МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Искусственные нейронные сети»

Тема: «Классификация обзоров фильмов»

Студентка гр. 7383 _____ Маркова А.В.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Прогноз успеха фильмов по обзорам (Predict Sentiment From Movie Reviews).

Классификация последовательностей – это проблема прогнозирующего моделирования, когда у вас есть некоторая последовательность входных данных в пространстве или времени, и задача состоит в том, чтобы предсказать категорию для последовательности.

Проблема усложняется тем, что последовательности могут различаться по длине, состоять из очень большого словарного запаса входных символов и могут потребовать от модели изучения долгосрочного контекста или зависимостей между символами во входной последовательности.

В данной лабораторной работе также будет использоваться датасет IMDb, однако обучение будет проводиться с помощью рекуррентной нейронной сети.

Порядок выполнения работы.

1. Ознакомиться с рекуррентными нейронными сетями;
2. Изучить способы классификации текста;
3. Ознакомиться с ансамблированием сетей;
4. Построить ансамбль сетей, который позволит получать точность не менее 97%.

Требования.

1. Найти набор оптимальных ИНС для классификации текста;
2. Провести ансамблирование моделей;
3. Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей;
4. Провести тестирование сетей на своих текстах (привести в отчете).

Ход работы.

Классификация – один из разделов машинного обучения, посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект – значит, указать номер (или наименование класса), к которому относится данный объект.

IMDb – набор данных для оценки моделей машинного обучения по задаче классификации отзывов к фильмам на положительные и отрицательные, опираясь на текст отзывов.

Набор данных представляет собой множество из 50000 самых разных отзывов к кинолентам в интернет-базе фильмов (Internet Movie Database). Набор разбит на 25000 обучающих и 25000 контрольных отзывов, каждый набор на 50% состоит из отрицательных и на 50% из положительных отзывов.

Положительные отзывы помечаются, как единицы, а отрицательные – нули. Классификация отзывов к фильмам – это пример бинарной классификации.

Была построена нейронная сеть, разработанный код представлен в приложении А.

Были созданы и обучены три модели искусственной нейронной сети, решающие задачу определения настроения обзора. Первая нейронная сеть является простой рекуррентной сетью. Её модель представлена на рис. 1.

```
def build_model_1():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
    model.add(LSTM(100))
    model.add(Dropout(0.3, noise_shape=None, seed=None))
    model.add(Dense(50, activation="relu"))
    model.add(Dropout(0.25, noise_shape=None, seed=None))
    model.add(Dense(1, activation="sigmoid"))
```

Рисунок 1 – Первая модель нейронной сети.

Вторая нейронная сеть является рекуррентной сверточной сетью. Её модель показана на рис. 2.

```
def build_model_2():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
    model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))
    model.add(LSTM(50))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation='sigmoid'))
```

Рисунок 2 – Вторая модель нейронной сети.

Третья нейронная сеть является сверточной сетью. Её модель представлена на рис. 3.

```
def build_model_3():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
    model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))
    model.add(Conv1D(filters=64, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.4))
    model.add(Flatten())
    model.add(Dense(1, activation='sigmoid'))
```

Рисунок 3 – Третья модель нейронной сети.

Поскольку это проблема двоичной классификации, в качестве функции потерь используется журнал потерь (`binary_crossentropy` в Керасе). Используется эффективный алгоритм оптимизации `Adam`. Модель подходит

только для 2 эпох, потому что она быстро решает проблему. Графики потери и точности обучения трёх архитектур приведены на рис. 4 - 9 соответственно

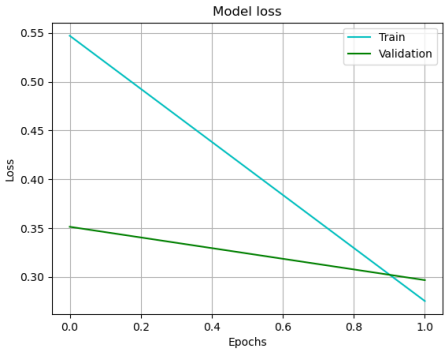

Рисунок 4 – График потери первой инс.

