

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

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

Студент гр. 7381

Кортев Ю. В.

Преподаватель

Жукова Н.А.

Санкт-Петербург

2020

Цель 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

- Найти набор оптимальных ИНС для классификации текста
- Провести ансамблирование моделей
- Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
- Провести тестирование сетей на своих текстах (привести в отчете)

Ход работы.

В мой ансамбль нейронных сетей будут входить: полносвязная нейронная сеть, сверточная нейронная сеть с рекуррентным слоем lstm и двунаправленная рекуррентная сеть. Для того, чтобы ансамблирование давало хорошие результаты, модели должны быть одинаково хороши, но основная идея подхода состоит в том, что каждая сеть должна иметь свое представление относительно элементов выборки.

1. Полносвязная нейронная сеть

Архитектура модели взята из прошлой работы, за исключением того, что сеть принимает не прямо закодированные обзоры, а векторные представления, которые генерирует слой embedding. Инициализация модели показана в листинге 1. Графики точности и потерь модели показаны на рисунках 1-2.

```
model=Sequential()
model.add(Embedding(num_wprds,3,input_length=500))
model.add(Flatten())
model.add(Dense(16, activation = "relu"))
model.add(Dropout(0.3))
model.add(Dense(16, activation = "relu"))
model.add(Dropout(0.3))
model.add(Dense(1, activation = "sigmoid"))

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

h=model.fit(data[10000:],labels[10000:], epochs=num_epochs,batch_size=512
            ,validation_data=(data[:5000],labels[:5000]),verbose=2)
```

