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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искусственные нейронные сети»

Тема: «Прогноз успеха фильмов по обзорам»

Студент гр. 7381 _____ Тарасенко Е.А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Задачи.

- Ознакомиться с задачей регрессии;
- Изучить способы представления текста для передачи в ИНС;
- Достигнуть точность прогноза не менее 95%.

Требования.

1. Построить и обучить нейронную сеть для обработки текста;
2. Исследовать результаты при различном размере вектора представления текста;
3. Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте).

Ход работы.

Сначала проанализируем поведение модели при разных размерностях векторов представления текста. Результаты обучения представлены на рисунках 1 – 3.

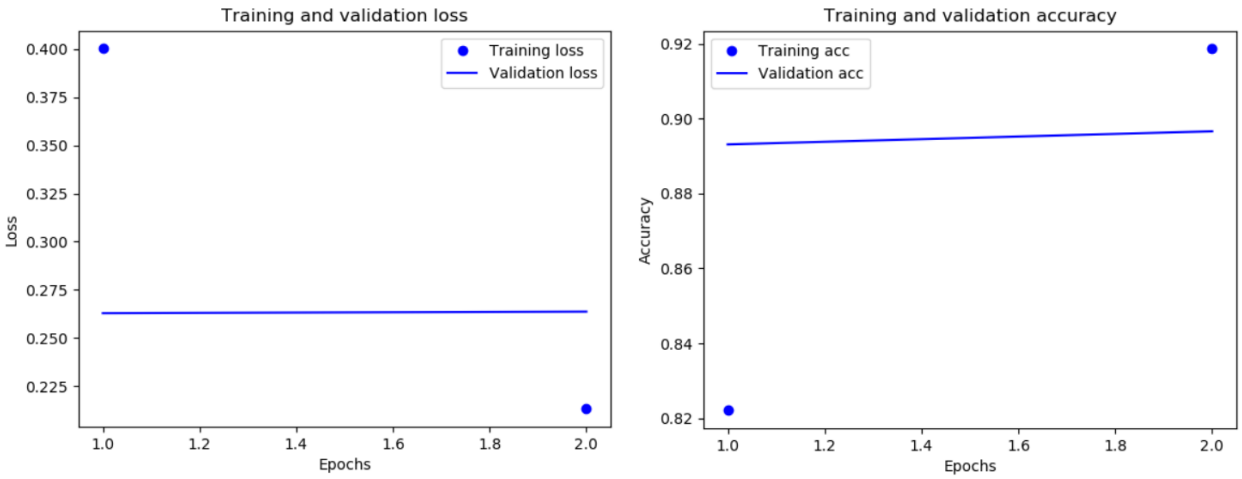

Рис. 1 – Графики ошибки и точности модели с размерностью вектора представления текста, равной 10000 (точность: 0.8948500022292136)