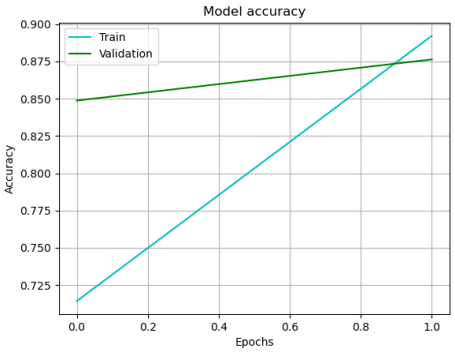

Рисунок 5 – График точности первой инс.

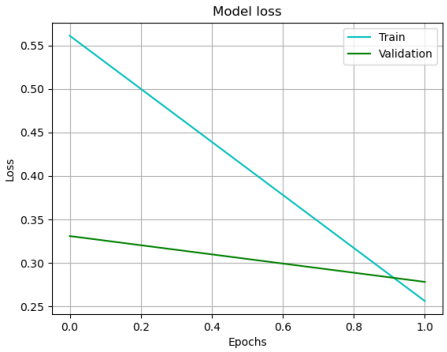

Рисунок 6 – График потери второй инс.

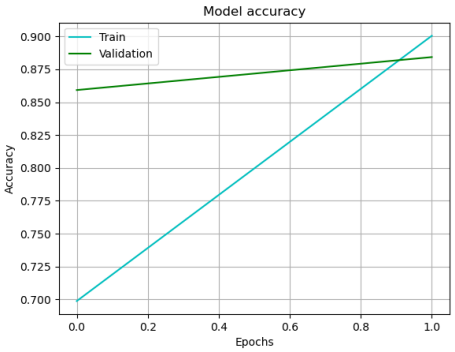

Рисунок 7 – График точности второй инс.

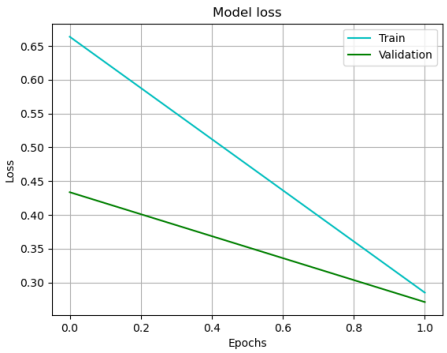

Рисунок 8 – График потери третьей инс.

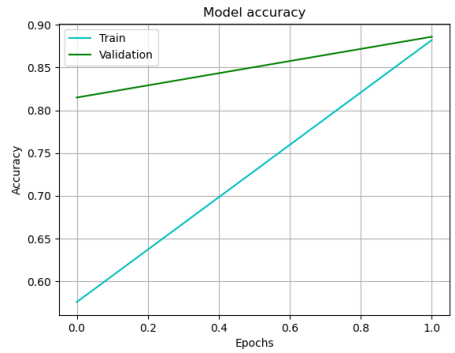

Рисунок 9 – График точности третьей инс.

Точность для каждой из нейронных сетей равна 87.64%, 88.43% и 88.61% соответственно.

Для ансамблирования моделей была написана функция `ensembling_models`. Точности ансамблей: ансамбль первой и второй сетей – 88.56%, ансамбль второй и третьей сетей – 88.83%, ансамбль первой и третьей сетей – 88.70%, ансамбль всех трёх сетей – 88.83%. Как видно, наиболее удачным ансамблем является комбинация второй и третьей.

Была написана функция для загрузки пользовательского текста `loading_and_testing`, она также позволяет прогнозировать успехи фильма. Фрагмент кода, в котором обрабатывается пользовательский датасет представлен на рис. 10.

```
def loading_and_testing():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    test_y = np.array(review_mood).astype("float64")
    for string in reviews:
        string = string.lower()
        words = string.replace(',', ' ').replace('.', ' ').replace('?', ' ').replace('\n', ' ').split()
        num_words = []
        for word in words:
            word = word.lower()
            word = dictionary.get(word)
            if word is not None and word < 10000:
                num_words.append(word)
        test_x.append(num_words)
    test_x = sequence.pad_sequences(test_x, maxlen=max_review_length)
```

Рисунок 10 – Загрузка пользовательского текста.