Листинг 1 - Инициализация и обучения полносвязной модели

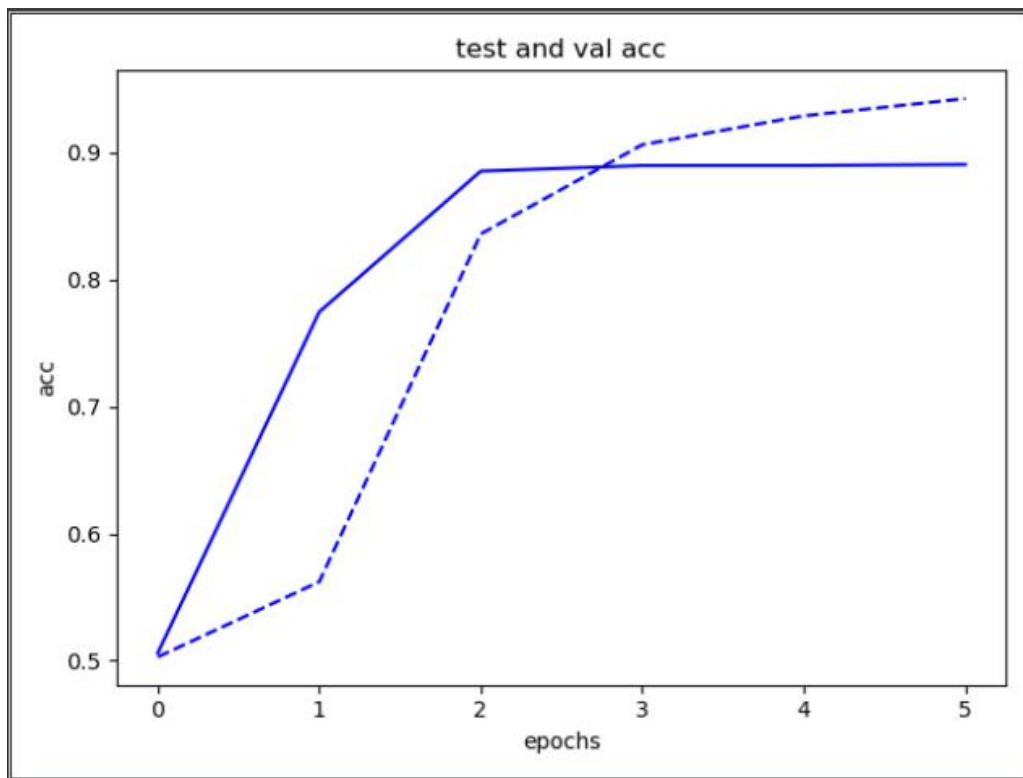


Рисунок 1 - Точность полносвязной модели

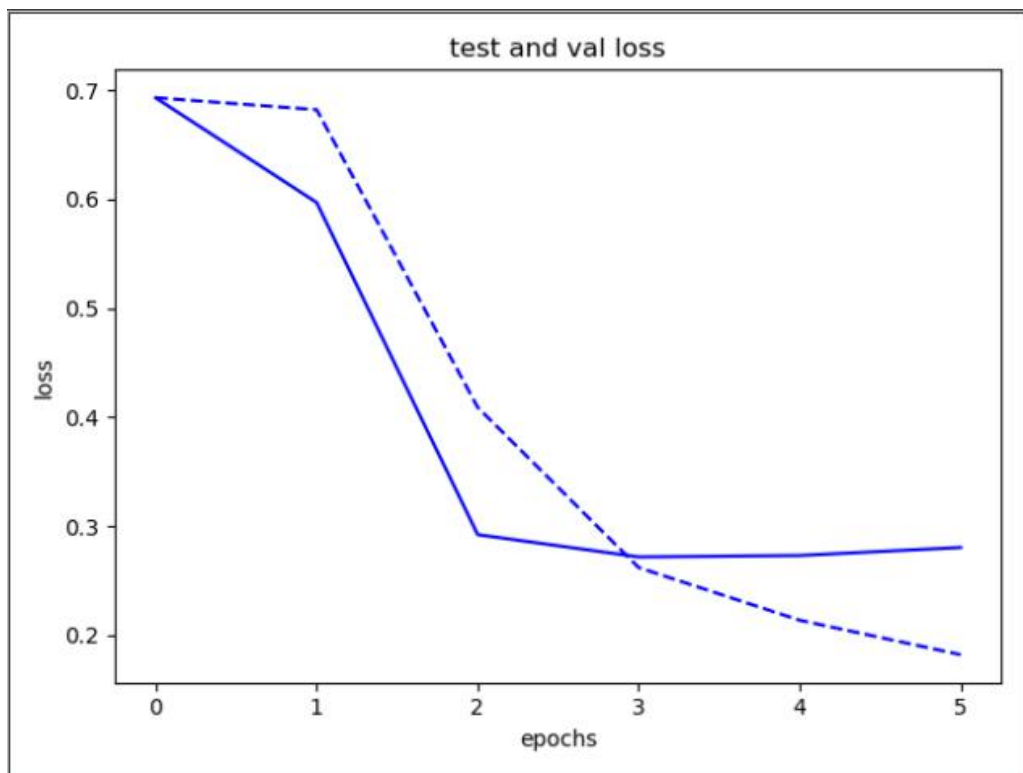


Рисунок 2 - Потери полносвязной модели

2. Сверточная сеть с рекуррентным слоем lstm.

Такая модель объединяет в себе скорость сверточных сетей, которая заключается в выделении высокоуровневых признаков, с чувствительностью рекуррентных сетей к порядку следования признаков. Инициализация модели показана в листинге 2. Графики точности и потерь показаны на рисунках 3-4.

```
model=Sequential()
model.add(Embedding(num_wprds,3,input_length=500))
model.add(Conv1D(filters=32, kernel_size=3, strides=2,padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100,recurrent_dropout=0.2,dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

h=model.fit(data[10000:],labels[10000:], epochs=num_epochs,batch_size=512
            ,validation_data=(data[:5000],labels[:5000]),verbose=2)
```

Листинг 2 - Инициализация сверточной сети с LSTM

Увеличение шага свертки до двух, а также добавления прореживания к слою LSTM положительно сказывается на итоговой точности модели.

