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

Студент гр. 8382 _____ Нечепуренко Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цели 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)

Выполнение работы.

Для выполнения работы будем использовать тот же набор данных, что и для предыдущей, поэтому рассмотрим архитектуры моделей, которые в дальшем будут ансамблироваться.

Входным слоем каждой модели будет слой Embedding, позволяющий векторизовать входные последовательности.

Модель LSTM.

Первая модель будет состоять из 1 слоя LSTM (Long Short Term Memory) со 100 нейронами.

```
    Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 32)           320000
_________________________________________________________________
lstm (LSTM)                  (None, 100)               53200
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 373,301
Trainable params: 373,301
Non-trainable params: 0
_________________________________________________________________
```

Получаем следующие результаты:

```
    Epoch 1/2
625/625 [==============================] - 22s 33ms/step - loss:
   0.5492 - accuracy: 0.7164 - val_loss: 0.2940 - val_accuracy:
   0.8810
    Epoch 2/2
625/625 [==============================] - 20s 32ms/step - loss:
```

```
0.2542 - accuracy: 0.9015 - val_loss: 0.2712 - val_accuracy:
0.8891
 Accuracy: 88.91%
```

Модель со сверточными слоями.

С помощью сверточных слоев можно хорошо изучить пространственную структуру во входных данных. Затем применим все тот же реккурентный слой LSTM.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           320000
_________________________________________________________________
conv1d (Conv1D)              (None, 500, 32)           3104
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 250, 32)           0
_________________________________________________________________
dropout (Dropout)            (None, 250, 32)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 250, 64)           6208
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 125, 64)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               66000
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 395,413
Trainable params: 395,413
Non-trainable params: 0
_________________________________________________________________
```

Обучим ее с теми же параметрами:

```
    Epoch 1/2
625/625 [==============================] - 12s 16ms/step - loss:
   0.4750 - accuracy: 0.7316 - val_loss: 0.2456 - val_accuracy:
   0.9015
    Epoch 2/2
625/625 [==============================] - 10s 15ms/step - loss:
   0.2098 - accuracy: 0.9199 - val_loss: 0.2386 - val_accuracy:
   0.9030
    Accuracy: 90.30%
```

Уже за 2 эпохи вторая модель достигла лучших результатов, хотя количество параметров отличается на 5%.

Модель с полносвязными слоями.

Рассмотрим еще одну архитектуру, в которой будем использовать слои Dense с функцией активации relu.

```
    Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 500, 32)           320000
_________________________________________________________________
lstm_5 (LSTM)                (None, 64)                24832
_________________________________________________________________
dense_7 (Dense)              (None, 128)               8320
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
```

```
dense_9 (Dense)              (None, 1)                 65
=================================================================
Total params: 361,473
Trainable params: 361,473
Non-trainable params: 0
_________________________________________________________________
```

Обучим модель на тех же параметрах:

```
Epoch 1/2
625/625 [==============================] - 19s 29ms/step - loss:
  0.5151 - accuracy: 0.7242 - val_loss: 0.2792 - val_accuracy:
  0.8837
Epoch 2/2
625/625 [==============================] - 18s 29ms/step - loss:
  0.2336 - accuracy: 0.9111 - val_loss: 0.2720 - val_accuracy:
  0.8916
Accuracy: 89.16%
```

Ансамблирование моделей.

Ансамблированная модель будет просто брать среднее по предсказанием моделей.

```
def get_ensemble_predictions(models, x_data):
    return np.mean(np.asarray([model.predict(x_data) for
      model in models]), 0).flatten()
```

Точность ансамблированной модели оказалась чуть хуже, чем точность второй модели, но с практической точки зрения, такая модель более устойчивая.

