Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

Отчет по лабораторной работе № 6
«Классификация текста»
по курсу “Методы машинного обучения”

Исполнитель:
Студент группы ИУ5-22М
Желанкина А.С.
15.05.2021

Москва, 2021

Задание лабораторной работы

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

1. На основе CountVectorizer или TfidfVectorizer.
2. На основе моделей word2vec или Glove или fastText.

Сравните качество полученных моделей.

Экранные формы с текстом программы и примерами её выполнения

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

CountVectorizer

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

```
Количество сформированных признаков - 37176
```

```
[5]  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
```

```
[6]  test_features = vocabVect.transform(data)
     test_features
```

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

```
[7]  len(test_features.todense()[0].getA1())
```

```
37176
```

```
[8]  vocabVect.get_feature_names()[37170:]
```

```
['zyg', 'zyklon', 'zz', 'zz_g9q3', 'zzz', 'íålittin']
```

```
[9]  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('============================')
```

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

```
============================
Векторизация - CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None,
                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(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
Accuracy = 0.9543812104787714
============================
```

word2vec

```
[20] # Using the stopwords.
     from nltk.corpus import stopwords

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

```
[25] # 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)
```

```
[28] %time model = word2vec.Word2Vec(corpus)
```

```
CPU times: user 8.02 s, sys: 72.1 ms, total: 8.1 s
Wall time: 5.35 s
```

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

```
[('badlands', 0.9109960794448853), ('reply', 0.9061744213104248), ('bill', 0.9060840606689453), ('itc', 0.9001675844192505),
```

```
[30] 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)
```

```
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)
            for words in X])
```

```python
# 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))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning:
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        0.7854984894259819
1        0.8188585607940446
2        0.639225181598063
3        0.7193460490463215
```