Каждая строка-обзор представляет собой массив индексов слов в IMDb. Пользовательский датасет показан на рис. 11.

```
reviews = [
    "I believe all the hype. Everything that every person said...yes, this show is AMAZING, WONDERFUL.",
    "The best movie I've ever seen. Great storyline and great acting.",
    "A awful and boring movie that is impossible to watch.",
    "It really is horribly inert, and every time Downey opens his mouth to say something unintelligible, the film dies a bit more. I do not advise watch
    "Dolittle is great fun for all the family, full of wonder and adventure, and it brings with it a great message of friendship against the odds."
]
```

Рисунок 11 – Пользовательские обзоры.

График точности оценки фильма при прогоне через пользовательский датасет из пяти отзывов представлен на рис. 12. Точность ансамбля составила

приблизительно 80%, то есть нейронная сеть правильно классифицировала четыре из пяти отзыва.

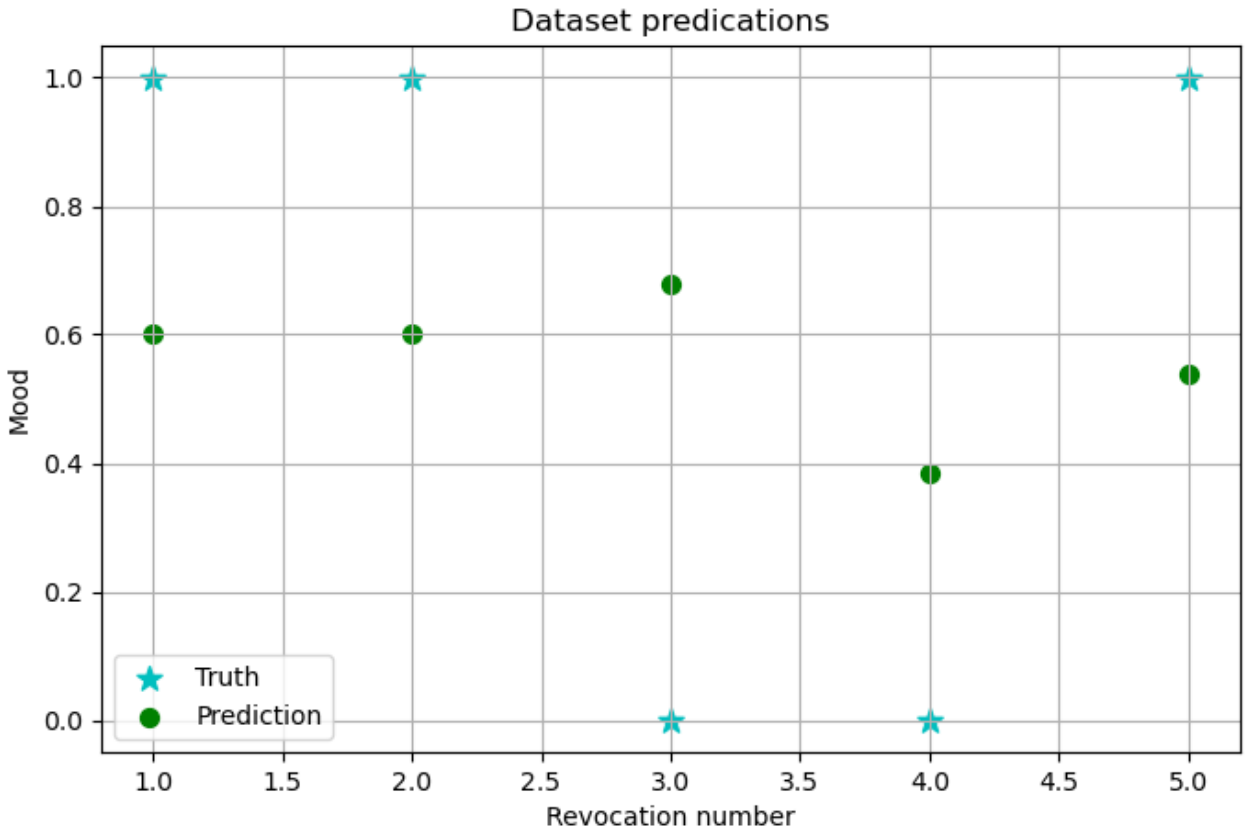

Рисунок 12 – График точности классификации отзывов к фильмам.

