Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Отчёт по лабораторной работе №4

Разработка нейронной сети

Выполнил:

студент гр. ИП-111

Кузьменок Д.В.

Проверил:

Старший преподаватель кафедры ПМиК

Дементьева К.И.

Новосибирск, 2024 г.

Задание

Целью данной лабораторной работы является разработка нейронной сети для решения задачи классификации или регрессии в зависимости от набора данных в рамках варианта. Лабораторная работа предполагает разработку на языке программирования Python с использованием библиотеки Keras.

Вариант задания:

3) Определение эмоционального окраса рецензии фильма (IMDB movie review sentiment classification dataset)

Все наборы данных доступны по ссылке: https://keras.io/api/datasets/

При разработке нейронной сети следует соблюсти наличие необходимых составляющих исходя из следующих вариантов:

3) Нейросеть должна состоять из пяти полносвязных слоёв, обязательное использование ActivityRegularization, в качестве оптимизатора использовать RMSprop.

Результаты

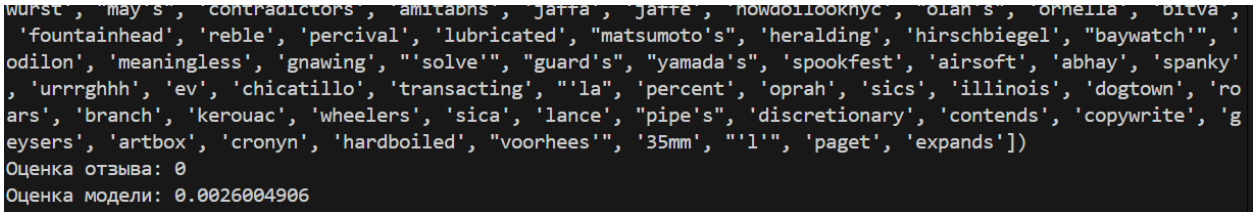
Программа у меня проходит по максимальному количеству слов, которые будут использоваться в модели (500000), с максимальной длиной одного отзыва в 50000 знаков. Такие результаты были получены:

```
400/400 ━━━━━━━━━━━━━━━━━━━━ 28s 67ms/step - accuracy: 0.5002 - loss: 613.7764 - val_accuracy: 0.5062 - val_lo
ss: 0.4026
Epoch 2/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 22s 55ms/step - accuracy: 0.4972 - loss: 0.1726 - val_accuracy: 0.5062 - val_loss
: 0.3856
Epoch 3/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 24s 60ms/step - accuracy: 0.4990 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 4/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 23s 57ms/step - accuracy: 0.4999 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 5/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.5018 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 6/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 22s 54ms/step - accuracy: 0.4948 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 7/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 22s 54ms/step - accuracy: 0.4967 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 8/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.5027 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 9/15
```

```
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.4930 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 10/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 22s 54ms/step - accuracy: 0.5030 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 11/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.5013 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 12/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.4986 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 13/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.5017 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 14/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.5014 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
Epoch 15/15
400/400 ━━━━━━━━━━━━━━━━━━━━ 21s 53ms/step - accuracy: 0.4980 - loss: 0.0000e+00 - val_accuracy: 0.5062 - val_
loss: 0.3856
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step
25000   25000
100.0
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 6ms/step - accuracy: 0.5073 - loss: 0.5203
Test Loss: 0.267
Test Accuracy: 0.500
```

Точность получилась ≈50%. После, чтобы визуализировать работу программы, я вывожу случайный отзыв с оценкой, и той оценкой, которая рассчитала моя модель:



Таким образом, модель максимально близко приблизилась к той оценке, которая была на самом деле, несмотря на общую точность предсказания.

Код программы

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence
import numpy as np


max_words = 500000
max_len = 50000


(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_words)
x_train = sequence.pad_sequences(x_train, maxlen=max_len)
x_test = sequence.pad_sequences(x_test, maxlen=max_len)

model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(max_len,)),
    layers.ActivityRegularization(l1=0.01),
    layers.Dense(64, activation='relu'),
    layers.ActivityRegularization(l1=0.01),
    layers.Dense(32, activation='relu'),
    layers.ActivityRegularization(l1=0.01),
    layers.Dense(16, activation='relu'),
    layers.ActivityRegularization(l1=0.01),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer=keras.optimizers.RMSprop(),
loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=15, batch_size=50,
validation_split=0.2)
L = len(y_test)
correct = 0
YP = model.predict(x_test)
for i in range(L):
    y1 = np.argmax(y_test[i])
    ypred = np.argmax(YP[i])
    if ypred == y1:
        correct += 1
print(correct, ' ', L)
print(correct/L*100)

test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test Loss: {test_loss:.3f}')
print(f'Test Accuracy: {test_acc:.3f}')

random_index = np.random.randint(0, len(x_test))
random_review = x_test[random_index]
predicted_rating = model.predict(np.array([random_review]))
print("Случайный отзыв:", imdb.get_word_index().keys())
print("Оценка отзыва:", y_test[random_index])
print("Оценка модели:", predicted_rating[0][0])
```