МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ВЫЧИСЛИТЕЛЬНЫХ КОМПЛЕКСОВ

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | В.В. Боженко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| Бинарный классификатор |
| по дисциплине: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4816 |  |  |  | Кулиев М.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

Содержание

[**Бинарный классификатор 1**](#_Toc60040799)

[**1. Постановка задачи 3**](#_Toc60040800)

[**3.3 Описание используемых библиотек 11**](#_Toc60040801)

[**3.5 Системные требования 12**](#_Toc60040802)

[**4. Тестирование разработанной программы 12**](#_Toc60040803)

[**4.2 Запуск программы 12**](#_Toc60040804)

[**4.3 Запуск обучения 13**](#_Toc60040805)

[**4.4 Результаты обучения 13**](#_Toc60040806)

[**Заключение 14**](#_Toc60040807)

[**Список использованных источников 15**](#_Toc60040808)

[**Приложение А. листинг 16**](#_Toc60040809)

**Введение**

Скорее всего, вы знаете, что Python — это самый популярный высокоуровневый язык программирования с динамической семантикой. Он довольно прост для работы и чтения: его использование снижает стоимость разработки и обслуживания программ. Python считается самым простым языком программирования — именно поэтому он самый распространенный. Почему Python так часто используется в машинном обучении?По сути, машинное обучение — это технология, которая помогает приложениям на основе искусственного интеллекта обучаться и выдавать результаты автоматически, без человеческого вмешательства. В чем состоит работа специалиста по машинному обучению? Он должен собирать, систематизировать и анализировать данные, а затем на основе полученной информации создавать алгоритмы для искусственного интеллекта. Python лучше всего подходит для выполнения таких задач, потому что он довольно понятный по сравнению с другими языками.

Цель курсовой работы является создание, бинарного классификатора на Python. При этом для классификации будут использоваться отзывы на сайте IMDB. Для достижения данной цели необходимо решить следующие задачи: изучить основы Python, познакомится с библиотеками, отвечающие за Frontend и Backend.

# Постановка задачи

Задачей курсовой работы является разработка программы на python. Это язык очень хорош в системном программировании, в графическом программировании, в создании программ для вычислений, разработке робот программ и игр соответственно. Python занимает 3 место в рейтинге TIOBE (индекс, оценивающий популярность использования различных языков программирования). Плюсы этого языка заключаются в том, что он простой для изучения, у него большая обширность, есть лицензия, доступность бесплатного использования, большая поддержка разработчиками, много библиотек, Python подходит для большинства типов операционных систем. Но у него также есть и свои минусы: невозможность написания драйверов, низкая скорость выполнения программ, несовместимость разных версий языка.

1. **Методы и алгоритм решения задачи**

Самый популярные для машинного обучения библиотеки это Keras. Keras является надстройкой над фреймворком tensorflow и нацелен на оперативную работу с [сетями глубинного обучения](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D1%83%D0%B1%D0%BE%D0%BA%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), при этом спроектирована так, чтобы быть компактной, модульной и расширяемой. Эта библиотека содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как слои, [целевые](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) и передаточные функции, [оптимизаторы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), и множество инструментов для упрощения работы с изображениями и текстом.

Для обучения нужен готовый датасет, с отзывами и оценкой отзыва.

Готовый датасет из 50000 отзывов и оценки был скачан с сайта Kaggle это сайт используется для обучения нейросетей. Далее этот датасет необходимо подготовить для обучения, то есть отзывы нужно преобразовать в читаемый для компьютера вид. Для этого будет применена токенизация слов по частоте, далее будет этап векторизации когда токенизированный текст проходит через входной слой и там он преобразуется в плотный вектор. Затем это вектор будет проходить через реккурентный слой нейросети который будет иметь 128 нейронов они будут обучаться. На последнем слое будет только функция активации которая определяет положительный отзыв или отрицательный.

Чтобы корректировать обучение будет выводиться график в котором будут показаны обучение на обучающем наборе и проверочном наборе.

Для тестирования на своих данных был создан веб скрапер который добывает отзывы и оценку, а потом сохраняет в csv файл. На уже обученной сети будет проходить тестирование собственных данных. В конце чтобы проверить работу можно написать свой отзыв и проверить положительный он или нет.

