Chapter

2

텍스트 분석

1. 한국어 형태소 분석
2. 텍스트 분석

학습 목표

✓ Python 텍스트 분석 이해한다.
✓ 텍스트 자료의 시각화 방법에 대해서 이해한다.
✓ 나이브 베이즈 분석 방법을 이해한다.
✓ 텍스트 유사성 측정 방법을 이해한다.

주요 내용

✓ 한글 형태소 분석
✓ 워드클라우드 작성
✓ 나이브 베이즈 분석을 이용한 분류 방법
✓ 유사성 측도를 이용한 문장의 유사성 판단 방법

1

한국어 형태소 분석

한국어 분석(형태소 분석)

❖ 형태소 분석

- 자연어의 문장을 "형태소"라는 의미를 갖는 최소 단위로 분할하고 품사를 판별하는 작업(의미를 가지는 요소로서는 더 이상 분석할 수 없는 가장 작은 말의 단위)
- 기계 번역, 텍스트 마이닝 등에서 활용
- 영어는 형태소마다 띄어쓰기로 하여 문장을 구성하기 때문에 어렵지 않음
- 아시아 언어의 분석은 문법 규칙에 의한 방법과 확률적 언어 모델을 사용하는 방법 등을 활용
  - 품사 사전과 문법 사전을 기반으로 대조하면서 형태소를 분석

❖ 한국 형태소 분석 라이브러리

- 다양한 형태소 분석 라이브러리가 오픈소스로 제공
- 파이썬에서는 KoNLPy(http://konlpy.org/ko/latest/)를 사용
- 운영체제별 KoNLPy 설치 방법

  https://konlpy-ko.readthedocs.io/ko/v0.4.3/install/#id2

  - Jpype 파일 다운로드

    https://www.lfd.uci.edu/~gohlke/pythonlibs/#jpype

한국어 분석(형태소 분석)

In [1]:
```
from konlpy.tag import Twitter

# twitter 객체를 생성
twitter=Twitter()

# pos 메소드로 형태소 분석(norm 옵션 : 단어 변환, stem 옵션 : 원형 단어 변환)
# norm 옵션 : "그래욬ㅋㅋㅋ?" -> "그래요" 변환, stem 옵션 : "그렇다" 변환
malist=twitter.pos("아버지 가방에 들어가신다.",norm=True,stem=True)
print(malist)
```

```
C:₩ProgramData₩Anaconda3₩envs₩pylecture₩lib₩site-packages₩konlpy₩tag₩_okt.py:16: UserWarning: "Twitter" has changed to "Okt" since KoNLPy v0.4.5.
  warn('"Twitter" has changed to "Okt" since KoNLPy v0.4.5.')
```

```
[('아버지', 'Noun'), ('가방', 'Noun'), ('에', 'Josa'), ('들어가다', 'Verb'), ('.', 'Punctuation')]
```

한국어 분석(형태소 분석)

❖ 출현 빈도 분석

- 각 형태소별로 빈도수를 측정

국립어학원 언어 정보 나눔터 말뭉치 데이터베이스
https://ithub.korean.go.kr/user/total/database/corpusManager.do

날짜 검색 ~

자료구분 외부용

검색어 ::: 전체 ::: 토지 검색

총 9건이 있습니다. 10개씩 보기

번호	제목	자료구분	올린사람	올린날짜	조회 수
9	토지 7, 전자파일	외부용	관리자	2014-01-09	1236
8	토지 1, 전자파일	외부용	관리자	2014-01-09	2221
7	토지 16, 전자파일	외부용	관리자	2014-01-09	679
6	토지12	외부용	관리자	2014-01-09	749
5	토지4	외부용	관리자	2014-01-09	658
4	토지3	외부용	관리자	2014-01-09	799
3	토지5	외부용	관리자	2014-01-09	588
2	토지2	외부용	관리자	2014-01-09	1503
1	한과 삶: 토지 비평1, 전자파일	외부용	관리자	2014-01-09	613

한국어 분석(형태소 분석)

In [2]:
```
import codecs
from bs4 import BeautifulSoup
from konlpy.tag import Twitter

# utf-16 인코딩으로 파일을 열고 글자를 출력하기 --- (※1)
fp = codecs.open("C:/Users/datam_000/Documents/Python/Module04/Ch02/BEXX0003.txt",
                 "r", encoding="utf-16")
soup = BeautifulSoup(fp, "html.parser")
body = soup.select_one("body > text")
text = body.getText()

# 텍스트를 한 줄씩 처리하기
twitter = Twitter()
word_dic = {}
lines = text.split("₩n")
for line in lines:
    malist = twitter.pos(line)
    for word in malist:
        if word[1] == "Noun": #  명사 확인하기
            if not (word[0] in word_dic):
                word_dic[word[0]] = 0
            word_dic[word[0]] += 1 # 카운트하기

# 많이 사용된 명사 출력하기
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()
```

한국어 분석(형태소 분석)

것(644) 그(554) 말(485) 안(304) 소리(196) 길(194) 용이(193) 눈(188) 놈(180) 내(174) 사람(167) 봉(165) 치수(160) 평산(160) 얼굴(156) 거(152) 네(151) 일(149) 이(148) 못(147) 댁(141) 생각(141) 때(139) 강청댁(137) 수(134) 서방(131) 집(131) 나(122) 더(120) 서희(119) 머(116) 어디(112) 마을(111) 최(110) 년(109) 김(99) 칠성(97) 구천이(96) 니(96) 뒤(91) 제(90) 날(90) 아이(88) 하나(84) 녀(83) 두(83) 참판(82) 월(82) 손(81) 임(79)

2

텍스트 분석

워드클라우드

❖ 워드 클라우드(Word Cloud)

- 워드 클라우드는 단어들을 구름모양으로 나열하여 시각화하는 방법
- 빈도가 높고 핵심단어일수록 가운데에 크게 표현
- 어떤 단어들을 자주 사용하고, 어떤 의미들이 내포되어 있는지 한 눈에 파악
- 영문인 경우는 wordcloud 패키지 사용하여 쉽게 작성
  - 파이썬에서는 다양한 이미지를 활용해 워드 클라우드 작성
- 한글의 경우는 KoNLPy 패키지를 사용하여 명사를 추출하고 wordcloud 패키지 사용하여 작성

