MYONGJI UNIVERSITY

Artificial Intelligence

Lecture 6. Text Mining Part 3

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail : jchun@mju.ac.kr
Lecture Note : https://lms.mju.ac.kr

Agenda

- Text Mining 개요
- 영문 텍스트 분석
- 한글 텍스트 처리 기법
- WordCloud

한글 텍스트 처리

KoNLPy

- KoNLPy (http://konlpy.org/ko/latest/)
  - 한국어 정보처리를 위한 Python Package
  - 다양한 종류의 한글 형태소 분석기 지원(Hannanum, Twitter, Kkma, Komoran, Mecab 등)
  - "코엔엘파이"라고 읽음

- 설치 절차
  - JDK 설치
  - JPype1 설치
    - 64bit OS의 경우 다음 사이트에서 JPype1-0.6.2-cp36-cp36m-win_amd64.whl 을 다운로드 한 후에 설치
    - https://www.lfd.uci.edu/~gohlke/pythonlibs/#jpype
    - **pip install JPype1-0.6.2-cp36-cp36m-win_amd64.whl**
  - Konlpy 설치
    - pip install konlpy

한글 영화평 데이터

- ratings_train.txt, ratings_test.txt
  - Train text data와 test text data 2개의 파일로 구성
  - id, document, label 3개의 컬럼으로 구성

```
id      document        label
6270596 굳 ㅋ     1
9274899 GDNTOPCLASSINTHECLUB    0
8544678 뭐야 이 평점들은.... 나쁘진 않지만 10점 짜리는 더더욱 아니잖아        0
6825595 지루하지는 않은데 완전 막장임... 돈주고 보기에는....        0
6723715 3D만 아니었어도 별 다섯 개 줬을텐데.. 왜 3D로 나와서 제 심기를 불편하게 하죠??   0
7898805 음악이 주가 된, 최고의 음악영화 1
6315043 진정한 쓰레기        0
6097171 마치 미국애니에서 튀어나온듯한 창의력없는 로봇디자인부터가,고개를 젖게한다  0
8932678 갈수록 개판되가는 중국영화 유치하고 내용없음 폼잡다 끝남 말도안되는 무기에 유치한cg남무 아 그립다 동사서독같은 영화가 이건 3류아류작이다  0
6242223 이별의 아픔뒤에 찾아오는 새로운 인연의 기쁨 But, 모든 사람이 그렇지는 않네..      1
7462111 괜찮네요오랜만포켓몬스터잼밌어요 1
8425305 한국독립영화의 한계 그렇게 아버지가 된다와 비교됨     0
6900881 청춘은 아름답다 그 아름다움은 이성을 흔들어 놓는다. 찰나의 아름다움을 잘 포착한 섬세하고 아름다운 수채화같은 퀴어영화이다.     1
9629375 눈에 보이는 반전이었지만 영화의 흡인력은 사라지지 않았다.      1
9037756 """스토리, 연출, 연기, 비주얼 등 영화의 기본 조차 안된 영화에 무슨 평을 해. 이런 영화 찍고도 김문옥 감독은 """"내가 영화 경력이 몇00인데 조무래기들이 내 영화를 평론해?"""" 같은 마인드에
빠져있겠지?"""      0
10268521        소위 조문가라는 평점은 뭐냐?   1
2968565 최고!!!!!!!!!!!!!!!!      1
10107726        발연기 도저히 못보겠다 진짜 이렇게 연기를 못할거라곤 상상도 못했네        0
6406912 나이스    1
4174028 별 재미도없는거 우려먹어 .... 챔프에서 방송 몇번했더라 ? ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ    0
9305768 '13일의 금요일', '나이트메어'시리즈와 함께 가장 많은 시리즈를 양산해냈던 헬레이저 시리즈의 첫편. 작가의 상상력이 돋보이는 작품이며, 갈고리로 사지찢는 고어씬은 지금보더라도 상당히 잔인하고
충격적이다.        1
6829990 나름 교훈돋는 하지만 어쩔수없이 저평점 받을수밖에 없는 저질섹스코미디    0
9727606 꽤 재밌게 본 영화였다!        1
1385942 민주화 시대의 억눌린 영혼의 관음적인 욕구 분출.인상적이다.     1
9504429 일본 천황이 미국으로부터 받은 면죄부의 긴박한 과정을 루즈하고 지저분하게 늘어놓았다. 0
9952932 괜히 나올 때 어미 알 건드려서 긁어 부스름 만들었다는 분이 저 아래 보이던데, 영화 제대로 안 봤네. 알이 딱 까지면서 새끼가 나오려 했음. 그냥 가면 그 놈 한테 당했을 것임. 한 놈, 두 놈 막 나올 게
뻔했으니 작살낼 수 밖에 없었다. 1
```

