlpy.tag import Twitter
from gensim.models import word2vec

fp = open("president_address.txt", "r", encoding="utf-8")
soup = BeautifulSoup(fp, "html.parser")
text = soup.string

# 텍스트를 한 줄씩 처리하기 --- (※2)
twitter = Twitter()
results = []
lines = text.split("\n")

for line in lines:
    # 형태소 분석하기 --- (※3)
    # 동사와 형용사는 단어의 기본형만 사용하도록 하였다.
    malist = twitter.pos(line, norm=True, stem=True)

    r = []
    for word in malist:
        # 어미/조사/구두점 등은 대상에서 제외
        if not word[1] in ["Josa", "Eomi", "Punctuation"]:
            r.append(word[0])

    rl = (" ".join(r)).strip()
    results.append(rl)
    print(rl)
```

```
# 파일로 출력하기  --- (※4)
file = 'president_result.txt'

with open(file, 'w', encoding='utf-8') as fp:
    fp.write("\n".join(results))

# Word2Vec 모델 만들기 --- (※5)

# LineSentence 함수로 텍스트 파일을 읽어 들인다.
data = word2vec.LineSentence( file )

# Word2Vec 메소드에 매개 변수 형태로 넣어 주면 모델이 만들어 진다.
model = word2vec.Word2Vec(data,
    size=200, window=10, hs=1, min_count=2, sg=1)

# save() 메소드로 모델을 저장한다.
model.save("president.model")

print("ok finished")
```

연설 모델 파일 살펴 보기

- Word2vec 모델을 읽어 들이고 간단히 살펴 보도록 한다.
- 파일 이름 : read_model.py

```
from gensim.models import word2vec

model = word2vec.Word2Vec.load('president.model')

print( model )
print( type(model) )
# Word2Vec(vocab=355, size=200, alpha=0.025)

# 모델 파일을 읽어 들이면 여러 단어를 추출해 볼 수 있다.
# most_similar : 유사한 단어를 확인하고자 할 때 사용한다.
result = model.most_similar(positive=['경제'])

# 이 결과를 이용하여 막대 그래프를 그려 보면 좋을 듯 하다.
print( result )

result = model.most_similar(positive=['취업'])

print( result )
```

출력 결과

```
Word2Vec(vocab=355, size=200, alpha=0.025)

<class 'gensim.models.word2vec.Word2Vec'>

[('있다', 0.9962364435195923), ('재', 0.996213436126709), ('성장', 0.9960838556289673), ('정부', 0.9957307577133179…

[('교육', 0.9937941431999207), ('더욱', 0.9930610656738281), ('사회', 0.9926791191101074), ('위', 0.9924912452697754), ('들', 0.9924020171165466), ('수', 0.9919396638870239…
```

위키피디아 데이터

- 덩치가 좀 큰 파일을 이용하여 테스트해보도록 한다.
- 파일 이름 : wiki_model.py, wiki_model, wiki.model.wv.syn0.npy, wiki.model.syn1neg.npy

```
from gensim.models import word2vec

model = word2vec.Word2Vec.load('wiki.model')
# 같이 첨부 되어야 할 파일 리스트
# wiki.model.wv.syn0.npy, wiki.model.syn1neg.npy

# most_similar : 유사한 단어를 확인하고자 할 때 사용한다.
result = model.most_similar(positive=['파이썬', 'Python'])
# 이 결과를 이용하여 막대 그래프를 그려 보면 좋을 듯 하다.
print( result )
result = model.most_similar(positive=['아빠', '여성'], negative=['남성'])[0]
print( result )
result = model.most_similar(positive=['왕자', '여성'], negative=['남성'])[0:5]
print( result )
result = model.most_similar(positive=['서울', '일본'], negative=['한국'])[0:5]
print( result )
result = model.most_similar(positive=['서울', '중국'], negative=['한국'])
print( result )
result = model.most_similar(positive=['오른쪽', '남자'], negative=['왼쪽'])[0]
print( result )
result = model.most_similar(positive=['서울', '맛집'])[0:5]
print( result )

# 신문사의 기사를 모아서 테스트해보기
# 트윗트 등의 데이터를 긁어서 활용하여 최신 트렌드 파악해보기
```

출력 결과

```
[('Perl', 0.9213456511497498), ('Java',
0.906911313533783), ('Tcl', 0.905478835105896),
('MATLAB', 0.8736516237258911), ('Lisp',
0.8692713975906372), ('자바스크립트',
0.8669256567955017), ('하스켈', 0.8633924722671509),
('JSP', 0.8586523532867432), ('IDL',
0.8562408685684204), ('CLI', 0.8507612943649292)]