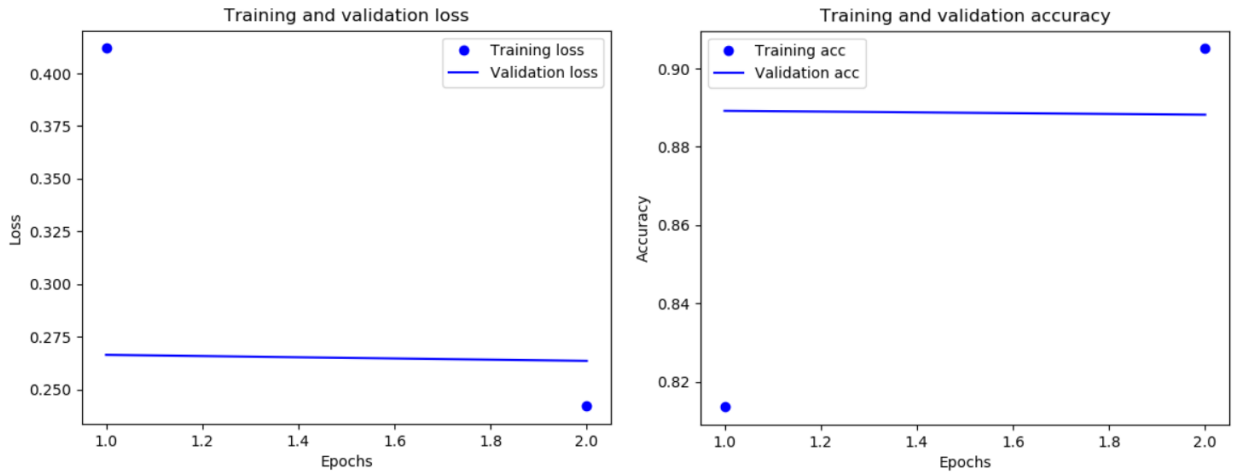

Рис. 2 – Графики ошибки и точности модели с размерностью вектора представления текста, равной 5000 (точность: 0.8886999934911728)

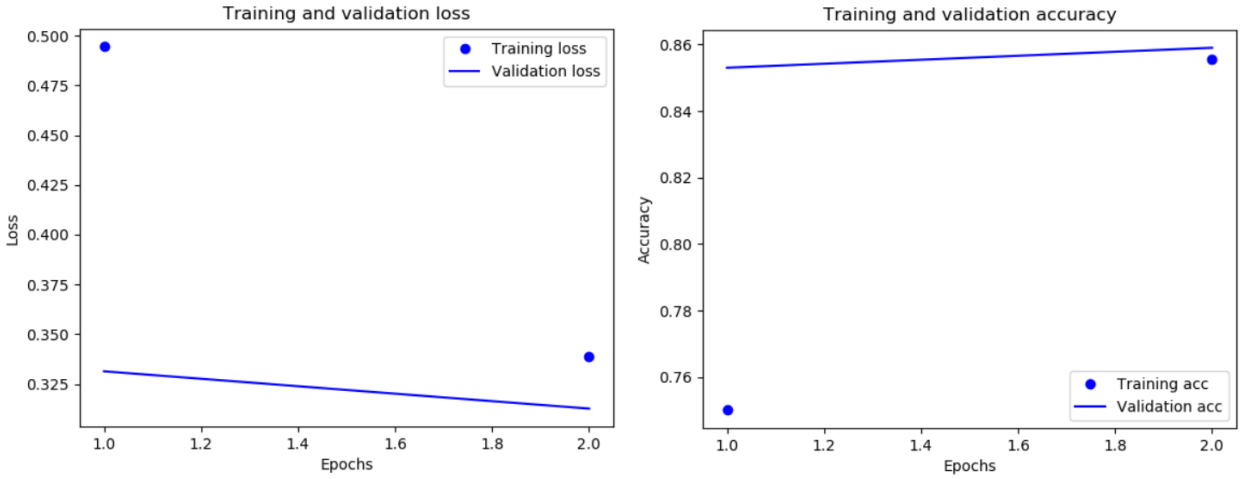

Рис. 3 – Графики ошибки и точности модели с размерностью вектора представления текста, равной 1000 (точность: 0.8560000061988831)

Таким образом, можно сделать вывод о том, что при уменьшении размерности векторов снижается точность, что обуславливается игнорированием сетью некоторых слов (с низкой частотой встречаемости в обзорах, однако все-таки придающих им дополнительный смысл). Решено оставить исходную размерность векторов представления текста (10000), так как при таком значении размерности достигается наибольшая точность. При большей размерности программа начинает обучаться слишком долго и требует много ресурсов процессора, использование векторов большей размерности нецелесообразно.

Далее был проведен ряд экспериментов с целью увеличить точность уже имеющейся модели, однако ни увеличение количества слоев, ни изменение числа нейронов на этих слоях, ни калибровка разрежения не помогли существенно улучшить результаты обучения сети. После чего было принято решение вернуться к исходной конфигурации модели, точность которой составляла почти 89,5%.