데이터 확인하기

```
import konlpy
import pandas as pd
import numpy as np

# train data와 test data 확인
df_train = pd.read_csv('data/ratings_train.txt', delimiter = '\t', keep_default_na = False)
df_test = pd.read_csv('data/ratings_test.txt', delimiter = '\t', keep_default_na = False)

print(df_train.head(n = 5), '\n')
print(df_test.head(n = 5))
```

구분자는 tab으로 구별함

빈 문자열을 NaN으로 저장하지 않고 빈 문자열 자체로 저장

```
        id                                           document  label
0  9976970                                아 더빙.. 진짜 짜증나네요 목소리      0
1  3819312                       흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나      1
2  10265843                                  너무재밋었다그래서보는것을추천한다      0
3  9045019                      교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정      0
4  6483659  사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...      1

        id                                           document  label
0  6270596                                              굳 ㅋ      1
1  9274899                                GDNTOPCLASSINTHECLUB      0
2  8544678                 뭐야 이 평점들은.... 나쁘진 않지만 10점 짜리는 더더욱 아니잖아      0
3  6825595                   지루하지는 않은데 완전 막장임... 돈주고 보기에는....      0
4  6723715  3D만 아니었어도 별 다섯 개 줬을텐데.. 왜 3D로 나와서 제 심기를 불편하게 하죠??      0
```

영화평 문자열과 평점(label) 확인하기

```
# 학습 데이터의 문자열(document)과 평점 정보(label)를 각각 추출하여 크기 확인
text_train = df_train['document']
y_train = df_train['label']

text_test = df_test['document']
y_test = df_test['label']

# np.bincount: 각 bin의 element를 count, 즉 distinct counting
#  label만 추출한 y_train, y_test의 0의 갯수, 1의 갯수를 추출하여 출력
print(len(text_train), np.bincount(y_train))
print(len(text_test), np.bincount(y_test))
```

```
150000 [75173 74827]
50000 [24827 25173]
```

Okt 사용하기

- Okt (Open Korean Text)
  - an open source Korean tokenizer written in Scala
  - Formerly known as Twitter

- morphs(*phrase*, *norm=False*, *stem=False*)
  - Parse phrase to morphemes.
- nouns(*phrase*): Noun extractor
- phrases(*phrase*): Phrase extractor
- pos(*phrase*, *norm=False*, *stem=False*, *join=False*)
  - POS(Part-of-speech) tagger
  - **norm** -- If True, normalize tokens
  - **stem** -- If True, stem tokens
  - **join** -- If True, returns joined sets of morph and tag

Okt 사용 예

```
from konlpy.tag import Okt
twitter = Okt()

print(twitter.morphs(u'단독입찰보다 복수입찰의 경우'))
print(twitter.nouns(u'유일하게 항공기 체계 종합개발 경험을 갖고 있는 KAI는'))
print(twitter.phrases(u'날카로운 분석과 신뢰감 있는 진행으로'))

print(twitter.pos(u'이것도 되나욬ㅋㅋ'))
print(twitter.pos(u'이것도 되나욬ㅋㅋ', norm = 'True'))
print(twitter.pos(u'이것도 되나욬ㅋㅋ', norm = 'True', stem = 'True'))
```

```
['단독', '입찰', '보다', '복수', '입찰', '의', '경우']
['항공기', '체계', '종합', '개발', '경험']
['날카로운 분석', '날카로운 분석과 신뢰감', '날카로운 분석과 신뢰감 있는 진행', '분석', '신뢰', '진행']
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나욬', 'Noun'), ('ㅋㅋ', 'KoreanParticle')]
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나요', 'Verb'), ('ㅋㅋ', 'KoreanParticle')]
[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되다', 'Verb'), ('ㅋㅋ', 'KoreanParticle')]
```

