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

ОТЧЕТ

по лабораторной работе №7
по дисциплине «Искусственные нейронные сети»
Тема: «Классификация обзоров фильмов»

Студентка гр. 8382 _____ Звегинцева Е.Н.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

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.

Задачи.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%

Требования.

- Найти набор оптимальных ИНС для классификации текста
- Провести ансамблирование моделей
- Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
- Провести тестирование сетей на своих текстах (привести в отчете)

Ход работы.

В ходе работы была создана и обучена модели искусственной нейронной сети в соответствии с условиями (код представлен в приложении).

Были построены три модели. Далее было реализовано обучение и тестирование моделей. Также была написана функция forecast, ассамблирующая имеющиеся модели нейронных сетей(через среднее).

Результат работы программы на тестовых данных представлен ниже.

```
Epoch 1/2
188/188 [==============================] - 8s 34ms/step - loss: 0.6493 - accuracy: 0.5989 - val_loss: 0.3817 - val_accuracy: 0.8411
Epoch 2/2
188/188 [==============================] - 6s 32ms/step - loss: 0.3374 - accuracy: 0.8677 - val_loss: 0.3484 - val_accuracy: 0.8643
105/105 - 1s - loss: 0.3579 - accuracy: 0.8533
Accuracy: 85.33%
Epoch 1/2
188/188 [==============================] - 5s 19ms/step - loss: 0.6564 - accuracy: 0.5634 - val_loss: 0.3723 - val_accuracy: 0.8328
Epoch 2/2
188/188 [==============================] - 3s 17ms/step - loss: 0.4823 - accuracy: 0.7654 - val_loss: 0.3454 - val_accuracy: 0.8681
105/105 - 0s - loss: 0.3476 - accuracy: 0.8644
Accuracy: 86.44%
Epoch 1/2
188/188 [==============================] - 3s 11ms/step - loss: 0.6703 - accuracy: 0.5557 - val_loss: 0.3050 - val_accuracy: 0.8733
Epoch 2/2
188/188 [==============================] - 2s 10ms/step - loss: 0.2570 - accuracy: 0.8948 - val_loss: 0.2632 - val_accuracy: 0.8928
105/105 - 0s - loss: 0.2838 - accuracy: 0.8866
Accuracy: 88.66%
Accuracy: 88.77%

Input string:
there are not many movies that make me actually laugh out loud, but this
one did on several occasions (usually comedies, even great ones, just keep
me grinning through out). This is easily one of the most enjoyable
experiences I have had at the cinema.
[[50, 26, 24, 111, 102, 15, 97, 72, 165, 462, 46, 1292, 21, 14, 31, 122,
23, 450, 5392, 1290, 60, 87, 663, 43, 401, 72, 143, 9, 714, 31, 7, 4, 91,
737, 2490, 28, 69, 33, 4, 438]]
[[0.82319105]]
[[0.66141206]]
[[0.96476525]]
[[1.]]
```

В итоге, наиболее удачным является ансамбль всех трех моделей. Точности ансамблей: первая сеть — 85.33%, вторая сеть — 86.44%, третья сеть — 88.66%, ансамбль всех трех сетей — 88,77%.

Также была написана функция загрузки собственного текста и прогнозирования успеха фильма по этому тексту - user_load.

Выводы.

В ходе выполнения данной работы было произведено ознакомление с рекуррентными нейронными сетями и ансамблированием сетей, а также классификация обзоров фильмов с помощью рекуррентной сети.

ПРИЛОЖЕНИЕ

Исходный код

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()


import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Dropout, LSTM, Conv1D,
MaxPool1D, Flatten, Embedding
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.datasets import imdb
from sklearn.metrics import accuracy_score


(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)

X_test = data[:10000]
Y_test = targets[:10000]
X_train = data[10000:]
Y_train = targets[10000:]

max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
embedding_vector_length = 32



model_lstm = Sequential([
    Embedding(10000, embedding_vector_length,
input_length=max_review_length),
    LSTM(128),
    Dropout(0.3),
    Dense(64, activation="relu"),
    Dropout(0.2),
    Dense(1, activation="sigmoid")])
```

```
model_cnn = Sequential([
    Embedding(10000, embedding_vector_length,
input_length=max_review_length),
    Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'),
    MaxPool1D(pool_size=2),
    Dropout(0.3),
    Conv1D(filters=64, kernel_size=3, padding='same',
activation='relu'),
    MaxPool1D(pool_size=2),
    Dropout(0.4),
    LSTM(128),
    Dropout(0.3),
    Dense(1, activation='sigmoid')])

model_cnl = Sequential([
    Embedding(10000, embedding_vector_length,
input_length=max_review_length),
    Conv1D(filters=16, kernel_size=3, padding='same',
activation='relu'),
    MaxPool1D(pool_size=2),
    Dropout(0.3),
    Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'),
    MaxPool1D(pool_size=2),
    Dropout(0.3),
    Flatten(),
    Dense(128),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
    ])


train_size = len(X_train) // 3
test_size = len(X_test) // 3
models = [model_lstm, model_cnn, model_cnl]

for i, mod in enumerate(models):
    x_train = X_train[i * train_size : (i + 1) * train_size]
    y_train = Y_train[i * train_size : (i + 1) * train_size]

    x_test = X_test[i * test_size : (i + 1) * test_size]
    y_test = Y_test[i * test_size : (i + 1) * test_size]
```

```
    mod.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    mod.fit(x_train, y_train, validation_split=0.1, epochs=2,
batch_size=64)
    scores = mod.evaluate(x_test, y_test, verbose=2)
    print("Accuracy: %.2f%%" % (scores[1]*100))


def forecast(models, x_test, load):
    combo = []

    for i, m in enumerate(models):
        if load:
          print(m.predict(x_test, verbose=0))
        combo.append(np.round(m.predict(x_test, verbose=0)))

    combo = np.asarray(combo)
    combo = np.round(np.mean(combo, 0))
    return combo


combo = forecast(models, X_test, False)
rating = accuracy_score(Y_test, combo)
print("Accuracy: %.2f%%" % (rating*100))

def user_load():
    print("Input string: ")
    words = input()
    words = words.replace(',', ' ').replace('.', ' ').replace('?', '
').replace('\n', ' ').split()
    dict_set = imdb.get_word_index()
    test_x = []
    test_y = []

    for word in words:
        if dict_set.get(word) in range(1, 10000):
            test_y.append(dict_set.get(word) + 3)
    test_x.append(test_y)
    print(test_x)
    result = sequence.pad_sequences(test_x, maxlen=max_review_length)
    print(forecast(models, result, True))

user_load()
```