# Описание разработанной программы

Использоваться в курсовой работе будет библиотеки keras, matplotlib, pandas, beautifulSoup, requests numpy,для backendа, для оболочки программы была использована библиотека PySimpleGUI. Для обучения нужен готовый датасет, с отзывами и оценкой отзыва.

Готовый датасет из 50000 отзывов и оценки был скачан с сайта Kaggle это сайт используется для обучения нейросетей. Далее этот датасет необходимо подготовить для обучения, то есть отзывы нужно преобразовать в читаемый для компьютера вид. Для этого будет применена токенизация слов по частоте, далее будет этап векторизации когда токенизированный текст проходит через входной слой и там он преобразуется в плотный вектор. Затем это вектор будет проходить через реккурентный слой нейросети который будет иметь 128 нейронов они будут обучаться. На последнем слое будет только функция активации которая определяет положительный отзыв или отрицательный.

Чтобы корректировать обучение будет выводиться график в котором будут показаны обучение на обучающем наборе и проверочном наборе.

Для тестирования на своих данных был создан веб скрапер который добывает отзывы и оценку, а потом сохраняет в csv файл. На уже обученной сети будет проходить тестирование собственных данных. В конце чтобы проверить работу можно написать свой отзыв и проверить положительный он или нет

* 1. **Фрагменты листинга**

В Листинге № 1 Все использованные библиотеки.

Листинг №1 – Начало программы

|  |
| --- |
| import keras  from keras import models  from keras import layers  from keras.datasets import imdb  from keras.callbacks import ModelCheckpoint  from keras.models import Sequential  from keras.layers import Dense, Embedding, GRU, LSTM  from keras import utils  from keras.preprocessing.sequence import pad\_sequences  from keras.preprocessing.text import Tokenizer  from keras.callbacks import ModelCheckpoint  import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  import random  from functools import partial  from bs4 import BeautifulSoup  import requests  import re  import csv  from datetime import datetime  import PySimpleGUI as sg |

В Листинге № 2 будет листинг вызова окна и все настройки.

Листинг №2 – Шарик

|  |
| --- |
| setup\_column = [  [sg.Text('Max review length:')],  [sg.Spin(key='-MAX-REVIEW-LENGTH-', values=[i for i in range(100, 200)], initial\_value = 150, size=(6, 1))],  [sg.Text('Choose model and encoding:')],  [sg.Combo(key='-MODEL-ENCODING-', values=['LSTM one hot', 'LSTM embedding', 'Convolutional one hot', 'Convolutional embedding'], size=(30, 1))],  [sg.Text('Choose optimizer:')],  [sg.Combo(key='-OPTIMIZER-', values=['RMSprop', 'SGD', 'Adam'], size=(30, 1))],  [sg.Text('# epochs'), sg.Spin(key='-EPOCHS-', values=[i for i in range(1, 1000)], initial\_value=200, size=(6, 1))],  [sg.Text('Ratio'), sg.Spin(key='-RATIO-', values=[i for i in range(0, 1)], initial\_value=200, size=(6, 1))],  [sg.Text('Package size'), sg.Spin(key='-PACKAGE-SIZE-', values=[i for i in range(1, 1000)], initial\_value=200, size=(6, 1))],    [sg.Text('IMDB url:')],  [sg.InputText(key='-URL-', size=(30, 1))],    [sg.Text('Review:')],  [sg.Multiline(key='-REVIEW-', size=(30, 5))],    [sg.Button(button\_text = 'Run')]  ]  layout = [  [  sg.Column(setup\_column)  ]  ]  # Create the window  window = sg.Window(  'IMDB review sentiment',  layout,  finalize=True,  element\_justification="center",  )  # Create an event loop  while True:  event, values = window.read()  print(event, values)  if event == 'Exit' or event == sg.WIN\_CLOSED:  break  window.close() |

* 1. **Описание функций**

Для разработки игры было написано 6 функций. А именно

1) def \_\_init\_\_ – зарезервированный метод в классах Python. Он известен как конструктор в концепциях ООП. Этот метод вызывается, когда объект создается из класса, и он позволяет классу инициализировать атрибуты класса. Первый аргумент относится к текущему объекту. Он связывает экземпляр с методом init()