Okt 형태소 분석기 사용

```
# 형태소 분석기 객체 생성
from konlpy.tag import Okt
twitter_tag = Okt()

# Okt morphs를 사용한 단순한 tokenizer 함수
def twitter_tokenizer(text):
    return twitter_tag.morphs(text)

# tokenizer 개선
# pos tagger의 normalization과 stemming 활용
def twitter_tokenizer_filter(text):
    malist = twitter_tag.pos(text, norm = True, stem = True)
    r = []
    for word in malist:
        if not word[1] in ["Josa", "Eomi", "Punctuation", "KoreanParticle"]:
            r.append(word[0])
    return r

print(twitter_tokenizer(u'유일하게 항공기 체계 종합개발 경험을 갖고 있는 KAI는'))
print(twitter_tokenizer_filter(u'유일하게 항공기 체계 종합개발 경험을 갖고 있는 KAI는'))
print(twitter_tokenizer_filter(u'이것도 되나욬ㅋㅋ'))
```

```
['유일하게', '항공기', '체계', '종합', '개발', '경험', '을', '갖고', '있는', 'KAI', '는']
['유일하다', '항공기', '체계', '종합', '개발', '경험', '갖다', '있다', 'KAI', '늘다']
['이', '것', '되다']
```

Naïve Bayesian Classifier

```
import konlpy
import pandas as pd
import numpy as np

from sklearn.feature_extraction.text import CountVectorizer
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB

from konlpy.tag import Okt
twitter_tag = Okt()

def twitter_tokenizer(text):
    malist = twitter_tag.pos(text, norm = True, stem = True)
    r = []
    for word in malist:
        if not word[1] in ["Josa", "Eomi", "Punctuation", "KoreanParticle"]:
            r.append(word[0])
    return r

df_train = pd.read_csv('data/ratings_train.txt', delimiter = '\t', keep_default_na = False)
df_test = pd.read_csv('data/ratings_test.txt', delimiter = '\t', keep_default_na = False)
```

Multinomial Naïve Bayesian 사용

```
# 학습 데이터의 문자열(document)과 평점 정보(label)를 각각 추출
text_train = df_train['document']
y_train = df_train['label']

text_test = df_test['document']
y_test = df_test['label']

vect = CountVectorizer(tokenizer = twitter_tokenizer).fit(text_train)

X_train = vect.transform(text_train)
clf_mult = MultinomialNB().fit(X_train, y_train)

X_test = vect.transform(text_test)

# 데이터 예측하기
pre = clf_mult.predict(X_test)

# 정답률 산출
ac_score = metrics.accuracy_score(y_test, pre)
print("정답률 = ", ac_score)
```

```
정답률 =  0.8402
```

WORDCLOUD

WordCloud

- 설치
  - Pip install wordcloud

WordCloud 예제 1

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

text = open('speech.txt', encoding = 'ISO8859').read()

# 자동으로 text의 term들을 추출하여 상대적인 출현 빈도수를 계산하고 array 형태로 이미지를 생성
wordcloud = WordCloud().generate(text)
print(type(wordcloud))

