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. 7383 _____ Лосев М.Л.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель.

Реализовать прогнозирование успеха фильмов по обзорам (Predict Sentiment From Movie Reviews).

Задачи.

- Ознакомиться с рекуррентными нейронными сетями 2.
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%

Выполнение работы.

Создадим две модели рекуррентной ИНС: со сверткой и без. Добавим слои разреживания.

Архитектура сети без свертки:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
model.add(LSTM(100))
model.add(Dropout(0.3, noise_shape=None, seed=None))
model.add(Dense(64, activation="relu"))
model.add(Dense(1, activation='sigmoid'))
```

Архитектура сети со слоями свертки:

```
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(Dropout(0.3, noise_shape=None, seed=None))
```

```
model.add(Conv1D(filters=32, kernel_size=3,
padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3, noise_shape=None, seed=None))
model.add(LSTM(100))
model.add(Dropout(0.3, noise_shape=None, seed=None))
model.add(Dense(1, activation='sigmoid'))
```

Графики обучения этих сетей представлены на рсунках 1 и 2 соответственно.

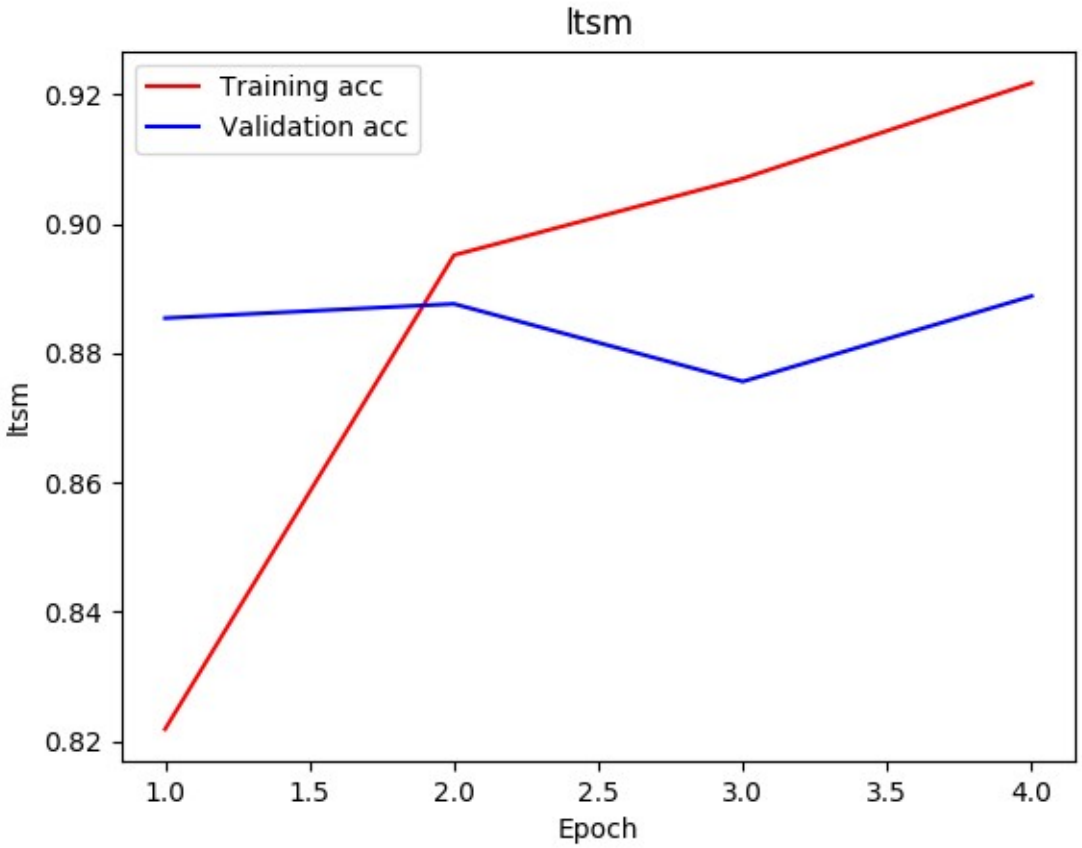

Рисунок 1 - обучение модели без сверточных слоев

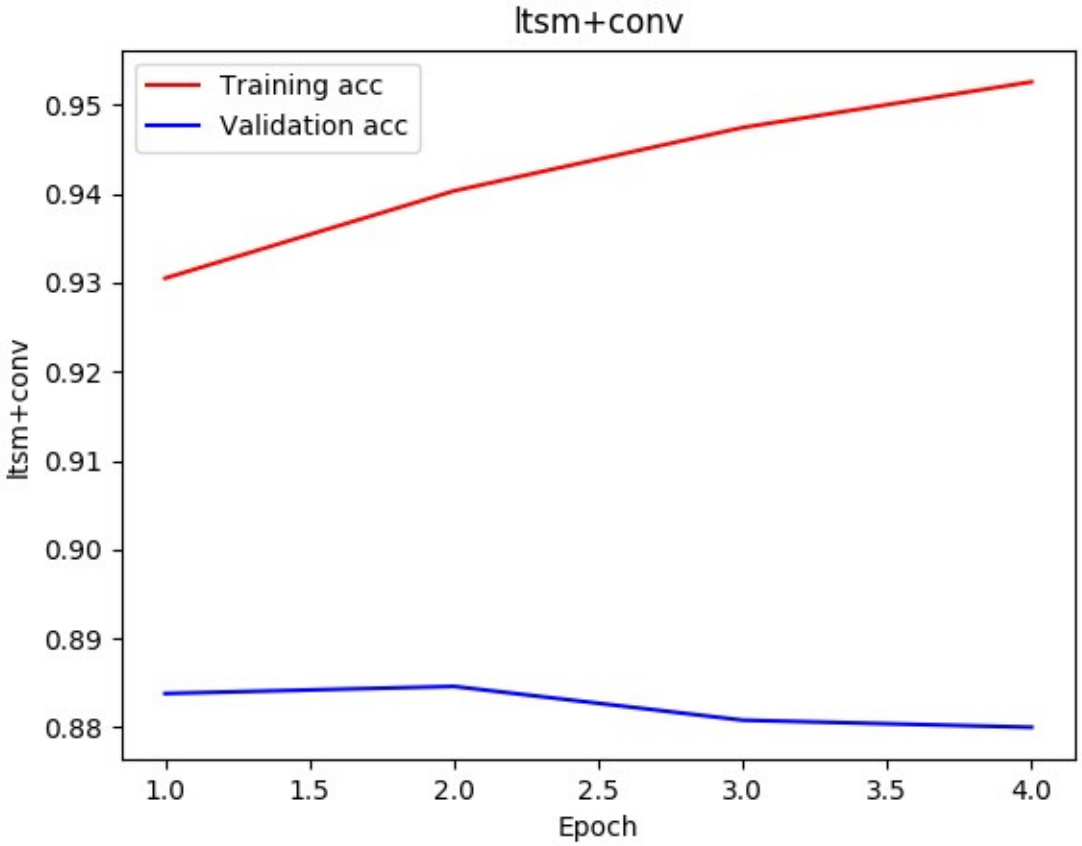

Рисунок 2 - обучение модели с двумя сверточными слоями

Чтобы ассамблировать эти две модели, найдем среднее арифметическое их предсказаний:

