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. 7381 _____ Тарасенко Е.А.

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.

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.

Сначала была составлена программа, использующая две модели, приведенные в методических указаниях к работе. Архитектура этих моделей следующая:

1)
```
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, validation_data=(X_test,
y_test), epochs=3, batch_size=64)
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100)
```

2)
```
model = Sequential()
model.add(Embedding(top_words, 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(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

Удалось достичь результатов точности в примерно 75% и 80% соответственно. Далее были предприняты попытки улучшить полученные показатели. В конечном итоге были построены и обучены модели со следующими архитектурами (полный код программы приведен в приложении А):

1)
```
model1 = tensorflow.keras.models.Sequential()
model1.add(tensorflow.keras.layers.Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
model1.add(tensorflow.keras.layers.Dropout(0.3))
model1.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
model1.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
model1.add(tensorflow.keras.layers.Dropout(0.3))
model1.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
model1.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
model1.add(tensorflow.keras.layers.Dropout(0.3))
model1.add(tensorflow.keras.layers.GRU(64, return_sequences=True))
model1.add(tensorflow.keras.layers.SimpleRNN(64))
model1.add(tensorflow.keras.layers.Dense(1, activation='sigmoid'))
```

```
   model1.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
   results1 = model1.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=5, batch_size=64, verbose=0)
```

2)
```
model2 = tensorflow.keras.models.Sequential()
   model2.add(tensorflow.keras.layers.Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
   model1.add(tensorflow.keras.layers.Dropout(0.3))
   model2.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
   model2.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
   model2.add(tensorflow.keras.layers.Dropout(0.2))
   model2.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
   model2.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
   model2.add(tensorflow.keras.layers.Dropout(0.2))
   model2.add(tensorflow.keras.layers.GRU(64, return_sequences=True))
   model2.add(tensorflow.keras.layers.Dropout(0.2))
   model2.add(tensorflow.keras.layers.GRU(32))
   model2.add(tensorflow.keras.layers.Dropout(0.2))
   model2.add(tensorflow.keras.layers.Dense(1, activation='sigmoid'))
   model2.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
   results2 = model2.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=5, batch_size=64, verbose=0)
```

Последующее увеличение количества эпох, расширение архитектуры и т. п. очень плохо сказывалось на времени обучения. Таким образом, эти архитектуры остались наилучшими вариантами.

При разных запусках обучения обеих сетей результаты работы давали точность от 83% до 89%. Так как стартовая инициализация весов так сильно влияет на результат обучения, то были предприняты следующие меры: перед созданием ансамбля этих двух сетей требуется получить наилучшие сборки. Программа создает по 3 модели каждой из двух архитектур и выбирает наилучшую из них (ту, что дает наивысшую точность). Для нее позже выводятся графики точности и ошибки, также лучшие из 3-х моделей участвуют в создании ансамбля, требуемого в условии. Графики ошибок и точностей сетей с приведенными выше архитектурами представлены на рис. 1 и 2.

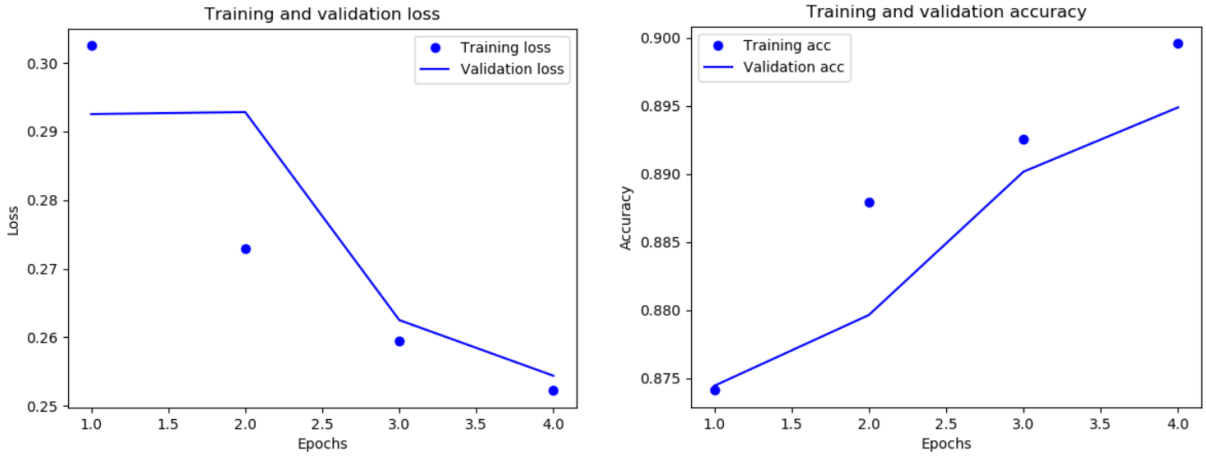

Рис. 1 – Графики ошибки и точности модели-1

Результат обучения этой сети (отбор лучшей сборки из 3-х созданных):

