Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6

«Классификация текста»

ИСПОЛНИТЕЛЬ:

Костян Алина Алексеевна
Группа ИУ5-21М

Москва 2021

Задание:

Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами:

1. Способ 1. На основе CountVectorizer или TfidfVectorizer.
2. Способ 2. На основе моделей word2vec или Glove или fastText.
3. Сравните качество полученных моделей.

Текст программы:

In [5]:
```
categories = ["talk.politics.guns", "alt.atheism", "sci.med", "rec.autos"]
newsgroups = fetch_20newsgroups(subset='train', categories=categories)
data = newsgroups['data']
```

In [6]:
```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [7]:
```
vocabVect = CountVectorizer()
vocabVect.fit(data)
corpusVocab = vocabVect.vocabulary_
print('Количество сформированных признаков - {}'.format(len(corpusVocab)))
```

```
Количество сформированных признаков - 37176
```

In [8]:
```
for i in list(corpusVocab)[1:10]:
    print('{}={}'.format(i, corpusVocab[i]))
```

```
thom=33375
morgan=23251
ucs=34360
mun=23527
ca=8754
thomas=33376
clancy=9784
subject=32210
re=28101
```

In [9]:
```
test_features = vocabVect.transform(data)
test_features
```

Out[9]:
```
<2214x37176 sparse matrix of type '<class 'numpy.int64'>'
        with 375168 stored elements in Compressed Sparse Row format>
```

In [10]:
```
len(test_features.todense()[0].getA1())
```

Out[10]:
```
37176
```

In [11]:
```
vocabVect.get_feature_names()[37170:]
```

Out[11]:
```
['zyg', 'zyklon', 'zz', 'zz_g9q3', 'zzz', 'îâlittin']
```

In [12]:
```
def VectorizeAndClassify(vectorizers_list, classifiers_list):
    for v in vectorizers_list:
        for c in classifiers_list:
            pipeline1 = Pipeline([("vectorizer", v), ("classifier", c)])
            score = cross_val_score(pipeline1, newsgroups['data'], newsgroups['target'], scoring='accuracy', cv=3).mean
            print('Векторизация - {}'.format(v))
            print('Модель для классификации - {}'.format(c))
            print('Accuracy = {}'.format(score))
            print('===========================')
```

In [13]:
```
vectorizers_list = [CountVectorizer(vocabulary = corpusVocab)]
classifiers_list = [LogisticRegression(C=3.0), LinearSVC(), KNeighborsClassifier()]
VectorizeAndClassify(vectorizers_list, classifiers_list)
```

```
/Users/lina/Documents/University Shit/Master/sem 2/ML/.venv/lib/python3.7/site-packages/sklearn/linear_model/_logisti
```

```
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200': 3,
                            '00014': 4, '000152': 5, '000406': 6,
                            '0005111312': 7, '0005111312na3em': 8, '000601': 9,
                            '000710': 10, '000mi': 11, '000miles': 12,
                            '000s': 13, '001': 14, '0010': 15, '001004': 16,
                            '001125': 17, '001319': 18, '001642': 19, '002': 20,
                            '002142': 21, '002651': 22, '003': 23,
                            '003258u19250': 24, '0033': 25, '003522': 26,
                            '004': 27, '004021809': 28, '004158': 29, ...})
Модель для классификации - LogisticRegression(C=3.0)
Accuracy = 0.951219512195122
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200': 3,
                            '00014': 4, '000152': 5, '000406': 6,
                            '0005111312': 7, '0005111312na3em': 8, '000601': 9,
                            '000710': 10, '000mi': 11, '000miles': 12,
                            '000s': 13, '001': 14, '0010': 15, '001004': 16,
                            '001125': 17, '001319': 18, '001642': 19, '002': 20,
                            '002142': 21, '002651': 22, '003': 23,
                            '003258u19250': 24, '0033': 25, '003522': 26,
                            '004': 27, '004021809': 28, '004158': 29, ...})
Модель для классификации - LinearSVC()
Accuracy = 0.9543812104787714
===========================
Векторизация - CountVectorizer(vocabulary={'00': 0, '000': 1, '0000': 2, '0000001200': 3,
                            '00014': 4, '000152': 5, '000406': 6,
                            '0005111312': 7, '0005111312na3em': 8, '000601': 9,
                            '000710': 10, '000mi': 11, '000miles': 12,
                            '000s': 13, '001': 14, '0010': 15, '001004': 16,
                            '001125': 17, '001319': 18, '001642': 19, '002': 20,
                            '002142': 21, '002651': 22, '003': 23,
                            '003258u19250': 24, '0033': 25, '003522': 26,
                            '004': 27, '004021809': 28, '004158': 29, ...})
Модель для классификации - KNeighborsClassifier()
Accuracy = 0.6820234869015357
===========================
```

In [14]:
```
# Using the stopwords.
from nltk.corpus import stopwords

# Initialize the stopwords
stoplist = stopwords.words('english')
```

In [15]:
```
# Подготовим корпус
corpus = []
stop_words = stopwords.words('english')
tok = WordPunctTokenizer()
for line in newsgroups['data']:
    line1 = line.strip().lower()
    line1 = re.sub("[^a-zA-Z]"," ", line1)
    text_tok = tok.tokenize(line1)
    text_tok1 = [w for w in text_tok if not w in stop_words]
    corpus.append(text_tok1)
```

In [16]:
```
corpus[:5]
```

```
    'mcgrdu'
```

In [17]:
```
%time model = word2vec.Word2Vec(corpus)
```

```
CPU times: user 2.91 s, sys: 28.5 ms, total: 2.94 s
Wall time: 1.24 s
```

In [18]:
```
# Проверим, что модель обучилась
print(model.wv.most_similar(positive=['subject'], topn=5))
```

```
[('bill', 0.8631049394607544), ('badlands', 0.8602226376533508), ('conner', 0.8579955697059631), ('itc', 0.8504862189
292908), ('burns', 0.8487685918807983)]
```

In [19]:
```
def sentiment(v, c):
    model = Pipeline(
        [("vectorizer", v),
         ("classifier", c)])
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print_accuracy_score_for_classes(y_test, y_pred)
```

In [20]:
```
class EmbeddingVectorizer(object):
    '''
    Для текста усредним вектора входящих в него слов
    '''
    def __init__(self, model):
        self.model = model
        self.size = model.vector_size

    def fit(self, X, y):
        return self

    def transform(self, X):
        return np.array([np.mean(
            [self.model[w] for w in words if w in self.model]
            or [np.zeros(self.size)], axis=0)
```

In [21]:
```
# Обучающая и тестовая выборки
boundary = 700
X_train = corpus[:boundary]
X_test = corpus[boundary:]
y_train = newsgroups['target'][:boundary]
y_test = newsgroups['target'][boundary:]
sentiment(EmbeddingVectorizer(model.wv), LogisticRegression(C=5.0))
```

```
/Users/lina/Documents/University Shit/Master/sem_2/ML/.venv/lib/python3.7/site-packages/sklearn/linear_model/_logisti
c.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Метка	Accuracy
0	0.8549848942598187
1	0.8535980148883374
2	0.7239709443099274
3	0.776566757493188