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

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Искусственные нейронные сети»
Тема: Прогноз успеха фильмов по обзорам

Студент гр. 7382 _____ Гаврилов А.В.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель работы.

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

Требования к выполнению задания.

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.

Была создана и обучена модель искусственной нейронной сети. Код предоставлен в приложении А.

Архитектура нейронной сети:

- batch_size = 500
- количество эпох – 2
- использовалось 3 скрытых слоя Dense с 70 нейронами, после каждого из которых кроме последнего следует слой dropout. Выходной слой Dense с 1 нейроном.
- Метрика – точность
- Оптимизатор – Adam
- Функция потерь – бинарная кроссэнтропия

Полученная модель достигает точности прогнозирования в 0.8926.

Увеличение слоев или количества нейронов не дает значительного увеличения точности, таким образом не получилось пересечь границу в 90%.

Затем исследовали точность прогнозирования модели в зависимости от длина обзора. На рис. 1 представлен график зависимости точности от длины обзора (при длине 10 – точность 0.5062).

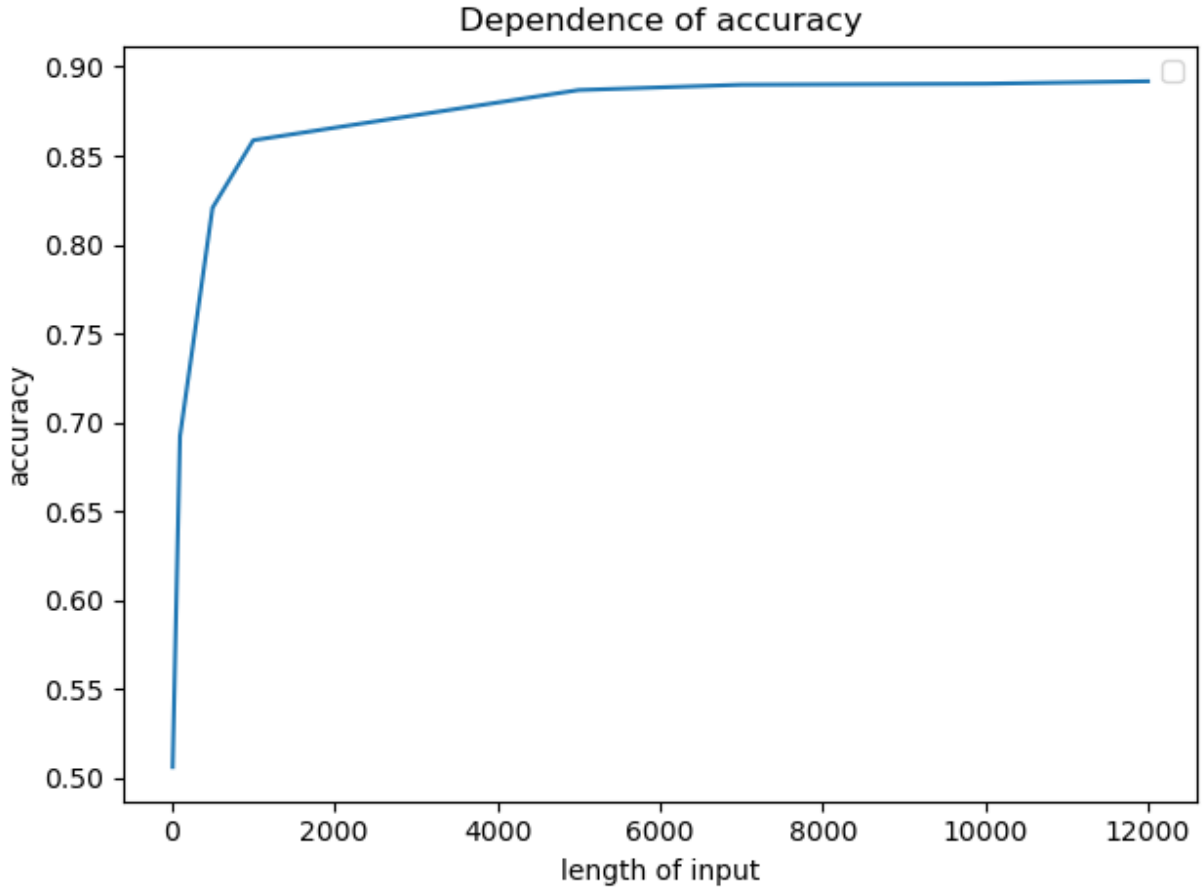


Рисунок 1 - Зависимость точности от длины входных данных

На данном рисунке можно заметить, что при маленькой длине входных данных точность значительно меньше, при длине в 1000 точность уже достигает 85% и медленно растет, таким образом можно сказать, что длине вектора входных данных в 1000 значений точности уже хватает.

При меньших размерностях количества признаков в каждом обзоре недостаточно для получения хорошей точности.

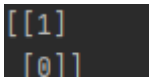
Была написана функция, которая кодирует пользовательские данные на английском языке в числа, которые соответствуют частоте встречи каждого

слова в изначальном датасете, на котором производилось обучение. Полученные вектора приводились к заданной длине и в бинарный вид.

Модель была протестирована на 2х предложений, относящихся к разным классам, но похожим по некоторым словам:

1) "I love it, it is the best film i have ever seen"
2) "Worst thing i have ever seen"

В результате прогнозирование получили результат, где первое число класс первого обзора, а второе – второго соответственно:



Данные результаты соответствуют действительности.

Выводы.

В ходе работы было изучено влияние длины обзора на точность модели, а также написана функция, которая позволяет предсказывать результат на пользовательских данных, написанных на английском языке.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras.datasets import imdb
from tensorflow.keras import Sequential


class lab6:
    def __init__(self):
        self.batch_size = 500
        self.epochs = 2
