**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №7**

**по дисциплине «Искусственные нейронные сети»**

Тема: Классификация обзоров фильмов

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Сосновский Д. 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

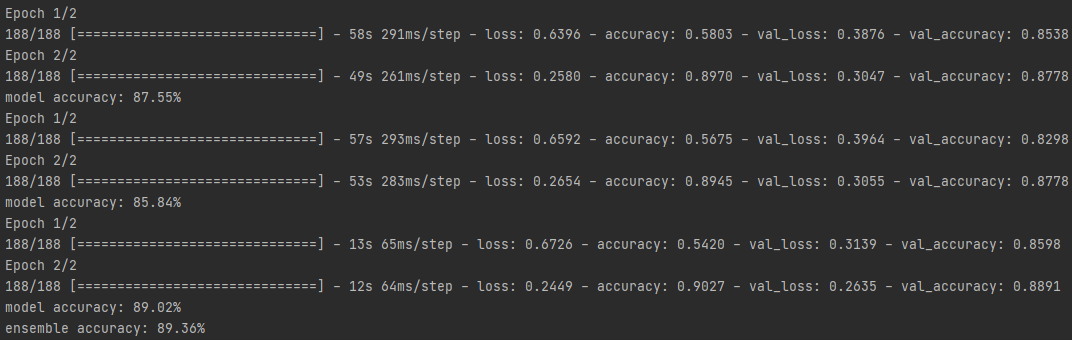
• Провести тестирование сетей на своих текстах (привести в отчете)

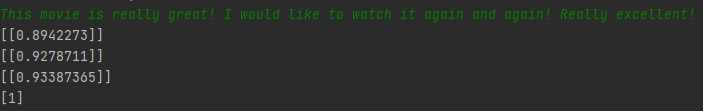
**Ход работы.**

Согласно методическим указаниям были построены и обучены нейронная сети. Исходный код программы приведён в приложении А.

Были построены три модели. Далее было реализовано обучение и тестирование моделей. Также была написана функция forecast, выполняющая ансамблирование моделей.

Ниже представлен результат тестирования программы на входных данных.





В итоге, наиболее удачным является ансамбль всех трех моделей. Точности ансамблей: первая сеть — 87.55%, вторая сеть — 85.84%, третья сеть — 89.02%, ансамбль всех трех сетей — 89,36%. Также была написана функция загрузки собственного текста и прогнозирования успеха фильма по этому тексту – read\_text\_from\_input()

**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 А**

В этом приложении приведён исходный код программы.

import numpy as np  
from keras.layers import Dense, Dropout, LSTM, Conv1D, MaxPool1D, Flatten  
from keras.models import Sequential  
from keras.layers.embeddings import Embedding  
from keras.preprocessing import sequence  
from sklearn.metrics import accuracy\_score  
import re  
from keras.datasets import imdb  
  
(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  
embedding\_vector\_length = 32  
top\_words = 10000  
  
X\_train = sequence.pad\_sequences(X\_train, maxlen=max\_review\_length)  
X\_test = sequence.pad\_sequences(X\_test, maxlen=max\_review\_length)  
  
  
def get\_model\_a():  
 model = Sequential()  
 model.add(Embedding(top\_words, embedding\_vector\_length, input\_length=max\_review\_length))  
 model.add(Conv1D(128, 3, padding='same', activation='relu'))  
 model.add(MaxPool1D(2))  
 model.add(Dropout(0.4))  
 model.add(LSTM(40, return\_sequences=True, dropout=0.5))  
 model.add(LSTM(20, dropout=0.3))  
 model.add(Dense(1, activation='sigmoid'))  
 return model  
  
  
def get\_model\_b():  
 model = Sequential()  
 model.add(Embedding(top\_words, embedding\_vector\_length, input\_length=max\_review\_length))  
 model.add(Conv1D(32, 3, padding='same', activation='relu'))  
 model.add(MaxPool1D(2))  
 model.add(Dropout(0.2))  
 model.add(LSTM(100))  
 model.add(Dense(1, activation='sigmoid'))  
 return model  
  
  
def get\_model\_c():  
 model = Sequential()  
 model.add(Embedding(top\_words, embedding\_vector\_length, input\_length=max\_review\_length))  
 model.add(Conv1D(32, 3, padding='same', activation='relu'))  
 model.add(MaxPool1D(2))  
 model.add(Dropout(0.3))  
 model.add(Conv1D(32, 3, padding='same', activation='relu'))  
 model.add(Dropout(0.2))  
 model.add(Flatten())  
 model.add(Dense(128))  
 model.add(Dropout(0.5))  
 model.add(Dense(1, activation='sigmoid'))  
 return model  
  
  
train\_size = len(X\_train) // 3  
test\_size = len(X\_test) // 3  
  
def get\_ensemble\_predictions(all\_models, x\_test, X):  
 result = []  
 for m in all\_models:  
 result.append(np.round(m.predict(x\_test, verbose=0)))  
 if X:  
 print(m.predict(x\_test, verbose=0))  
 result = np.asarray(result)  
 result = [np.round(np.mean(x)) for x in zip(\*result)]  
 return np.asarray(result).astype('int')  
  
  
all\_models = [get\_model\_a(), get\_model\_b(), get\_model\_c()]  
  
for i, model in enumerate(all\_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]  
 model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])  
 model.fit(x\_train, y\_train, validation\_split=0.1, epochs=2, batch\_size=64)  
 scores = model.evaluate(x\_test, y\_test, verbose=0)  
 print("model accuracy: %.2f%%" % (scores[1] \* 100))  
  
result\_predictions = get\_ensemble\_predictions(all\_models, X\_test, False)  
acc = accuracy\_score(Y\_test, result\_predictions)  
print("ensemble accuracy: %.2f%%" % (acc \* 100))  
  
  
def read\_text\_from\_input():  
 dictionary = imdb.get\_word\_index()  
 words = input()  
 words = re.sub(r"[^\w']", " ", words).split(' ')  
  
 valid = []  
 for word in words:  
 word = dictionary.get(word)  
 if word in range(1, 10000):  
 valid.append(word + 3)  
  
 X = []  
 X.append(valid)  
 result = sequence.pad\_sequences(X, maxlen=max\_review\_length)  
 print(get\_ensemble\_predictions(all\_models, result, True))  
  
  
read\_text\_from\_input()