[Lab 10]

제출과제

이번 실습에서는 텍스트데이터를 이용한 영화리뷰 감성분석 모델을 학습합니다. 자료실에 게시된 “Introduction to machine learning with python”의 7장(텍스트 데이터 다루기 1절~5절의 내용을 참고하여 구현합니다.

- http://ai.stanford.edu/~amaas/data/sentiment/ 의 데이터 이용
- LogisticRegression 분류기를 사용하여 train data로 학습하고 test data에 대한 성능 평가
- 다음의 전처리 적용 -> train과 test set에 대해 모두 적용해야 함
  - Bag of words 표현법 사용
  - Stopwords removal 사용

제출과제

Large Movie Review Dataset

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided. See the README file contained in the release for more details.

Large Movie Review Dataset v1.0

When using this dataset, please cite our ACL 2011 paper [bib].

Contact

For comments or questions on the dataset please contact Andrew Maas. As you publish papers using the dataset please notify us so we can post a link on this page.

Publications Using the Dataset

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

데이터 읽기(1)

- ! 기호는 셸(Shell) 명령어를 실행해주는 Ipython의 매직 명령어입니다.
- 매직 명령어는 파이썬에서 작업의 유연성을 높이고 시스템 동작을 간편하게 제어할 수 있게 도와줍니다.
- test와 train 디렉토리가 존재하고 각각의 디렉토리 안에 neg(부정), pos(긍정) 디렉토리가 존재합니다.
- neg와 pos 디렉토리 안에 각각의 영화에 관한 리뷰 텍스트 파일이 존재합니다.
- 꼭 unsup 디렉토리 삭제 필요!

```
!dir aclImdb₩test
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: F65E-7C0B

 C:₩Users₩YC₩aclImdb₩test 디렉터리

2020-11-05  오후 05:47    <DIR>          .
2020-11-05  오후 05:47    <DIR>          ..
2011-04-13  오전 02:25        20,205,283 labeledBow.feat
2020-11-05  오후 05:47    <DIR>          neg
2020-11-05  오후 05:47    <DIR>          pos
2011-04-12  오후 06:48           612,500 urls_neg.txt
2011-04-12  오후 06:48           612,500 urls_pos.txt
               3개 파일          21,430,283 바이트
               4개 디렉터리  136,544,366,592 바이트 남음
```

```
!tree aclImdb
```

```
폴더 PATH의 목록입니다.
볼륨 일련 번호는 F65E-7C0B입니다.
C:₩USERS₩YC₩ACLIMDB
├─test
│   ├─neg
│   └─pos
└─train
    ├─neg
    ├─pos
    └─unsup
```

데이터 읽기 (2)

- Load_files() 함수를 이용한 데이터 읽기
- Train 데이터 살펴보기

```
from sklearn.datasets import load_files

reviews_train = load_files("data/aclImdb/train/")
# 텍스트와 레이블을 포함하고 있는 Bunch 오브젝트를 반환합니다.
text_train, y_train = reviews_train.data, reviews_train.target
print("text_train의 타입:", type(text_train))
print("text_train의 길이:", len(text_train))
print("text_train[6]:₩n", text_train[6])
```

"실제 데이터가 존재하는 경로"

```
text_train의 타입: <class 'list'>
text_train의 길이: 25000
text_train[6]:
 b"This movie has a special way of telling the story, at first i found it rather odd as it jumped through time and I had no idea whats happening.<br
/><br />Anyway the story line was although simple, but still very real and touching. You met someone the first time, you fell in love completely, but
broke up at last and promoted a deadly agony. Who hasn't go through this? but we will never forget this kind of pain in our life. <br /><br />I would
say i am rather touched as two actor has shown great performance in showing the love between the characters. I just wish that the story could be a ha
ppy ending."
```

데이터 읽기 (3)