        self.results = []


    def get_data(self, length = 10000):
        (training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=length)
        data = np.concatenate((training_data, testing_data), axis=0)
        targets = np.concatenate((training_targets, testing_targets),
axis=0)
        data = self.vectorize(data, length)
        targets = np.array(targets).astype("float32")
        test_x = data[:10000]
        test_y = targets[:10000]
        train_x = data[10000:]
        train_y = targets[10000:]
        return  train_x, train_y, test_x, test_y


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


    def build_model(self):
        model = Sequential()
        model.add(layers.Dense(70, activation="relu"))
        model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
        model.add(layers.Dense(70, activation="relu"))
        model.add(layers.Dropout(0.4, noise_shape=None, seed=None))
        model.add(layers.Dense(70, activation="relu"))
        model.add(layers.Dense(1, activation="sigmoid"))
        model.compile(optimizer="adam", loss="binary_crossentropy",
```

```
metrics=["accuracy"])
        return model


    def fit(self, model, length = 10000):
        model = self.build_model()
        x_train, y_train, x_test, y_test = self.get_data(length)
        history = model.fit(x_train, y_train, epochs=self.epochs,
batch_size=self.batch_size,
                            validation_data=(x_test, y_test))
        self.results.append([length,
np.mean(history.history["val_acc"])])
        return model


    def start(self):
        model = self.build_model()
        lengths = [10, 100, 500, 1000, 5000, 7000, 10000, 12000]
        for l in lengths:
            self.fit(model, l)
        self.plot_results()
        model = self.fit(model)
        self.predict_reviews(model, ["I love it, it is the best film i
have ever seen", "Worst thing i have ever seen"])
        print(self.results)


    def plot_results(self):
        x = [el[0] for el in self.results]
        y = [el[1] for el in self.results]
        plt.plot(x, y)
        plt.title('Dependence of accuracy')
        plt.ylabel('accuracy')
        plt.xlabel('length of input')
        plt.legend()
        plt.show()


    def predict_reviews(self, model, rev):
        encoded_rev = self.encode_review(rev)
        print(model.predict_classes(encoded_rev))


    def encode_review(self, rev):
        for i, el in enumerate(rev):
            el = el.split()
            rev[i] = el
            for j, word in enumerate(el):
                code = imdb.get_word_index().get(word)
                if code is None:
                    code = 0
```

```
                rev[i][j] = code
        return self.vectorize(rev)


lab = lab6()
lab.start()
```