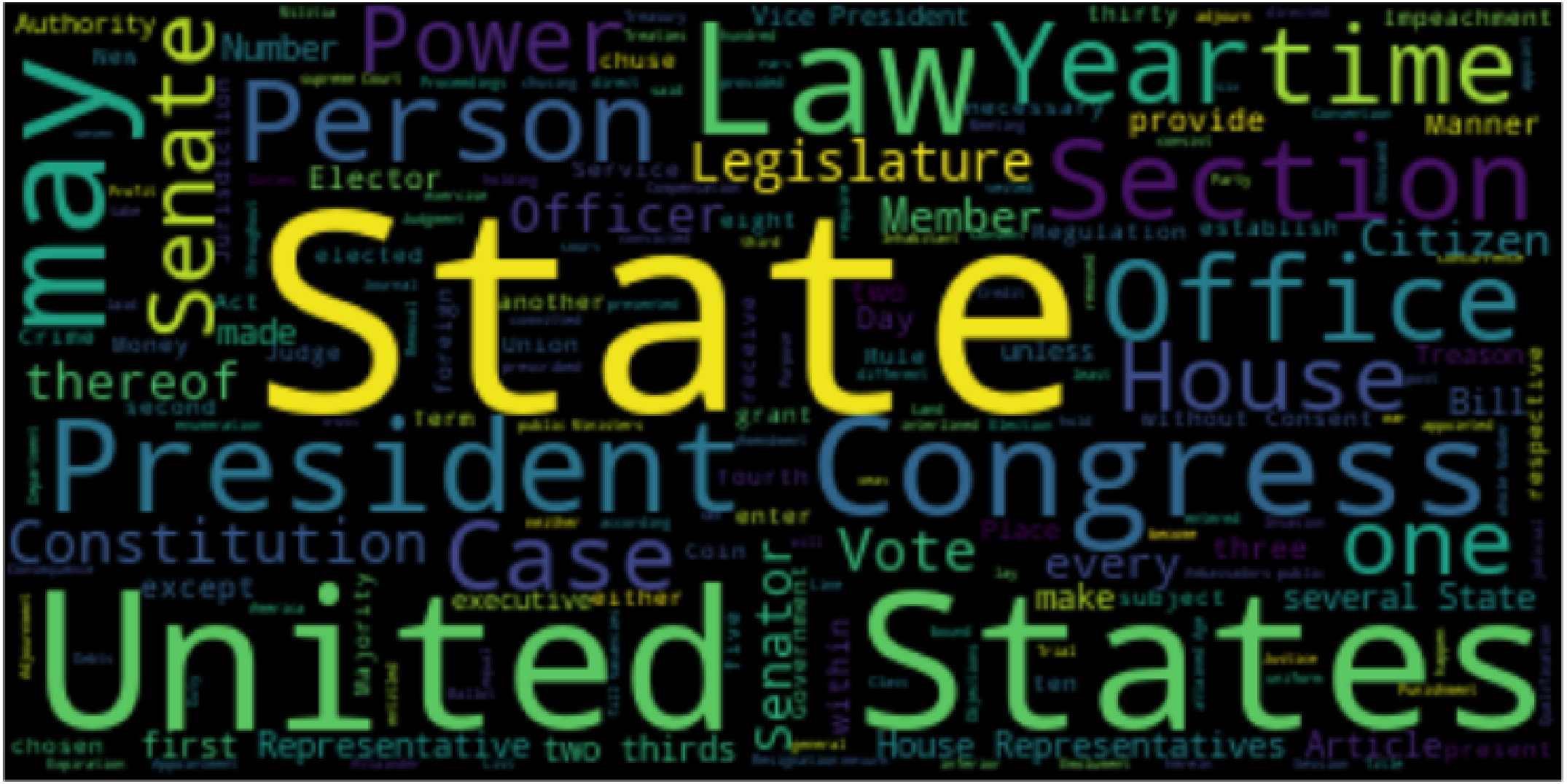
워드클라우드

In [3]:
```
import matplotlib.pyplot as plt
%matplotlib inline

from wordcloud import WordCloud

# 텍스트 파일 불러오기
text=open("C:/Users/datam_000/Documents/Python/Module04/Ch02/constitution.txt").read()
# 단어별 빈도 계산(공백으로 분리된 단어)
wordcloud=WordCloud().generate(text)
wordcloud.words_
```

Out[3]:
```
{'State': 1.0,
 'United States': 0.8181818181818182,
 'Law': 0.5151515151515151,
```

In [4]:
```
plt.figure(figsize=(12,12))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

워드클라우드



워드클라우드

In [5]:
```
# 단어별 빈도 계산
wordcloud=WordCloud(max_font_size=40).generate(text)

plt.figure(figsize=(12,12))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

워드클라우드

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
from wordcloud import WordCloud
from wordcloud import STOPWORDS

# 이미지 불러오기
alice_mask=np.array(Image.open("C:/Users/datam_000/Documents/Python/Module04/Ch02/alice

# 삭제할 단어와 추가
stopwords=set(STOPWORDS)
stopwords.add("said")

# 텍스트 파일 불러오기
text=open("C:/Users/datam_000/Documents/Python/Module04/Ch02/constitution.txt").read()

# 단어별 빈도 계산(공백으로 분리된 단어)
wordcloud=WordCloud(background_color="white", max_words=2000, mask=alice_mask,
                    stopwords=stopwords)
wordcloud=wordcloud.generate(text)
```

워드클라우드


```
# 색상 함수
r=lambda: np.random.randint(0,255)
#color=lambda: (r(), r(), r())
def wc_color(word, font_size, position, orientation, randomstate=None, **kwargs) :
    return (r(), r(), r())

# 워드클라우드 작성
plt.figure(figsize=(12,12))
plt.imshow(wordcloud.recolor(color_func=wc_color, random_state=2),
           interpolation="bilinear")
plt.axis("off")
plt.show()
```

워드클라우드

이 자료는 혁신성장 청년인재 집중양성 사업 강의 자료로
개인 학습 자료로만 사용가능 합니다.

워드클라우드

In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image

from wordcloud import WordCloud
from wordcloud import STOPWORDS
from wordcloud import ImageColorGenerator # 이미지에서 색 추출

# 이미지 불러오기
alice_color=np.array(Image.open("C:/Users/datam_000/Documents/Python/Module04/Ch02/alic
image_colors=ImageColorGenerator(alice_color) # 이미지에서 색 추출

# 삭제할 단어와 추가
stopwords=set(STOPWORDS)
stopwords.add("said")

# 텍스트 파일 불러오기
text=open("C:/Users/datam_000/Documents/Python/Module04/Ch02/constitution.txt").read()

# 단어별 빈도 계산(공백으로 분리된 단어)
wordcloud=WordCloud(background_color="white", max_words=2000, mask=alice_color,
                    stopwords=stopwords)
wordcloud=wordcloud.generate(text)
```

워드클라우드


```
# 워드클라우드 작성
plt.figure(figsize=(12,12))
plt.imshow(wordcloud.recolor(color_func=image_colors), interpolation="bilinear")
plt.axis("off")
plt.show()
```

워드클라우드

이 자료는 혁신성장 청년인재 집중양성 사업 강의 자료로
개인 학습 자료로만 사용가능 합니다.