- Train, Test 데이터 안의 HTML태그 삭제
- Test 데이터 살펴보기

```
text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
```

```
print("클래스별 샘플 수 (훈련 데이터):", np.bincount(y_train))
```

```
클래스별 샘플 수 (훈련 데이터): [12500 12500]
```

```
reviews_test = load_files("data/aclImdb/test/")
text_test, y_test = reviews_test.data, reviews_test.target
print("테스트 데이터의 문서 수:", len(text_test))
print("클래스별 샘플 수 (테스트 데이터):", np.bincount(y_test))
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]
```

```
테스트 데이터의 문서 수: 25000
클래스별 샘플 수 (테스트 데이터): [12500 12500]
```

LogisticRegression()

- 유방암 데이터셋을 이용한 LogisticRegression

```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, stratify=cancer.target,
random_state=42)
logreg = LogisticRegression().fit(X_train, y_train)
print("훈련 세트 점수 : {:.3f}".format(logreg.score(X_train, y_train)))
print("테스트 세트 점수 : {:.3f} " .format(logreg.score(X_test, y_test)))
```

- 훈련 세트 점수 : 0.948
- 테스트 세트 점수 : 0.958

Sklearn CountVectorizer()

- CountVectorizer()

```
from sklearn.feature_extraction.text import CountVectorizer

CountVectorizer(*, input='content', encoding='utf-8', decode_error='strict', strip_accents=None,
lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b',
ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False,
dtype=<class 'numpy.int64'>)
```

Corpus에서 단어 사전을 구성하고 각 문서에 대해 단어의 등장 횟수를 카운팅하여 수치 벡터화합니다.

- Parameter

- input : CountVectorizer를 할 데이터를 입력합니다.
- stop_words : 문자열 형태('english')로 들어가면 영어용 Stop Word가 사용됩니다.
  리스트 형태로 들어가면 직접 지정한 리스트로 Stop Word가 사용됩니다.
- tokken_pattern : 기본 패턴은 두 글자로 한 글자로 된 경우는 count가 되지 않습니다.
- analyzer : 문자열 또는 함수가 들어갈 수 있습니다.
  -> 예시 : *{'word', 'char', 'char_wb'}*
- min/max_df : 단어장에 포함되기 위한 최소빈도 / 비율

텍스트 데이터를 BOW로 표현하기

- 예제 문장을 활용한 텍스트 데이터 BOW로 표현하기

```
bards_words =["The fool doth think he is wise,",
              "but the wise man knows himself to be a fool"]
```

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
vect.fit(bards_words)
```

```
CountVectorizer()
```

```
print("어휘 사전의 크기:", len(vect.vocabulary_))
print("어휘 사전의 내용:₩n", vect.vocabulary_)
```

```
어휘 사전의 크기: 13
어휘 사전의 내용:
 {'the': 9, 'fool': 3, 'doth': 2, 'think': 10, 'he': 4, 'is': 6, 'wise': 12, 'but': 1, 'man': 8, 'knows': 7, 'himself': 5, 'to': 11, 'be': 0}
```

```
bag_of_words = vect.transform(bards_words)
print("BOW:", repr(bag_of_words))
```

```
BOW: <2x13 sparse matrix of type '<class 'numpy.int64'>'
        with 16 stored elements in Compressed Sparse Row format>
```

```
print("BOW의 밀집 표현:₩n", bag_of_words.toarray())
```

```
BOW의 밀집 표현:
 [[0 0 1 1 1 0 1 0 0 1 1 0 1]
 [1 1 0 1 0 1 0 1 1 1 0 1 1]]
```

텍스트 데이터를 BOW로 표현하기

- 영화 리뷰에 대한 BOW

```
vect = CountVectorizer().fit(text_train)
X_train = vect.transform(text_train)
print("X_train:₩n", repr(X_train))
```

```
X_train:
 <25000x74849 sparse matrix of type '<class 'numpy.int64'>'
        with 3431196 stored elements in Compressed Sparse Row format>
```

```
feature_names = vect.get_feature_names()
print("특성 개수:", len(feature_names))
print("처음 20개 특성:₩n", feature_names[:20])
print("20010에서 20030까지 특성:₩n", feature_names[20010:20030])
print("매 2000번째 특성:₩n", feature_names[::2000])
```

