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

ОТЧЕТ

по лабораторной работе №7
по дисциплине «Искусственные нейронные сети»
Тема: Классификация обзоров фильмов

Студент гр. 8383 _____ Бессуднов Г. И.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2021

Цель работы

Прогноз успеха фильмов по обзорам с помощью рекуррентной нейронной сети.

Основные теоретические положения

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.

Мы отобразим каждый обзор фильма в реальную векторную область, популярную технику при работе с текстом, которая называется встраивание слов. Это метод, в котором слова кодируются как действительные векторы в многомерном пространстве, где сходство между словами в смысле смысла приводит к близости в векторном пространстве.

Keras предоставляет удобный способ преобразования положительных целочисленных представлений слов в вложение слов слоем Embedding.

Мы сопоставим каждое слово с вещественным вектором длиной 32. Мы также ограничим общее количество слов, которые нам интересны в моделировании, до 5000 наиболее часто встречающихся слов и обнуляем остальные. Наконец, длина последовательности (количество слов) в каждом обзоре варьируется, поэтому мы ограничим каждый обзор 500 словами, укорачивая длинные обзоры и дополняя короткие обзоры нулевыми значениями.

Теперь, когда мы определили нашу проблему и то, как данные будут подготовлены и смоделированы, мы готовы разработать модель LSTM.

Выполнение работы

Было решено для выполнения работы использовать модели двух типов: первый тип содержит сверточный слой, второй такового не содержит. Всего было создано и обучено 4 сети, по 2 каждого типа. Данные для обучения были разделены в процентом соотношении 80:20 и далее были разделены на равные порции, тем самым все модели обучались на разных данных. Код Программы

представлен в Приложении А. Ниже представлены схемы моделей первого и второго типа соответственно:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 32)           640000
_________________________________________________________________
conv1d (Conv1D)              (None, 500, 32)           3104
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 250, 32)           0
_________________________________________________________________
dropout (Dropout)            (None, 250, 32)           0
_________________________________________________________________
lstm (LSTM)                  (None, 64)                24832
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 668,001
Trainable params: 668,001
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           640000
_________________________________________________________________
```

```
lstm_1 (LSTM)                (None, 500, 64)           24832
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                12416
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
```

Все модели были сохранены и далее использованы в программе `semantic_analyzer.py` (код представлен в Приложении Б) для возможности ввода пользовательского обзора и предсказания его настроения.

Далее были проведены 3 эксперимента:

1. Положительный обзор. Программе был указан файл со следующим текстом:

   Today I've watched American Psycho seven times in a row and can't be bored of it. This movie is very great and intense with interesting characters, thought provoking and aesthetic scenes. Christian Bale play is top notch and this film features very good soundtrack.

   Из содержания понятно, что обзор положительный, программа успешно смогла это определить.

2. Отрицательный обзор. Программе был указан файл со следующим текстом:

   Yesterday I've watched Revolver and this film is very very bad and meaningless. I didn't understand anything and everything seems to be out of context. I felt like this film was pathos for the sake of pathos and that's it. Didn't like it at all, worst film by Guy Ritchie.

   Из содержания понятно, что обзор отрицательный, программа успешно смогла это определить.

3. Обзор-ловушка. Была проведена попытка обмануть сеть. Программе был указан файл со следующим текстом:

I've wanted to watch this film so bad and after sitting through it I can say that it was way beyond what expected. Never I've seen such diverse cast of characters and deep storyline as in the No Country For Old Man. Film catches you from the first frame and never let you down. It is all here: evil, pleasantly disgusting and cruel main villain, cowboy-like protagonist and, of course, the old man himself, who can't handle the events of this story. I can't recommend this film enough.

Данный текст наполнен негативными словами, но несет в себе положительный смысл. Вывод сетей был следующим: [0.41597152 0.86582929 0.06422675 0.7945531]. Таким образом по среднему значению было установлено, что обзор положительный, ансамбль сетей обмануть не удалось.

Выводы

В ходе работы была реализована рекуррентная сеть, прогнозирующая успех фильмов по обзорам. Так же был применен метод ансамблирования моделей для более точного предсказания обзора фильма.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ MAIN.PY

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


def generateModel1():
    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(64, dropout=0.25))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model

def generateModel2():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
    model.add(LSTM(64, return_sequences=True, dropout=0.3))
```

```
        model.add(LSTM(32, dropout=0.2))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy',        optimizer='adam',
metrics=['accuracy'])
        return model

    (training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets), axis=0)

    top_words = 20000
    trainSize = len(data) * 8 // 10
    embedding_vector_length = 32
    max_review_length = 500
    modelsCount = 4
    trainBlockSize = trainSize // modelsCount
    testBlockSize = (len(data) - trainSize) // modelsCount

    X_train = data[:trainSize]
    Y_train = targets[:trainSize]
    X_test = data[trainSize:]
    Y_test = targets[trainSize:]

    X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
    X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)

    scores = []

    for i in range(modelsCount):
        X_block_train  =  X_train[i  *  trainBlockSize  :  (i  +  1)  *
trainBlockSize]
        Y_block_train  =  Y_train[i  *  trainBlockSize  :  (i  +  1)  *
trainBlockSize]
```

```
        X_block_test = X_test[i * testBlockSize : (i + 1) * testBlockSize]
        Y_block_test = Y_test[i * testBlockSize : (i + 1) * testBlockSize]

        if (i % 2 == 0):
            print("type 1")
            model = generateModel1()
        else:
            print("type 2")
            model = generateModel2()

        model.summary()
        model.fit(X_block_train,                                Y_block_train,
validation_data=(X_block_test, Y_block_test), epochs=2, batch_size=16)
        score = model.evaluate(X_block_test, Y_block_test, verbose=0)
        print(score)
        scores.append(score[1])
        #  model.save("model_lb7_"  +  str(i  %  2)  +  "_"  +  str(i  //  2)  +
".h5")


   print("Accuracy: %.2f%%" % (np.mean(scores)*100))
```

ПРИЛОЖЕНИЕ Б

КОД ПРОГРАММЫ SEMANTIC_ANALYZER.PY

```
import numpy as np
from keras.models import load_model
from pathlib import Path
from tensorflow.keras.datasets import imdb
from keras.preprocessing import sequence

def predictReview(filename):
    path = Path(filename)
    if (not path.exists()):
        print("Oops! Can't find file", filename)
        return

    reviewStr = ""
    with open(path.absolute()) as f:
        reviewStr = f.read()

    reviewStr = "".join(char for char in reviewStr if
char.isalpha() or char.isspace() or char == "'").lower().strip().split()
    indicies = []
    wordsIndicies = imdb.get_word_index()

    for word in reviewStr:
        i = wordsIndicies.get(word)
        if i is not None and i < 10000:
            indicies.append(i + 3)

    indicies = np.array([indicies])
    indicies = sequence.pad_sequences(indicies, maxlen=500)

    scores = np.zeros(len(models))
    for i in range(len(models)):
        scores[i] = models[i].predict(indicies)
```

```
        return scores


    models = []
    for i in range(4):
        model = load_model("model_lb7_" + str(i % 2) + "_" + str(i //
2) + ".h5")
        models.append(model)

    print("Enter file name with review: ")
    inp = str(input())

    res = predictReview(inp)
    print(res)

    answer = np.mean(res)
    print(answer)

    if answer >= 0.5:
        print("good")
    else:
   print("bad")
```