```
Ensemble accuracy: 90.13%
```

Предсказание произвольного комментария.

Возьмем те же отзывы, что и в ЛР 6.

Позитивные:

1. A true masterpiece of storytelling.

2. An outstanding and well-thought-out story of a perspective of one's suffering by its acknowledged ideology. The acting and directing are finest bringing the film an absolute classic. The emotions are well intact throughout the whole movie generating a realistic impression. I swear this film will strike you with feels and other senses depending on your notions.

Негативные:

1. Not an exploration of the racial divide for the modern era; there is far too much prominently displayed nazi paraphernalia, and far too many white people yelling the n-word for an explicitly stated thesis of "hate is bad."It's hard to take Edward Norton's Derek Vinyard seriously as we move back and forth between his portrayals of a bright-eyed high-schooler, a murderous skin-head, and a hardened ex-con. The core narrative is simple, but the film is over-ambitious in its execution. The characters mostly fall flat, which makes it difficult to take their grappling with extremism seriously. The use of black and white is far too on the nose, and embodies the simplemindedness with which this complex subject matter is handled.

2. Tries to tackle racism head-on and in a raw, brutal way. But ends up treating the audience like juvenile idiots. Cheesy operatic music (screams: look at me, I'm a cool art house movie! This is meant to be a dramatic scene!) and pointless usage of slow-mo shots. Characters are caricatures. Derek Vinyard's sweeping shift from hardcore skinhead to reformed man is unrealistic and almost comical.

Получаем предсказания:

```
[10] classify_text("good1.txt", models, 500)

     Result for the text good1.txt is [0.84713435]

[7]  classify_text("good2.txt", models, 500)

     Result for the text good2.txt is [0.93171555]

[8]  classify_text("bad1.txt", models, 500)

     Result for the text bad1.txt is [0.42003992]

[9]  classify_text("bad2.txt", models, 500)

     Result for the text bad2.txt is [0.09253382]
```

Рисунок 1 – Предсказания для отзывов

Отзыв bad1 длинный и сложный, для него было получено значение практически на границе, но даже при демократичном выборе порога в 0.5, отзыв будет классифицирован правильно.

Выводы.

В результате выполнения работы была реализована ансамблированная модель, позволяющая анализировать настроение человека по его комментарию к фильму. Были рассмотрены несколько архитектур с использованием реккурентных, сверточных и полносвязных слоев. Была произведена классификация «реального» текста, взятого с сайта Rotten Tomatoes.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.layers import Dense, LSTM, Embedding,
   Conv1D, MaxPooling1D, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import sequence


def get_ensemble_predictions(models, x_data):
    return np.mean(np.asarray([model.predict(x_data) for model in
        models]), 0).flatten()


def evaluate_ensemble(models, x_data, y_data):
    predictions = np.round(get_ensemble_predictions(models,
       x_data))
    accuracy = predictions == y_data
    return np.count_nonzero(accuracy) / y_data.shape[0]


def sequence_from_file(filepath, max_words=10000):
    with open(filepath, "r") as f:
        content = f.read().lower()
        words = [word.strip(":;?!.,'\"-_").lower() for word in
           content.strip().split()]
        index = imdb.get_word_index()
        return list(map(lambda x: 2 if x >= max_words else x + 3,
            [1] + [index.get(word, -1) for word in words]))


def classify_text(filepath, models, max_review_length, max_words
```

```
   =10000):
    coded_text = sequence.pad_sequences([sequence_from_file(
       filepath, max_words)], maxlen=max_review_length)
    prediction = get_ensemble_predictions(models, coded_text)
    print(f"Result for the text {filepath} is {prediction}")


def build_simple_lstm_model():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vector_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'])
    return model


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

```
        metrics=['accuracy'])
    return model


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


top_words = 10000
(training_data, training_targets), (testing_data, testing_targets
    ) = imdb.load_data(num_words=top_words)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets),
    axis=0)
X_train = data[:40000]
y_train = targets[:40000]
X_test = data[40000:]
y_test = targets[40000:]

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)

embedding_vector_length = 32
models = [build_simple_lstm_model(), build_model_with_conv_pool()
   , build_lstm_dense_model()]

for index, model in enumerate(models):
    print(f"Training model #{index+1}")
    model.fit(X_train, y_train, validation_data=(X_test, y_test),
                      epochs=2, batch_size=64)
    scores = model.evaluate(X_test, y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1] * 100))

print("Ensemble accuracy: %.2f%%" % (evaluate_ensemble(models,
   X_test, y_test) * 100))

classify_text("reviews/good1.txt", models, 500)
```