Выводы.

В ходе выполнения лабораторной работы были найдены оптимальные сети, некоторые сети были объединены в ансамбли. Была также продемонстрирована работа ансамбля на собственном тексте.

ПРИЛОЖЕНИЕ А

```
    # Подключение модулей
    import numpy as np
    import matplotlib.pyplot as plt
    from keras.datasets import imdb
    from keras.preprocessing import sequence
    from keras.layers.embeddings import Embedding
    from keras.models import Sequential, load_model, Input
    from keras.layers import Dense, LSTM, Dropout, Conv1D,
MaxPooling1D, Flatten

    reviews = ["I believe all the hype. Everything that every person
said...yes, this show is AMAZING, WONDERFUL.",
               "The best movie I've ever seen. Great storyline and
great acting.",
               "A awful and boring movie that is impossible to
watch.",
               "It really is horribly inert, and every time Downey
opens his mouth to say something unintelligible, the film dies a bit
more. I do not advise watching this film, it is disgusting.",
               "Dolittle is great fun for all the family, full of
wonder and adventure, and it brings with it a great message of
friendship against the odds."
    ]

    epochs = 2
    batch_size = 256
    top_words = 10000
    max_review_length = 500
    embedding_vector_length = 32
    review_mood = [1.0, 1.0, 0.0, 0.0, 1.0]

    # Загрузка набора данных IMDb
    def load_data():
        (training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=10000)
        training_data = sequence.pad_sequences(training_data,
maxlen=max_review_length)
        testing_data = sequence.pad_sequences(testing_data,
maxlen=max_review_length)
        return (training_data, training_targets), (testing_data,
testing_targets)
```

```
    # Создание моделей
    def build_model_1():
        model = Sequential()
        model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
        model.add(LSTM(100))
        model.add(Dropout(0.3, noise_shape=None, seed=None))
        model.add(Dense(50, activation="relu"))
        model.add(Dropout(0.25, noise_shape=None, seed=None))
        model.add(Dense(1, activation="sigmoid"))

        # Инициализация параметров обучения
        model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
        return model

    def build_model_2():
        model = Sequential()
        model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
        model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
        model.add(MaxPooling1D(pool_size=2))
        model.add(Dropout(0.3))
        model.add(LSTM(50))
        model.add(Dropout(0.3))
        model.add(Dense(1, activation='sigmoid'))

        # Инициализация параметров обучения
        model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
        return model

    def build_model_3():
        model = Sequential()
        model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
        model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
        model.add(MaxPooling1D(pool_size=2))
        model.add(Dropout(0.3))
        model.add(Conv1D(filters=64, kernel_size=3, padding='same',
activation='relu'))
        model.add(MaxPooling1D(pool_size=2))
```

```
            model.add(Dropout(0.4))
            model.add(Flatten())
            model.add(Dense(1, activation='sigmoid'))

            # Инициализация параметров обучения
            model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
            return model

        # Построение графиков
        def plotting(hist):
            loss = hist.history['loss']
            val_loss = hist.history['val_loss']
            acc = hist.history['accuracy']
            val_acc = hist.history['val_accuracy']

            plt.plot(loss, 'c', label='Train')
            plt.plot(val_loss, 'g', label='Validation')
            plt.title('Model loss')
            plt.xlabel('Epochs')
            plt.ylabel('Loss')
            plt.legend()
            plt.grid()
            plt.show()
            plt.clf()

            plt.plot(acc, 'c', label='Train')
            plt.plot(val_acc, 'g', label='Validation')
            plt.title('Model accuracy')
            plt.xlabel('Epochs')
            plt.ylabel('Accuracy')
            plt.legend()
            plt.grid()
            plt.show()

        def train_models():
            (training_data, training_targets), (testing_data,
testing_targets) = load_data()
            model1 = build_model_1()
            model2 = build_model_2()