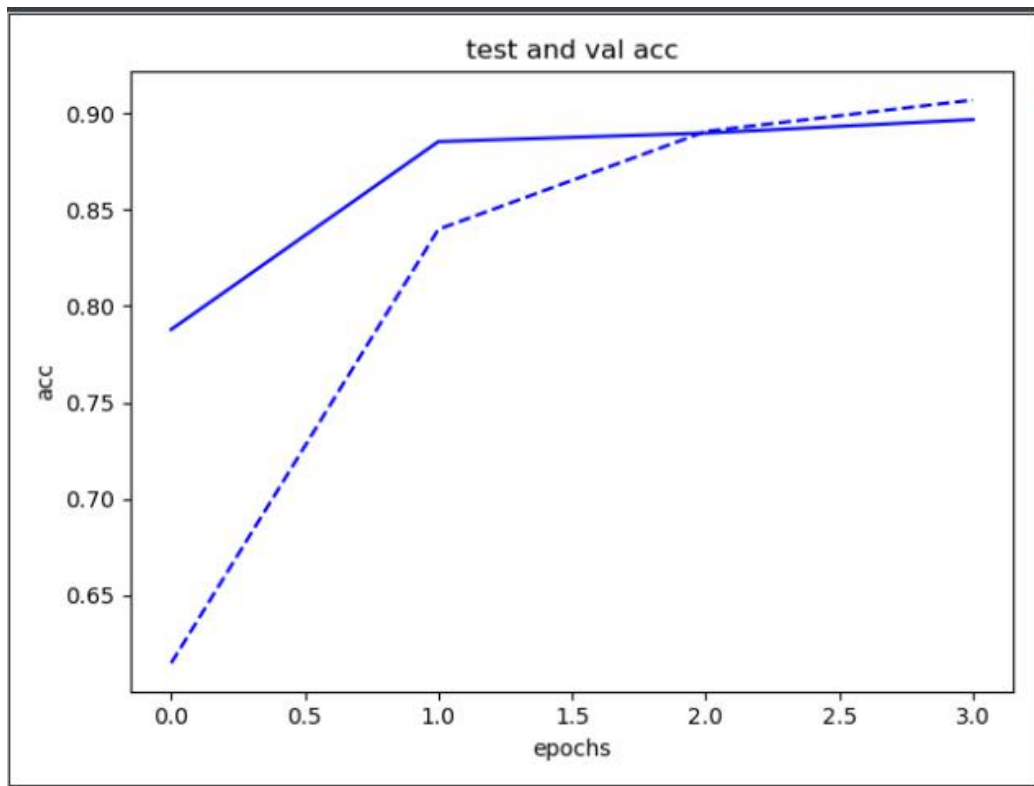


Рисунок 3 - Точность сверточной сети с слоем LSTM

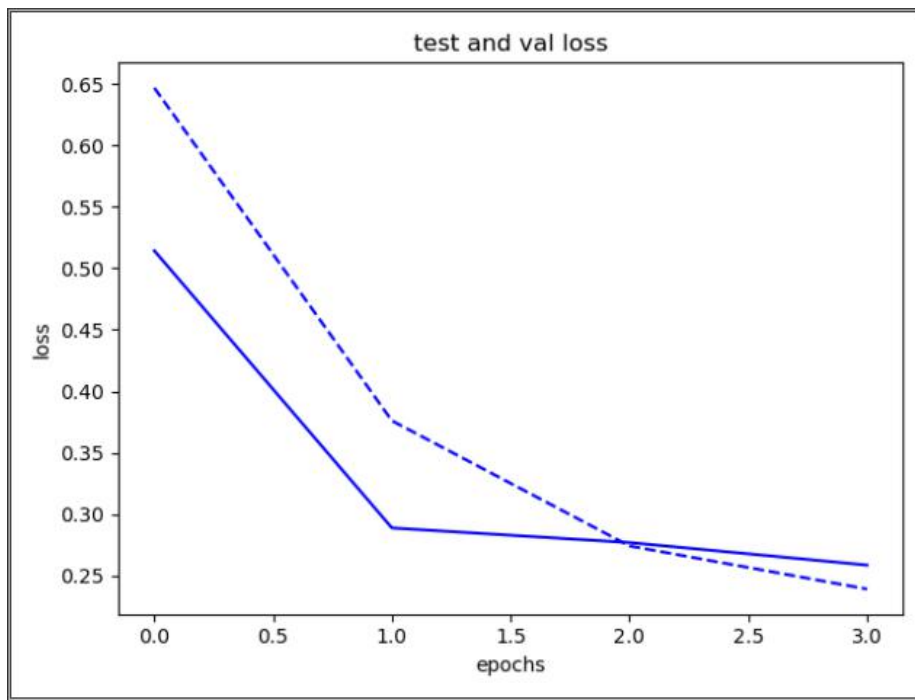


Рисунок 4 - Потери сверточной сети с слоем LSTM

3. Двухнаправленная рекуррентная сеть.

Слой LSTM чувствителен к порядку следования признаков, но его память не долгосрочна. Признаки, идущие в конце будут вносить больший вклад в результат, чем начальные. В некоторых типах задач этот эффект оказывает положительное влияние, например в предсказании прогноза погоды. В ином случае для предотвращения этого эффекта используется двухнаправленная нейронная сеть, которая представляет из себя две одинаковых нейронных сети, одна из которых принимает признаки в прямом порядке, а вторая в обратном. На выходе представления объединяются. Схематичный принцип работы сети показан на рисунке 5.

