Министерство науки и высшего образования
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 (ИУ5)_____

Отчет по лабораторной работе №5

«Классификация текста»

по курсу «Методы машинного обучения».

ИСПОЛНИТЕЛЬ: Группа ИУ5-24М
Уралова Е.А.
ФИО
_____
подпись

"26" апреля 2024 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО
_____
подпись

"__"_____2024 г.

Москва – 2024

Задание:

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

Способ 1. На основе CountVectorizer или TfidfVectorizer.

Способ 2. На основе моделей word2vec или Glove или fastText.

Сравните качество полученных моделей.

Для поиска наборов данных в поисковой системе можно использовать ключевые слова "datasets for text classification".

Выполнение:

```
[7]  from sklearn.datasets import fetch_20newsgroups
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
     from sklearn.model_selection import train_test_split
     from gensim.models import Word2Vec
     from gensim.models.doc2vec import TaggedDocument
     import numpy as np
```

Загрузка набора данных "20 Newsgroups"

```
[8]  data = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)
```

Способ 1: CountVectorizer + наивный байесовский классификатор

Преобразование текстов с помощью CountVectorizer

```
[9]  vectorizer = CountVectorizer()
     X = vectorizer.fit_transform(data.data)
     y = data.target
```

Разделение данных на обучающую и тестовую выборки

```
[10] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Обучение модели логистической регрессии

```
[11] lr_classifier = LogisticRegression()
     lr_classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: 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
  n_iter_i = _check_optimize_result(
```

LogisticRegression()

Предсказание на тестовой выборке

```
[12] y_pred_cv = lr_classifier.predict(X_test)
```

Оценка качества

```
[13] accuracy_cv = accuracy_score(y_test, y_pred_cv)
     precision_cv = precision_score(y_test, y_pred_cv, average='weighted')
     recall_cv = recall_score(y_test, y_pred_cv, average='weighted')
     f1_cv = f1_score(y_test, y_pred_cv, average='weighted')
```

Способ 2: Word2Vec

Создание словаря для модели Word2Vec

```
[14] tagged_data = [doc.split() for doc in data.data]
     w2v_model = Word2Vec(sentences=tagged_data, vector_size=100, window=5, min_count=1, workers=4)
     w2v_model.train(tagged_data, total_examples=w2v_model.corpus_count, epochs=10)
```

```
WARNING:gensim.models.word2vec:Effective 'alpha' higher than previous training cycles
(45393581, 53457810)
```

Преобразование текстов в векторное представление

```
[15] X_w2v = np.array([np.mean([w2v_model.wv[word] for word in doc if word in w2v_model.wv] or [np.zeros(100)], axis=0) for doc in tagged_data])
     y = data.target
```

Разделение данных на обучающую и тестовую выборки

```
[16] X_train, X_test, y_train, y_test = train_test_split(X_w2v, y, test_size=0.2, random_state=42)
```

Обучение модели логистической регрессии

```
[17] lr_classifier_w2v = LogisticRegression()
     lr_classifier_w2v.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: 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
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

Предсказание на тестовой выборке

```
[18] y_pred_w2v = lr_classifier_w2v.predict(X_test)
```

Оценка качества

```
[19] accuracy_w2v = accuracy_score(y_test, y_pred_w2v)
     precision_w2v = precision_score(y_test, y_pred_w2v, average='weighted')
     recall_w2v = recall_score(y_test, y_pred_w2v, average='weighted')
     f1_w2v = f1_score(y_test, y_pred_w2v, average='weighted')
```

Сравнение результатов двух способов

```
[20] print("CountVectorizer + Logistic Regression:")
     print("Accuracy:", accuracy_cv)
     print("Precision:", precision_cv)
     print("Recall:", recall_cv)
     print("F1-score:", f1_cv)

     print("\nWord2Vec + Logistic Regression:")
     print("Accuracy:", accuracy_w2v)
     print("Precision:", precision_w2v)
     print("Recall:", recall_w2v)
     print("F1-score:", f1_w2v)
```

```
CountVectorizer + Logistic Regression:
Accuracy: 0.8777188328912466
Precision: 0.8788846100820348
Recall: 0.8777188328912466
F1-score: 0.8776216264713581

Word2Vec + Logistic Regression:
Accuracy: 0.5830238726790451
Precision: 0.5797508675360127
Recall: 0.5830238726790451
```

```
[22] from tabulate import tabulate

     data = [
         ["CountVectorizer + Logistic Regression", accuracy_cv, precision_cv, recall_cv, f1_cv],
         ["Word2Vec + Logistic Regression", accuracy_w2v, precision_w2v, recall_w2v, f1_w2v]
     ]

     headers = ["Model", "Accuracy", "Precision", "Recall", "F1-score"]

     print(tabulate(data, headers=headers, tablefmt='fancy_grid'))
```

Model	Accuracy	Precision	Recall	F1-score
CountVectorizer + Logistic Regression	0.877719	0.878885	0.877719	0.877622
Word2Vec + Logistic Regression	0.583024	0.579751	0.583024	0.578596

Вывод:

В данной лабораторной работе рассмотрена классификация текста двумя способами: CountVectorizer и word2vec. В результате были получены метрики, по которым можно судить о качестве моделей. CountVectorizer показала себя намного лучше. Точность данной модели составляет 87%. word2vec не дала хороших результатов. Точность только 58%.