워드클라우드

❖ 한글 워드클라우드

- NLTK(Natural Language Toolkit) 자연어 처리 패키지
  - NLTK 패키지는 교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 패키지
  - 다양한 기능 및 예제를 가지고 있으며 실무 및 연구에서도 많이 사용
  - NLTK 패키지가 제공하는 주요 기능
    - 말뭉치
    - 토큰 생성
    - 형태소 분석
    - 품사 태깅

워드클라우드

In [8]:
```
import nltk
from konlpy.corpus import kolaw
from konlpy.tag import Twitter
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator # 이미지에서 색 추출

t=Twitter()

# 텍스트 파일 불러오기
ko_con_text=open("C:/Users/datam_000/Documents/Python/Module04/Ch02/speech.txt",
                 encoding="utf-8").read()
ko_con_text

# 명사 추출
tokens_ko=t.nouns(ko_con_text)
tokens_ko
```

워드클라우드


```
# 단어 삭제
stop_words=[]
f=open("C:/Users/datam_000/Documents/Python/Module04/Ch02/stop_word.txt",
       encoding="utf-8")
lines=f.readlines()
for x in lines :
    stop_words.append(x.strip())
stop_words
tokens_ko=[each_word for each_word in tokens_ko if each_word not in stop_words]
tokens_ko

sel_word=nltk.Text(tokens_ko)
data=sel_word.vocab().most_common(1000)
tmp_data=dict(data)


# 단어별 빈도 계산(공백으로 분리된 단어)
wordcloud=WordCloud(font_path="C:/Windows/Fonts/HYBDAM.ttf",
                    background_color="white")
wordcloud=wordcloud.generate_from_frequencies(tmp_data)
```

워드클라우드


```
# 워드클라우드 작성
plt.figure(figsize=(12,12))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

워드클라우드

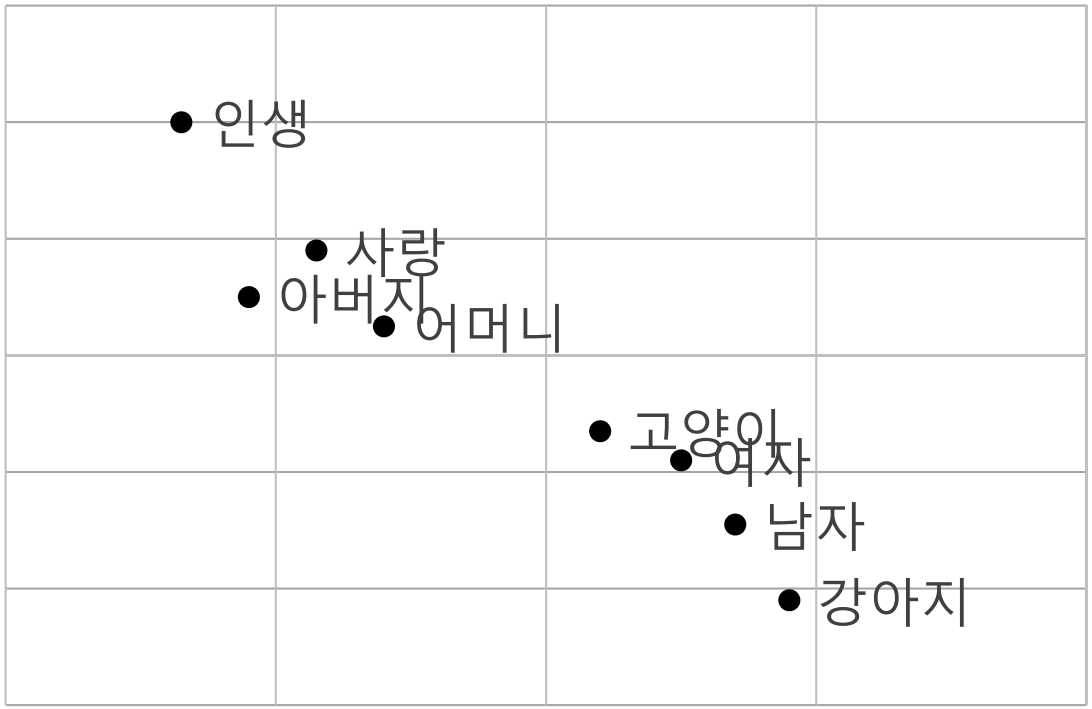
In [9]:
```
# 이미지 불러오기
#korea_color=np.array(Image.open("C:/Users/datam_000/Documents/Python/Module04/Ch02/ko
#image_colors=ImageColorGenerator(korea_color) # 이미지에서 색 추출
korea_color=np.array(Image.open(
    "C:/Users/datam_000/Documents/Python/Module04/Ch02/south-korea-flag.png"))
image_colors=ImageColorGenerator(korea_color) # 이미지에서 색 추출

# 단어별 빈도 계산(공백으로 분리된 단어)
wordcloud=WordCloud(font_path="C:/Windows/Fonts/HYBDAM.ttf", relative_scaling=0.2,
                    mask=korea_color, background_color="white",
                    min_font_size=1, max_font_size=80)

wordcloud=wordcloud.generate_from_frequencies(tmp_data)

# 워드클라우드 작성
plt.figure(figsize=(12,12))
#plt.imshow(korea_color, interpolation="bilinear")
plt.imshow(wordcloud.recolor(color_func=image_colors), interpolation="bilinear")
plt.axis("off")
plt.show()
```

워드클라우드

문장을 벡터로 변환

❖ Word2Vec

- 문장 내부의 단어를 벡터로 변환(임베딩(Embedding))
- 단어의 연결을 기반으로 단어의 연관성을 벡터로 생성하여 단어의 의미를 파악



- Word2Vec의 결과를 2차원 그래프로 작성 결과 아버지, 어머니는 사랑과 가깝고, 인생과 개, 고양이는 멀리 있다는 것을 알 수 있음
- 이를 이용해 의미를 선형으로 계산 가능 아빠-남자+여자의 계산 결과는 엄마

문장을 벡터로 변환

- genism 라이브러리를 설치하고 말뭉치(corpus)라고 불리는 단어 사전을 생성하여 데이터를 준비하고 학습
- KoNLPy의 Twitter 형태소 분석기로 형태소로 나누고 Word2Vec으로 읽어 들여 데이터 생성

In [10]:
```
import codecs
from bs4 import BeautifulSoup
from konlpy.tag import Twitter
from gensim.models import word2vec

# utf-16 인코딩으로 파일을 열고 글자를 출력하기
fp=codecs.open("C:/Users/datam_000/Documents/Python/Module04/Ch02/BEXX0003.txt",
               "r", encoding="utf-16")
soup=BeautifulSoup(fp, "html.parser")
body=soup.select_one("body > text")
text=body.getText()
```

문장을 벡터로 변환