2) В листинг ниже привед набор функций для сохранения данных с сайта

Листинг № 3 – Описания сохранения csv файла

|  |
| --- |
| def get\_input():  with open(\_\_INPUT\_FILE\_NAME, mode='r', encoding='utf-8') as reader:  return reader.read().splitlines()      def create\_output\_storage(name):  with open(resolve\_file\_name(name), mode='w+', encoding='utf-8', newline='') as csv\_file:  writer = csv.writer(csv\_file, \*\*\_\_FORMAT)  writer.writerow(\_\_HEADER)    def to\_filename(name):  return re.sub(re.compile('[^A-z]'), '', str(name))[0:10]    def resolve\_file\_name(name):  return f'{\_\_FILE\_NAME\_BASE}'    def store\_data(name, data):  with open(resolve\_file\_name(name), mode='a', encoding='utf-8', newline='') as csv\_file:  writer = csv.writer(csv\_file, \*\*\_\_FORMAT)  writer.writerows(data) |

3) Это набор вспомогательных функций которые проверяют на ответ 200 сайта, для проверки на наличия добываемых отзывов.

Листинг № 4 – вспомогательные функциии

|  |
| --- |
| def withLog(msg, fn=None, \*args):  result = fn(\*args) if callable(fn) else None  print(f'>>>> [LOG]: {msg}\n{result if result is not None else ""}')  return result    def get\_html(url, params=None):  response = requests.get(url, headers={'Accept-Language': 'en-US'})  return response.text if response.status\_code == 200 else ' '    def is\_valid\_url(url):  return re.match(re.compile('https://www\.imdb\.com/title/[A-z0-9]+/reviews'), url) is not None |

4) Основные функции которые добывают информацию они ищут по заданному шаблону отзыв и оценку

Листинг № 5 – шаблоны

|  |
| --- |
| def has\_more\_reviews(soup):  return len(soup.select(LOAD\_MORE)) > 0    def get\_more\_url(base\_url, soup):  more\_element = soup.select(LOAD\_MORE)[0]  return f'{base\_url}/\_ajax?paginationKey={more\_element["data-key"]}'    def get\_title(soup):  return soup.select(TITLE)[0].text if len(soup.select(TITLE)) > 0 else None    def get\_review(soup):  return soup.select(REVIEW)[0].text if len(soup.select(REVIEW)) > 0 else None    def get\_rating(soup):  return soup.select(RATING)[0].text if len(soup.select(RATING)) > 0 else None    def get\_data(soup):  return[  get\_rating(soup),  get\_review(soup)  ]    def get\_review\_items(soup):  return soup.select(REVIEW\_ITEM) |

4) Следующие функции отвечают за циклы скрапинга и за завершение

Листинг № 6 – Цикл

|  |
| --- |
| def shouldContinue(data, step):  return data and step <= MAX\_FETCHES    def run(url):  # Setup data  fetchCount = 1  html = get\_html(url)  soup = BeautifulSoup(html, 'html.parser')  title = get\_title(soup)  create\_output\_storage(title)  store\_data\_for\_title = partial(store\_data, title)    # Fetch loop  while shouldContinue(html, fetchCount):  soup = BeautifulSoup(html, 'html.parser')  withLog(  f'Fetch {fetchCount}',  store\_data\_for\_title,  map(  get\_data,  get\_review\_items(soup)  )  )    # Next step condition  fetchCount += 1  html = get\_html(get\_more\_url(url, soup)) if has\_more\_reviews(soup) else ''    for url in get\_input():  withLog(  f'\n-------------------------\nComplete for url {url}\n-------------------------',  run,  url  ) if is\_valid\_url(url) else withLog(f'Invalid url {url}') |

# Описание используемых библиотек

Pandas используется для работы с таблицами(датафреймами) данных.

Numpy для работы с преобразованиями данных и мат операциями.

Keras для работы с нейросетями и глубоким обучением

BeautifulSoup библиотека для добычи информации с интернета

Requests библиотека для отправки запросов сайтам

Csv для работы с csv файлами

PySimpleGUI Для создания оболочек программ

* 1. Особенности реализации приложения

Для обучения нужен готовый датасет, с отзывами и оценкой отзыва.

