

기계학습 텀프로젝트 2

19011494 조국희

02. SPAM 문자 분류기

■ 텍스트 데이터 전처리

No I'm in the same boat. Still here at my moms. Check me out on yo. I'm half naked.

re.sub : (패턴, 바꿀문자열, 문자열, 횟수) -> 알파벳을 제외한 문제 공백으로 만들

No I m in the same boat Still here at my moms Check me out on yo I m half naked

대문자를 소문자로 변경

no i m in the same boat still here at my moms check me out on yo i m half naked

문장 토큰화

```
['no', 'i', 'm', 'in', 'the', 'same', 'boat', 'still', 'here', 'at',  
'my', 'moms', 'check', 'me', 'out', 'on', 'yo', 'i', 'm', 'half', 'naked']
```

***불용어 : I, me, 조사, 접미사 같이 문장에서는 자주 등장하지만
실제 의미 분석을 하는데는 거의 기여하는 바가 없는 단어**

불용어 골라냄

PorterStemmer -> stem() : 어간 추출 함수

['boat', 'still', 'moms', 'check', 'yo', 'half', 'naked']

['boat', 'still', 'mom', 'check', 'yo', 'half', 'nake']

```
# [1] A-Z, a-z를 제외한 문자는 공백으로 만들어 pre_words에 저장  
pre_words = re.sub('[^A-Za-z]', ' ', text)
```

```
# [2] pre_words의 문자들을 소문자로 변경  
pre_words = pre_words.lower()
```

```
# [3] pre_words의 문장 토큰화해 tokenized_words에 저장  
tokenized_words = word_tokenize(pre_words)
```

```
# [4] stops에 불용어 set 저장  
stops = set(stopwords.words('english'))
```

```
tokenized_words_remove=[]  
for w in tokenized_words:  
    # [5] tokenized_words의 단어가 불용어가 아니라면  
    if w not in stops:  
        # tokenized_words_remove에 해당 단어 추가  
        tokenized_words_remove.append(w)
```

```
stemmer = PorterStemmer() # 어간추출  
for i in range(len(tokenized_words_remove)):  
    # [6] stem 내장 함수를 이용해 어간 추출  
    tokenized_words_remove[i] = stemmer.stem(tokenized_words_remove[i])
```

02. SPAM 문자 분류기

▪ Bag of Word

- ✓ CountVectorizer : 각 텍스트에서 단어 출현 횟수를 카운팅한 벡터
- ✓ TfidfVectorizer : TF-IDF 값을 사용하여 CountVectorizer의 단점을 보완함
문장 안에서는 많이 등장할수록, 문서 전체에서는 적게 등장할수록 가중치를 부여하는 것
- ✓ HashingVectorizer : CountVectorizer에서 해시 함수를 사용하여 속도를 높임

```
# [1] CountVectorizer 생성
vectorizer = CountVectorizer(max_features = 200)

# [2] numpy array로 변환 후 데이터 타입을 "U"로 변경해 저장
X_train = np.asarray(X_train).astype("U")
X_test = np.asarray(X_test).astype("U")

# [3] X_train은 학습 및 변환을 하고, X_test는 변환
X_train_features = vectorizer.fit_transform(X_train)
X_test_features = vectorizer.transform(X_test)
```

CountVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features = 200)

X_train = np.asarray(X_train).astype("U")
X_test = np.asarray(X_test).astype("U")

X_train_features = vectorizer.fit_transform(X_train)
X_test_features = vectorizer.transform(X_test)
```

TfidfVectorizer

```
from sklearn.feature_extraction.text import HashingVectorizer

vectorizer = HashingVectorizer()

X_train = np.asarray(X_train).astype("U")
X_test = np.asarray(X_test).astype("U")

X_train_features = vectorizer.fit_transform(X_train)
X_test_features = vectorizer.transform(X_test)
```

HashingVectorizer

02. SPAM 문자 분류기

■ 예측

```
params = {
    'C' : [3, 4, 5, 6, 7],
    'kernel' : ['rbf'],
}
clf = GridSearchCV(SVC(random_state = 0), params, cv=5)
clf.fit(X, y)
print(clf.best_params_)

y_pred = clf.predict(test)
df_pred={"ID": range(np.array(y_pred).shape[0]), "Class":y_pred}
df_pred=pd.DataFrame(df_pred)
df_pred.to_csv("baseline_TF.csv", index=False)
```

GridSearchCV를 통해 최적의 파라미터를 찾아 모델학습 후 예측

■ 성능

베이스라인	성능
HashingVectorizer	0.86547
TfidfVectorizer	0.94439
CountVectorizer	0.94439

모델	성능
SVC + HashingVectorizer	0.97937
SVC + TfidfVectorizer	0.97937
SVC + CountVectorizer	0.97937