# wordcloud.words_에 'dict' type으로 빈도수를 저장
print(type(wordcloud.words_))
print(wordcloud.words_)
```

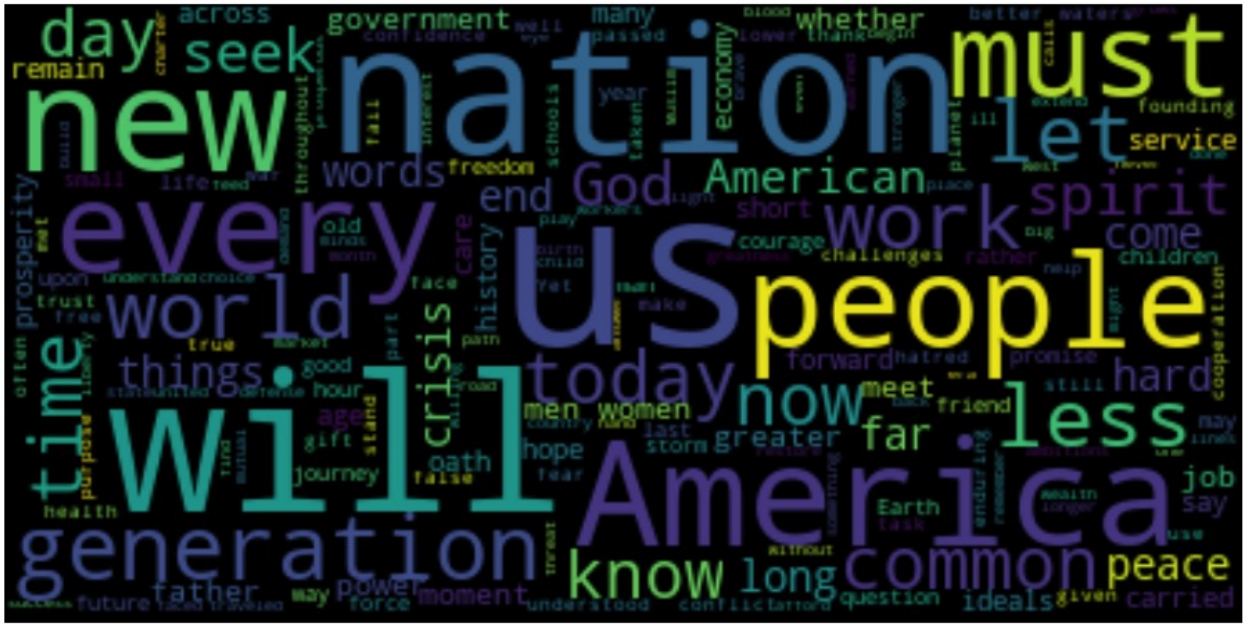
```
<class 'wordcloud.wordcloud.WordCloud'>
<class 'dict'>
{'us': 1.0, 'will': 0.782608695652174, 'nation': 0.6521739130434783, 'new': 0.4782608695652174, 'America': 0.43478260
869565216, 'every': 0.34782608695652173, 'people': 0.34782608695652173, 'must': 0.34782608695652173, 'generation': 0.
34782608695652173, 'less': 0.30434782608695654, 'work': 0.30434782608695654, 'world': 0.30434782608695654, 'let': 0.3
0434782608695654, 'today': 0.2608695652173913, 'now': 0.2608695652173913, 'time': 0.2608695652173913, 'common': 0.260
8695652173913, 'day': 0.21739130434782608, 'know': 0.21739130434782608, 'spirit': 0.21739130434782608, 'God': 0.21739
130434782608, 'seek': 0.21739130434782608, 'American': 0.21739130434782608, 'words': 0.17391304347826086, 'peace': 0.
17391304347826086, 'crisis': 0.17391304347826086, 'far': 0.17391304347826086, 'hard': 0.17391304347826086, 'come': 0.
17391304347826086, 'end': 0.17391304347826086, 'long': 0.17391304347826086, 'things': 0.17391304347826086, 'greater':
0.17391304347826086, 'meet': 0.17391304347826086, 'whether': 0.17391304347826086, 'government': 0.17391304347826086,
'power': 0.17391304347826086, 'father': 0.17391304347826086, 'moment': 0.17391304347826086, 'job': 0.1739130434782608
6, 'men women': 0.17391304347826086, 'service': 0.13043478260869565, 'oath': 0.13043478260869565, 'prosperity': 0.130
43478260869565, 'carried': 0.13043478260869565, 'ideals': 0.13043478260869565, 'economy': 0.13043478260869565, 'age':
```

WordCloud 예제 1

```
# Display the generated image
# 이미지 생성
# WordCloud 객체의 option은 https://amueller.github.io/word_cloud/
# max_font_size: Maximum font size for the largest word
wordcloud = WordCloud(max_font_size = 70).generate(text)

# figure size 설정
plt.figure(figsize = (16, 9))

# plt.imshow: nparray를 이미지로 표현하는 method
# interpolation: pixel간의 경계를 얼마나 부드럽게 하느냐를 나타냄
# https://matplotlib.org/devdocs/gallery/images_contours_and_fields/interpolation_methods.html 참고
plt.imshow(wordcloud, interpolation = "bilinear")
plt.axis("off")
plt.show()
```



WordCloud 예제 2

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS

text = open('speech.txt',encoding = 'ISO8859').read()

# 그림 모양을 나타내는 이미지 지정
alice_mask = np.array(Image.open("alice.jpg"))

# Background 이미지, stopword 등 지정
wc = WordCloud(background_color = "white", max_words = 200, mask = alice_mask, stopwords = STOPWORDS)
wc.generate(text)

# show
plt.figure(figsize = (10, 10)) # 이미지 크기 지정(inch)
plt.imshow(wc, interpolation = 'bilinear')
plt.axis("off")

plt.show()
```

한글 WordCloud 예제 1

```
import konlpy
from konlpy.tag import Okt
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from collections import Counter

text = open('moon.txt', encoding = 'utf-8').read()

t = Okt()
tokens = t.nouns(text) # 명사 추출
count = Counter(tokens) # 추출된 명사의 빈도수 계산
tags = count.most_common(80) # 빈도수 높은 순서로 80개만 추출
print(type(tags))

dict_tags = dict(tags) # dict type으로 변환
print(dict_tags)

# wordcloud 이미지 생성
# 한글 폰트 지정이 필요
# 이미 빈도수를 계산해 놓은 상태이므로 generate_from_frequencies를 사용
wc = WordCloud(font_path = "/Library/Fonts/AppleGothic.ttf",
               background_color = "white",
               max_font_size = 40,
               ).generate_from_frequencies(dict_tags)

plt.figure(figsize = (10, 10))
plt.imshow(wc, interpolation = "bilinear")
plt.axis("off")
plt.show()
```

