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

Студентка гр. 8383	________________	Кормщикова А.О.
Преподаватель	________________	Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы..

Классификация обзоров фильмов

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.

Ход работы

Были загружены обучающие и тестовые данные из набора IMDb. Данные были объединены, для последующего разделения 80/20. Входные последовательности были дополнены нулями/обрезаны до 500 слов.

Программа может загрузить 5 готовых моделей либо же создать новые 5 моделей.

Была написана специальная функция, которая создает модель и сохраняет ее. Функция может создавать два типа моделей:

Первый тип

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 32)           320000
_________________________________________________________________
conv1d (Conv1D)              (None, 500, 32)           3104
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 250, 32)           0
_________________________________________________________________
dropout (Dropout)            (None, 250, 32)           0
_________________________________________________________________
lstm (LSTM)                  (None, 100)               53200
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
```

```
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 376,405
Trainable params: 376,405
Non-trainable params: 0
_________________________________________________________________
```

Второй тип

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           320000
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 373,301
Trainable params: 373,301
Non-trainable params: 0
```

Функция:

```
def createModel(type = 1):
    if(type == 1):
        print("First type")
        model = Sequential()
        model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
        model.add(LSTM(100, dropout=0.2))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    else:
        print("Second type")
        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(0.2))
        model.add(LSTM(100))
        model.add(Dropout(0.2))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy', optimizer='adam', 
metrics=['accuracy'])
    return model
```

Всего было создано 5 моделей с именами model_i.h5, где i = [0,4]: две модели второго типа и три модели первого типа. Каждая сеть тренировалась на своей части данных.

Полученные значения точности:

model_0.h5. Accuracy: 0.8586000204086304

model_1.h5. Accuracy: 0.8458999991416931

model_2.h5. Accuracy: 0.8547000288963318

model_3.h5. Accuracy: 0.862500011920929

model_4.h5. Accuracy: 0.8781999945640564

Mean accuracy: 0.8599800109863281

Была написана функция, позволяющая считать пользовательский текст из файла. Функция получает на вход имя файла, считывает данные и подготавливает их к виду, удобному для ИНС, функция возвращает эти данные.

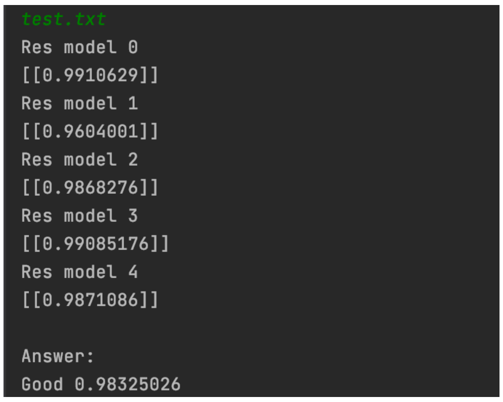
```
def textFromFile(fileName, dimension=10000):
    file = open(fileName, "r")
    text = file.read()
    file.close()
    text.lower()
    tt = str.maketrans(dict.fromkeys("!\"#$%&()*+,-./:;<=>?
@[\]^_`{|}~"))
    text = text.translate(tt).split()
    index = imdb.get_word_index()
    codedText = []
    for word in text:
        i = index.get(word)
        if i is not None and i < dimension:
            codedText.append(i+3)
    codedText = np.array([codedText])
    codedText = sequence.pad_sequences(codedText, 
maxlen=max_review_length)
```

```
    return codedText
```

Для тестирования были выбраны два обзора на фильм "Druk" - один отрицательный другой положительный:

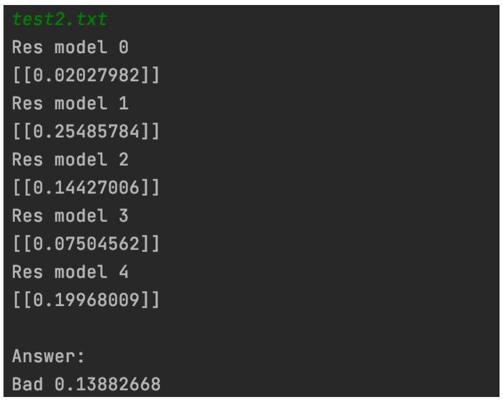
Положительный отзыв находится в файле test.txt, отрицательный в файле test2.txt. Тексты отзывов предоставлены в приложении Б.

