

형태소 분석 실습

Natural Language Processing & AI Lab.,
Korea University



KOREA
UNIVERSITY



Natural Language
Processing
& Artificial Intelligence



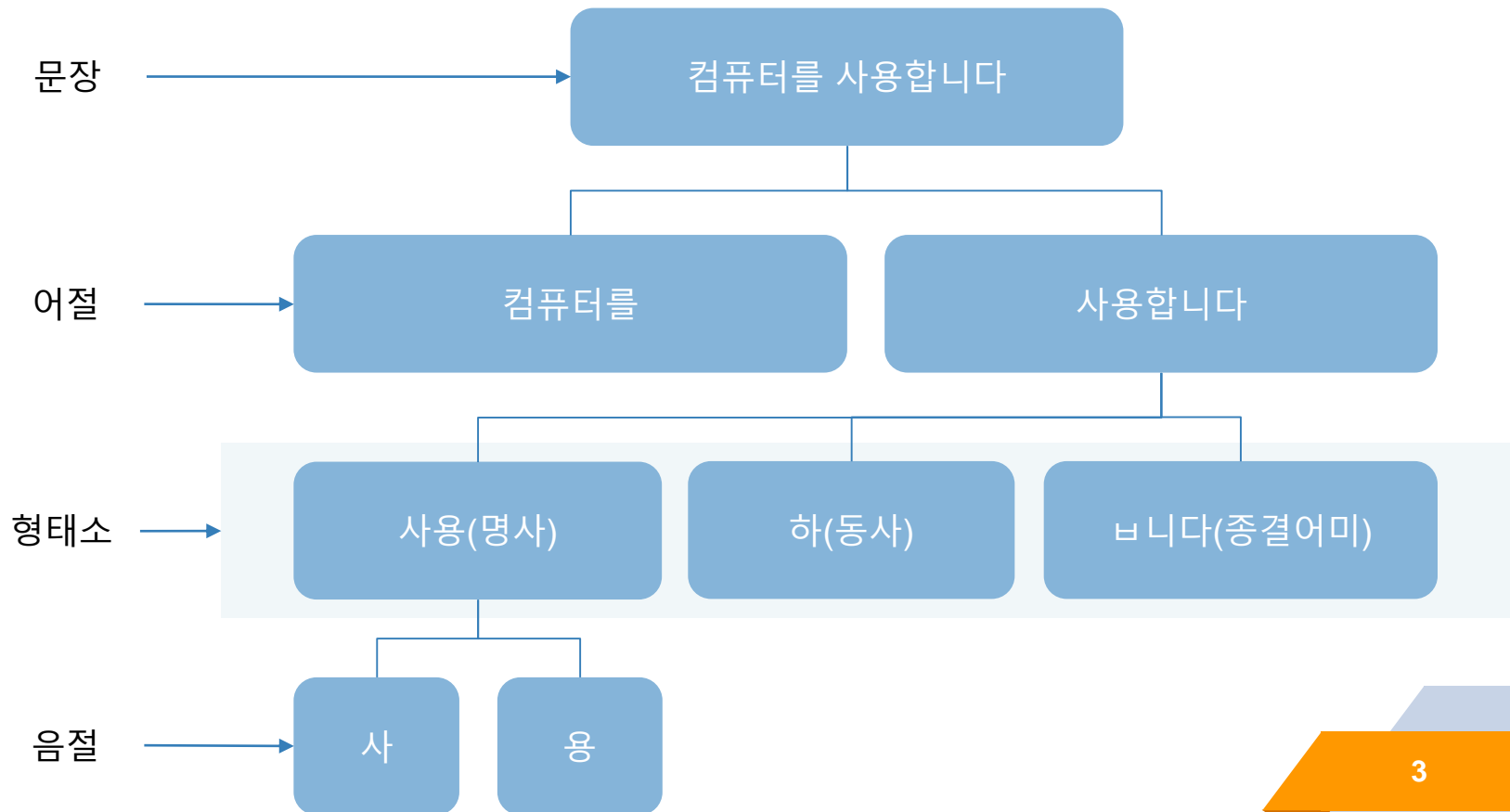
자연어 처리

- ❑ 텍스트 데이터는 비정형데이터로 숫자나 통계 데이터와 같이 정형화 되어 있지 않기 때문에 분석 과정이 매우 복잡
- ❑ 다양한 표현방식을 사용하는 인간 언어를 분석할 수 있도록 해야함
- ❑ 자연어처리는 이러한 인간 언어를 해석하는 기술



형태소 분석

- 형태소란 의미를 가지는 최소 단위로서 더 이상 의미적으로 분석이 불가능한 단위.