```
# 텍스트를 한 줄씩 처리하기
twitter=Twitter()
results=[]
lines=text.split("₩r₩n")
for line in lines:
    # 형태소 분석하기
    # 단어의 기본형 사용
    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
wakati_file='toji.wakati'
with open(wakati_file, 'w', encoding='utf-8') as fp:
    fp.write("₩n".join(results))

# Word2Vec 모델 만들기
data=word2vec.LineSentence(wakati_file) # 텍스트 읽어 들이기
model=word2vec.Word2Vec(data, size=200, window=10, hs=1, min_count=2, sg=1) # 모델생성
model.save("C:/Users/datam_000/Documents/Python/Module04/Ch02/toji.model")
print("ok")
```

문장을 벡터로 변환

제 1 편 어둠 발 소리
서다 序
1897년 의 한가위
까치 들 울타리 안 감나무 오다 아침 인사 하다 전 무색 옷 댕기 꼬리 늘 아이 들 송편 입 물 마을 길 쏘다 기쁘다 날뛰다 어른 들 해 중천 좀 기울어지다 무렵 이 래야 차례 치르다 하다 성묘 하다 하다 이웃 끼리 음식 나누다 보다 한나절 넘다 이 때 타 작 마당 사람 들 모 이기 시작 들뜨다 시작 남정 노인 들 아낙 들 채비 아무래도 더디다 그렇다 수 없다 것 식구 들 시중 음식 간수 끝내다 제 자신 치장 남아 있다 이 바람 고개 무겁다 벼 이삭 황금 빛 물결 이루다 들판 마음 놓다 새 떼 들 모여들다 풍 성하다 향연 벌이다
후우 이이 이 놈 새 떼 들 극성 새 쫓다 할망구 와삭 와삭 풀발 선 출입 옷 갈아입다 타

작 마당 굿 보고 있다 것 추석 마을 남녀 노유 사람 들 뿐 아니다 강아지 돼지 소나 말 새 들 시궁창 드나들다 쥐 새끼 포식 날인 보다
빠르다 장단 꽹과리 소리 느리다 장단 둔중하다 여음 울리다 징 소리 타 작 마당 거리 멀다 최 참판 댁 사랑 흐느낌 슬프다 들려오다 농부 들 지금 꽃 달리다 고깔 흔들다 신명 내다 괴롭다 하다 일상 日常 잊다 굿 놀이 열중 있다 것 최 참판 댁 섭섭찮 전곡 錢穀 이 나가다 풍년 미치다 못 하다 실하다 평작 임 틀림 없다 것 모처럼 허리 끈 풀다 쌀밥 식구 들 배 두드리다 테 하루 근심 잊다 만 하다 것
이 날 수 수 개비 꺾다 아이 들 매 맞다 않다 여러 달 솟다 증 素症 풀다 느긋하다 늙은이 들 뒷간 출입 잦다 힘 좋다 젊은이 들 벌써 읍내 가다 없다 황소 하다 마리 끌 돌아오다 꿈 꾸미다 읍내 씨름판 몰리다 간 것
최 참판 댁 사랑 무인 지경 적막하다 햇빛 맑다 뜰 비치다 사람 들 모두 어디 가버리다 새 롭다 바르다 방문 장지 낯설다

문장을 벡터로 변환

In [11]:
```
model=word2vec.Word2Vec.load(
    "C:/Users/datam_000/Documents/Python/Module04/Ch02/toji.model")
model.most_similar("땅") # 입력한 단어와 유사한 단어
```

```
C:₩ProgramData₩Anaconda3₩envs₩pylecture₩lib₩site-packages₩smart_open₩smart_open_lib.py:398: UserWarning: This function is deprecated, use smart_open.open instead. See the migration notes for details: https://github.com/RaRe-Technologies/smart_open/blob/master/README.rst#migrating-to-the-new-open-function
  'See the migration notes for details: %s' % _MIGRATION_NOTES_URL
C:₩ProgramData₩Anaconda3₩envs₩pylecture₩lib₩site-packages₩ipykernel_launcher.py:3: DeprecationWarning: Call to deprecated `most_similar` (Method will be removed in 4.0.0, use self.wv.most_similar() instead).
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[11]:
```
[('각', 0.9285707473754883),
 ('꾼', 0.916225016117096),
 ('봇짐', 0.8953264355659485),
 ('조상', 0.8933743238449097),
 ('기틀', 0.8869537115097046),
 ('총', 0.8847390413284302),
 ('작정', 0.8830253481864929),
 ('백', 0.881824970245361 3),
 ('벼슬길', 0.8799862861633301),
 ('대가', 0.8788503408432007)]
```

문장을 벡터로 변환

In [12]:
```
model.similarity('땅', '조상') # 두 단어 사이의 유사도
```

```
C:₩ProgramData₩Anaconda3₩envs₩pylecture₩lib₩site-packages₩ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `similarity` (Method will be removed in 4.0.0, use self.wv.similarity() instead).
  """Entry point for launching an IPython kernel.
```

Out[12]:
```
0.89337426
```

In [13]:
```
# (땅+조상-젊은이 ) 선형결과
model.most_similar(positive=["땅", "조상"], negative=["젊은이"], topn=2)
```

```
C:₩ProgramData₩Anaconda3₩envs₩pylecture₩lib₩site-packages₩ipykernel_launcher.py:2: DeprecationWarning: Call to deprecated `most_similar` (Method will be removed in 4.0.0, use self.wv.most_similar() instead).
```

Out[13]:
```
[('송장', 0.8368602395057678), ('각', 0.8294554948806763)]
```

베이즈 정리로 텍스트 분류하기

❖ 베이지안 필터(Basian filter)

- 베이즈 정리를 이용한 텍스트 분류 방법으로 지도학습
  - 확률에 기반한 classification 알고리즘
- 나이브 베이즈 분류(Naïve bayes classfier)알고리즘을 사용
- 학습을 많이 시키면 시킬수록 필터의 분류 능력이 상승
- 메일 서비스의 스팸 메일을 구분하거나 커뮤니티 사이트에서 스팸 글을 구분할 때 사용
  - 이전의 스팸 메일은 키워드 입력을 통하여 구분

