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

Студентка гр. 8383 _____ Кормщикова А.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы..

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

Ход работы

Были загружены обучающие и тестовые данные из набора IMDb

Датасет IMDb состоит из 50 000 обзоров фильмов от пользователей, помеченных как положительные (1) и отрицательные (0).

```
(training_data, training_targets), (testing_data, testing_targets)
= imdb.load_data(num_words=10000)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets),
axis=0)
```

Данные были векторизованы, вектор содержит ровно 10 000 чисел, т.к. Входные данные должны быть одинакового размера. Для векторизации написана функция vectorize. Также датасет был разделен на 40 000 обучающих данных, и 10 000 тестовых данных.

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

    return results
```

Была создана и обучена модель ИНС: модель состоит их входного слоя, двух скрытых слоев с двумя слоями dropout и выходного слоя.

```
model = models.Sequential()
# Input - Layer
model.add(layers.Dense(50, activation="relu",
input_shape=(10000, )))
# Hidden - Layers
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
```

```
model.add(layers.Dense(50, activation="relu"))
# Output- Layer
model.add(layers.Dense(1, activation="sigmoid"))
model.summary()

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

print(np.mean(history.history["val_accuracy"]))
```

Схема модели

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                500050
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                2550
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
```

Total params: 505,201

Trainable params: 505,201

Non-trainable params: 0

Модель была обучена на 15 эпохах, графики показали что модель действительно переобучается после 2й эпохи, поэтому количество эпох было уменьшено до 2х.

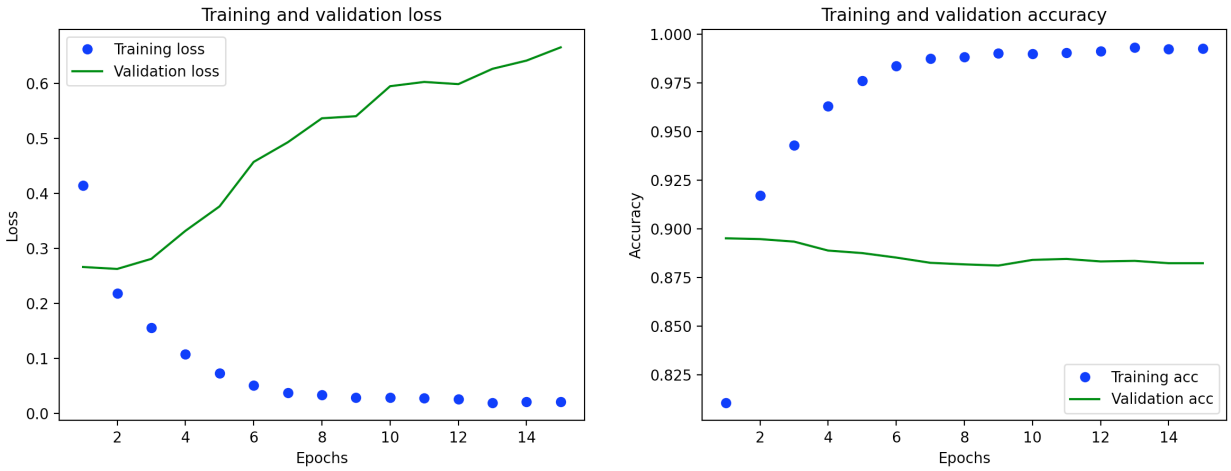


Рисунок 1 -Графики точности и потерь

Модель была протестирована при различных размерах вектора представления текста:

1) Длина вектора 10 000:

Epoch 1/2

80/80 [==============================] - 10s 109ms/step - loss: 0.5311 - accuracy: 0.7325 - val_loss: 0.2607 - val_accuracy: 0.8954

Epoch 2/2

80/80 [==============================] - 3s 41ms/step - loss: 0.2158 - accuracy: 0.9197 - val_loss: 0.2627 - val_accuracy: 0.8955

0.8954499959945679

2) Длина вектора 20 000:

Total params: 1,005,201

Trainable params: 1,005,201

Non-trainable params: 0

Epoch 1/2

80/80 [==============================] - 11s 127ms/step - loss: 0.5026 - accuracy: 0.7625 - val_loss: 0.2535 - val_accuracy: 0.8992