형태소 분석

- 다양한 언어표현의 중의성을 해소.
 - (사용/NNG+하/XSV+ㄴ 다/EFN) (사용/NNG+하/XSV+ㅂ니다/EFN)
 - 사용/NNG+하/XSV 형태소 단위로 쪼개면 '사용한다', '사용합니다'의 의미를 같은 태그로써 묶을 수 있다.



형태소 분석

□ 형태소 분석의 한계점

- 신조어, 은어, 오탈자, 띄어쓰기 오류들이 형태소 분석을 하는데 큰 영향을 미침.
- 다의어에 대해서 모호성을 해결하지 못함.
 - 밝이 어둡다, 밝이 맛있다



KoNLPy

- ❑ 한국어 자연어처리를 위한 대표적인 파이썬 라이브러리
- ❑ Twitter, Komoran, Hannanum, Twitter, Mecab 등 5개의 클래스가 있다.



□ 성능 비교 - 띄어쓰기

- “아버지가방에들어가신다”

Hannanum	Kkma	Komoran	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시ㄴ다 / E	에 / JKM		방 / NNG	에 / Josa
	들어가 / VV		에 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	ㄴ다 / EFN		신다 / EP+EC	

- 띄어쓰기에 대한 성능 비교로써 “아버지가 방에 들어가신다”로
어절이 구분되어야 함.



□ 성능 비교 – 의미 파악

- “나는 밥을 먹는다”

Hannanum	Kkma	Komoran	Mecab	Twitter
나 / N	나 / NP	나 / NP	나 / NP	나 / Noun
는 / J	는 / JX	는 / JX	는 / JX	는 / Josa
밥 / N	밥 / NNG	밥 / NNG	밥 / NNG	밥 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
먹 / P	먹 / VV	먹 / VV	먹 / VV	먹는 / Verb
는다 / E	는 / EPT	는다 / EC	는다 / EC	다 / Eomi
	다 / EFN			



□ 성능 비교 – 의미 파악

- “하늘을 나는 자동차”

Hannanum	Kkma	Komoran	Mecab	Twitter
하늘 / N	하늘 / NNG	하늘 / NNG	하늘 / NNG	하늘 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
나 / N	날 / VV	나 / NP	나 / NP	나 / Noun
는 / J	는 / ETD	는 / JX	는 / JX	는 / Josa
자동차 / N	자동차 / NNG	자동차 / NNG	자동차 / NNG	자동차 / Noun



□ 성능 비교 – 의미 파악

- “나는 밥을 먹는다”와 “하늘을 나는 자동차” 에서 주변 문맥에 따라 ‘나는’이 어떻게 해석되는지 비교분석 한 것이며, 첫번째 문장에서는 명사로 두번째 문장에서는 동사로 해석되는 것이 맞음.



□ 성능 비교 - 신조어

- “아이폰 기다리다 지쳐
애플공홈에서
언락폰질러버렸다 6+
128 실버”