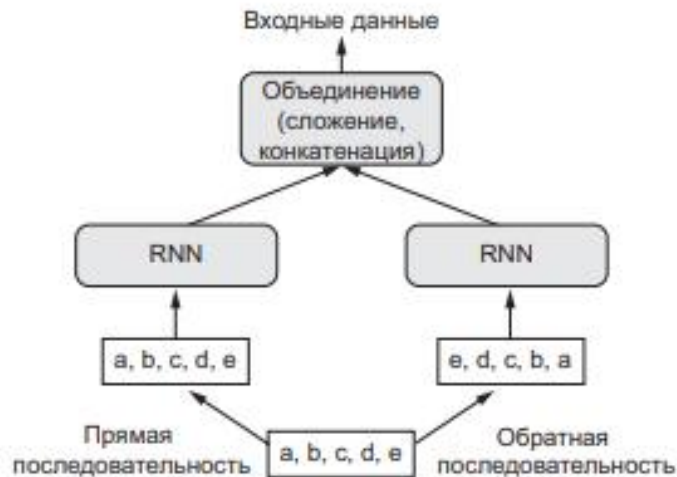


Рисунок 5 - Схема работы двунаправленной рекуррентной сети

Инициализация и обучение модели показано в листинге 3. Графики точности и потерь показаны на рисунках 6-7.

```
model=Sequential()
model.add(Embedding(num_wprds,3,input_length=500))
model.add(Bidirectional(LSTM(32, return_sequences = True)))
model.add(GlobalMaxPool1D())
model.add(Dense(20, activation="relu"))
model.add(Dropout(0.05))
model.add(Dense(1, activation="sigmoid"))

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

h=model.fit(data[10000:],labels[10000:], epochs=num_epochs,batch_size=512
            ,validation_data=(data[:5000],labels[:5000]),verbose=2)
```

