[Lab 11]

지난 과제 수정

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

-> Train , Test 데이터를 각각 Bag of words 방법으로 나타낼 경우 각각의 Feature값이 달라 각각에 대해 Bag of words 표현법을 사용하지 못합니다.

지난 과제 수정

- Train과 test 데이터를 모두 사용하여 단어 사전 구축과 불용어 처리를 해야 함
- vect = CountVectorizer(stop_words="english").fit(text_train, text_test)
- 단어 빈도수를 이용해 문서 행렬을 구성할 때는 vect.transform() 함수를 train 데이터와 test 데이터에 대해 따로 적용

제출과제

이번 과제의 마감일은 12월 1일입니다. (두번째 마감일은 없습니다)
이번 학기 마지막 실습과제입니다.

문제. 지난 실습에서는 텍스트데이터를 이용한 영화리뷰 감성분석 모델을 학습하고 성능을 평가하였습니다. 이번 실습에서는 BBC 스포츠 뉴스 기사를 모은 데이터셋을 이용하여 카테고리 예측 모델을 학습합니다.

BBC 스포츠 뉴스 기사 데이터 세트는 사이트 http://mlg.ucd.ie/datasets/bbc.html 에서 다운받을 수 있습니다.

"raw text files"을 다운 받아 다음을 수행합니다.
- scikit-learn의 Tfidfvectorizer를 이용하여 전처리 (불용어 처리, tf-idf 적용, L2 regularization)
- Linear SVM 으로 5-cross validation 수행하여 성능 평가

제출과제

Dataset: BBCSport

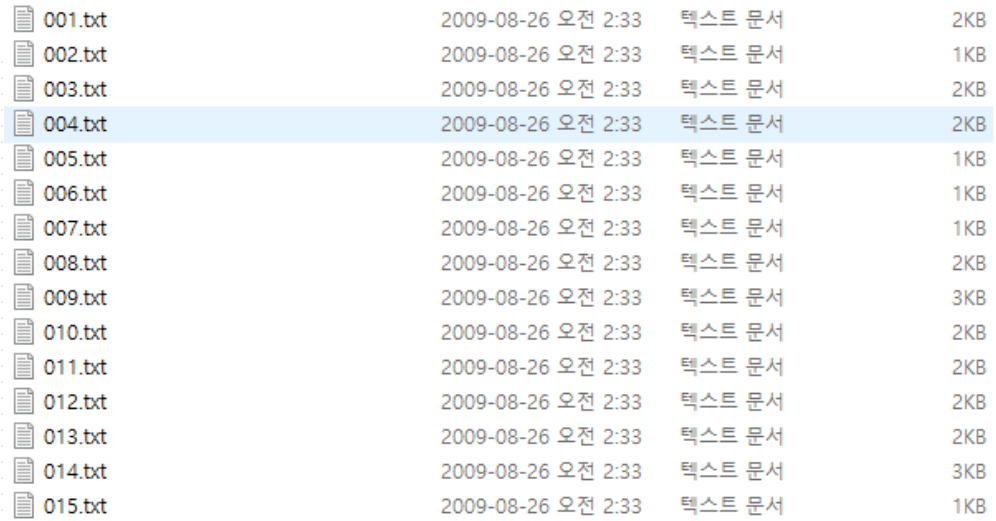
All rights, including copyright, in the content of the original articles are owned by the BBC.

- Consists of 737 documents from the BBC Sport website corresponding to sports news articles in five topical areas from 2004-2005.
- Class Labels: 5 (athletics, cricket, football, rugby, tennis)

>> Download pre-processed dataset

>> Download raw text files

C:₩USERS₩GUEST₩BBCSPORT-FULLTEXT
└──bbcsport
    ├──athletics
    ├──cricket
    ├──football
    ├──rugby
    └──tennis



001.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
002.txt	2009-08-26 오전 2:33	텍스트 문서	1KB
003.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
004.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
005.txt	2009-08-26 오전 2:33	텍스트 문서	1KB
006.txt	2009-08-26 오전 2:33	텍스트 문서	1KB
007.txt	2009-08-26 오전 2:33	텍스트 문서	1KB
008.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
009.txt	2009-08-26 오전 2:33	텍스트 문서	3KB
010.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
011.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
012.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
013.txt	2009-08-26 오전 2:33	텍스트 문서	2KB
014.txt	2009-08-26 오전 2:33	텍스트 문서	3KB
015.txt	2009-08-26 오전 2:33	텍스트 문서	1KB

제출과제

원문서가 개별 파일로 이루어져 있음

```
import chardet
files =load_files("./bbcsport-fulltext/bbcsport/")
X,y = files.data,files.target


# 데이터 인코딩 방식 변경
for i in range(len(X)):
    if(chardet.detect(X[i])!="utf-8"):
        X[i]= X[i].decode(chardet.detect(X[i])['encoding']).encode('utf8’)


# 데이터 전처리
X = [doc.replace(b"<br />",b"") for doc in X]
X = [doc.replace(b"\n",b" ") for doc in X]
```

TfildVectorizer()

TfildVectorizer() : 단어사전 구축, 불용어처리, tf-idf 백터 생성, L2

```
from sklearn.feature_extraction.text import TfidVectorizer

TfidfVectorizer(input='content', encoding='utf-8', decode_error='strict', strip_accents=None,
lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words='English',
token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None,
vocabulary=None, binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True,
smooth_idf=True, sublinear_tf=False)
```

- tf-idf (term frequency-inverse document frequency 단어빈도 – 역문서 빈도)
- tf-idf는 말뭉치의 다른 문서보다 특정 문서에 자주 나타나는 단어에 높은 가중치를 주는 방법입니다.

- Parameter

input : TfidVectorizer를 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.

min/max_df : 단어장에 포함되기 위한 최소빈도 / 비율

TfildVectorizer()

- from sklearn.feature_extraction.text import TfidfVectorizer

```
corpus = [
    'This is the first document.',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
print(X.shape)
(4, 9)
```

TfildVectorizer()

- build_analyzer() : Return a callable that handles preprocessing, tokenization and n-grams generation.
- build_preprocessor() :Return a function to preprocess the text before tokenization.
- build_tokenizer() : Return a function that splits a string into a sequence of tokens.
- decode(doc) : Decode the input into a string of unicode symbols.
- fit(raw_documents[, y]) : Learn vocabulary and idf from training set.
- fit_transform(raw_documents[, y]) : Learn vocabulary and idf, return document-term matrix.
- get_feature_names() : Array mapping from feature integer indices to feature name.
- get_params([deep]) : Get parameters for this estimator.
- get_stop_words() : Build or fetch the effective stop words list.
- inverse_transform(X) : Return terms per document with nonzero entries in X.
- set_params(**params) : Set the parameters of this estimator.
- transform(raw_documents[, copy]) : Transform documents to document-term matrix.

다양한 문서 전처리 기능

- DictVectorizer:
  각 단어의 수를 세어서 만든 사전에서 BOW 인코딩 벡터를 만든다.

- CountVectorizer:
  문서 집합에서 단어 토큰을 생성하고 각 단어의 수를 세어 BOW 인코딩 벡터를 만든다.

- TfidfVectorizer:
  CountVectorizer와 비슷하지만 TF-IDF 방식으로 단어의 가중치를 조정한 BOW 인코딩 벡터를 만든다.

- HashingVectorizer:
  해시 함수(hash function)을 사용하여 적은 메모리와 빠른 속도로 BOW 인코딩 벡터를 만든다.