('엄마', 0.8517739772796631)
… 이하 생략
```

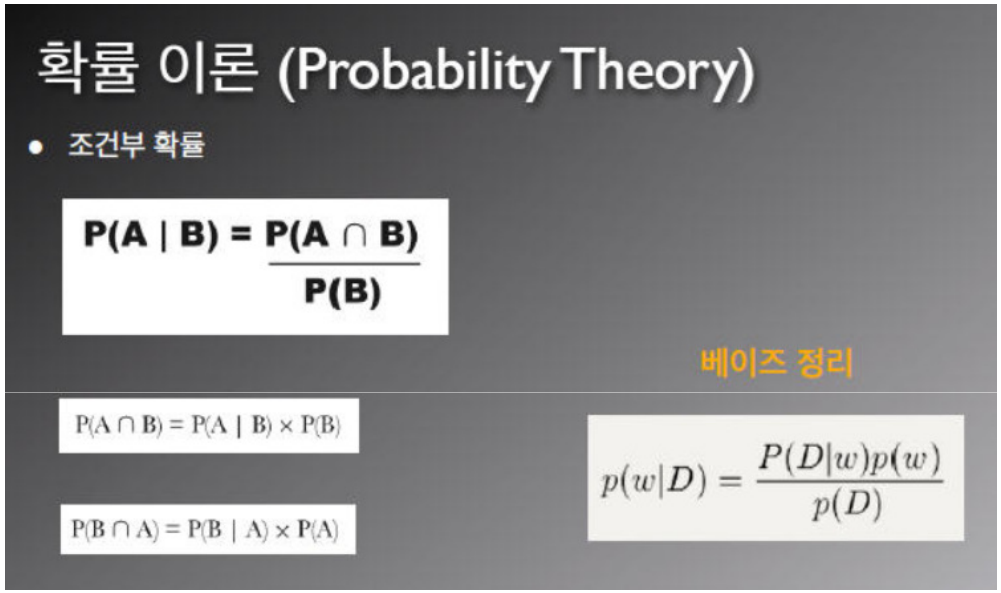
텍스트 분류

- 텍스트를 분류하는 방법에는 여러 가지가 있다.

분류 방법	텍스트
베이지안 필터 (Bayseian filter)	베이즈 정리를 이용한 텍스트 분류 방법이다. 나이브 베이즈 분류 (Naïve Bayes Classifier) 알고리즘을 사용한다. 학습을 많이 할 수록 필터의 분류 기능이 향상된다. 스팸 메일/스팸 글 등을 구분하는 데 사용한다. 문장들의 카테고리 분류에도 사용된다.
MLP(Multi Layer Perceptron)	입력 층과 출력 층 사이에 은닉층을 넣은 뉴럴 네트워크이다. 텍스트 데이터를 숫자로 표현하는 벡터로 변환한다.

베이즈 정리

- 베이즈 정리는 "조건부 확률"과 관련된 이론이다.
- 토마스 베이스에 의하여 정립된 이론이다.



- 조건부 확률 예시
  - 주사위를 2번 던져서 처음에 3이 나오고, 두 번째에 짝수가 나올 확률은?
  - 3이 나올 확률은 1/6이고, 짝수가 나올 확률은 3/6이므로 동시에 발생하려면 곱하면 된다.

$$\frac{1}{6} \times \frac{1}{2} = \frac{1}{12}$$

베이즈 정리

- 어느 문방구의 매출이 다음과 같다.

전체 손님 수 : 100 공책을 구입한 손님 수 : 50 사인펜을 구입한 손님 수 : 20 두 가지를 모두 구입한 손님 수 : 10

- 문제 1
  - 공책 구입한 사람 중에 사인펜을 구입할 손님의 확률은 ?

- 문제 2
  - 사인펜을 구입한 사람 중에 공책을 구입할 손님의 확률은 ?

$$\frac{50}{100} \times \frac{10}{50} = \frac{500}{5000} = \frac{1}{10}$$

$$\frac{20}{100} \times \frac{10}{20} = \frac{200}{2000} = \frac{1}{10}$$

베이지안 필터 사용하기

- 메일에 '광고', '중요' 메일 2가지가 있다고 가정한다.
- 해당 메일의 제목을 이용하여 광고 스팸 메일인지 아닌지를 판단하는 프로그램을 작성하세요.
- 새로운 메일이 들어 왔을 때 잘 판단할 수 있는 가?
- 파일 이름 : bayes.py(필터 구현해 놓은 파일), bayes_test.py(실행 파일)

변수 : words

{'진행', '확인', '배송', '따뜻하다', '회의', '프로젝트', '일정', '인기', '무료', '할인', '되어다', '백화점', '세일', '부탁드리다', '신제품', '등록', '자다', '기간', '쿠폰', '계', '함께', '한정', '계약', '파격', '보고', '소식', '오늘', '봄', '제품', '30', '현데', '찾아오다', '상황', '없다', '선물'}

변수 : word_dict