Листинг 3 - Инициализация и обучения двунаправленной рекуррентной сети

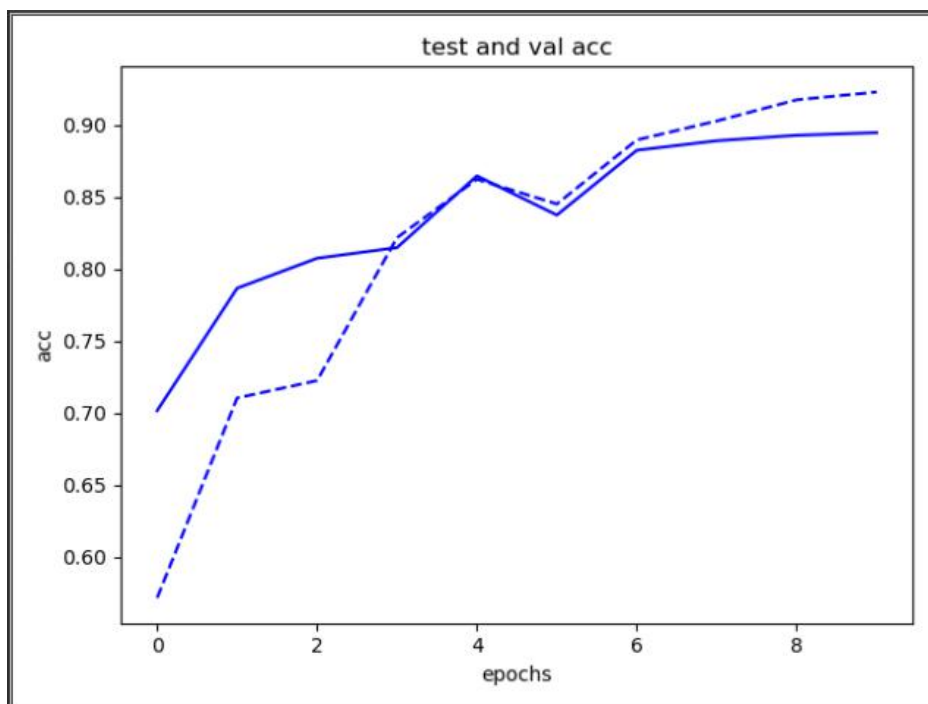


Рисунок 6 - Точность двунаправленной рекуррентной сети

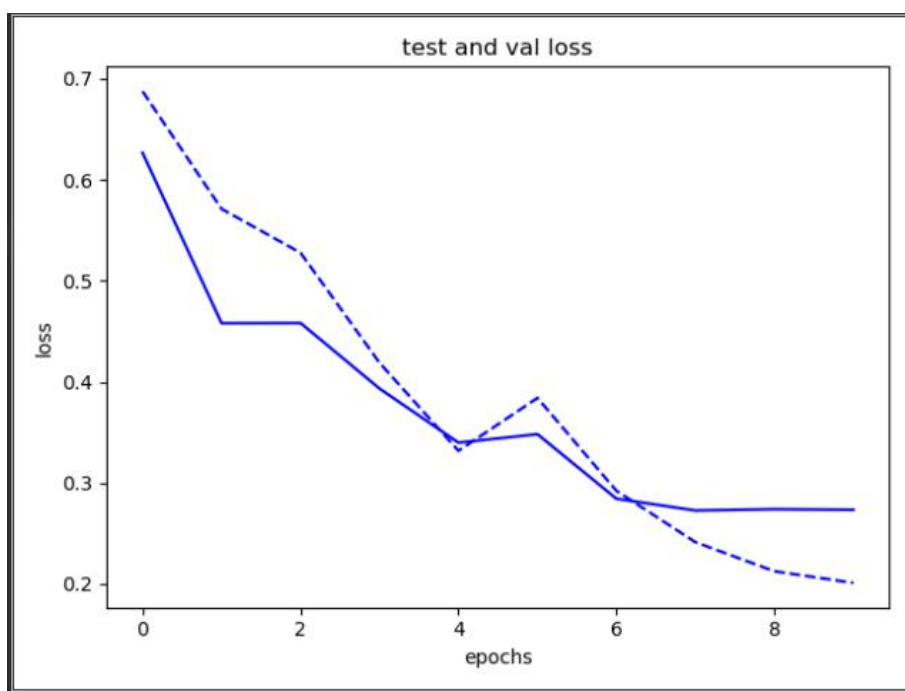


Рисунок 7 - Потери двунаправленной рекуррентной сети

Ансамблирование.