Hannanum	Kkma	Komorani	Mecab	Twitter
아이폰 / N	아이 / NNG	아이폰 / NNP	아이폰 / NNP	아이폰 / Noun
기다리 / P	폰 / NNG	기다리 / VV	기다리 / VV	기다리 / Verb
다 / E	기다리 / VV	다 / EC	다 / EC	다 / Eomi
지쳐 / P	다 / ECS	지쳐 / VV	지쳐 / VV+EC	지쳐 / Verb
어 / E	지쳐 / VV	어 / EC	애플 / NNP	애플 / Noun
애플공홈 / N	어 / ECS	애플 / NNP	공 / NNG	공홈 / Noun
에서 / J	애플 / NNP	공 / NNG	홈 / NNG	에서 / Josa
언락폰질러버 렸다 / N	공 / NNG	홈 / NNG	에서 / JKB	언락폰 / Noun
6+ / N	홈 / NNG	에서 / JKB	언락 / NNG	질 / Verb
128기가실버 / N	에서 / JKM	언 / NNG	폰 / NNG	러 / Eomi
	언락 / NNG	락 / NNG	질러버렸 / VV+EC+VX+EP	버렸 / Verb
	폰 / NNG	폰 / NNG	다 / EC	다 / Eomi
	질르 / VV	지르 / VV	6 / SN	6 / Number
	어 / ECS	어 / EC	+ / SY	+ / Punctuation
	버리 / VXV	버리 / VX	128 / SN	128 / Number
	었 / EPT	었 / EP	기 / NNG	기 / Noun
	다 / ECS	다 / EC	가 / JKS	가 / Josa
	6 / NR	6 / SN	실버 / NNP	실버 / Noun
	+ / SW	+ / SW	ㅋ / UNKNOWN	ㅋ / KoreanParticle
	128 / NR	128기가실버 / NA		
	기가 / NNG			
	실버 / NNG			
	ㅋ / UN			



□ 시간 분석

□ 로딩시간 (사전로딩과 클래스 로딩시간)

- Kkma : 5.6988 secs
- Komoran : 5.4866 secs
- Hannanum : 0.6591 secs
- Twitter : 1.4870 secs
- Mecab : 0.0007 secs

□ 처리시간(10만 문자의 문서 대상)

- Kkma : 35.7163 secs
- Komoran : 25.6008 secs
- Hannanum : 8.8251 secs
- Twitter : 2.4714 secs
- Mecab : 0.2838 secs



- ❑ 설치
 - ❑ 서버에 Java, g++ 설치
 - !apt-get update
 - !apt-get install g++ openjdk-8-jdk



KoNLPy

❑ 설치

❑ 파이썬 패키지 설치

```
#!pip install JType1==0.5.7
```

```
#!pip install JType1-py3==0.5.5
```

```
!pip install konlpy
```



- 실행
 - 문장 분류



```
from konlpy.tag import Kkma
from konlpy.utils import pprint
kkma = Kkma()
pprint(kkma.sentences(u'네, 안녕하세요. 반갑습니다.'))
```



```
[네, 안녕하세요.,
반갑습니다.]
```



- 실행
 - 명사 추출



```
from konlpy.tag import Kkma
from konlpy.utils import pprint
kkma = Kkma()
pprint(kkma.nouns(u'하늘을 나는 자동차'))
```



[하늘, 자동차]



□ 실행

□ 형태소 분석



```
from konlpy.tag import Kkma  
from konlpy.utils import pprint  
kkma = Kkma()  
pprint(kkma.pos(u'하늘을 나는 자동차'))
```



```
[(하늘, NNG),  
(을, JKO),  
(날, VV),  
(는, ETD),  
(자동차, NNG)]
```



- 품사 사전

- <http://kkma.snu.ac.kr/documents/?doc=postag>

- Hannanum

```
from konlpy.tag import Hannanum
```

```
han = Hannanum()
```

```
# 각 token에 대한 다양한 형태학적 후보를 보여줌
```

```
pprint(han.analyze(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

```
pprint(han.morphs(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

```
pprint(han.nouns(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

```
pprint(han.pos(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

- Komoram

```
from konlpy.tag import Komoran  
  
ko = Komoran()  
  
pprint(ko.morphs(u"코모란도 똑같아요"))  
pprint(ko.nouns(u"코모란도 똑같아요"))  
pprint(ko.pos(u"코모란도 똑같아요"))
```

- Twitter

```
from konlpy.tag import Hannanum
```

```
han = Hannanum()
```

```
# 각 token에 대한 다양한 형태학적 후보를 보여줌
```

```
pprint(han.analyze(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