```
{
        '광고':
                {'쿠폰': 1, '계' '한정': 1, '파격': 1, '할인': 1, '백화점': 1, '세일': 3, '신제품, '선물': : 1},
        '중요':
                {'진행': 1, '등록': 1, '계약': 1, '확인': 1, '보고': 1, '상황': 1, '회의': 1, '일정': 3}
}
```

변수 : category_dict

{'광고': 5, '중요': 5}

베이지안 필터 사용하기

- 해당 메일의 제목을 이용하여 광고 스팸 메일인지 아닌지를 판단하는 프로그램을 작성하세요.

```
from bayes import BayesianFilter
bf = BayesianFilter()

# 텍스트 학습
bf.fit("파격 세일 - 오늘까지만 30% 할인", "광고")
bf.fit("쿠폰 선물 & 무료 배송", "광고")
bf.fit("현데계 백화점 세일", "광고")
bf.fit("봄과 함께 찾아온 따뜻한 신제품 소식", "광고")
bf.fit("인기 제품 기간 한정 세일", "광고")
bf.fit("오늘 일정 확인", "중요")
bf.fit("프로젝트 진행 상황 보고","중요")
bf.fit("계약 잘 부탁드립니다","중요")
bf.fit("회의 일정이 등록되었습니다.","중요")
bf.fit("오늘 일정이 없습니다.","중요")

print('-----------------------------------')
print( bf.showInfo())
print('-----------------------------------')

# 예측
pre, scorelist = bf.predict("재고 정리 할인, 무료 배송")
print("결과 =", pre)
print(scorelist)
```

MLP로 텍스트 분류하기

- 단어 하나에 ID를 부여하고, ID의 출현 빈도와 정렬 순서를 기반으로 벡터를 만드는 방법이다.
- 문장에 어떠한 단어들이 있는 지를 수치로 나타내는 방법을 마치 "가방 안에 단어들을 넣어서 사용하는 것"과 같은 개념으로 보아 Bow(Bag of words)라고 한다.

몇|번|을|쓰러지다|몇|번|을|무너지다|다시|일어나다

twitter.pos(text, norm=True, stem=True) 실행 후 결과

형태소	몇	번	을	쓰러지다	몇	번	을	무너지다	다시	일어나다
ID	1	2	3	4	1	2	3	5	6	7

추가적으로 단어의 출현 횟수를 세어 보자.

형태소	몇	번	을	쓰러지다	무너지다	다시	일어나다
ID	1	2	3	4	5	6	7
출현 횟수	2	2	2	1	1	1	1

텍스트 분류하기

- 텍스트를 분류하는 과정은 다음과 같이 수행하면 된다.
  - (1) 불필요한 품사를 제거한다.
  - (2) 사전으로 기반으로 단어를 숫자로 변환한다.
  - (3) 파일 내부의 단어 출현 비율을 계산한다.
  - (4) 데이터를 학습시킨다.
  - (5) 텍스트 데이터를 넣어 성공률을 확인한다.

단어를 ID로 변환하고 출현 횟수 구하기

- 신문 기사와 카테고리를 학습시켜 모델을 만들고, 해당 모델을 이용하여 새로운 신문 기사의 카테고리를 파악하는 예제이다.

- 2017/04/01 ~ 0207/04/17까지 각 신문사의 정치, 경제, 사회, 생활/문화, 세계, IT/과학과 관련된 신문 기사 1만개를 추출하고, 형태소 분석을 수행한 파일 제공(word-dic.json)

- 파일 이름 : mlp2-seq.py
  - 실습을 수행하고 나면, 아래 폴더에 3개의 파일이 자동으로 생성된다.
  - data.json, data-mini.json, word-dic.json
  - 나중에 해당 파일들을 덮어 쓰기 하도록 한다.

- 파일 이름 : mlp3-classify.py