Branch: master ▾ | Find file | Copy path

Machine-learning-python / Анализ текстов.py

sergeybelov должен теперь работать и давать правильный ответ на ДЗ

32f6878 on 2 Mar 2017

1 contributor

Raw | BlameHistory

107 lines (87 sloc) | 7.03 KB

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Feb 28 15:08:44 2017
4
5  @author: tehn-11
6  """
7
8  #============================================================================
9  # Анализ текстов
10 #============================================================================
11 from   sklearnimportdatasets
12 from sklearn.feature_extraction.text import TfidfVectorizer
13 from sklearn.model_selection import GridSearchCV
14 from sklearn.svm import SVC
15 import numpy as np
16 import pandas as pd
17 from sklearn.model_selection import KFold
18
19 #============================================================================
20 # Для начала вам потребуется загрузить данные.
21 # В этом задании мы воспользуемся одним из датасетов, доступных в scikit-learn'e – 20 newsgroup
22 # Для этого нужно воспользоваться модулем datasets:
23 #============================================================================
24 newsgroups = datasets.fetch_20newsgroups(
25                     subset='all',
26                     categories=['alt.atheism', 'sci.space']
```

```
            )

#========================================================================
# Вычислите TF-IDF-признаки для всех текстов.
# Обратите внимание, что в этом задании мы предлагаем вам вычислить TF-IDF по всем данным.
# При таком подходе получается, что признаки на обучающем множестве используют информацию из те
# поскольку мы не используем значения целевой переменной из теста.
# На практике нередко встречаются ситуации, когда признаки объектов тестовой выборки известны н
# и поэтому можно ими пользоваться при обучении алгоритма.
#========================================================================

#В Scikit-Learn это реализовано в классе
#sklearn.feature_extraction.text.TfidfVectorizer.
#Преобразование обучающей выборки нужно делать с помощью функции fit_transform, тестовой — с по
y_train = newsgroups.target#Класс
X_train = newsgroups.data#Характеристики

vectorizer = TfidfVectorizer()
dataMatrix=vectorizer.fit_transform(X_train).toarray()#матрица объектов по словам, в ячейках ве

#idf = vectorizer.idf_
words=vectorizer.get_feature_names()
#tf_idf=dict(zip(words, idf))#токены с весами


#========================================================================
# Подберите минимальный лучший параметр C из множества [10^-5, 10^-4, ... 10^4, 10^5] для SVM с
# при помощи кросс-валидации по 5 блокам. Укажите параметр random_state=241 и для SVM, и для KF
# В качестве меры качества используйте долю верных ответов (accuracy).
#========================================================================

grid = {'C': np.power(10.0, np.arange(-5, 6))}
cv = KFold(n_splits=5, shuffle=True, random_state=241)
clf = SVC(kernel='linear', random_state=241)
gs = GridSearchCV(clf, grid, scoring='accuracy', cv=cv)
gs.fit(dataMatrix, y_train)#внимание очень долго работает на тестовых данных

#записываем параметры в массив
#validationTest=dict(zip(gs.cv_results_['mean_test_score'], gs.cv_results_['params']))#получаем
#validationTestData=pd.DataFrame(data=validationTest).transpose()#создаем датафрейм с транспони
#validationTestData.sort_index(ascending=[False],inplace=True)

#У GridSearchCV есть поле best_estimator_,
#которое можно использовать, чтобы не обучать заново классификатор с оптимальным параметром.
bestC=gs.best_estimator_.C

#устарело
#for a in gs.grid_scores_:
 #   validationTest[a.mean_validation_score]=a.parameters# — оценка качества по кросс-валидации

#========================================================================
# Обучите SVM по всей выборке с оптимальным параметром C, найденным на предыдущем шаге.
```

```
#============================================================================
clf = SVC(C=bestC,kernel='linear', random_state=241)
result=clf.fit(dataMatrix, y_train)


#============================================================================
# Найдите 10 слов с наибольшим абсолютным значением веса (веса хранятся в поле coef_ у svm.SVC)
# Они являются ответом на это задание. Укажите эти слова через запятую или пробел, в нижнем рег
#в лексикографическом порядке.
#============================================================================
weights=[]
for element in result.coef_.T:#с транспонированием, иначе все коэффициенты идут строкой
    weights.append(abs(element[0]))

combine=dict(zip(words,weights))
bestWords=pd.DataFrame(data=combine,index=[0]).transpose()#с транспонированием, иначе все коэфф
bestWords.columns = ['weights']#переименуем колонки
bestWords.sort_values(['weights'], ascending=[False],inplace=True)

topTenWordsCollection=bestWords.head(10).index.values#берем значения 10 индексов (там слова)
newWords=pd.DataFrame(data=topTenWordsCollection)
newWords.columns = ['words']#переименуем колонки
newWords.words.astype(str)
newWords=newWords.apply(lambda x: x[0].lower(),axis =1)#приводим к нижнему регистру, имя колонк
newWords.sort_values(0, ascending=[True],inplace=True)#сортируем в лексикографическом порядке.

collection=newWords.values#получаем значения массива
print (' '.join(map(lambda x: x, collection)))#сливаем в строку каждый элемент
```