Epoch 2/2

80/80 [==============================] - 6s 68ms/step - loss: 0.1829 - accuracy: 0.9333 - val_loss: 0.2629 - val_accuracy: 0.8946

0.8968999981880188

3) Длина вектора 30 000:

Total params: 1,505,201

Trainable params: 1,505,201

Non-trainable params: 0

Epoch 1/2

80/80 [==============================] - 19s 218ms/step - loss: 0.5162 - accuracy: 0.7553 - val_loss: 0.2603 - val_accuracy: 0.8963

Epoch 2/2

80/80 [==============================] - 11s 133ms/step - loss: 0.2074 - accuracy: 0.9221 - val_loss: 0.2652 - val_accuracy: 0.8964

0.8963499963283539

4) Длина вектора 5 000

Total params: 255,201

Trainable params: 255,201

Non-trainable params: 0

Epoch 1/2

80/80 [==============================] - 3s 26ms/step - loss: 0.5551 - accuracy: 0.7007 - val_loss: 0.2727 - val_accuracy: 0.8892

Epoch 2/2

80/80 [==============================] - 1s 13ms/step - loss: 0.2439 - accuracy: 0.9052 - val_loss: 0.2692 - val_accuracy: 0.8878

0.8884999752044678

5) Длина вектора 2500

Total params: 130,201

Trainable params: 130,201

Non-trainable params: 0

Epoch 1/2

80/80 [==============================] - 2s 19ms/step - loss: 0.5578 - accuracy: 0.6955 - val_loss: 0.2954 - val_accuracy: 0.8750

Epoch 2/2

80/80 [==============================] - 1s 13ms/step - loss: 0.2760 - accuracy: 0.8870 - val_loss: 0.2796 - val_accuracy: 0.8833

0.879150003194809

Анализируя полученные данные, можно сказать что при увеличении размера вектора точность не увеличилась: однако можно сказать что количество параметров сильно возрастает, что увеличивает расходуемую память. При уменьшении размера вектора точность уменьшается. Для вектора была выбрана оптимальная длина - 10000.

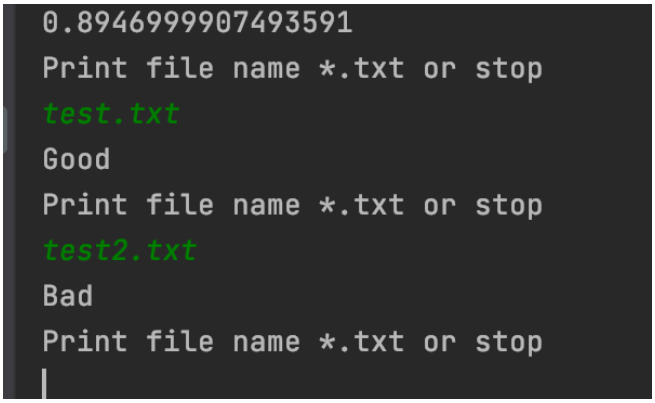
Была написана функция, позволяющая считать пользовательский текст из файла. Функция получает на вход имя файла, считывает данные и подготавливает их к виду, удобному для ИНС, функция возвращает эти данные.

```
def textFromFile(fileName, dimension=10000):
    file = open(fileName, "r")
    text = file.read()
    file.close()
    text.lower()
    tt = str.maketrans(dict.fromkeys("!\"#$%&()*+,-./:;<=>?
@[\]^_`{|}~"))
    text = text.translate(tt).split()
    index = imdb.get_word_index()
    codedText = []
    for word in text:
        i = index.get(word)
        if i is not None and i < dimension:
            codedText.append(i+3)
    codedText = vectorize(np.asarray([codedText]))
    return codedText
```

Для тестирования были выбраны два обзора на фильм "Druk" - один отрицательный другой положительный:

Положительный отзыв находится в файле test.txt, отрицательный в файле test2.txt. Тексты отзывов предоставлены в приложении Б.

Результаты тестирования:



Выводы.

Во время выполнения лабораторной работы была реализована ИНС, которая прогнозирует успех фильма по обзорам. Реализована функция для пользовательского ввода из файла. Было исследовано влияние размера вектора представления текста на работу сети.

