**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 |  | Киреев К.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2021

**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**

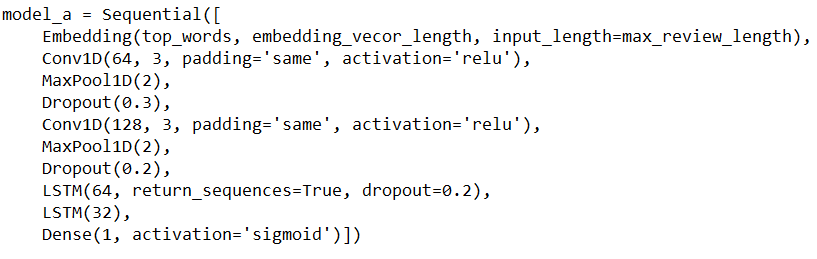
* Найти набор оптимальных ИНС для классификации текста
* Провести ансамблирование моделей
* Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
* Провести тестирование сетей на своих текстах (привести в отчете)

**Ход работы**

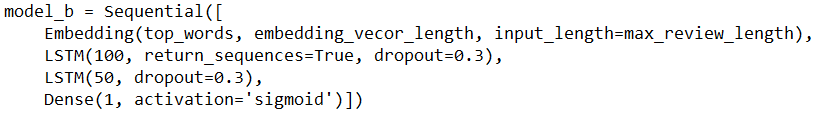
***Модели для классификации текста***

Были выбраны и реализованы три различные архитектуры нейронной сети для классификации текста. Архитектуры моделей представлены ниже.

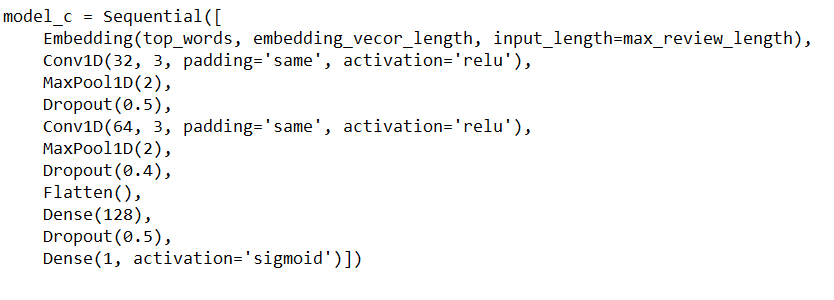
Модель 1:



Модель 2:

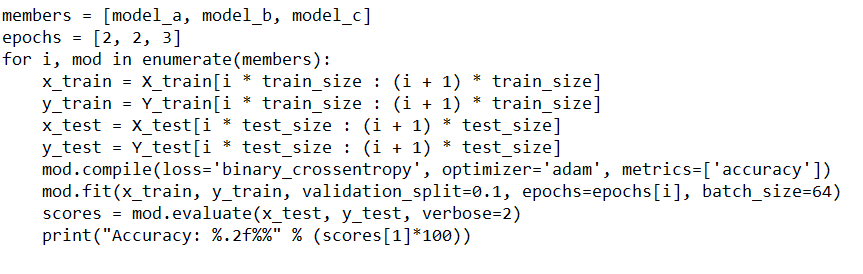


Модель 3:

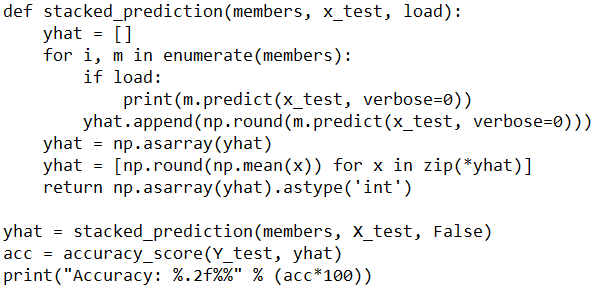


***Ансамблирование моделей***

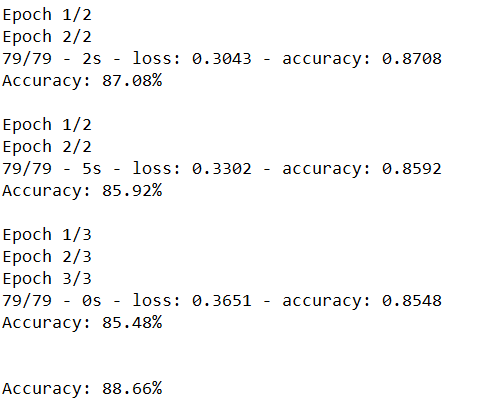
Было реализовано обучение моделей и их оценка на тестовых данных. В цикле выводится точность для каждой отдельной модели.



Была написана функция для получения результата предсказания ансамбля моделей. Далее производится оценка точности ансамбля моделей.



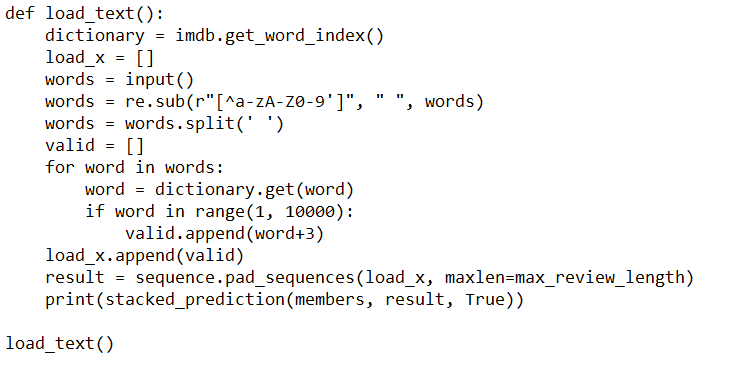
Точность отдельных моделей и ансамбля представлена ниже.



