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

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине «Искусственные нейронные сети»

Тема: «Классификация обзоров фильмов»

Студент гр. 7381 ____ Дорох С.В.

Преподаватель ____ Жукова Н. А.

Санкт-Петербург
2020

Цель работы.

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.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- 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.

1. Были созданы и обучены две модели искусственной нейронной

сети, решающей задачу определения настроения обзора. Первая нейронная сеть рекуррентная с добавлением полносвязных слоев и слоев разреживания. Вторая нейронная сеть рекуррентная с добавлением свёрточного слоя и слоя пуллинга. Функция создания моделей представлена на рисунке 1.

```
def build_models(num=1):
    model = models.Sequential()
    model.add(layers.Embedding(top_words, embedding_vector_length, input_length=max_review_length))
    if num == 1:
        model.add(layers.Dropout(0.3))
    else:
        model.add(layers.Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
        model.add(layers.MaxPooling1D(pool_length=2))
    model.add(layers.LSTM(100))
    if num == 1:
        model.add(layers.Dropout(0.3))
    model.add(layers.Dense(1, activation="sigmoid"))
    model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
    return model
```

Рисунок 1 – Функция создания моделей

Результаты точности обученных моделей представлены на рисунках 2 и 3.

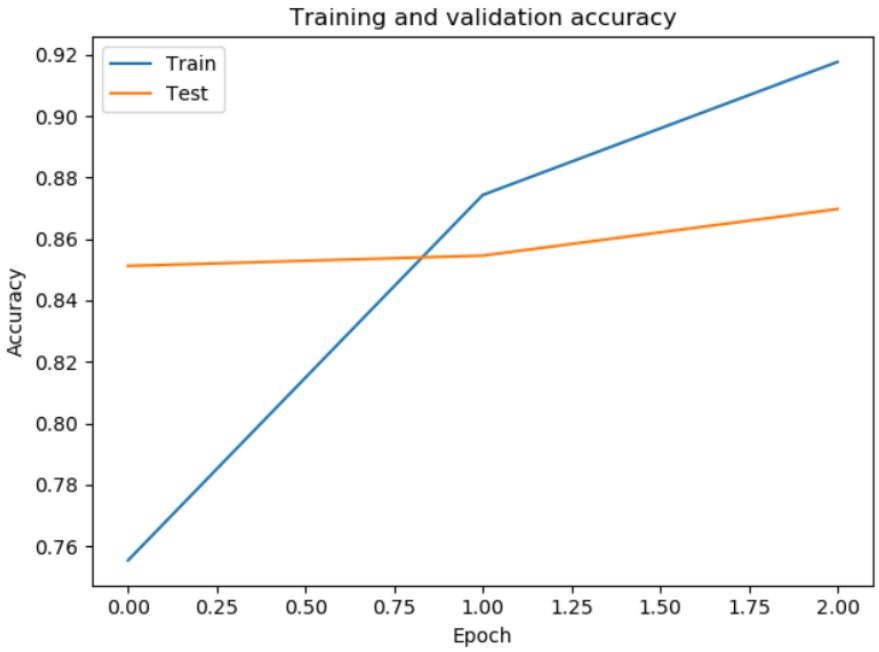

Рисунок 2 – График точности первой модели

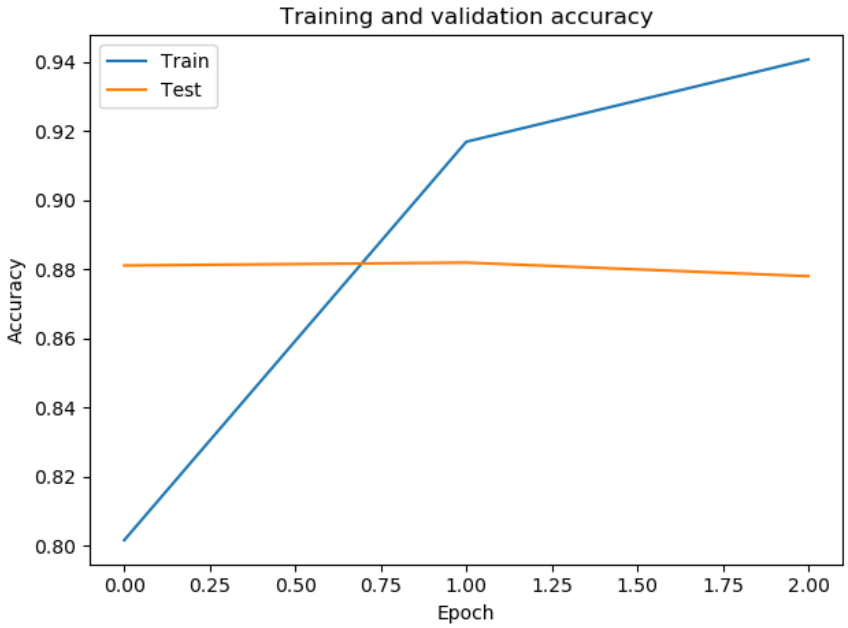

Рисунок 3 – График точности второй модели

Как можно заметить у второй модели заметно небольшое переобучение на последней эпохе. Результат точности у первой модели составил 0.8585333228111267, у второй 0.8803333242734274.