Готовый датасет из 50000 отзывов и оценки был скачан с сайта Kaggle это сайт используется для обучения нейросетей. Далее этот датасет необходимо подготовить для обучения, то есть отзывы нужно преобразовать в читаемый для компьютера вид. Для этого будет применена токенизация слов по частоте, далее будет этап векторизации когда токенизированный текст проходит через входной слой и там он преобразуется в плотный вектор. Затем это вектор будет проходить через реккурентный слой нейросети который будет иметь 128 нейронов они будут обучаться. На последнем слое будет только функция активации которая определяет положительный отзыв или отрицательный.

Чтобы корректировать обучение будет выводиться график в котором будут показаны обучение на обучающем наборе и проверочном наборе.

Для тестирования на своих данных был создан веб скрапер который добывает отзывы и оценку, а потом сохраняет в csv файл. На уже обученной сети будет проходить тестирование собственных данных. В конце чтобы проверить работу можно написать свой отзыв и проверить положительный он или нет.

# Системные требования

В состав технических средств должен входить IBM-совместный персональный компьютер, включающий в себя:

процессор intel-core-i7, не менее;

оперативную память объемом, 4 Гигабайт, не менее;

HDD, 256 МБ, не менее;

Операционную систему windows 10;

# Тестирование разработанной программы

# Запуск программы

Программа прошла тестирование с использованием нескольких прогонов, по всей функциональности.

При запуске программы выводится окно, для настройки гиперпараметров сети(Рисунок 1).

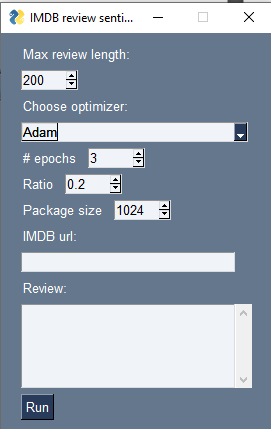


Рисунок 1 – Запуск программы

# Запуск обучения

После проведения настройки сети будет произведен запуск обучения

В выбранное количество эпох.

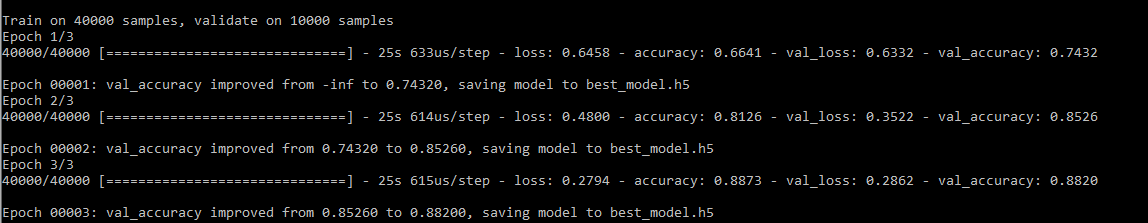


Рисунок 2 – Результат обучения

# Результаты обучения

Когда шарик упал на пол, то игра завершается и выводится соответствующее проигрышу сообщение(Рисунок 3).