Результаты тестирования:



```
test.txt
Res model 0
[[0.9910629]]
Res model 1
[[0.9604001]]
Res model 2
[[0.9868276]]
Res model 3
[[0.99085176]]
Res model 4
[[0.9871086]]

Answer:
Good 0.98325026
```



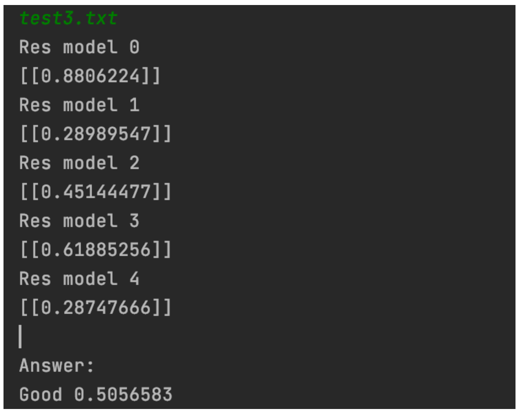
```
test2.txt
Res model 0
[[0.02027982]]
Res model 1
[[0.25485784]]
Res model 2
[[0.14427006]]
Res model 3
[[0.07504562]]
Res model 4
[[0.19968009]]

Answer:
Bad 0.13882668
```

Также для тестирования был выбран обзор, который может показаться для нейросети плохим, т.к. содержит в себе негативные отзывы, которые опровергаются потом.

"This film is bad," - said all my friends who saw it. "It has strange characters." Despite bad reviews, I decided to watch this film. On the contrary, he turned out to be amazing and interesting. Its whole point is not in the plot, but in the wonderful atmosphere and small details, I liked the fullness of this film. I was very pleased with this film, it brought me joy. He's not for everyone but it's not bad.

Результат:



```
test3.txt
Res model 0
[[0.8806224]]
Res model 1
[[0.28989547]]
Res model 2
[[0.45144477]]
Res model 3
[[0.61885256]]
Res model 4
[[0.28747666]]

Answer:
Good 0.5056583
```

Видно, что разные модели оценивали обзор по разному. Минимальным значением было 0.287, а максимальным 0.881. Положительно обзор оценили две модели. Из этого можно сделать вывод, что ансамблирование моделей позволяет более точно анализировать текст.

Выводы.

Во время выполнения лабораторной работы был реализован ансамбль нейронный сетей, который классифицирует фильм. Реализована функция для пользовательского ввода из файла.

ПРИЛОЖЕНИЕ А

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

def textFromFile(fileName, dimension=10000):
    file = open(fileName, "r")
    text = file.read()
    file.close()
    text.lower()
    tt = str.maketrans(dict.fromkeys("!\"#$%&()*+,-./:;<=>?
@[\]^_`{|}~"))
    text = text.translate(tt).split()
    index = imdb.get_word_index()
    codedText = []
    for word in text:
        i = index.get(word)
        if i is not None and i < dimension:
            codedText.append(i+3)
    codedText = np.array([codedText])
    codedText = sequence.pad_sequences(codedText,
maxlen=max_review_length)
    return codedText


def createModel(type = 1):
    if(type == 1):
        print("First type")
        model = Sequential()
        model.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
        model.add(LSTM(100, dropout=0.2))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    else:
        print("Second type")
        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(0.2))
        model.add(LSTM(100))
        model.add(Dropout(0.2))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model

def createEnsemble(count = 5):
    for i in range(count):
        print("Creating model "+ str(i)+"...")
        X_batch_train = X_train[i * train_batch:(i + 1) *
train_batch]
        X_batch_test = X_test[i * test_batch:(i + 1) * test_batch]
        Y_batch_train = Y_train[i * train_batch:(i + 1) *
train_batch]
        Y_batch_test = Y_test[i * test_batch:(i + 1) * test_batch]
        model = createModel(i % 2)
        # print(model.summary())
        model.fit(X_batch_train, Y_batch_train,
validation_data=(X_batch_test, Y_batch_test), epochs=2,
batch_size=64)
        score = model.evaluate(X_test, Y_test, verbose=0)
        scores.append(score[1])
        print("Accuracy:", score, score[1], "\n")
        model.save("model_" + str(i) + ".h5")
    print("Mean accuracy:", np.mean(scores))