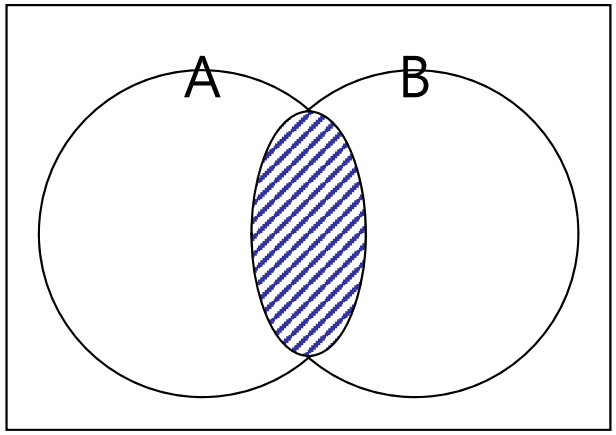
베이즈 정리로 텍스트 분류하기

❖ 조건부 확률

- 어떤 한 사상 A가 이 발생하였다는 조건에서 다른 사상 B가 발생할 확률

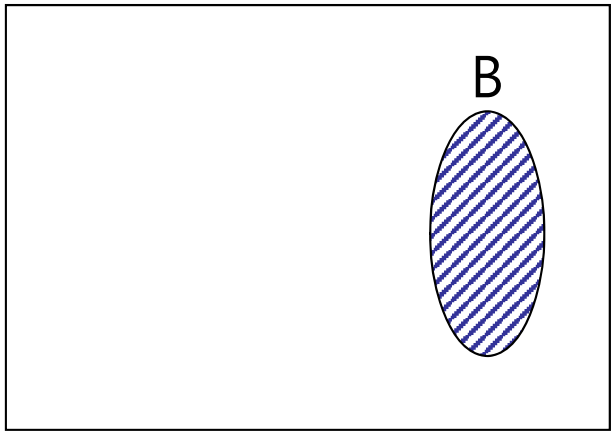
$P(A \cap B)$

S



$P(B|A)$

A



$$P(B|A) = \frac{P(A \cap B)/n(S)}{n(A)/n(S)} = \frac{P(A \cap B)}{P(A)}$$

- 조건부 확률의 성질

$$P(\emptyset|A) = 0$$

$$P(B^c|A) = 1 - P(B|A)$$

$$P[(A \cup B)|C] = P(A|C) + P(B|C) - P[(A \cap B)|C]$$

베이즈 정리로 텍스트 분류하기

❖ 예제

- 어느 고등학교 학생 100명을 대상으로 헌혈을 한 경험이 있는지 조사하였더니 다음 표와 같았다. 이 중에서 여학생 한 명을 선택했을 때, 이 학생이 헌혈을 한 적이 있는 학생일 확률은?
  - A : 선택된 학생이 여학생 사상
  - B : 선택된 학생이 헌혈한 적이 있을 사상
  - A∩B : 선택된 학생이 여학생이고 헌혈한 적이 있을 사상

성별 \ 헌혈	헌혈을 한 적이 있다	헌혈을 한 적이 없다	합계
남학생	32	20	52
여학생	25	23	48
합계	57	43	100

$$P(A) = \frac{n(A)}{n(S)} = \frac{48}{100} = 0.48$$

$$P(A \cap B) = \frac{n(A \cap B)}{n(S)} = \frac{25}{100} = 0.25$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{0.25}{0.48} = 0.52$$

베이즈 정리로 텍스트 분류하기

❖ 조건부 확률의 곱셈법칙

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A \cap B) = P(A)P(B|A)$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(A \cap B) = P(B)P(A|B)$$

$$P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$$

❖ 베이즈 정리

- 영국 철학자 Thomas Bayes의 이름에서 유래된 정리로 사건 A의 확률을 새로운 정보 B에 의해 Update하는 방법
  - 즉, Bayes Theorem은 Pr(A|B)을 구하는데 유용한 정리
  - Update하기 전 사건 A 확률[Pr(A)]을 사전적 확률(Prior Probability)
  - Update 한 후에 구해진 확률[Pr(A|B)]을 사후적 확률(Posterior Probability)

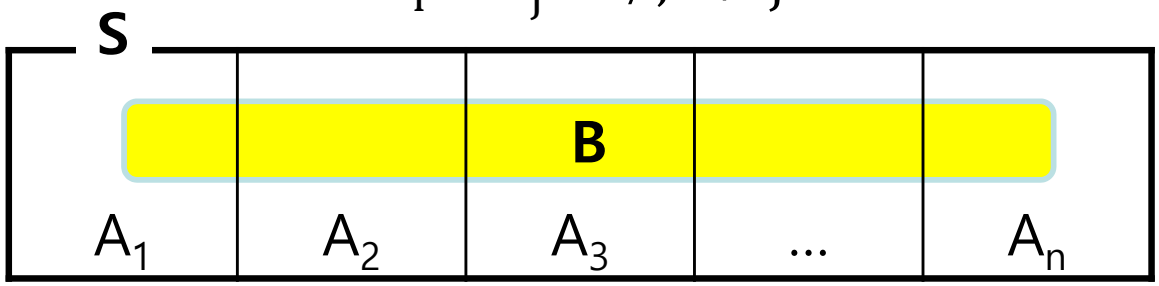
베이즈 정리로 텍스트 분류하기

- 표본공간 S에 대해서 다음 조건을 만족시키는 사상 $A_1, A_2, \ldots, A_n$을 S의 분할

$$A_i \neq \phi$$

$$S = \bigcup_{i=1}^{n} A_i$$

$$A_i \cap A_j = \emptyset, i \neq j$$



- $P(B|A_k)$뿐만 아니라 $P(A_k|B)$과 같은 조건부 확률에 관심

$$P(A_k|B) = \frac{P(A_k \cap B)}{P(B)}$$

  - 여기에서 $P(B)$는 $P(A_1 \cap B)$, $P(A_2 \cap B)$, ..., $P(A_n \cap B)$의 합사상

$$P(B) = P(A_1 \cap B) + P(A_2 \cap B) + \cdots + P(A_n \cap B)$$

  - 또한 $P(A_i \cap B)$는 조건부 확률의 곱셈법칙을 이용

$$P(A_i \cap B) = P(A_i)P(B|A_i)$$

베이즈 정리로 텍스트 분류하기

- 전확률 공식

$$P(B) = \sum_{i=1}^{n} P(A_i)P(B|A_i)$$

- 베이즈 정리

$$P(A_k|B) = \frac{P(A_k \cap B)}{P(B)} = \frac{P(A_k)P(B|A_k)}{\sum_{i=1}^{n} P(A_i)P(B|A_i)}$$