ПРИЛОЖЕНИЕ А

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

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

    return results

def textFromFile(fileName, dimension=10000):
    file = open(fileName, "r")
    text = file.read()
    file.close()
    text.lower()
    tt = str.maketrans(dict.fromkeys("!\"#$%&()*+,-./:;<=>?
@[\]^_`{|}~"))
    text = text.translate(tt).split()
    index = imdb.get_word_index()
    codedText = []
    for word in text:
        i = index.get(word)
        if i is not None and i < dimension:
            codedText.append(i+3)
    codedText = vectorize(np.asarray([codedText]))
    return codedText


dimension=10000
(training_data, training_targets), (testing_data, testing_targets)
= imdb.load_data(num_words=dimension)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets),
axis=0)
#
# 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)
data = vectorize(data, dimension)
targets = np.array(targets).astype("float32")
test_x = data[:10000]
test_y = targets[:10000]
train_x = data[10000:]
train_y = targets[10000:]
print(data[0])

model = models.Sequential()
# Input - Layer
model.add(layers.Dense(50, activation="relu",
input_shape=(dimension, )))
# Hidden - Layers
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
# Output- Layer
model.add(layers.Dense(1, activation="sigmoid"))
model.summary()

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

print(np.mean(history.history["val_accuracy"]))

while(1):
    print("Print file name *.txt or stop")
    fileName = input()
    if fileName == "stop":
        break
    text = textFromFile(fileName, dimension)
    res = model.predict(text)
    if res >= 0.5:
        print("Good")
    else:
        print("Bad")
```

ПРИЛОЖЕНИЕ Б

Положительный обзор, файл test.txt

Shouldn't we slap a glass?

What if a person from the very birth lacks a little alcohol in their blood ?! So, in any case, according to one of the heroes of the new film Thomas Winterberg "Druk", the Norwegian psychiatrist Finn Skarderud thinks. Alcohol helps a person to become more relaxed, joyful and gives the opportunity to enjoy all the delights of life. This does not mean that you need to thump like hell, you just need to drink a few glasses a day and keep those same magic 0.5 ppm in your blood. Moreover, only on weekdays and during working hours.

"Druk" is a wonderful tragicomedy, the protagonists of which are four teachers of the Danish school, who decided on themselves (exclusively as a scientific experiment) to test the above theory. This picture is not so much about drunkenness as about the midlife crisis and the extinct gaze of our teachers. It is no coincidence that it is teachers, role models, guides to the world of knowledge, and indeed life itself, that are in the foreground. After all, it is absolutely not clear which teacher is better / worse, the one who mumbles a paragraph from the textbook without enthusiasm (and even the one that has already been passed in the last lesson), or who gushes with ideas and tries to teach his students (who already like to drink beer on a cozy lake) necessary knowledge, albeit slightly under the fly ?!

"Druk" is not a solemn song to drunkenness, showing that drinking is cool, that alcohol is the solution to all problems, but also not a moralizing propaganda that explains to every fool that drinking intoxicating drinks is the last thing, and sobriety is above all. It so happened that alcohol is a part of our life. And as each of us copes with this in his own way, the heroes of the film will have to go through different stages of alcoholic intoxication and get out of them in different ways. Well, the subtle humor with the excellent acting of the alcoholic quartet led by the star of the pan-European cinema Mads Mikkelsen, and the hilarious inserts of chronicle frames with famous politicians who put the collars, should move both an inveterate teetotaler and an ardent advocate of a healthy lifestyle. 9 out of 10

Отрицательный обзор, файл test2.txt

When I moved on to watching this film, I expected to see what our directors and screenwriters had failed to realize - a literate story about where alcohol leads. And it seems that an excellent starting point was provided for this - an experiment that assumes the positive effects of alcohol.

But no! Do not be fooled, the film is ideologically absolutely identical to those Russian films that the same Russian viewers did not like. He is completely devoid of morality. The characters in it do not change qualitatively - they do not get rid of their flaws. Moreover, some of the characters do not have these flaws at all, which cancels even their fictitious development.

And this is my main bewilderment: why did they love a picture with the same idea and the same mediocre development of the heroes, but they hate the ideologically similar pictures filmed in Russia? An alcohol-related package that is not disclosed. Unpleasant and naturalistic scenes of behavior of drunk people. Is it really all about the actors, and the fact of the presence of a famous person in the picture qualitatively changes its message and the development of the characters? Mikkelsen alone automatically guarantees a film 90% positive reviews?

I don't think so.