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)

max_review_length = 500
train_len = len(data) * 8 // 10
X_train = data[:train_len]
X_test = data[train_len:]
Y_train = targets[:train_len]
Y_test = targets[train_len:]

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

count_model = 5;
```

```
scores = []
train_batch = train_len//count_model
test_batch = (len(data) - train_len)//count_model
# print(train_batch, test_batch)
embedding_vector_length = 32


print("Type 1 to create models, another input to load models")
inp = input()
if(inp == "1"):
    createEnsemble(count_model)

models = []
score_load = []
for i in range(count_model):
    print("Loading model_"+str(i)+".h5...")
    model = load_model("model_"+str(i)+".h5")
    models.append(model)
#     score = model.evaluate(X_test, Y_test, verbose=0)
#     score_load.append(score[1])
#     print("Load success. Accuracy:", score[1])
#
# print("Mean accuracy:", np.mean(score_load))


while(1):
    print("Print file name *.txt or stop")
    fileName = input()
    if fileName == "stop":
        break
    codedText = textFromFile(fileName, top_words)
    res = []
    for i in range(count_model):
        r = models[i].predict(codedText)
        print("Res model "+str(i))
        print(r)
        res.append(r)
    result = np.mean(res)
    print("\nAnswer:")
    if result >= 0.5:
        print("Good", result)
    else:
        print("Bad", result)
```

ПРИЛОЖЕНИЕ Б

Положительный обзор, файл test.txt

Shouldn't we slap a glass?

What if a person from the very birth lacks a little alcohol in their blood ?! So, in any case, according to one of the heroes of the new film Thomas Winterberg "Druk", the Norwegian psychiatrist Finn Skarderud thinks. Alcohol helps a person to become more relaxed, joyful and gives the opportunity to enjoy all the delights of life. This does not mean that you need to thump like hell, you just need to drink a few glasses a day and keep those same magic 0.5 ppm in your blood. Moreover, only on weekdays and during working hours.

"Druk" is a wonderful tragicomedy, the protagonists of which are four teachers of the Danish school, who decided on themselves (exclusively as a scientific experiment) to test the above theory. This picture is not so much about drunkenness as about the midlife crisis and the extinct gaze of our teachers. It is no coincidence that it is teachers, role models, guides to the world of knowledge, and indeed life itself, that are in the foreground. After all, it is absolutely not clear which teacher is better / worse, the one who mumbles a paragraph from the textbook without enthusiasm (and even the one that has already been passed in the last lesson), or who gushes with ideas and tries to teach his students (who already like to drink beer on a cozy lake) necessary knowledge, albeit slightly under the fly ?!

"Druk" is not a solemn song to drunkenness, showing that drinking is cool, that alcohol is the solution to all problems, but also not a moralizing propaganda that explains to every fool that drinking intoxicating drinks is the last thing, and sobriety is above all. It so happened that alcohol is a part of our life. And as each of us copes with this in his own way, the heroes of the film will have to go through different stages of alcoholic intoxication and get out of them in different ways. Well, the subtle humor with the excellent acting of the alcoholic quartet led by the star of the pan-European cinema Mads Mikkelsen, and the hilarious inserts of chronicle frames with famous politicians who put the collars, should move both an inveterate teetotaler and an ardent advocate of a healthy lifestyle. 9 out of 10

Отрицательный обзор, файл test2.txt

When I moved on to watching this film, I expected to see what our directors and screenwriters had failed to realize - a literate story about where alcohol leads. And it seems that an excellent starting point was provided for this - an experiment that assumes the positive effects of alcohol.

But no! Do not be fooled, the film is ideologically absolutely identical to those Russian films that the same Russian viewers did not like. He is completely devoid of morality. The characters in it do not change qualitatively - they do not get rid of their flaws. Moreover, some of the characters do not have these flaws at all, which cancels even their fictitious development.

And this is my main bewilderment: why did they love a picture with the same idea and the same mediocre development of the heroes, but they hate the ideologically similar pictures filmed in Russia? An alcohol-related package that is not disclosed. Unpleasant and naturalistic scenes of behavior of drunk people. Is it really all about the actors, and the fact of the presence of a famous person in the picture qualitatively changes its message and the development of the characters? Mikkelsen alone automatically guarantees a film 90% positive reviews?

I don't think so.