2. Ансамблирование сетей представлено как среднее арифметическое от результатов предсказаний двух моделей. Конечный результат ансамблирования составил 0.8738799989. Это немного меньше, чем точность второй модели, но на 1.5 процента больше точности модели первой.

3. Был написан декоратор, позволяющий считывать данные из текста и подготавливать их для корректного обучения модели. Для текста «quality action with amazing and exciting stunt work , as in 1999's the matrix , can be a real gem.» результат работы составил 0.8700242, оценив обзор как положительный, этот показатель на двадцать процентов выше, чем результат полученный в предыдущей работе.

Код функций и приложения в целом можно увидеть в приложении.

Выводы.

В ходе выполнения данной работы построены несколько сетей, построен ансамбль сетей. Была продемонстрирована работа ансамбля и сравнена с результатами сетей по отдельности. Результат предсказания по пользовательским данных оказался в ансамбле значительно выше, чем в 6-й работе.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
import matplotlib.pyplot as plt
import numpy as np
from keras import layers, models
from keras.datasets import imdb
from keras.preprocessing.text import text_to_word_sequence
from keras.preprocessing import sequence

(train_x, train_y), (test_x, test_y) = imdb.load_data(num_words=10000)
data = np.concatenate((train_x, test_x), axis=0)
targets = np.concatenate((train_y, test_y), axis=0)

max_review_length = 500
train_x = sequence.pad_sequences(train_x, maxlen=max_review_length)
test_x = sequence.pad_sequences(test_x, maxlen=max_review_length)

top_words = 10000
embedding_vector_length = 32


def vectorize(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results


def build_models(num=1):
    model = models.Sequential()
    model.add(layers.Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
    if num == 1:
        model.add(layers.Dropout(0.3))
    else:
        model.add(layers.Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
        model.add(layers.MaxPooling1D(pool_length=2))
    model.add(layers.LSTM(100))
    if num == 1:
        model.add(layers.Dropout(0.3))
    model.add(layers.Dense(1, activation="sigmoid"))
    model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])
    return model


models = [build_models(1), build_models(2)]

ens_acc = 0
```

```
for model in models:
    history = model.fit(train_x, train_y, validation_data=(test_x, test_y),
epochs=3, batch_size=64)
    print(model.evaluate(test_x, test_y, verbose=0))
    ens_acc += model.evaluate(test_x, test_y, verbose=0)[1]
    print("Test-Accuracy:", np.mean(history.history["val_accuracy"]))

    plt.plot(history.history["accuracy"])
    plt.plot(history.history["val_accuracy"])
    plt.title("Training and validation accuracy")
    plt.ylabel("Accuracy")
    plt.xlabel("Epoch")
    plt.legend(["Train", "Test"], loc="upper left")
    plt.show()

    plt.plot(history.history["loss"])
    plt.plot(history.history["val_loss"])
    plt.title("Training and validation loos")
    plt.ylabel("Loss")
    plt.xlabel("Epoch")
    plt.legend(["Train", "Test"], loc="upper left")
    plt.show()

print(f" Ensemble accuracy: {ens_acc / 2}")


def prepare_text(func):
    def wrapper(*args, **kwargs):
        with open(args[0]) as f:
            text = f.read().lower()
            text = text_to_word_sequence(text)
        indexes = imdb.get_word_index()
        text_indexes = []
        for item in text:
            if item in indexes and indexes[item] < 10000:
                text_indexes.append(indexes[item])
        text_indexes = sequence.pad_sequences([text_indexes],
maxlen=max_review_length)
        return func(text_indexes, **kwargs)

    return wrapper


@prepare_text
def predict_for_text(text):
    results = []
    for model in models:
        model.fit(
            train_x, train_y, epochs=3, batch_size=64, validation_data=(test_x,
test_y),
        )
```

```
        results.append(model.predict(text))
    result = sum(results) / 2
    print(result)


predict_for_text("review.txt")
```