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

ЛАБОРАТОРНАЯ РАБОТА №7

по дисциплине «Искусственные нейронные сети»

Тема: «Классификация обзоров фильмов»

Студент гр. 7381 _____ Ильясов А.В.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург
2020

Цели.

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%

Ход работы.

Для ансамбля были выбраны 2 модели:

Первая представляет из себя рекуррентную сеть с добавлением dropout и полносвязных слоев:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model.add(LSTM(128))
model.add(Dropout(0.3, noise_shape=None, seed=None))
model.add(Dense(64, activation="relu"))
```

```
model.add(Dropout(0.2, noise_shape=None, seed=None))
model.add(Dense(1, activation="sigmoid"))
```

Вторая модель представляет из себя рекуррентную сеть с добавлением слоев свертки:

```
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=64, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))
model.add(LSTM(128))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

Ниже представлены результаты обучения первой и второй модели.

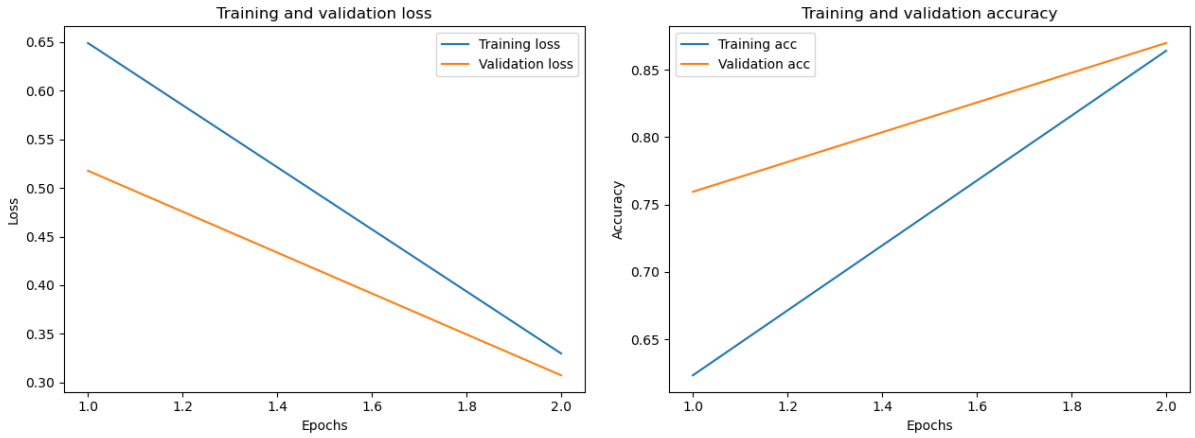

Рисунок 1 – графики функции потерь и точности первой модели

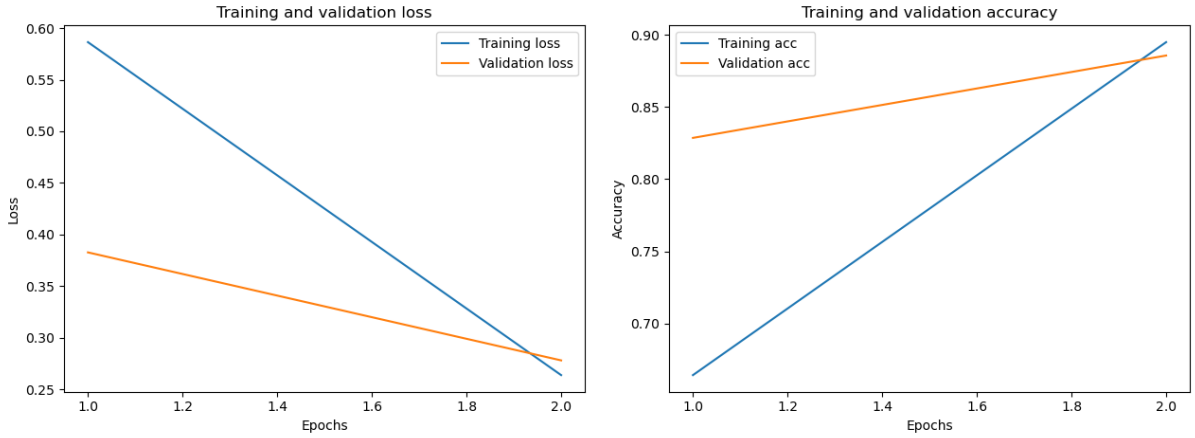

Рисунок 2 – графики функции потерь и точности второй модели

Ансамблирование моделей происходит путем вычисления среднего арифметического значений предсказаний.

Для загрузки пользовательского текста была написана функция, которая приводит входной текст в приемлемый для модели вид и подается на вход обоим моделям, результат которых усредняется и выдается за ответ. Результат работы функции представлен ниже.