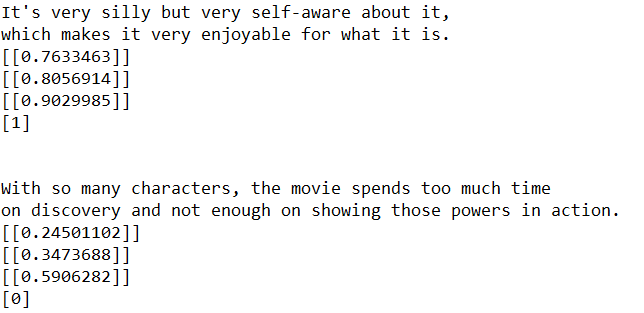
Точность ансамбля моделей составила 88.66%, что превосходит показатели отдельных моделей.

***Функция ввода пользовательского текста***

Была написана функция для ввода пользовательского текста load\_text()



Получаем словарь со словами и их индексами. Далее обрабатываем введенный текст, удаляя лишние символы. Заменяем числа на их индексы, оставляя только 10000 самых частых слов. Далее расширяем текст до длины 500 и выводим результат предсказания ансамбля нейронных сетей. Пример работы представлен ниже.



**Выводы**

В ходе лабораторной работы был реализован прогноз успеха фильмов по обзорам. Также был применен метод ансамблирования моделей для более точного семантического анализа текста.

Приложение А. Исходный код программы

import re

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

os.environ['TF\_FORCE\_GPU\_ALLOW\_GROWTH'] = 'true'

from matplotlib import gridspec

import matplotlib

import matplotlib.pyplot as plt

import numpy as np

from keras.utils import to\_categorical

from keras.layers import Dense, Dropout, LSTM, Conv1D, MaxPool1D, GRU, Flatten

from keras.models import Sequential

from keras.layers.embeddings import Embedding

from keras.preprocessing import sequence

from sklearn.metrics import accuracy\_score

from keras.datasets import imdb

(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)

X\_test = data[:10000]

Y\_test = targets[:10000]

X\_train = data[10000:]

Y\_train = targets[10000:]

max\_review\_length = 500 # макс. длина текста

embedding\_vecor\_length = 32 # 32-мерное векторное представление

top\_words = 10000 # количество слов, рассматриваемых как признаки

# print(X\_train[0])

X\_train = sequence.pad\_sequences(X\_train, maxlen=max\_review\_length)

# print(X\_train[0])

X\_test = sequence.pad\_sequences(X\_test, maxlen=max\_review\_length)

model\_a = Sequential([

Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length),

Conv1D(64, 3, padding='same', activation='relu'),

MaxPool1D(2),

Dropout(0.3),

Conv1D(128, 3, padding='same', activation='relu'),

MaxPool1D(2),

Dropout(0.2),

LSTM(64, return\_sequences=True, dropout=0.2),

LSTM(32),

Dense(1, activation='sigmoid')])

model\_b = Sequential([

Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length),

LSTM(100, return\_sequences=True, dropout=0.3),

LSTM(50, dropout=0.3),

Dense(1, activation='sigmoid')])

model\_c = Sequential([

Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length),

Conv1D(32, 3, padding='same', activation='relu'),

MaxPool1D(2),

Dropout(0.5),

Conv1D(64, 3, padding='same', activation='relu'),

MaxPool1D(2),

Dropout(0.4),

Flatten(),

Dense(128),

Dropout(0.5),

Dense(1, activation='sigmoid')])

ans\_len = 4

train\_size, test\_size = len(X\_train) // ans\_len, len(X\_test) // ans\_len

members = [model\_a, model\_b, model\_c]

epochs = [2, 2, 3]

for i, mod in enumerate(members):

x\_train = X\_train[i \* train\_size : (i + 1) \* train\_size]

y\_train = Y\_train[i \* train\_size : (i + 1) \* train\_size]

x\_test = X\_test[i \* test\_size : (i + 1) \* test\_size]

y\_test = Y\_test[i \* test\_size : (i + 1) \* test\_size]

mod.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

mod.fit(x\_train, y\_train, validation\_split=0.1, epochs=epochs[i], batch\_size=64)

scores = mod.evaluate(x\_test, y\_test, verbose=2)

print("Accuracy: %.2f%%" % (scores[1]\*100))

def stacked\_prediction(members, x\_test, load):

yhat = []

for i, m in enumerate(members):

if load:

print(m.predict(x\_test, verbose=0))

yhat.append(np.round(m.predict(x\_test, verbose=0)))

yhat = np.asarray(yhat)

yhat = [np.round(np.mean(x)) for x in zip(\*yhat)]

return np.asarray(yhat).astype('int')

yhat = stacked\_prediction(members, X\_test, False)

acc = accuracy\_score(Y\_test, yhat)

print("Accuracy: %.2f%%" % (acc\*100))

def load\_text():

dictionary = imdb.get\_word\_index()

load\_x = []

words = input()

words = re.sub(r"[^a-zA-Z0-9']", " ", words)

words = words.split(' ')

valid = []

for word in words:

word = dictionary.get(word)

if word in range(1, 10000):

valid.append(word+3)

load\_x.append(valid)

result = sequence.pad\_sequences(load\_x, maxlen=max\_review\_length)

print(stacked\_prediction(members, result, True))

load\_text()