```
pprint(han.morphs(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

```
pprint(han.nouns(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

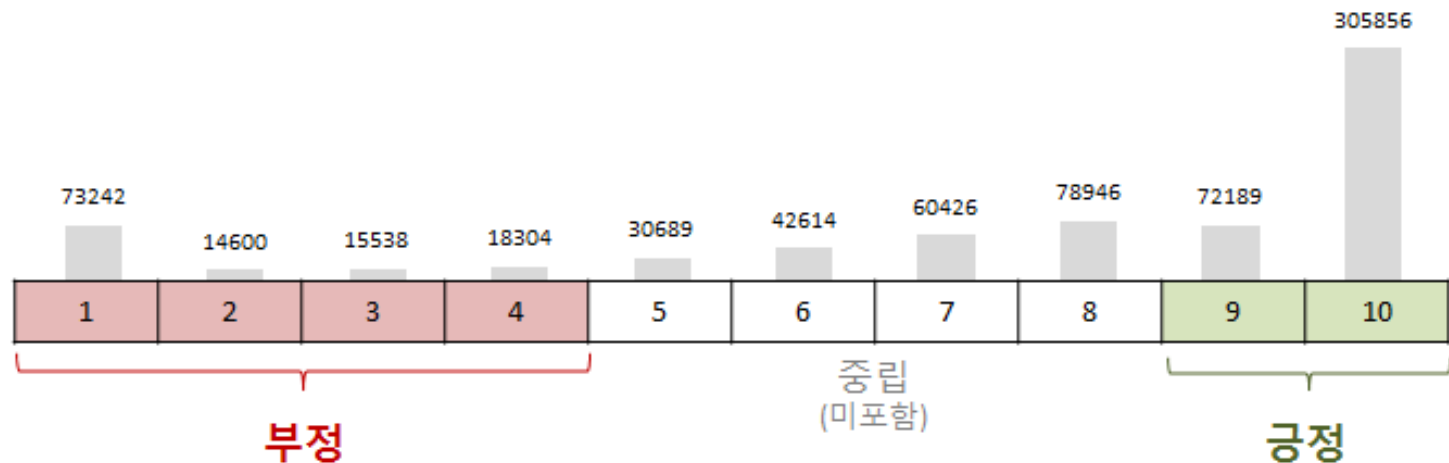
```
pprint(han.pos(u"아버지가 방에 들어가신다. 안녕하세요 나는 허윤아입니다"))
```

KoNLPy를 통한 한국어 리뷰 분석

- 이 데이터셋은 네이버 영화의 리뷰 중 영화당 **100개의 리뷰**를 모아 총 200,000개의 리뷰(train: 15만, test: 5만)로 이루어져있음

KoNLPy를 통한 한국어 리뷰 분석

- 1~10점까지의 평점 중에서 중립적인 평점(5~8점)은 제외하고 **1~4점을 긍정**으로, **9~10점을 부정**으로 동일한 비율로 데이터에 포함



KoNLPy를 통한 한국어 리뷰 분석

- Naver sentiment movie corpus 다운

- `!git clone https://github.com/e9t/nsmc.git`

KoNLPy를 통한 한국어 리뷰 분석

- Train data 확인
 - id, document, label 세 개의 열로 이루어져있음
 - id는 리뷰의 고유한 key 값이고, document는 리뷰의 내용, label은 긍정(0)인지 부정(1)인지

```
!cat ./nsmc/ratings_train.txt | head -n 10
```

KoNLPy를 통한 한국어 리뷰 분석

- Train/Test data를 list로 넣기

```
def read_data(filename):  
    with open(filename, 'r') as f:  
        data = [line.split('\t') for line in f.read().splitlines()]  
        # txt 파일의 헤더(id document label)는 제외하기  
        data = data[1:]  
    return data
```

```
train_data = read_data('./nsmc/ratings_train.txt')  
test_data = read_data('./nsmc/ratings_test.txt')  
print(train_data[0:3]) #[['9976970', '아 더빙.. 진짜 짜증나네요 목소리', '0'],  
['3819312', '흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나', '1'], ['10265843',  
'너무재밌었다그래서보는것을추천한다', '0']]
```

KoNLPy를 통한 한국어 리뷰 분석

- Train/Test data 제대로 불러왔는지 len로 확인

```
print(len(train_data))  
print(len(train_data[0]))  
print(len(test_data))  
print(len(test_data[0]))
```

KoNLPy를 통한 한국어 리뷰 분석

- KoNLPy는 띄어쓰기 알고리즘과 정규화를 이용해서 맞춤법이 틀린 문장도 어느 정도 고쳐주면서 형태소 분석과 품사를 태깅해주는 여러 클래스를 제공
- Okt(Open Korean Text)

