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: Прогноз успеха фильмов по обзорам

Студент гр. 7383 _____ Бергалиев М.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы: прогноз успеха фильмов по обзорам.

Порядок выполнения работы.

1. Найти набор оптимальных ИНС для классификации текста
2. Провести ансамблирование моделей
3. Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей

Ход работы.

0. Исходный код программы представлен в приложении.
1. Выберем оптимальные модели сети.

Первая модель является простой рекуррентной сетью:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length,
                    input_length=max_review_length))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

Вторая модель является рекуррентной сверточной сетью:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length,
                    input_length=max_review_length))
model.add(Conv1D(filters=32, kernel_size=3, padding='same',
                 activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(LSTM(50))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

Третья модель является сверточной сетью:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length,
                    input_length=max_review_length))
```

```
model.add(Conv1D(filters=32, kernel_size=3, padding='same',
                 activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(Conv1D(filters=32, kernel_size=3, padding='same',
                 activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
```

2. Обучим модели и ансамблируем их. Результаты показаны на рис. 2.

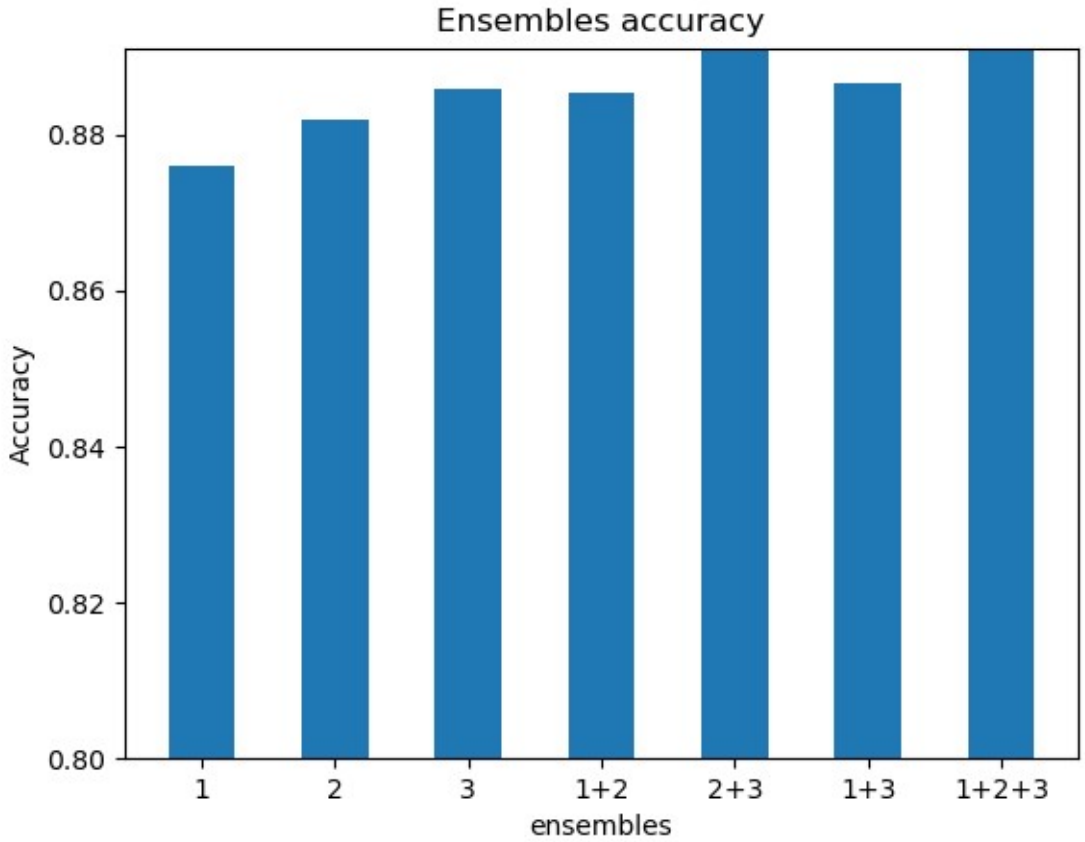

Рисунок 2 — Точность на тестовых данных различных комбинаций сетей

Как видно, наиболее удачным ансамблем является комбинация второй и третьей модели. Точности ансамблей: первая сеть — 87.6%, вторая сеть — 88.2%, третья сеть — 88.6%, ансамбль первой и второй сетей — 88.5%, ансамбль второй и третьей сетей — 89,1%, ансамбль первой и третьей сетей — 88,6%, ансамбль всех трех сетей — 89,1%.

3. Была написана функция загрузки собственного текста `loadText`. Проверим работу ансамбля на следующем тексте:

Scenario of this movie is really DUMP! But actors played very well, so it saved this film.

Ансамбль в качестве ответа выдала значение 0.9253897. Эта оценка ближе, чем полученная в предыдущей работе.

Выводы: Были найдены оптимальные сети, построен ансамбль сетей. Была продемонстрирована работа ансамбля на собственном тексте. Результат оказался лучше, чем в предыдущей работе.

ПРИЛОЖЕНИЕ