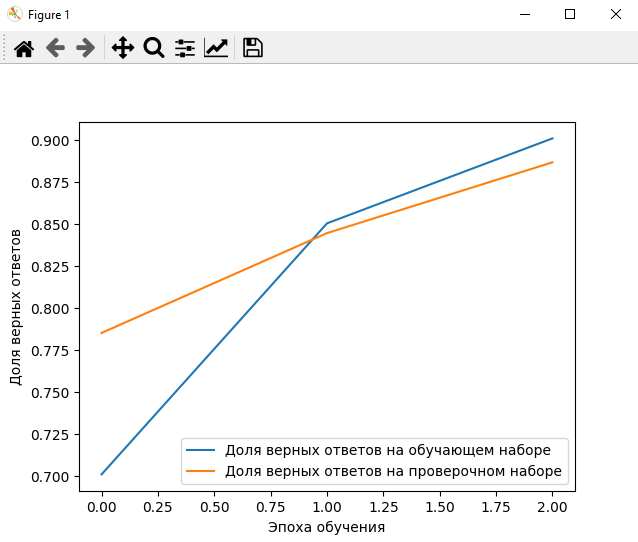


Рисунок 3 – Вывод результатов обучения

# Заключение

В процессе работы были закреплены навыки в разработке простых нейросетей с глубоким обучением на языке Python, а также была проведена работа по документированию программного обеспечения, согласно ГОСТ. Разработанный классификатор имеет точность в 87 процентов, что является на сегодняшний день не таким большим показателем но это зависит от обучающего датасета тк датасет состоял всего лишь из 50000 слов то и результат не такой высокий, но с другой стороны нейросеть обучается достаточно быстро. Если бы был датасет на 1000000 слов то и результат был бы выше, но скорость бы была намного ниже.

# Список использованных источников

1. Пола Бэрри, Изучаем программирование на Python / Бомбора, 2016. - 611 с.
2. Андреас Мюллер. Введение в машинное обучение с помощью Python. 2017. - 480с.