```
!pip install konlpy
import konlpy

konlpy.__version__ # '0.5.1'
```

KoNLPy를 통한 한국어 리뷰 분석

- Okt(Open Korean Text) Test해보기

```
from konlpy.tag import Okt
```

```
okt = Okt()
```

```
print(okt.pos(u'이 밤 그날의 반딧불을 당신의 창 가까이  
보낼게요'))
```

KoNLPy를 통한 한국어 리뷰 분석

- 아까 불러온 데이터에 형태소 분석을 통해서 품사를 태깅해주는 작업

```
import json
import os
from pprint import pprint

train_data = train_data[:10000]
test_data = test_data[:1000]

def tokenize(doc):
    # norm은 정규화, stem은 근어로 표시하기를 나타냄
    return ['/'.join(t) for t in okt.pos(doc, norm=True, stem=True)]
```

KoNLPy를 통한 한국어 리뷰 분석

```
if os.path.isfile('./nsmc/train_docs.json'):
    with open('./nsmc/train_docs.json') as f:
        train_docs = json.load(f)
    with open('./nsmc/test_docs.json') as f:
        test_docs = json.load(f)
else:
    # train_data에 있는걸 한줄 씩 받아서 리스트로 만들어라
    train_docs = [(tokenize(row[1]), row[2]) for row in train_data]
    test_docs = [(tokenize(row[1]), row[2]) for row in test_data]
    # JSON 파일로 저장
    with open('./nsmc/train_docs.json', 'w', encoding="utf-8") as make_file:
        json.dump(train_docs, make_file, ensure_ascii=False, indent="\t")
    with open('./nsmc/test_docs.json', 'w', encoding="utf-8") as make_file:
        json.dump(test_docs, make_file, ensure_ascii=False, indent="\t")

# 예쁘게(?) 출력하기 위해서 pprint 라이브러리 사용
pprint(train_docs[0])
```

KoNLPy를 통한 한국어 리뷰 분석

- 분석된 token의 가장 작은 단위인 형태소가 몇 개 있는지 확인

```
# train_doc에 있는걸 d로 받고 d 중 0번째(리뷰)만 가져와서 list를 만들어라
tokens = [t for d in train_docs for t in d[0]]
print(len(tokens))
```


KoNLPy를 통한 한국어 리뷰 분석

- 이제 이 데이터를 nltk 라이브러리를 통해서 전처리를 해볼텐데요
- Text 클래스는 문서를 편리하게 탐색할 수 있는 다양한 기능을 제공합니다.

```
import nltk
text = nltk.Text(tokens, name='NMSC')
print(text)
```

KoNLPy를 통한 한국어 리뷰 분석

- 여기에서는 `vocab().most_common` 메서드를 이용해서 데이터에서 가장 자주 사용되는 단어를 가져올 때 사용하겠습니다.

전체 토큰의 개수

```
print(len(text.tokens))
```

중복을 제외한 토큰의 개수

```
print(len(set(text.tokens)))
```

출현 빈도가 높은 상위 토큰 10개

```
pprint(text.vocab().most_common(10))
```

상위 10000개 token을 벡터화시킴

```
selected_words = [f[0] for f in text.vocab().most_common(10000)]
```

Bag of words를 이용하여 count vector를 만드는 함수

```
def term_frequency(doc):  
    return [doc.count(word) for word in selected_words]
```

x에는 count vector가 적용된 문장을

y에서는 정답 셋을

```
train_x = [term_frequency(d) for d, _ in train_docs]
```

```
test_x = [term_frequency(d) for d, _ in test_docs]
```

```
train_y = [c for _, c in train_docs]
```

```
test_y = [c for _, c in test_docs]
```

KoNLPy를 통한 한국어 리뷰 분석

- 데이터를 float로 형 변환 시켜주면 데이터 전처리 과정은 끝납니다.