```
특성 개수: 74849
처음 20개 특성:
 ['00', '000', '0000000000001', '00001', '00015', '000s', '001', '003830', '006', '007', '0079', '0080', '0083', '0093638', '00am', '00pm', '00s', '01', '01pm', '02']
20010에서 20030까지 특성:
 ['dratted', 'draub', 'draught', 'draughts', 'draughtswoman', 'draw', 'drawback', 'drawbacks', 'drawer', 'drawers', 'drawing', 'drawings', 'drawl', 'drawled', 'drawling', 'drawn', 'draws', 'draza', 'dre', 'drea']
매 2000번째 특성:
 ['00', 'aesir', 'aquarian', 'barking', 'blustering', 'bête', 'chicanery', 'condensing', 'cunning', 'detox', 'draper', 'enshrined', 'favorit', 'freezer', 'goldman', 'hasan', 'huitieme', 'intelligible', 'kantrowitz', 'lawful', 'maars', 'megalunged', 'mostey', 'norrland', 'padilla', 'pincher', 'promisingly', 'receptionist', 'rivals', 'schnaas', 'shunning', 'sparse', 'subset', 'temptations', 'treatises', 'unproven', 'walkman', 'xylophonist']
```

텍스트 데이터를 BOW로 표현하기

- 영화 리뷰에 대한 BOW
- 영화 리뷰 데이터를 바탕으로 LogisticRegression을 통한 cross_val_score 평가

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
vect = CountVectorizer().fit(text_train)
X_train = vect.transform(text_train)
scores = cross_val_score(LogisticRegression(max_iter=1000), X_train, y_train, cv=5)
print("크로스 밸리데이션 평균 점수: {:.4f}".format(np.mean(scores)))
```

-> 크로스 밸리데이션 평균 점수 : 0.8813

- min_df 파라미터 값 추가

```
vect = CountVectorizer(min_df=5).fit(text_train)
X_train = vect.transform(text_train)
scores = cross_val_score(LogisticRegression(max_iter=1000), X_train, y_train, cv=5)
```

- -> 크로스 밸리데이션 평균 점수 : 0.8793

불용어

- 불용어 제거
- ENGLISH_STOP_WORDS를 사용한 불용어 제거

```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
print("불용어 개수:", len(ENGLISH_STOP_WORDS))
print("매 10번째 불용어:₩n", list(ENGLISH_STOP_WORDS)[::10])
```

```
불용어 개수: 318
매 10번째 불용어:
 ['often', 'we', 'less', 'a', 'back', 'none', 'full', 'empty', 'cant', 'too', 'than', 'fifteen', 'ie', 'everywhere', 'some', 'afterwards', 'cannot',
'thereafter', 'more', 'twenty', 'an', 'below', 'nowhere', 'somehow', 'noone', 'already', 'upon', 'meanwhile', 'whoever', 'its', 'sometime', 'others']
```

```
# stop_words="english"라고 지정하면 내장된 불용어를 사용합니다.
# 내장된 불용어에 추가할 수도 있고 자신만의 목록을 사용할 수도 있습니다.
vect = CountVectorizer(min_df=5, stop_words="english").fit(text_train)
X_train = vect.transform(text_train)
print("불용어가 제거된 X_train:₩n", repr(X_train))
```

```
불용어가 제거된 X_train:
 <25000x26966 sparse matrix of type '<class 'numpy.int64'>'
        with 2149958 stored elements in Compressed Sparse Row format>
```

불용어

- 불용어 제거 후 LogisticRegression을 통한 cross_val_score 평가

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
vect = CountVectorizer(min_df=5,stop_words="english").fit(text_train)
X_train = vect.transform(text_train)
scores = cross_val_score(LogisticRegression(max_iter=1000), X_train, y_train, cv=5)
print("크로스 밸리데이션 평균 점수: {:.4f}".format(np.mean(scores)))
```

-> 크로스 밸리데이션 평균 점수 : 0.8754