```
1 ) Accuracy: 84.36%
2 ) Accuracy: 87.82%
3 ) Accuracy: 89.49%
Best accuracy: 89.49%
```

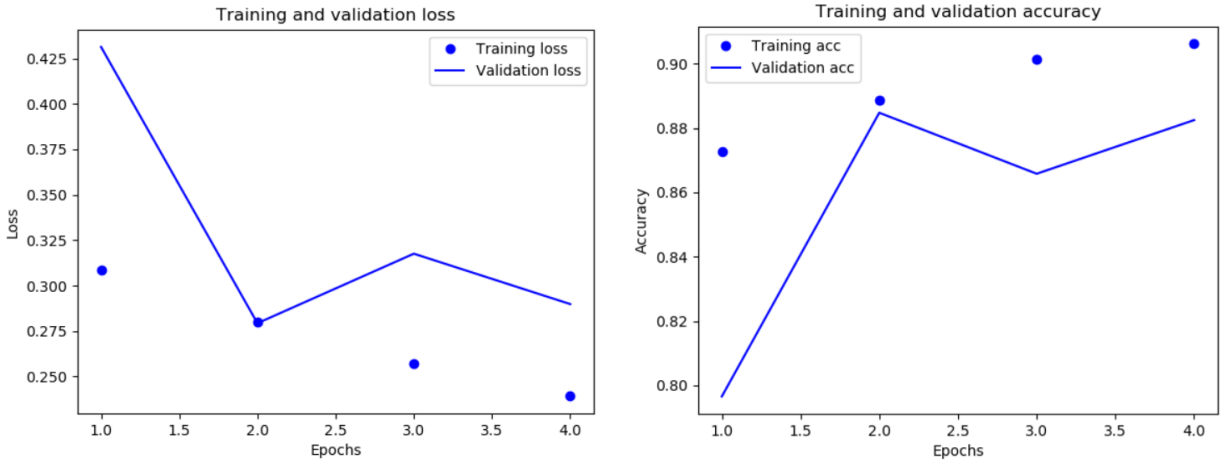

Рис. 2 – Графики ошибки и точности модели-2

Результат обучения этой сети (отбор лучшей сборки из 3-х созданных):

```
1 ) Accuracy: 86.06%
2 ) Accuracy: 88.24%
3 ) Accuracy: 87.32%
Best accuracy: 88.24%
```

Далее была реализована функция для обработки текста пользовательского обзора фильма ансамблем из лучших сборок моделей каждой из двух архитектур:

```
def user_review(review, models):
    print(review)
    words = review.replace(',', '').replace('.', '').replace(':', '').replace('!',
'').replace('?', '').replace('(', '').replace(')', '').replace('"', '').replace(' -
', '').lower().split()
    indexes = dict(imdb.get_word_index())
    codes, num_words = [], []
    for word in words:
        word = indexes.get(word)
        if word and (word < 10000):
            num_words.append(word)
    codes.append(num_words)
    codes = sequence.pad_sequences(codes, maxlen=max_review_length)
    avg_prediction = 0
    for model in models:
        avg_prediction += model.predict(codes)[0][0]
    avg_prediction /= len(models)
    print(1 - avg_prediction)
```

Результат работы функции:

```
Bad example: This was the worst movie I saw at WorldFest and it
also received the least amount of applause afterwards!
0.28067144751548767
Good example: I thought this was a wonderful way to spend time on
a too hot summer weekend, sitting in the air conditioned theater and
watching a light-hearted comedy.
0.8298462927341461
```

Выводы.

В ходе выполнения данной лабораторной работы были обобщены некоторые навыки по работе с рекуррентными нейронными сетями, по их созданию и обучению. В рамках выполнения задания была написана программа, формирующая ансамбль двух сетей, нацеленных на классификацию обзоров фильмов, и дающая возможность использования полученной модели в классификации пользовательских обзоров.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow
from keras.datasets import imdb
from keras.preprocessing import sequence


def draw_plots(results):
    # получение ошибки и точности в процессе обучения
    loss = results.history['loss']
    val_loss = results.history['val_loss']
    acc = results.history['acc']
    val_acc = results.history['val_acc']
    epochs = range(1, len(loss) + 1)

    # построение графика ошибки
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    # построение графика точности
    plt.clf()
    plt.plot(epochs, acc, 'bo', label='Training acc')
    plt.plot(epochs, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


def user_review(review, models):
    print(review)
    words = review.replace(',', '').replace('.', '').replace(':', '').replace('!',
'').replace('?', '').\
        replace('(', '').replace(')', '').replace('"', '').replace(' - ',
'').lower().split()
    indexes = dict(imdb.get_word_index())
    codes, num_words = [], []
    for word in words:
        word = indexes.get(word)
        if word and (word < 10000):
            num_words.append(word)
    codes.append(num_words)
    codes = sequence.pad_sequences(codes, maxlen=max_review_length)
    avg_prediction = 0
    for model in models:
        avg_prediction += model.predict(codes)[0][0]
    avg_prediction /= len(models)
    print(1 - avg_prediction)


# исправление ошибки с данными:
```

```
# ValueError: Object arrays cannot be loaded when allow_pickle=False
np_load_old = np.load
# modify the default parameters of np.load
np.load = lambda *a, **k: np_load_old(*a, allow_pickle=True, **k)
# call load_data with allow_pickle implicitly set to true
(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)
# restore np.load for future normal usage
np.load = np_load_old

# перестроение массивов данных
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)

max_review_length = 500
training_data = sequence.pad_sequences(training_data, maxlen=max_review_length)
testing_data = sequence.pad_sequences(testing_data, maxlen=max_review_length)

# подготовка и обучение моделей, вывод результатов и графиков
top_words = 10000
embedding_vector_length = 32

# модель - 1
max1, best_res1, best_model1 = 0, None, None
for i in range(3):
    model1 = tensorflow.keras.models.Sequential()
    model1.add(tensorflow.keras.layers.Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
    model1.add(tensorflow.keras.layers.Dropout(0.3))
    # model1.add(tensorflow.keras.layers.LSTM(200))
    # model1.add(tensorflow.keras.layers.Dropout(0.3, noise_shape=None, seed=None))
    model1.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
    model1.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
    model1.add(tensorflow.keras.layers.Dropout(0.3))
    model1.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
    model1.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
    model1.add(tensorflow.keras.layers.Dropout(0.3))
    model1.add(tensorflow.keras.layers.GRU(64, return_sequences=True))
    model1.add(tensorflow.keras.layers.SimpleRNN(64))
    model1.add(tensorflow.keras.layers.Dense(1, activation='sigmoid'))
    model1.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
    results1 = model1.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets),
                          epochs=5, batch_size=64, verbose=0)
    scores1 = model1.evaluate(testing_data, testing_targets, verbose=0)
    print(i + 1, ") Accuracy: %.2f%%" % (scores1[1]*100))
    if (scores1[1]*100) > max1:
        max1 = (scores1[1]*100)
        best_res1 = results1
        best_model1 = model1
print("Best accuracy: %.2f%%" % max1)
if best_res1: draw_plots(best_res1)

# модель - 2
max2, best_res2, best_model2 = 0, None, None
for i in range(3):
    model2 = tensorflow.keras.models.Sequential()
    model2.add(tensorflow.keras.layers.Embedding(top_words, embedding_vector_length,
```

```
input_length=max_review_length))
    model2.add(tensorflow.keras.layers.Dropout(0.3))
    model2.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
    model2.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
    model2.add(tensorflow.keras.layers.Dropout(0.2))
    model2.add(tensorflow.keras.layers.Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
    model2.add(tensorflow.keras.layers.MaxPooling1D(pool_size=2))
    model2.add(tensorflow.keras.layers.Dropout(0.2))
    model2.add(tensorflow.keras.layers.GRU(64, return_sequences=True))
    model2.add(tensorflow.keras.layers.Dropout(0.2))
    model2.add(tensorflow.keras.layers.GRU(32))
    model2.add(tensorflow.keras.layers.Dropout(0.2))
    model2.add(tensorflow.keras.layers.Dense(1, activation='sigmoid'))
    model2.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
    results2 = model2.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets),
                          epochs=5, batch_size=64, verbose=0)
    scores2 = model2.evaluate(testing_data, testing_targets, verbose=0)
    print(i + 1, ") Accuracy: %.2f%%" % (scores2[1]*100))
    if (scores2[1]*100) > max2:
        max2 = (scores2[1]*100)
        best_res2 = results2
        best_model2 = model2
print("Best accuracy: %.2f%%" % max2)
if best_res2: draw_plots(best_res2)

scores = 0
for sc in [max1, max2]: scores += sc
scores /= len([max1, max2])
print("Ens. accuracy: %.2f%%" % scores)

# тестирование программы на пользовательских обзорах
bad_review = 'This was the worst movie I saw at WorldFest and it also received the
least amount of applause afterwards!'
good_review = 'I thought this was a wonderful way to spend time on a too hot summer
weekend, sitting in the air conditioned theater and watching a light-hearted comedy.'
print('Bad example:', end=' ')
user_review(bad_review, [best_model1, best_model2])
print()
print('Good example:', end=' ')
user_review(good_review, [best_model1, best_model2])
```