베이즈 정리로 텍스트 분류하기

❖ 예제

- 두 개의 항아리가 있다.
  - 항아리1 : 노란 구슬 2개와 빨간 구슬 4개
  - 항아리2 : 노란 구슬 3개와 빨간 구슬 5개
- 항아리2에서 하나의 구슬을 꺼내어 항아리1에 넣고 항아리1에서 하나의 구슬을 꺼낼 때 노란 구슬이 나올 확률은?
  - Y : 항아리1에서 꺼낸 구슬이 노란색일 사상
  - A : 항아리2에서 꺼낸 구슬이 노란색일 사상
  - B : 항아리2에서 꺼낸 구슬이 빨간색일 사상

$$P(A) = \frac{3}{8} \qquad P(B) = \frac{5}{8}$$

베이즈 정리로 텍스트 분류하기

- 노란 구슬을 첫 번째 항아리에 넣고 첫 번째 항아리에서 노란 구슬이 나올 확률

$$P(Y|A) = \frac{3}{7}$$

- 빨간 구슬을 첫 번째 항아리에 넣고 첫 번째 항아리에서 노란 구슬이 나올 확률

$$P(Y|B) = \frac{2}{7}$$

- 따라서 전확률의 법칙에 의해 첫 번째 항아리에서 노랑 구슬을 뽑을 확률

$$P(Y) = P(A)P(Y|A) + P(B)P(Y|B)$$
$$= \frac{3}{8} \times \frac{3}{7} + \frac{5}{8} \times \frac{2}{7} = \frac{19}{56}$$

베이즈 정리로 텍스트 분류하기

❖ 나이브 베이즈안 분류법

- 나이브 베이지안(Naive Bayesian) 분류법이란 속성변수들과 범주변수가 확률분포를 따른다고 간주하여 베이즈 정리와 조건부 독립성을 활용한 분류기법
- 속성변수들이 범주형일 때 주로 사용되나, 연속형인 경우에도 확률분포의 형태를 가정하여 사용 가능
- 어떤 객체 x에 대한 범주를 Y
  - x가 주어질 때 Y의 조건부 확률분포는 베이즈 정리에 의해($x=(x_1,x_2,\ldots,x_p)^T$)

$$f(y|\mathbf{x}) \propto f(y)f(\mathbf{x}|y) \quad , \quad y = 1,2,\cdots,J$$

    - 여기서, f(y)는 Y의 사전분포(prior),
      f(y|x)는 x의 관측 후 사후분포(posterior)
- 나이브 베이지안 분류법은 이 사후분포를 산출하여 가장 큰 값을 갖는 범주를 객체 x에 부여하는 것
- 사후분포를 정확한 계산을 위해서 비례상수가 필요하나 범주를 부여하기 위해서는 오른쪽의 두 항을 산출하면 가능(모든 범주 Y에 대해 P(x)는 공통)

베이즈 정리로 텍스트 분류하기

- f(x|y)는 속성변수들의 조건부 결합확률분포
  - 나이브 베이지안에서는 조건부 독립성을 이용하여 이를 평가
  - 확률변수 X_1, X_2, Y에 대하여 다음이 성립하면 Y가 주어질 때 X_1과 X_2 가 조건부 독립

$$P(X_1 = x_1 | X_2, Y) = P(X_1 = x_1 | Y)$$

- 따라서 속성변수들의 범주가 주어질 때(동일한 범주 내에서) 서로 독립이라 가정하면

$$f(y|\mathbf{x}) \propto f(y) \prod_{i=1}^{p} f_{X_i}(x_i|y)$$

  - 사후확률(또는 비례항)을 평가하기 위해 학습표본으로부터 Y의 사전확률 및 각 속성변수별 분포를 추정
- Y의 사전확률 및 각 속성변수별 분포
  - f(y)=범주 y의 상대도수, y=1,2,...,J
  - $f_{xi}(x_i|y)$=범주 y인 객체 중에서 $X_i=x_i$인 객체들의 비율
- 사후확률이 산출된 후에는 가장 큰 값을 갖는 범주를 선택하여 객체 x에 부여

베이즈 정리로 텍스트 분류하기

❖ 예제

- 9명의 고객에 대한 학습표본이 있다. 나이브 베이지안 분류법을 적용하여 분류하시오.

고객	1	2	3	4	5	6	7	8	9
성별 (X_1)	남	남	남	남	여	여	여	여	여
나이 (X_2)	20대	20대	30대	40대	10대	20대	20대	30대	40대
범주 (Y)	구매	비구매	구매	구매	구매	비구매	구매	비구매	비구매

- Y의 사전분포

$$f(\text{구매}) = \frac{5}{9} \quad , \quad f(\text{비구매}) = \frac{4}{9}$$

베이즈 정리로 텍스트 분류하기

- 변수별 조건부 확률분포

$$f(\text{남자}|\text{구매}) = \frac{3}{5} \quad , \quad f(\text{여자}|\text{구매}) = \frac{2}{5}$$

$$f(\text{남자}|\text{비구매}) = \frac{1}{4} \quad , \quad f(\text{여자}|\text{비구매}) = \frac{3}{4}$$

$$f(10\text{대}|\text{구매}) = \frac{1}{5} \quad , \quad f(20\text{대}|\text{구매}) = \frac{2}{5}$$

$$f(30\text{대}|\text{구매}) = \frac{1}{5} \quad , \quad f(40\text{대}|\text{구매}) = \frac{1}{5}$$

$$f(10\text{대}|\text{비구매}) = 0 \quad , \quad f(20\text{대}|\text{비구매}) = \frac{2}{4}$$

$$f(30\text{대}|\text{비구매}) = \frac{1}{4} \quad , \quad f(40\text{대}|\text{비구매}) = \frac{1}{4}$$

베이즈 정리로 텍스트 분류하기

- 고객 1의 범주별 사후확률

$$f(\text{구매}|X_1 = \text{남자}, X_2 = 20\text{대}) = f(\text{구매})f(\text{남자}|\text{구매})f(20\text{대}|\text{구매})$$
$$= \frac{5}{9} \times \frac{3}{5} \times \frac{2}{5} = \frac{2}{15}$$

$$f(\text{비구매}|X_1 = \text{남자}, X_2 = 20\text{대}) = f(\text{비구매})f(\text{남자}|\text{비구매})f(20\text{대}|\text{비구매})$$
$$= \frac{4}{9} \times \frac{1}{4} \times \frac{2}{4} = \frac{1}{18}$$

  - 구매에 대한 사후확률이 더 크므로 고객 1의 범주는 구매

- 결과

고객	1	2	3	4	5	6	7	8	9
성별(X_1)	남	남	남	남	여	여	여	여	여
나이(X_2)	20대	20대	30대	40대	10대	20대	20대	30대	40대
범주(Y)	구매	비구매	구매	구매	구매	비구매	구매	비구매	비구매
f(구매|X_1,X_2)	2/15	2/15	1/15	1/15	2/45	4/45	4/45	2/45	2/45
f(비구매|X_1,X_2)	1/18	1/18	1/36	1/36	0	1/6	1/6	1/12	1/12
결과	구매	구매	구매	구매	구매	비구매	비구매	비구매	비구매