```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import GRU
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Conv1D
from tensorflow.keras.layers import MaxPooling1D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.datasets import imdb
import matplotlib.pyplot as plt

(X_train,        y_train),        (X_test,        y_test)        =
imdb.load_data(num_words=10000)
data = np.concatenate((X_train, X_test), axis=0)
targets = np.concatenate((y_train, y_test), axis=0)
print(X_train[0])

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)

top_words = 10000
embedding_vector_length = 32
def model1():
    model = Sequential()
          model.add(Embedding(top_words,   embedding_vector_length,
input_length=max_review_length))
    model.add(LSTM(100))
    model.add(Dropout(0.3))
    model.add(Dense(50, activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation='sigmoid'))
       model.compile(loss='binary_crossentropy',  optimizer='adam',
metrics=['accuracy'])
    return model

def model2():
    model = Sequential()
          model.add(Embedding(top_words,   embedding_vector_length,
input_length=max_review_length))
```

```
      model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))
    model.add(LSTM(50))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation='sigmoid'))
       model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model

def model3():
    model = Sequential()
          model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
      model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))
      model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))
    model.add(Flatten())
    model.add(Dense(1, activation='sigmoid'))
       model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model

def plot_acc(accs, labels):
    plt.bar(np.arange(len(accs)) * 2, accs, width=1)
    plt.xticks([2*i for i in range(0,len(accs))],
               labels=labels)
    plt.title('Ensembles accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('ensembles')
    plt.ylim([0.8, max(accs)])
    plt.show()

def ensemble(models):
    results = []
    accs = []
    labels = []
    for model in models:
        results.append(model.predict(X_test))
        result = np.array(results[-1])
```

```
        result = np.reshape(result, result.shape[0])
              result  =  np.greater_equal(result,  np.array([0.5]),
dtype=np.float64)
        acc = 1 - np.abs(y_test-result).mean(axis=0)
        accs.append(acc)
        labels.append(str(len(results)))
    pairs = [(0, 1), (1, 2), (0, 2)]
    for (i, j) in pairs:
        result = np.array([results[i], results[j]]).mean(axis=0)
        result = np.reshape(result, result.shape[0])
              result  =  np.greater_equal(result,  np.array([0.5]),
dtype=np.float64)
        acc = 1 - np.abs(y_test-result).mean(axis=0)
        accs.append(acc)
        labels.append(str(i+1) + '+' + str(j+1))
    result = np.array(results).mean(axis=0)
    result = np.reshape(result, result.shape[0])
           result   =   np.greater_equal(result,   np.array([0.5]),
dtype=np.float64)

    acc = 1 - np.abs(y_test-result).mean(axis=0)
    accs.append(acc)
    labels.append('1+2+3')
    print(accs)
    plot_acc(accs, labels)


def train_model(model):
     model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=2, batch_size=64)

def loadText(filename):
    punctuation = ['.',',',':',';','!','?','(',')']
    text = []
    with open(filename, 'r') as f:
        for line in f.readlines():
              text += [s.strip(''.join(punctuation)).lower() for s
in line.strip().split()]
    indexes = imdb.get_word_index()
    encoded = []
    for w in text:
        if w in indexes and indexes[w] < 10000:
            encoded.append(indexes[w])
    return np.array(encoded)

def testOwnText(models, text):
```

```
    results = []
    for model in models:
        results.append(model.predict(text))
    result = np.array(results).mean(axis=0)
    result = np.reshape(result, result.shape[0])
    print(result)

model1 = model1()
model2 = model2()
model3 = model3()

train_model(model1)
train_model(model2)
train_model(model3)
ensemble([model1, model2, model3])
filename = 'text.txt'
text=loadText(filename)
text = sequence.pad_sequences([text], maxlen=max_review_length)
testOwnText([model1, model2, model3], text)
```