Позже была написана функция для обработки текста пользовательского обзора при помощи полученной нейронной сети. Она сначала преобразует текст для того, чтобы было удобнее обрабатывать отдельные лексемы (удалит знаки препинания и переведет все в нижний регистр); потом составит вектор представления текста и отправит его в сеть. В конечном итоге – выведет результат. Код полученной программы приведен в Приложении А.

Результаты работы сети на пользовательских обзорах:

```
Bad example: this was the worst movie i saw at worldfest and it also received the least amount of applause afterwards

0.2995906472206116

Good example: i thought this was a wonderful way to spend time on a too hot summer weekend sitting in the air conditioned theater and watching a light-hearted comedy

0.5583906471729279
```

Выводы.

В ходе выполнения данной лабораторной работы было изучено векторное представление текста для возможности работы с ним нейронных сетей. Была построена модель для оценки (ранжирования) пользовательских обзоров фильмов (положительных и отрицательных).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow
from keras.datasets import imdb


def user_review(review_text):
    review_text = review_text.replace(',', '').replace('.', '').replace(':',
'').replace('!', '').replace('?', '')
    review_text = review_text.replace('(', '').replace(')', '').replace('"',
'').replace(' - ', '').lower()
    print(review_text)
    codes = []
    for word in review_text:
        code = imdb.get_word_index().get(word)
        if code: codes.append(code)
    codes = np.array(codes)

    print(1 - model.predict(vectorize([codes]))[0][0])


# исправление ошибки с данными:
# ValueError: Object arrays cannot be loaded when allow_pickle=False
np_load_old = np.load
# modify the default parameters of np.load
np.load = lambda *a, **k: np_load_old(*a, allow_pickle=True, **k)
# call load_data with allow_pickle implicitly set to true
(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)
# restore np.load for future normal usage
np.load = np_load_old

# перестроение массивов данных
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)

# изучение данных
print("Categories:", np.unique(targets))
print("Number of unique words:", len(np.unique(np.hstack(data))))
length = [len(i) for i in data]
print("Average Review length:", np.mean(length))
print("Standard Deviation:", round(np.std(length)))
print("Label:", targets[0])
print(data[0])

index = imdb.get_word_index()
reverse_index = dict([(value, key) for (key, value) in index.items()])
decoded = " ".join([reverse_index.get(i - 3, "#") for i in data[0]])
print(decoded)


# подготовка данных
def vectorize(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
```

```
        results[i, sequence] = 1
    return results


data = vectorize(data)
targets = np.array(targets).astype("float32")

test_x = data[:10000]
test_y = targets[:10000]
train_x = data[10000:]
train_y = targets[10000:]

# создание и обучение модели
model = tensorflow.keras.models.Sequential()

# Input - Layer
model.add(tensorflow.keras.layers.Dense(50, activation="relu", input_shape=(10000,
)))

# Hidden - Layers
model.add(tensorflow.keras.layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(tensorflow.keras.layers.Dense(50, activation="relu"))
model.add(tensorflow.keras.layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(tensorflow.keras.layers.Dense(50, activation="relu"))

# Output- Layer
model.add(tensorflow.keras.layers.Dense(1, activation="sigmoid"))
model.summary()

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
results = model.fit(train_x, train_y, epochs=2, batch_size=500,
validation_data=(test_x, test_y), verbose=0)

# оценка работы модели
print(np.mean(results.history["val_acc"]))

# получение ошибки и точности в процессе обучения
loss = results.history['loss']
val_loss = results.history['val_loss']
acc = results.history['acc']
val_acc = results.history['val_acc']
epochs = range(1, len(loss) + 1)

# построение графика ошибки
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# построение графика точности
plt.clf()
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
bad_review = 'This was the worst movie I saw at WorldFest and it also received the least amount of applause afterwards!'
good_review = 'I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching a light-hearted comedy.'
print('Bad example:', end=' ')
user_review(bad_review)
print()
print('Good example:', end=' ')
user_review(good_review)

'''print('Put your review here:')
review = input()
user_review(review.lower())'''
```