베이즈 정리로 텍스트 분류하기

- 남자이고 10대인 새로운 고객 범주

$$f(\text{구매}|X_1 = \text{남자}, X_2 = 10\text{대}) = f(\text{구매})f(\text{남자}|\text{구매})f(10\text{대}|\text{구매})$$
$$= \frac{5}{9} \times \frac{3}{5} \times \frac{1}{5} = \frac{1}{15}$$

$$f(\text{비구매}|X_1 = \text{남자}, X_2 = 20\text{대}) = f(\text{비구매})f(\text{남자}|\text{비구매})f(20\text{대}|\text{비구매})$$
$$= \frac{4}{9} \times \frac{1}{4} \times 0 = 0$$

  - 구매에 대한 사후확률이 더 크므로 새로운 고객의 범주는 구매

베이즈 정리로 텍스트 분류하기

- 장점
  - 단순하고 빠르며 효과적
  - 노이즈와 결측 데이터가 있어도 잘 수행
  - 훈련에 대해 상대적으로 적은 예제가 필요(많은 예제에서도 잘 수행)
  - 예측에 대한 추정된 확률을 얻기 쉽움
- 단점
  - 모든 속성은 동등하게 증요하고 독립적이라는 알려진 결함(open-faulty assumption)
  - 연속형 수치 속성으로 구성된 데이터셋에 대해 이상적이지 않음(discretization 필요)
  - 추정된 확률은 예측된 범주 보다 덜 신뢰적

베이즈 정리로 텍스트 분류하기

❖ 라플라스 추정값(Laplas estimator)

$$\mathbf{x} = \left(x_1, x_2, \cdots, x_{p-1}, \text{None}\right)^T$$

- 위의 경우와 같이 훈련 데이터에 없는 경우가 테스트 케이스에 포함 되어 있다면

$$f(\text{구매}|X_1 = \text{남자}, X_2 = 50\text{대}) = f(\text{구매})f(\text{남자}|\text{구매})f(50\text{대}|\text{구매})$$
$$= \frac{5}{9} \times \frac{3}{5} \times 0 = 0$$

$$f(\text{비구매}|X_1 = \text{남자}, X_2 = 50\text{대}) = f(\text{비구매})f(\text{남자}|\text{비구매})f(50\text{대}|\text{비구매})$$
$$= \frac{4}{9} \times \frac{1}{4} \times 0 = 0$$

  - 나이를 제외한 나머지 정보를 가지고도 분류가 가능한데 0이 곱해져서 다른 정보도 쓸모 없게 됨
- 이를 방지하기 위해 확률이 0 이 되지 않도록 임의의 값(대부분 1 을 사용)을 부여하여 계산
- 라플라스 추정값은 각 속성별로 서로 다른 값을 부여할 수 도 있음
- 속성의 중요도에 따라 서로 다른 값을 부여할 수 도 있음
- 훈련 데이터셋이 충분히 크다면 라플라스 추정값은 고려하지 않아도 됨

베이즈 정리로 텍스트 분류하기

❖ 베이지안 필터 파이썬 프로그램(파이썬 파일 ".py" 로 저장)

```
import math, sys
from konlpy.tag import Twitter
class BayesianFilter:
    """ 베이지안 필터 """
    def __init__(self):
        self.words = set() # 출현한 단어 기록
        self.word_dict = {} # 카테고리마다의 출현 횟수 기록
        self.category_dict = {} # 카테고리 출현 횟수 기록
    # 형태소 분석하기 --- (※2)
    def split(self, text):
        results = []
        twitter = Twitter()
        # 단어의 기본형 사용
        malist = twitter.pos(text, norm=True, stem=True)
        for word in malist:
            # 어미/조사/구두점 등은 대상에서 제외
            if not word[1] in ["Josa", "Eomi", "Punctuation"]:
                results.append(word[0])
        return results
```

베이즈 정리로 텍스트 분류하기


```
# 단어와 카테고리의 출현 횟수 세기 --- (※2)
def inc_word(self, word, category):
    # 단어를 카테고리에 추가하기
    if not category in self.word_dict:
        self.word_dict[category] = {}
    if not word in self.word_dict[category]:
        self.word_dict[category][word] = 0
    self.word_dict[category][word] += 1
    self.words.add(word)

def inc_category(self, category):
    # 카테고리 계산하기
    if not category in self.category_dict:
        self.category_dict[category] = 0
    self.category_dict[category] += 1
```

베이즈 정리로 텍스트 분류하기


```
# 텍스트 학습하기 --- (※3)
def fit(self, text, category):
    """ 텍스트 학습 """
    word_list = self.split(text)
    for word in word_list:
        self.inc_word(word, category)
    self.inc_category(category)


# 단어 리스트에 점수 매기기--- (※4)
def score(self, words, category):
    score = math.log(self.category_prob(category))
    for word in words:
        score += math.log(self.word_prob(word, category))
    return score
```

베이즈 정리로 텍스트 분류하기


```
# 예측하기 --- (※5)
def predict(self, text):
    best_category = None
    max_score = -sys.maxsize
    words = self.split(text)
    score_list = []
    for category in self.category_dict.keys():
        score = self.score(words, category)
        score_list.append((category, score))
        if score > max_score:
            max_score = score
            best_category = category
    return best_category, score_list

# 카테고리 내부의 단어 출현 횟수 구하기
def get_word_count(self, word, category):
    if word in self.word_dict[category]:
        return self.word_dict[category][word]
    else:
        return 0
```

베이즈 정리로 텍스트 분류하기


```
# 카테고리 계산
def category_prob(self, category):
    sum_categories = sum(self.category_dict.values())
    category_v = self.category_dict[category]
    return category_v / sum_categories


# 카테고리 내부의 단어 출현 비율 계산 --- (※6)
def word_prob(self, word, category):
    n = self.get_word_count(word, category) + 1 # ---(※6a)
    d = sum(self.word_dict[category].values()) + len(self.words)
    return n / d
```

베이즈 정리로 텍스트 분류하기

In [14]:
```
# "bayes.py" 불러오기
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")

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

```
결과 = 광고
[('광고', -19.485641988358296), ('중요', -20.63806741338132)]
```

베이즈 정리로 텍스트 분류하기

In [15]:
```
# spam과 ham 메일 분류
from pandas import Series, DataFrame
from numpy import nan as NA
import pandas as pd
import numpy as np
from bayes import BayesianFilter

df=pd.read_csv("C:/Users/datam_000/Documents/Python/Module04/Ch02/sms_spam.csv",
               header=0, encoding="ansi")
df.head(7)

# 베이지안 필터 학습
bf= BayesianFilter()
for i in df.index :
    bf.fit(df.text[i],df.type[i])
```

베이즈 정리로 텍스트 분류하기

In [16]:
```
# 예측
df_test=pd.read_csv(
    "C:/Users/datam_000/Documents/Python/Module04/Ch02/sms_spam_test.csv",
                header=0, encoding="ansi")
print(df_test.head(7),"₩n")

prediced=[]
for x in df_test.text :
    pre, scorelist=bf.predict(x)
    prediced.append(pre)
    print("예측 결과 : {0}".format(pre))
    print("예측 점수₩n", scorelist, "₩n")
print(prediced,"₩n")

df_test["predict"]=prediced
print("예측률 : {0:5.1f}%".format(df_test.type.eq(df_test.predict).mean()*100))
df_test.head(7)
```

베이즈 정리로 텍스트 분류하기


```
   type                                               text
0  spam  Free msg: Single? Find a partner in your area!...
1  spam  Want to funk up ur fone with a weekly new tone...
2   ham  Even my brother is not like to speak with me. ...
3   ham  Wow. I never realized that you were so embaras...
4  spam  URGENT This is our 2nd attempt to contact U. Y...
5   ham          Yes..gauti and sehwag out of odi series.
6  spam  Free Msg: Ringtone!From: http://tms. widelive....