Все описанные выше модели сохраняются в соответствующие файлы, модуль `ensemble.py` реализует ансамблирование моделей. Тестирование производится по заранее выделенным образцам. Реализация ансамблирования показана в листинге 4.

```
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=num_wprds)
data = np.concatenate((train_data, test_data), axis=0)
labels = np.concatenate((train_labels, test_labels), axis=0)

data = sequence.pad_sequences(data, maxlen=500)
labels = np.asarray(labels).astype('float32')

cnn_lstm = load_model('model_cnn_lstm.h5')
fnn = load_model('model.h5')
bi_cnn = load_model('model_bi.h5')

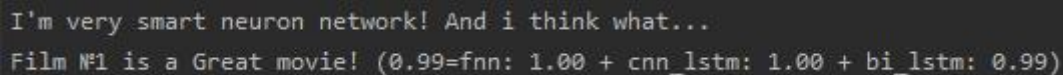
lstm_pred = cnn_lstm.evaluate(data[5000:10000], labels[5000:10000])
fnn_pred = fnn.evaluate(data[5000:10000], labels[5000:10000])
bi_pred = bi_cnn.evaluate(data[5000:10000], labels[5000:10000])
print((lstm_pred[1] + fnn_pred[1] + bi_pred[1]) / 3)
```

Листинг 4 - Реализация ансамблирования

Точность подхода составляет 0.8934.

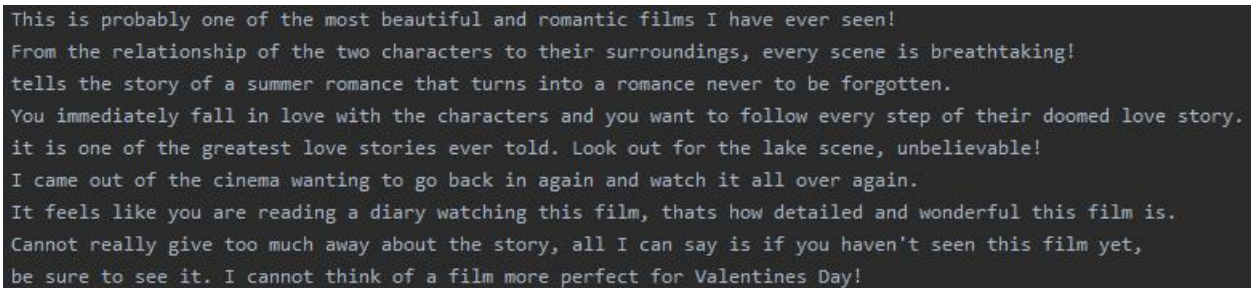
Тестирование на своих текстах.

Для загрузки пользовательских обзоров взят модуль из прошлой работы. Отличие лишь в том, что не требуется прямое кодирование образцов, они обрезаются до длины в 500 слов, а слой embedding уже векторизует индексы слов. Также добавлен вывод уверенности в результате, как группы моделей, так и каждой модели в отдельности. Пример работы модуля показан на рисунке 8, сам обзор на рисунке 9.



```
I'm very smart neuron network! And i think what...  
Film №1 is a Great movie! (0.99=fnn: 1.00 + cnn_lstm: 1.00 + bi_lstm: 0.99)
```

Рисунок 8 - Пример работы модуля предсказания пользовательских обзоров



```
This is probably one of the most beautiful and romantic films I have ever seen!  
From the relationship of the two characters to their surroundings, every scene is breathtaking!  
tells the story of a summer romance that turns into a romance never to be forgotten.  
You immediately fall in love with the characters and you want to follow every step of their doomed love story.  
it is one of the greatest love stories ever told. Look out for the lake scene, unbelievable!  
I came out of the cinema wanting to go back in again and watch it all over again.  
It feels like you are reading a diary watching this film, thats how detailed and wonderful this film is.  
Cannot really give too much away about the story, all I can say is if you haven't seen this film yet,  
be sure to see it. I cannot think of a film more perfect for Valentines Day!
```

Рисунок 9 - Пользовательский обзор

Особенность метода ансамблирования заключается в том, что каждая отдельная модель не просто хорошо справляется с задачей, а имеет свое особое представления, относительно одного образца. Так например мы знаем, что рекуррентная сеть чувствительно к последовательности признаков, а сверточная переходит к более высокоуровневым признаком. Тогда работу сверточной модели с рекуррентным слоем можно истолковать, как нахождение закономерности в последовательности словосочетаний. А работу двунаправленной рекуррентной сети, как нахождение закономерности в положении слов. С помощью описанного выше модуля, наглядно покажу, что каждая модель подходит к получению результата по-своему. Для этого к обзору на рис. 9 добавлю в начало выбросы и посмотрю как они повлияют на