            model3 = build_model_3()
```

```
        # Обучение первой сети
        hist = model1.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=epochs,
batch_size=batch_size)
        scores = model1.evaluate(testing_data, testing_targets,
verbose=0)
        print("-------------------------------------------------")
        print("Accuracy №1: %.2f%%" % (scores[1] * 100))
        model1.save('model1.h5')
        plotting(hist)

        # Обучение второй сети
        hist = model2.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=epochs,
batch_size=batch_size)
        scores = model2.evaluate(testing_data, testing_targets,
verbose=0)
        print("-------------------------------------------------")
        print("Accuracy №2: %.2f%%" % (scores[1] * 100))
        model2.save('model2.h5')
        plotting(hist)

        # Обучение третьей сети
        hist = model3.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=epochs,
batch_size=batch_size)
        scores = model3.evaluate(testing_data, testing_targets,
verbose=0)
        print("-------------------------------------------------")
        print("Accuracy №3: %.2f%%" % (scores[1] * 100))
        model3.save('model3.h5')
        plotting(hist)

    def ensembling_models():
        (_, _), (testing_data, testing_targets) = load_data()
        model1 = load_model("model1.h5")
        model2 = load_model("model2.h5")
        model3 = load_model("model3.h5")
        models = [model1, model2, model3]
        results = []
        accs = []
        labels = []
        for model in models:
            results.append(model.predict(testing_data))
```

```
            result = np.array(results[-1])
            result = np.reshape(result, result.shape[0])
            result = np.greater_equal(result, np.array([0.5]),
dtype=np.float64)
            acc = 1 - np.abs(testing_targets-result).mean(axis=0)
            accs.append(round(acc, 4))
            labels.append(str(len(results)))
        pairs = [(0, 1), (1, 2), (0, 2)]
        for (i, j) in pairs:
            result = np.array([results[i], results[j]]).mean(axis=0)
            result = np.reshape(result, result.shape[0])
            result = np.greater_equal(result, np.array([0.5]),
dtype=np.float64)
            acc = 1 - np.abs(testing_targets-result).mean(axis=0)
            accs.append(round(acc, 4))
        result = np.array(results).mean(axis=0)
        result = np.reshape(result, result.shape[0])
        result = np.greater_equal(result, np.array([0.5]),
dtype=np.float64)

        acc = 1 - np.abs(testing_targets-result).mean(axis=0)
        accs.append(round(acc, 4))
        print("-------------------------Accuracy-----------------")
        print(accs)

    def loading_and_testing():
        dictionary = dict(imdb.get_word_index())
        test_x = []
        test_y = np.array(review_mood).astype("float64")
        for string in reviews:
            string = string.lower()
            words = string.replace(',', ' ').replace('.', '
').replace('?', ' ').replace('\n', ' ').split()
            num_words = []
            for word in words:
                word = word.lower()
                word = dictionary.get(word)
                if word is not None and word < 10000:
                    num_words.append(word)
            test_x.append(num_words)
        test_x = sequence.pad_sequences(test_x,
maxlen=max_review_length)

        model1 = load_model("model1.h5")
```

```
        model2 = load_model("model2.h5")
        model3 = load_model("model3.h5")

        predictions1 = model1.predict(test_x)
        predictions2 = model2.predict(test_x)
        predictions3 = model3.predict(test_x)
        predictions = np.divide(np.add(predictions2, predictions3),
2)

        print("-----------------------------------------------------")
        print("1:")
        print(predictions1, "\n")
        print("2:")
        print(predictions2, "\n")
        print("3:")
        print(predictions3, "\n")
        print("Ensembling:")
        print(predictions, "\n")

        plt.figure(3, figsize=(8, 5))
        plt.title("Dataset predications")
        plt.scatter([1, 2, 3, 4, 5], review_mood, marker='*', c='c',
s=100, label='Truth')
        plt.scatter([1, 2, 3, 4, 5], predictions, c='g', s=50,
label='Prediction')
        plt.xlabel('Revocation number')
        plt.ylabel('Mood')
        plt.legend()
        plt.grid()
        plt.show()
        plt.clf()

    train_models()
    ensembling_models()
    loading_and_testing()
```