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

ОТЧЕТ

по лабораторной работе 7

по дисциплине «Искусственные нейронные сети»

Тема: «Классификация обзоров фильмов»

Студентка гр. 7383 Чемова К.А.

Преподаватель Жукова Н.А.

Санкт-Петербург
2020

Цель.

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.

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;
4. Провести тестирование сетей на своих текстах (привести в отчете).

Ход работы.

Была построена нейронная сеть, код которой представлен в приложении А.

После запуска нейронной сети были получены графики точности работы сети в разных комбинациях. График точности представлен на рис 1.

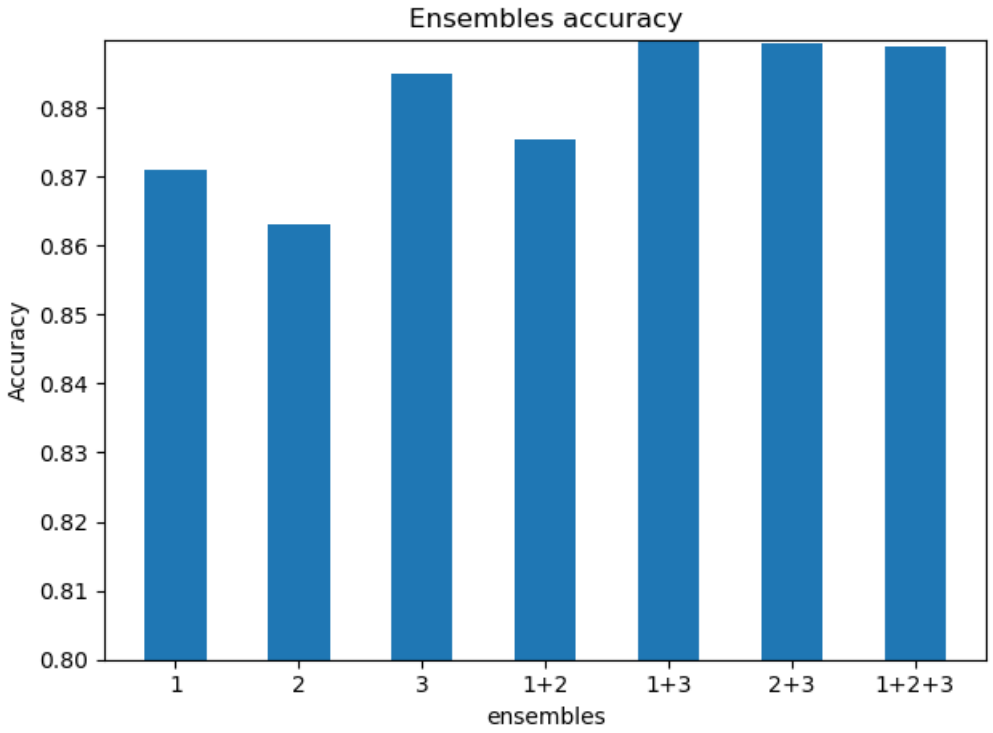

Рисунок 1 – График точности

Из графиков видно, что лучшей моделью является модель под номером 3. Ее точность составила 88,49%, точность двух других 87,09% и 86,3% соответственно. Большую эффективность показал ансамбль первой и третьей моделей его точность 88,97%.

Была проверена работа функции, позволяющей оценивать пользовательские комментарии. Комментарий «Very good film with amazing cast» сеть оценила на 76,379%, что говорит о том, что отзыв положителен. Точность стала выше по сравнению предыдущей лабораторной работой (72,32%).

Выводы.

В ходе выполнения лабораторной работы были изучены рекуррентные нейронные сети. Были построены и обучены оптимальные нейронные сети. Был построен ансамбль из нейронных сетей. Была написана функция, позволяющая пользователю вводить свой текст.

ПРИЛОЖЕНИЕ А

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

top_words = 10000
embedding_vector_length = 32
max_review_length = 500

(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)

x_train = sequence.pad_sequences(x_train, maxlen=max_review_length)
x_test = sequence.pad_sequences(x_test, maxlen=max_review_length)

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()]
    print(text)
    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 Own_Com(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)

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'])
    print(model.summary())
    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(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'])
    print(model.summary())
    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 train_model(model):
    results = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=2,
batch_size=64)
    return results

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), (0, 2), (1, 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)


text = loadText('text.txt')
model1 = model1()
model2 = model2()
model3 = model3()

train_model(model1)
train_model(model2)
train_model(model3)

ensemble([model1, model3, model3])

text = sequence.pad_sequences([text], maxlen=max_review_length)
Own_Com([model1, model2, model3], text)
```