итоговую точность. Получившийся обзор с ничего не значащими выбросами показан на рисунке 10.

```
VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE
VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE

This is probably one of the most beautiful and romantic films I have ever seen!
From the relationship of the two characters to their surroundings, every scene is breathtaking!
tells the story of a summer romance that turns into a romance never to be forgotten.
You immediately fall in love with the characters and you want to follow every step of their doomed love story.
it is one of the greatest love stories ever told. Look out for the lake scene, unbelievable!
I came out of the cinema wanting to go back in again and watch it all over again.
It feels like you are reading a diary watching this film, thats how detailed and wonderful this film is.
Cannot really give too much away about the story, all I can say is if you haven't seen this film yet,
be sure to see it. I cannot think of a film more perfect for Valentines Day!
```

Рисунок 10 - Обзор с ничего не значащими выбросами

Предсказание ансамбля сетей:

```
I'm very smart neuron network! And i think what...
Film №1 is a Great movie! (0.59=fnn: 0.58 + cnn_lstm: 0.98 + bi_lstm: 0.20)
```

Каждая модель имеет свое представление относительно этого обзора. Начало обзора вносит большой вклад в результат полносвязной модели и двунаправленной рекуррентной модели, чем однонаправленной. Теперь перенесу выбросы в конец обзора и снова сравню предсказания.

```
This is probably one of the most beautiful and romantic films I have ever seen!
From the relationship of the two characters to their surroundings, every scene is breathtaking!
tells the story of a summer romance that turns into a romance never to be forgotten.
You immediately fall in love with the characters and you want to follow every step of their doomed love story.
it is one of the greatest love stories ever told. Look out for the lake scene, unbelievable!
I came out of the cinema wanting to go back in again and watch it all over again.
It feels like you are reading a diary watching this film, thats how detailed and wonderful this film is.
Cannot really give too much away about the story, all I can say is if you haven't seen this film yet,
be sure to see it. I cannot think of a film more perfect for Valentines Day!

VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE
VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE VERY BAD HATE
```

Предсказание моделей:

```
I'm very smart neuron network! And i think what...
Film №1 is a horrible movie!!! (0.22=fnn: 0.01 + cnn_lstm: 0.34 + bi_lstm: 0.30)
```

Из проделанных тестов следует, что каждая модель, входящая в ансамбль имеет свое представление относительно одного и того же образца, из этого следует, что каждая отдельная модель может увидеть какую-то

закономерность упущенную другой моделью, от чего конечный результат ансамбля будет более полный, чем результат любой отдельной модели.