```
def get_ensembled_predictions(model1, model2, data):
    predictions1 = model1.predict(testing_data)
    predictions2 = model2.predict(testing_data)
    predictions = np.mean([predictions1, predictions2], axis=0)
    return predictions
```

Чтобы оценить полученный результат, возьмем тестовые данные и получим предсказания для них, а затем сравним с правильными ответами и посчитаем количество ошибок и найдем долю верных предсказаний:

```
def evaluate_ensemble(data, testing_targets):
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    ensembled_predictions =
get_ensembled_predictions(model1, model2, data)
    testing_targets = np.reshape(testing_targets,
(len(testing_targets), 1))
    categ = np.greater_equal(ensembled_predictions,
np.array([0.5]))
    mistakes = np.logical_xor(categ, testing_targets)
    return  1 – mistakes.mean()
```

Достигнутая точность равна 0.8898.

Чтобы делать предсказания для пользовательского текста, нужно его закодировать индексами слов, которые были использованы при обучении (их можно взять из датасета):

```
    def encode(text):
    text = text.translate(
str.maketrans('',  '',  string.punctuation))  #  remove
punctuation
     text = text.lower() # to lowercase cuz there's no
indexes for non-lowercase words
    words = text.split(" ")
    index = imdb.get_word_index()
    encoded = [index.get(word, 0) if index.get(word, 0) <
top_words else 0 for word in words]
    return encoded
```

Сделаем теперь предсказания для следующих строк:

"As a noble use of the medium and as a tireless expression of national anguish, it towers over everything that has been attempted by an American filmmaker in a very long time",

"While much of the footage is breathtaking, the movie is emotionally obtuse and intellectually empty",

"That is the best film i have ever seen",

"It's a fucked up shit i will never watch again",

"Take my strong advice buddies, remember never to watch it",

"The beauty of the movie makes it a masterwork of cinema".

Результат совпадает с ожиданиями:

[[0.76443934]
[0.18299055]
[0.68349713]
[0.3955919]
[0.13416032]
[0.5171615]]

Вывод.

Были созданы и обучены модели реккурентных сетей, прогнозирующих оценку фильма по обзорам, и проведено ансамблирование этих моделей. Также была написана функция прогнозирования оценки по пользовательскому тексту с помощью ансамблированных моделей.

Приложение А

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

top_words = 5000 # this much most frequent words to be kept
max_review_length = 500
embedding_vecor_length=32
picdir = './pics/'

review_parts = ["As a noble use of the medium and as a tireless expression of
national anguish, it towers over everything that has been attempted by an
American filmmaker in a very long time",
       "While much of the footage is breathtaking, the movie is emotionally
obtuse and intellectually empty",
       "That is the best film i have ever seen",
       "It's a fucked up shit i will never watch again",
       "Take my strong advice buddies, remember never to watch it",
       "The beauty of the movie makes it a masterwork of cinema"
       ]


def build_model():
    model = Sequential()
    model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
    model.add(LSTM(100))
    model.add(Dropout(0.3, noise_shape=None, seed=None))
    model.add(Dense(64, activation="relu"))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model

def build_conv_model():
    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(Dropout(0.3, noise_shape=None, seed=None))

    model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3, noise_shape=None, seed=None))

    model.add(LSTM(100))
    model.add(Dropout(0.3, noise_shape=None, seed=None))

    model.add(Dense(1, activation='sigmoid'))

def save_plot(label, history):
```

```
    plt.clf()
    arg = range(1, len(history['accuracy']) + 1)
    plt.plot(arg, history['accuracy'], 'r', label='Training acc')
    plt.plot(arg, history['val_accuracy'], 'b', label='Validation acc')
    plt.title(label)
    plt.xlabel('Epoch')
    plt.ylabel(label)
    plt.legend()
    plt.savefig(label + '.png')
    plt.savefig(picdir + label + '.png')

def encode(text):
    text = text.translate(str.maketrans('', '', string.punctuation)) # remove
punctuation
    text = text.lower() # to lowercase cuz there's no indexes for non-lowercase
words
    words = text.split(" ")
    index = imdb.get_word_index()
    encoded = [index.get(word, 0) if index.get(word, 0) < top_words else 0 for
word in words]
    return encoded

def get_ensembled_predictions(model1, model2, data):
    predictions1 = model1.predict(testing_data)
    predictions2 = model2.predict(testing_data)
    predictions = np.mean([predictions1, predictions2], axis=0)
    return predictions

def evaluate_ensemble(data, testing_targets):
    model1 = load_model("model1.h5")
    model2 = load_model("model2.h5")
    ensembled_predictions = get_ensembled_predictions(model1, model2, data)
    testing_targets = np.reshape(testing_targets, (len(testing_targets), 1))
    categ = np.greater_equal(ensembled_predictions, np.array([0.5]))
    mistakes = np.logical_xor(categ, testing_targets)
    return  1 - mistakes.mean()

def predict_user_reviews():
    test_x = []
    for r in review_parts:
        test_x.append(encode(r))
    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)
    predictions = np.divide(np.add(predictions1, predictions2), 2)
    print(predictions)

def split_data(data, targets, val_split):
    assert(len(data) == len(targets))
    border = m.floor(len(data) * val_split)
    training_data = data[border:]
    testing_data = data[:border]
    training_targets = targets[border:]
    testing_targets = targets[:border]
    return (training_data, training_targets), (testing_data, testing_targets)

from keras.datasets import imdb
(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)
data = sequence.pad_sequences(data, maxlen=max_review_length)
```

```
(training_data, training_targets), (testing_data, testing_targets) =
split_data(data, targets, 0.1)

# train the models and than save it
model = build_model()
print(model.summary())
H = model.fit(training_data, training_targets, validation_data=(testing_data,
testing_targets), epochs=4, batch_size=64)
save_plot('ltsm', H.history)
model.save('model1.h5')

model = build_model()
print(model.summary())
H = model.fit(training_data, training_targets, validation_data=(testing_data,
testing_targets), epochs=4, batch_size=64)
save_plot('ltsm+conv', H.history)
model.save('model2.h5')

print(evaluate_ensemble(testing_data, testing_targets))

predict_user_reviews()
```