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»

Студентка гр. 7381 _____ Судакова П.С.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

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.

Порядок выполнения работы.

1. Ознакомится с рекуррентными нейронными сетями
2. Изучить способы классификации текста
3. Ознакомится с ансамблированием сетей
4. Построить ансамбль сетей, который позволит получать точность не менее 97%

Требования.

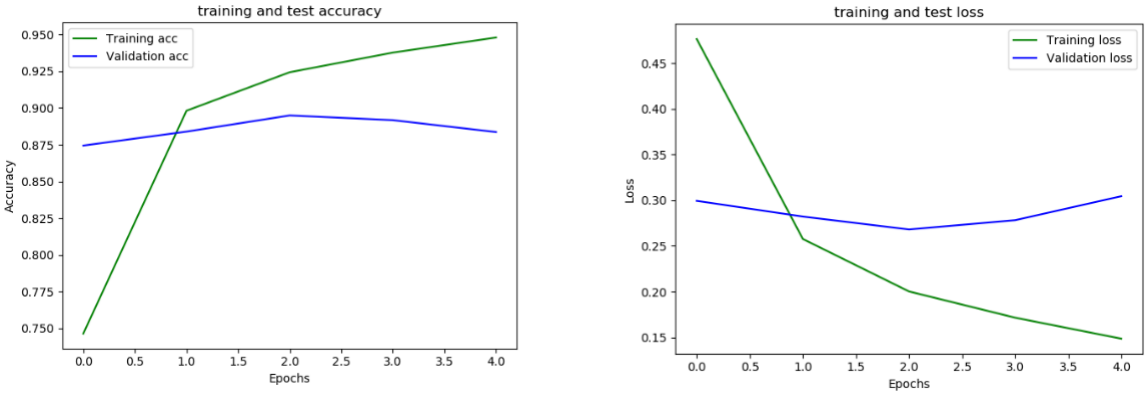
1. Найти набор оптимальных ИНС для классификации текста
2. Провести ансамблирование моделей
3. Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
4. Провести тестирование сетей на своих текстах (привести в отчете)

Ход работы.

Набор данных IMDB – множество из 50 000 самых разных отзывов к кинолентам в интернет-базе фильмов (Internet Movie Database). Набор разбит на 25 000 обучающих и 25 000 контрольных отзывов, каждый набор на 50 % состоит из отрицательных и на 50 % из положительных отзывов.

1. Была построена нейронная сеть со следующей архитектурой:

- Оптимизатор – `adam`
- `batch_size=256`
- `loss='binary_crossentropy'`
- `epochs=5`



а б

Рисунок 1 – Графики точности и потерь данной архитектуры

В данном случае используется ансамблирование слоёв разных моделей, благодаря чему к 5-й эпохе достигается результат в ~96% на обучающих данных и ~90% на тестовых.

2. Функция для ввода пользовательского текста:

```
def gen_custom_x():
    def get_index(a, _index):
        new_list = a.split()
        for j, v in enumerate(new_list):
            new_list[j] = _index.get(v)
        return new_list

    for i in range(len(custom_x)):
        custom_x[i] = get_index(custom_x[i],
        imdb.get_word_index())
    for index_j, i in enumerate(custom_x):
        for _index, value in enumerate(i):
            if value is None:
                custom_x[index_j]
    [_index] = 0 return custom_x
```

Используем функцию для следующих оценок:

```
custom_x = [
    "An excellent, well-researched film",
    "It is, perhaps, most playful and ambitious film
    to date", "This is the boring, stupid movie",
"You don't really care who done it, you just want it to
                                                    be over
```

```
with",

    "Great movie with a great story"
```

Полученные результаты:

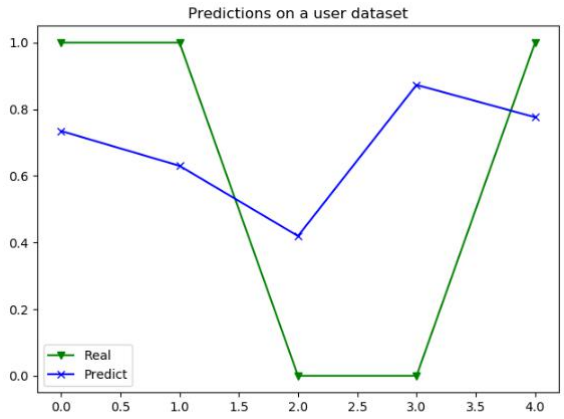


Рисунок 2 – Предсказания пользовательского датасета.