한글 WordCloud 예제 1

```
<class 'list'>
{'국민': 18, '우리': 17, '것': 14, '수': 11, '새해': 6, '해': 6, '대한민국': 6, '더': 6, '정부': 6, '평화': 6, '일': 5, '작년': 4, '저': 4, '모든': 4, '나라': 4, '대화': 4, '노력': 4, '북한': 4, '여러분': 3, '대표': 3, '분': 3, '해결': 3, '지난해': 3, '혁명': 3, '세계': 3, '위': 3, '대통령': 3, '최선': 3, '뜻': 3, '사회': 3, '평창올림픽': 3, '비롯': 2, '감사': 2, '또': 2, '환영': 2, '박수': 2, '촛불': 2, '국가': 2, '정상': 2, '회담': 2, '때': 2, '수출': 2, '여러': 2, '가지': 2, '지난': 2, '내': 2, '올해': 2, '해소': 2, '경쟁': 2, '기원': 2, '삶': 2, '소망': 2, '하나': 2, '한반도': 2, '남북': 2, '안전': 2, '무술년': 2, '복': 1, '오늘': 1, '부요': 1, '각계': 1, '각층': 1, '원로': 1, '또한': 1, '기억': 1, '수능': 1, '지진': 1, '때문': 1, '어려움': 1, '포항': 1, '여고': 1, '학생': 1, '정규직': 1, '꿈': 1, '비정규직': 1, '노동자': 1, '장기': 1, '미제': 1, '사건': 1, '경찰관': 1}
```

한글 WordCloud 예제 2

```
from konlpy.tag import Okt
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
import matplotlib.pyplot as plt
from collections import Counter
from PIL import Image
import numpy as np

text = open('moon.txt', encoding = 'utf-8').read()

korea_mask = np.array(Image.open("korea_mask.jpg"))
# Korea_mask 이미지에 있는 color만을 사용
colors = ImageColorGenerator(korea_mask)

t = Okt()
tokens = t.nouns(text) # 명사 추출
count = Counter(tokens) # 추출된 명사의 빈도수 계산
tags = count.most_common(80) # 빈도수 높은 순서로 80개만 추출
print(type(tags))

dict_tags = dict(tags) # dict type으로 변환
print(dict_tags)

# wordcloud 이미지 생성
# 한글 폰트 지정이 필요
# 이미 빈도수를 계산해 놓은 상태이므로 generate_from_frequencies를 사용
wc = WordCloud(font_path = "/Library/Fonts/AppleGothic.ttf",
               background_color = "white",
               max_font_size = 40,
               mask = korea_mask,
               color_func = colors, # color 함수 지정
               ).generate_from_frequencies(dict_tags)

plt.figure(figsize = (10, 10))
plt.imshow(wc, interpolation = "bilinear")
plt.axis("off")
plt.show()
```

한글 WordCloud 예제 2

```
<class 'list'>
{'국민': 18, '우리': 17, '것': 14, '수': 11, '새해': 6, '해': 6, '대한민국': 6, '더': 6, '정부': 6, '평화': 6, '일': 5, '작년': 4, '저': 4, '모든': 4, '나라': 4, '대화': 4, '노력': 4, '북한': 4, '여러분': 3, '대표': 3, '분': 3, '해결': 3, '지난해': 3, '혁명': 3, '세계': 3, '위': 3, '대통령': 3, '최선': 3, '뜻': 3, '사회': 3, '평창올림픽': 3, '비롯': 2, '감사': 2, '또': 2, '환영': 2, '박수': 2, '촛불': 2, '국가': 2, '정상': 2, '회담': 2, '때': 2, '수출': 2, '여러': 2, '가지': 2, '지난': 2, '내': 2, '올해': 2, '해소': 2, '경쟁': 2, '기원': 2, '삶': 2, '소망': 2, '하나': 2, '한반도': 2, '남북': 2, '안전': 2, '무술년': 2, '복': 1, '오늘': 1, '부요': 1, '각계': 1, '각층': 1, '원로': 1, '또한': 1, '기억': 1, '수능': 1, '지진': 1, '때문': 1, '어려움': 1, '포항': 1, '여고': 1, '학생': 1, '정규직': 1, '꿈': 1, '비정규직': 1, '노동자': 1, '장기': 1, '미제': 1, '사건': 1, '경찰관': 1}
```

END