예측 결과 : spam
예측 점수

 [('ham', -253.26336787943671), ('spam', -219.50440600523493)]

예측 결과 : spam
예측 점수
 [('ham', -246.76792140374928), ('spam', -207.6443336201288)]
```

베이즈 정리로 텍스트 분류하기

예측률 : 98.0%

Out[16]:

	type	text	predict
0	spam	Free msg: Single? Find a partner in your area!...	spam
1	spam	Want to funk up ur fone with a weekly new tone...	spam
2	ham	Even my brother is not like to speak with me. ...	ham
3	ham	Wow. I never realized that you were so embaras...	ham
4	spam	URGENT This is our 2nd attempt to contact U. Y...	spam
5	ham	Yes..gauti and sehwag out of odi series.	ham
6	spam	Free Msg: Ringtone!From: http://tms. widelive....	spam

문장의 유사도

❖ 문장의 유사도 분석

- 두 개의 문장이 비슷한 것인지 또는 관련이 있는 것인지 분석

❖ 레벤슈타인 거리(Lvenshtein Distance)

- 레벤슈타인 거리는 두 개의 문자열이 어느 정도 다른지를 나타내는 것
- 편집 거리(Edit Distance)라고 부르기도 함
- 철자 오류 수정, 비슷한 어구 검색, 의학 분야에서 DNA 배열의 유사성 등을 판단할 때 사용
- 어떤 문자열을 비교하려는 문자열로 편집할 때 몇 번의 문자열 조작이 필요한지에 주목하여 단어의 거리를 계산

가나다라 → 가마바라

횟수	편집 조작	결과
0	-	가나다라
1	'나' 를 '마' 로 조작	가마다라
2	'다'를 '바'로 조작	가마바라

- 2회 조작으로 변경하였으므로 편집 비용은 2(레벤슈타인 거리)

문장의 유사도

In [17]:
```
# 레벤슈타인 거리 구하기
def calc_distance(a, b):
    ''' 레벤슈타인 거리 계산하기 '''
    if a == b: return 0
    a_len = len(a)
    b_len = len(b)
    if a == "": return b_len
    if b == "": return a_len

    # 2차원 표 (a_len+1, b_len+1) 준비하기 --- (※1)
    matrix = [[] for i in range(a_len+1)]
    for i in range(a_len+1): # 0으로 초기화
        matrix[i] = [0 for j in range(b_len+1)]
    # 0일 때 초깃값을 설정
    for i in range(a_len+1):
        matrix[i][0] = i
    for j in range(b_len+1):
        matrix[0][j] = j
```

문장의 유사도


```
    # 표 채우기 --- (※2)
    for i in range(1, a_len+1):
        ac = a[i-1]
        for j in range(1, b_len+1):
            bc = b[j-1]
            cost = 0 if (ac == bc) else 1
            matrix[i][j] = min([
                matrix[i-1][j] + 1,     # 문자 삽입
                matrix[i][j-1] + 1,     # 문자 제거
                matrix[i-1][j-1] + cost # 문자 변경
            ])
    return matrix[a_len][b_len]

# "가나다라"와 "가마바라"의 거리 --- (※3)
print(calc_distance("가나다라","가마바라"))

# 지하철 역 유사 정도
samples = ["신촌역","신천군","신천역","신발","마곡역"]
base = samples[0]
r = sorted(samples, key=lambda n: calc_distance(base, n))
for n in r:
    print(calc_distance(base, n), n)
```

문장의 유사도


```
2
0 신촌역
1 신천역
2 신천군
2 신발
2 마곡역
```

문장의 유사도

❖ N-gram 유사도

- N-gram은 텍스트에서 이웃한 N개의 문자를 의미
- t서로 다른 2개의 문장을 N-gram으로 비교하면 출현하는 단어의 종류와 빈도를 확인
  - 논문 도용, 라이선스가 있는 프로그램 코드의 복사여부 등 확인

문장의 유사도

In [18]:
```
# 문장을 N-gram으로 나누기
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

# 두 문장의 유사도 측정
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
```

문장의 유사도


```
a = "파이썬으로 하는 빅데이터 분석과 머신러닝은 매우 쉽습니다."
b = "빅데이터 분석과 머신러닝은 파이썬을 이용하여 매우 쉽게 할 수 있습니다."

# 2-gram 유사도

r_2gram, word_2gram = diff_ngram(a, b, 2)
print("2-gram:", r_2gram, word_2gram)

# 3-gram 유사도
r_3gram, word_3gram  = diff_ngram(a, b, 3)
print("3-gram:", r_3gram, word_3gram)
```

```
2-gram: 0.7419354838709677 ['파이', '이썬', '빅데', '데이', '이터', '터 ', ' 분', '분
석', '석과', '과 ', ' 머', '머신', '신러', '러닝', '닝은', '은 ', ' 매', '매우', '우
', ' 쉽', '습니', '니다', '다.']
3-gram: 0.6333333333333333 ['파이썬', '빅데이', '데이터', '이터 ', '터 분', ' 분석',
'분석과', '석과 ', '과 머', ' 머신', '머신러', '신러닝', '러닝은', '닝은 ', ' 매우',
'매우 ', '우 쉽', '습니다', '니다.']
```

3

요약

요약

- **Python은 정형데이터 이외에 비정형 데이터 분석을 위한 다양한 라이브러리를 제공하고 있음**
- **비정형 자료를 이용한 예측 및 분류 등을 수행 할 수 있음**
  - Python에서 비정형 자료의 분석 기법을 습득하므로서 다양한 데이터 분석 방법의 기본지식 확보
  - 비정형 자료 분석을 위한 기술 확보
    - 비정형 자료의 시각화 및 분석 기술 습득
    - 현장 문제에 적용 가능한 수준까지 프로그래밍 기술 습득