```
import numpy as np
```

```
x_train = np.asarray(train_x).astype('float32')
```

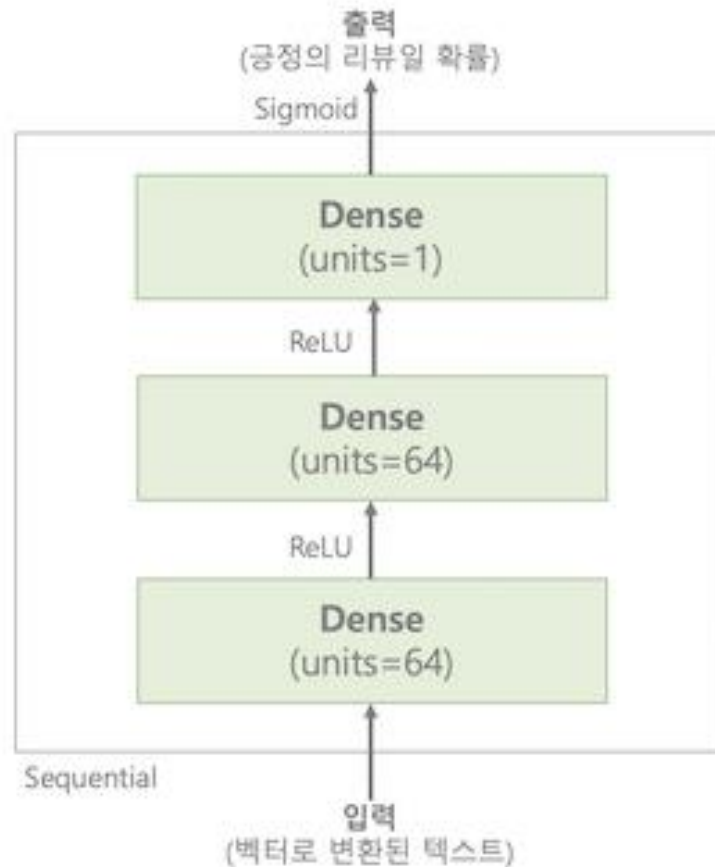
```
x_test = np.asarray(test_x).astype('float32')
```

```
y_train = np.asarray(train_y).astype('float32')
```

```
y_test = np.asarray(test_y).astype('float32')
```

KoNLPy를 통한 한국어 리뷰 분석


- Unit : 신경망 데이터를 얼마나 복잡하게 할 것인가
- 몇 개의 layer를 사용할 것인가



K

```
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras import losses
from tensorflow.keras import metrics
```

```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu',
input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```



- # 손실 함수로는 `binary_crossentropy`를 사용
- # `RMSProp` 옵티마이저를 통해서 `gradient descent`을 진행
- # 또한 배치 사이즈를 512로, 에포크를 10번으로 학습

```
model.compile(optimizer=optimizers.RMSprop(lr=0.001),  
              loss=losses.binary_crossentropy,  
              metrics=[metrics.binary_accuracy])
```

```
model.fit(x_train, y_train, epochs=10, batch_size=512)  
results = model.evaluate(x_test, y_test)
```

KoNLPy를 통한 한국어 리뷰 분석

- 성능확인

results


```
def predict_pos_neg(review):  
    token = tokenize(review)  
    tf = term_frequency(token)  
    # 데이터 형태 맞추기 위해 np.expand_dims 메서드를 이용해 array의 축을 확장  
    data = np.expand_dims(np.asarray(tf).astype('float32'), axis=0)  
    score = float(model.predict(data))  
    # 최종 확률이 0.5 보다 크면 긍정이고, 그렇지 않으면 부정이라고 예측  
    if(score > 0.5):  
        print("[{}]는 {:.2f}% 확률로 긍정 리뷰이지 않을까  
추측해봅니다.^\\n".format(review, score * 100))  
    else:  
        print("[{}]는 {:.2f}% 확률로 부정 리뷰이지 않을까  
추측해봅니다.^;\\n".format(review, (1 - score) * 100))
```

KoNLPy를 통한 한국어 리뷰 분석

- 새로운 데이터 넣기

```
predict_pos_neg("올해 최고의 영화! 세 번 넘게 봐도 질리지  
않네요.")  
predict_pos_neg("배경 음악이 영화의 분위기랑 너무 안  
맞았습니다. 몰입에 방해가 됩니다.")  
predict_pos_neg("주연 배우가 신인인데 연기를 진짜 잘 하네요.  
몰입감 ㅎㅎ")  
predict_pos_neg("믿고 보는 감독이지만 이번에는 아니네요")  
predict_pos_neg("주연배우 때문에 봤어요")
```



THANKS!

허윤아
yj72722@korea.ac.kr