Точность составила 80%, то есть 4/5 оценок было угадано.

Выводы.

В ходе выполнения данной работы ознакомились с рекуррентными нейронными сетями, изучили способы классификации текста, а также ознакомились с ансамблированием сетей.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
      from keras import Sequential
      from keras.datasets import imdb
      import matplotlib.pyplot as plt
      from keras.layers import Embedding, Conv1D,
  Dropout, MaxPooling1D, LSTM, Dense

      from keras_preprocessing import
      sequence import numpy as np

      top_words = 10000

      custom_x = [

          "An excellent, well-researched film",

          "It is, perhaps, most playful and ambitious film to
          date", "This is the boring, stupid movie",

          "You don't really care who done it, you just want it
  to be over with",

          "Great movie with a great story"

      ]

      custom_y = [1., 1., 0., 0., 1.]

      def gen_custom_x():

          def get_index(a, _index):
              new_list = a.split()
              for j, v in enumerate(new_list):
                  new_list[j] = _index.get(v)
              return new_list
          for i in range(len(custom_x)):

              custom_x[i] = get_index(custom_x[i],
  imdb.get_word_index())

for index_j, i in enumerate(custom_x):

for _index, value in enumerate(i):

                  if value is None:

                      custom_x[index_j][_index] = 0
          return custom_x

      max_review_length = 500
```

```
     (training_data, training_targets),
(testing_data, testing_targets) =
imdb.load_data(num_words=top_words)
     training_data =
sequence.pad_sequences(training_data,
maxlen=max_review_length)

     testing_data =
sequence.pad_sequences(testing_data,
maxlen=max_review_length)

     data = np.concatenate((training_data, testing_data))

     targets = np.concatenate((training_targets,
     testing_targets))

     targets = np.array(targets).astype("float32")
     x_test = data[:10000]
     y_test = targets[:10000]
     x_train = data[10000:]
     y_train = targets[10000:]

     embedding_vecor_length = 32 model
     = Sequential()
     model.add(Embedding(input_dim=top
     _words,
output_dim=embedding_vecor_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(0.2))

     model.add(Dense(256, activation='relu'))

     model.add(Conv1D(filters=32, kernel_size=3,
     padding='same',

activation='relu'))

     model.add(LSTM(100, recurrent_dropout=0.3))
     model.add(Dense(1, activation='sigmoid'))
     model.compile(loss='binary_crossentropy',
     optimizer='adam',
```

```
metrics=['accuracy'])

     print(model.summary())

     h = model.fit(x_train, y_train, validation_data=(x_test,
y_test), epochs=5, batch_size=256)

     scores = model.evaluate(x_test, y_test, verbose=0)

     print('Test loss:', scores[0])
     print('Test accuracy:', scores[1])
     plt.title("training and test accuracy")
     plt.plot(h.history['accuracy'], 'g', label='Training
     acc')
     plt.plot(h.history['val_accuracy'], 'b',
     label='Validation acc')
     plt.xlabel('Epochs')
     plt.ylabel('Accuracy')
     plt.legend()
     plt.show()

     plt.clf()
     plt.title("training and test loss")
     plt.plot(h.history['loss'], 'g', label='Training loss')
     plt.plot(h.history['val_loss'], 'b', label='Validation
     loss')
     plt.xlabel('Epochs')
     plt.ylabel('Loss')
     plt.legend()
     plt.show()
     plt.clf()

     X =
sequence.pad_sequences(gen_custom_x(),
maxlen=max_review_length)
     custom_score = model.evaluate(X, custom_y)
     print('custom_acc:', custom_score[1])
     predict = model.predict(X)
     plt.title("Predictions on a user dataset")
     plt.plot(custom_y, 'g', marker='v', label='Real')
     plt.plot(predict, 'b', marker='x', label='Predict')
     plt.legend()
     plt.show()
     plt.clf()
```