Примеры отзывов на фильм:

"Hated was looking forward to it. Its not for anyone to watch. Promotes hate violence suicide and Murder. Please pick another movie to go to. Not sure what movie the other people were watching.",

"The acting, cinematography, sound design, and the script itself is phenomenal. This movie is a triumph. Joaquin Pheonix deserves an Oscar win for this tbh."

Результаты моделей:

```
Validation accuracy of 1st model is 0.5
Validation accuracy of 2nd model is 1.0
Validation accuracy of ensembling models is 1.0
```

Вывод.

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
ИСХОДНЫЙ КОД

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

reviews = ["Hated was looking forward to it. Its not for anyone to
watch. Promotes hate violence suicide and Murder. "
           "Please pick another movie to go to. Not sure what movie
the other people were watching.",
           "The acting, cinematography, sound design, and the script
itself is phenomenal. This movie is a triumph. "
           "Joaquin Pheonix deserves an Oscar win for this tbh."]

rating = [0, 1]
max_review_length = 500
top_words = 10000
embedding_vector_length = 32


def load_data():
    (training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=10000)
    training_data = sequence.pad_sequences(training_data,
maxlen=max_review_length)
    testing_data = sequence.pad_sequences(testing_data,
maxlen=max_review_length)
    return (training_data, training_targets), (testing_data,
testing_targets)


def build_LSTM_model():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
    model.add(LSTM(128))
    model.add(Dropout(0.3))
    model.add(Dense(64, activation="relu"))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation="sigmoid"))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

    return model
```

```
def build_CNN_model():
    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=64, kernel_size=3, padding='same',
activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.4))
    model.add(LSTM(128))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

    return model


def draw_plot(history):
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    epochs = range(1, len(loss) + 1)

    plt.plot(epochs, loss, label='Training loss')
    plt.plot(epochs, val_loss, label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()

    plt.plot(epochs, acc, label='Training acc')
    plt.plot(epochs, val_acc, label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


def train_models():
    (training_data, training_targets), (testing_data,
testing_targets) = load_data()

    model1 = build_LSTM_model()
```

```
    model2 = build_CNN_model()

    history = model1.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=2,
                         batch_size=256)
    scores = model1.evaluate(testing_data, testing_targets,
verbose=0)
    print("Accuracy: %.2f%%" % (scores[1] * 100))
    model1.save('model1.h5')
    draw_plot(history)

    history = model2.fit(training_data, training_targets,
validation_data=(testing_data, testing_targets), epochs=2,
                         batch_size=256)
    scores = model2.evaluate(testing_data, testing_targets,
verbose=0)
    print("Accuracy: %.2f%%" % (scores[1] * 100))
    model2.save('model2.h5')
    draw_plot(history)


def ensembling_models():
    (_, _), (testing_data, testing_targets) = load_data()
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    predictions1 = model1.predict(testing_data)
    predictions2 = model2.predict(testing_data)
    predictions = np.divide(np.add(predictions1, predictions2), 2)
    testing_targets = np.reshape(testing_targets, (25000, 1))
    predictions = np.greater_equal(predictions, np.array([0.5]))
    predictions = np.logical_not(np.logical_xor(predictions,
testing_targets))
    acc = predictions.mean()
    print("Accuracy of ensembling models is %s" % acc)


def predict_text():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    test_y = np.array(rating).astype("float32")
    for review in reviews:
        review = review.lower()
        words = review.replace(',', ' ').replace('.', '
').replace('?', ' ').replace('\n', ' ').split()
        num_words = []
        for word in words:
            word = dictionary.get(word)
            if word is not None and word < 10000:
                num_words.append(word)
        test_x.append(num_words)
```

```
    test_x = sequence.pad_sequences(test_x,
maxlen=max_review_length)
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")

    predictions1 = model1.predict(test_x)
    predictions2 = model2.predict(test_x)
    print(predictions1)
    print(predictions2)
    predictions = np.divide(np.add(predictions1, predictions2), 2)
    print(predictions)

    plt.title("Predictions")
    plt.plot(test_y, label='Real rating')
    plt.plot(predictions, label='Predictions')
    plt.legend()
    plt.show()
    plt.clf()

    predictions = np.greater_equal(predictions, np.array([0.5]))
    test_y = np.reshape(test_y, (2, 1))
    predictions = np.logical_not(np.logical_xor(predictions,
test_y))
    _, acc1 = model1.evaluate(test_x, test_y)
    _, acc2 = model2.evaluate(test_x, test_y)
    print("Validation accuracy of 1st model is %s" % acc1)
    print("Validation accuracy of 2nd model is %s" % acc2)
    acc = predictions.mean()
    print("Validation accuracy of ensembling models is %s" % acc)


if __name__ == '__main__':
    # train_models()
    # ensembling_models()
    predict_text()
```