Вывод

В итоге данной лабораторной работы, был получен ансамбль нейронных сетей предсказывающий успех фильма по обзорам, разработан модуль для обработки пользовательских обзоров.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ ТЕКСТ

```
import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from keras.layers import Dense, Dropout, Embedding, Flatten, LSTM,
Conv1D, MaxPooling1D, GlobalMaxPool1D, Bidirectional
from keras.models import Sequential
from matplotlib import pyplot as plt
from keras.utils import to_categorical

from keras.preprocessing import sequence

from keras.datasets import imdb

num_wprds=10000
num_epochs=4

(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_w
ords=num_wprds)
data = np.concatenate((train_data, test_data), axis=0)
labels = np.concatenate((train_labels, test_labels), axis=0)

data = sequence.pad_sequences(data, maxlen=500, truncating='post')

labels = np.asarray(labels).astype('float32')

model = Sequential()
model.add(Embedding(num_wprds, 3, input_length=500))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(64, activation="relu"))
model.add(Dropout(0.2))
model.add(Dense(1, activation="sigmoid"))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['ac
curacy'])

h = model.fit(data[10000:], labels[10000:],
epochs=num_epochs, batch_size=512
, validation_data=(data[:5000], labels[:5000]), verbose=2)

model.save('model.h5')

plt.plot(range(num_epochs), h.history['val_accuracy'], 'b-', label='val')
plt.plot(range(num_epochs), h.history['accuracy'], 'b--', label='test')
plt.title('test and val acc')
plt.xlabel('epochs')
```

```
plt.ylabel('acc')
plt.show()

plt.plot(range(num_epochs),h.history['val_loss'],'b-',label='val')
plt.plot(range(num_epochs),h.history['loss'],'b--',label='test')
plt.title('test and val loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.show()
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ ТЕКСТ МОДУЛЯ ОБРАБОТКИ ПОЛЬЗОВАТЕЛЬСКИХ ОБЗОРОВ

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from keras.models import load_model
from keras.datasets.imdb import get_word_index
from sys import argv
import numpy as np

from keras.preprocessing import sequence
from keras.preprocessing.text import text_to_word_sequence

def filter(list_of_revs):
    for i in range(len(list_of_revs)):
        j=0
        while j<len(list_of_revs[i]):
            if list_of_revs[i][j] > 10000:
                del list_of_revs[i][j]
            else:
                j+=1

def decode(rev):
    reverse_word_index = dict(
        [(value, key) for (key, value) in word_index.items()])
    decoded_review = ' '.join(
        [reverse_word_index.get(i - 3, '?') for i in rev])
    return decoded_review

reviews=argv[1:]
word_index=get_word_index()

for i, nameOfFile in enumerate(reviews):
    with open(nameOfFile, 'r') as file:
        strings = text_to_word_sequence(file.read())
        for j, word in enumerate(strings):
            if word in word_index:
                #print(word, word_index[word])
                strings[j]=word_index[word]+3
            else:
                strings[j]=2
        reviews[i]=strings

filter(reviews)

reviews=sequence.pad_sequences(reviews,maxlen=500)
```

```

cnn_lstm=load_model('model_cnn_lstm.h5')
fnn=load_model('model.h5')
bi_cnn=load_model('model_bi.h5')

cnn_lstm_pred=cnn_lstm.predict(reviews)
fnn_pred=fnn.predict(reviews)
bi_cnn_pred=bi_cnn.predict(reviews)

prediction=(cnn_lstm_pred+fnn_pred+bi_cnn_pred)/3

print("I'm very smart neuron network! And i think what...")
for i, pr in enumerate(prediction):
    if np.round(pr[0])==1:
        print('Film №{0} is a Great movie!'.format(i+1),'({:.2f}=fnn: {:.2f} + cnn_lstm: {:.2f} + bi_lstm: {:.2f})'.format(pr[0],fnn_pred[i][0],cnn_lstm_pred[i][0],bi_cnn_pred[i][0]))
    else:
        print('Film №{0} is a horrible movie!!!'.format(i+1),'({:.2f}=fnn: {:.2f} + cnn_lstm: {:.2f} + bi_lstm: {:.2f})'.format(pr[0],fnn_pred[i][0],cnn_lstm_pred[i][0],bi_cnn_pred[i][0]))

```


ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ ТЕКСТ МОДУЛЯ ДЛЯ АНСАМБЛИРОВАНИЯ

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from keras.models import load_model
import numpy as np
from keras.datasets import imdb
from keras.preprocessing import sequence

num_wprds=10000

(train_data,train_labels),(test_data,test_labels)=imdb.load_data(num_w
ords=num_wprds)
data=np.concatenate((train_data,test_data),axis=0)
labels=np.concatenate((train_labels,test_labels),axis=0)

data=sequence.pad_sequences(data,maxlen=500)
labels=np.asarray(labels).astype('float32')

cnn_lstm=load_model('model_cnn_lstm.h5')
fnn=load_model('model.h5')
bi_cnn=load_model('model_bi.h5')

lstm_pred=cnn_lstm.evaluate(data[5000:10000],labels[5000:10000])
fnn_pred=fnn.evaluate(data[5000:10000],labels[5000:10000])
bi_pred=bi_cnn.evaluate(data[5000:10000],labels[5000:10000])
print((lstm_pred[1]+fnn_pred[1]+bi_pred[1])/3)
```