# Приложение А. листинг

Код написанной программы

|  |
| --- |
| import keras  from keras import models  from keras import layers  from keras.datasets import imdb  from keras.callbacks import ModelCheckpoint  from keras.models import Sequential  from keras.layers import Dense, Embedding, GRU, LSTM  from keras import utils  from keras.preprocessing.sequence import pad\_sequences  from keras.preprocessing.text import Tokenizer  from keras.callbacks import ModelCheckpoint  import numpy as np  import matplotlib.pyplot as plt  import pandas as pd  import random  from functools import partial  from bs4 import BeautifulSoup  import requests  import re  import csv  from datetime import datetime  import PySimpleGUI as sg  setup\_column = [  [sg.Text('Max review length:')],  [sg.Spin(key='-MAX-REVIEW-LENGTH-', values=[i for i in range(100, 200)], initial\_value = 200, size=(6, 1))],  [sg.Text('Choose optimizer:')],  [sg.Combo(key='-OPTIMIZER-', values=['RMSprop', 'SGD', 'Adam'], size=(30, 1))],  [sg.Text('# epochs'), sg.Spin(key='-EPOCHS-', values=[i for i in range(1, 25)], initial\_value=3, size=(6, 1))],  [sg.Text('Ratio'), sg.Spin(key='-RATIO-', values=[i for i in range(0, 1)], initial\_value=0.2, size=(6, 1))],  [sg.Text('Package size'), sg.Spin(key='-PACKAGE-SIZE-', values=[i for i in range(1, 1000)], initial\_value=200, size=(6, 1))],    [sg.Text('IMDB url:')],  [sg.InputText(key='-URL-', size=(30, 1))],    [sg.Text('Review:')],  [sg.Multiline(key='-REVIEW-', size=(30, 5))],    [sg.Button(button\_text = 'Run')]  ]  layout = [[sg.Column(setup\_column)]]  #Create the window  window = sg.Window(  'IMDB review sentiment',  layout,  finalize=True,  element\_justification="center",  )  #Create an event loop  while True:  event, values = window.read()  print(event, values)    if event == 'Run':    rev = pd.read\_csv("IMDB Dataset.csv",sep =',')  train\_review = rev['review']  rate = pd.read\_csv("rate\_train.csv")  train\_rate = rate['sentiment']      num\_words = 10000  max\_review\_len = int(values['-MAX-REVIEW-LENGTH-'])    tokenizer = Tokenizer(num\_words = num\_words)  tokenizer.fit\_on\_texts(train\_review)  sequences = tokenizer.texts\_to\_sequences(train\_review)    x\_train = pad\_sequences(sequences, maxlen = max\_review\_len)    model = Sequential()  model.add(Embedding(num\_words, 64, input\_length = max\_review\_len))  model.add(LSTM(128))  model.add(Dense(1, activation='sigmoid'))  print(type(values['-PACKAGE-SIZE-']),type(values['-EPOCHS-']),type(values['-RATIO-']))  model.compile(optimizer = values['-OPTIMIZER-'],  loss='binary\_crossentropy',  metrics=['accuracy'])    model\_save\_path = 'best\_model.h5'    checkpoint\_callback = ModelCheckpoint(model\_save\_path,  monitor='val\_accuracy',  save\_best\_only=True,  verbose=1)    history = model.fit(x\_train,  train\_rate,  epochs = int(values['-EPOCHS-']),  batch\_size = int(values['-PACKAGE-SIZE-']),  validation\_split = float(values['-RATIO-']),  callbacks = [checkpoint\_callback])    import matplotlib.pyplot as plt      plt.plot(history.history['accuracy'],  label='Доля верных ответов на обучающем наборе')  plt.plot(history.history['val\_accuracy'],  label='Доля верных ответов на проверочном наборе')  plt.xlabel('Эпоха обучения')  plt.ylabel('Доля верных ответов')  plt.legend()  plt.show()  f = open('input.txt', 'w')  f.write(values['-URL-'])  f.close()  # TODO: this should be input  # Number of 'Load more' 'clicks' per movie  MAX\_FETCHES = 5  \_\_INPUT\_FILE\_NAME = 'input.txt'  \_\_HEADER = ['rating', 'review']  \_\_FILE\_NAME\_BASE = 'reviews\_data.csv'  \_\_FORMAT = { 'delimiter': ';', 'quotechar': '"'}    # Selectors #######################################################################################  LOAD\_MORE = '.load-more-data[data-key]'  TITLE = '[itemprop=name] > a'  REVIEW\_ITEM = '[data-review-id]'  REVIEW = '[data-review-id] .content .text'  RATING = '[data-review-id] .rating-other-user-rating > span:first-of-type'  ###################################################################################################    # Utils ###########################################################################################    def withLog(msg, fn=None, \*args):  result = fn(\*args) if callable(fn) else None  print(f'>>>> [LOG]: {msg}\n{result if result is not None else ""}')  return result    def get\_html(url, params=None):  response = requests.get(url, headers={'Accept-Language': 'en-US'})  return response.text if response.status\_code == 200 else ''    def is\_valid\_url(url):  return re.match(re.compile('https://www\.imdb\.com/title/[A-z0-9]+/reviews'), url) is not None    ##################################################################################################    # Data ###########################################################################################    def has\_more\_reviews(soup):  return len(soup.select(LOAD\_MORE)) > 0    def get\_more\_url(base\_url, soup):  more\_element = soup.select(LOAD\_MORE)[0]  return f'{base\_url}/\_ajax?paginationKey={more\_element["data-key"]}'    def get\_title(soup):  return soup.select(TITLE)[0].text if len(soup.select(TITLE)) > 0 else None    def get\_review(soup):  return soup.select(REVIEW)[0].text if len(soup.select(REVIEW)) > 0 else None    def get\_rating(soup):  return soup.select(RATING)[0].text if len(soup.select(RATING)) > 0 else None    def get\_data(soup):  return[  get\_rating(soup),  get\_review(soup)  ]    def get\_review\_items(soup):  return soup.select(REVIEW\_ITEM)    #################################################################################################    # Storage #######################################################################################    def get\_input():  with open(\_\_INPUT\_FILE\_NAME, mode='r', encoding='utf-8') as reader:  return reader.read().splitlines()      def create\_output\_storage(name):  with open(resolve\_file\_name(name), mode='w+', encoding='utf-8', newline='') as csv\_file:  writer = csv.writer(csv\_file, \*\*\_\_FORMAT)  writer.writerow(\_\_HEADER)    def to\_filename(name):  return re.sub(re.compile('[^A-z]'), '', str(name))[0:10]    def resolve\_file\_name(name):  return f'{\_\_FILE\_NAME\_BASE}'    def store\_data(name, data):  with open(resolve\_file\_name(name), mode='a', encoding='utf-8', newline='') as csv\_file:  writer = csv.writer(csv\_file, \*\*\_\_FORMAT)  writer.writerows(data)    #################################################################################################    # Main ##########################################################################################  def shouldContinue(data, step):  return data and step <= MAX\_FETCHES    def run(url):  # Setup data  fetchCount = 1  html = get\_html(url)  soup = BeautifulSoup(html, 'html.parser')  title = get\_title(soup)  create\_output\_storage(title)  store\_data\_for\_title = partial(store\_data, title)    # Fetch loop  while shouldContinue(html, fetchCount):  soup = BeautifulSoup(html, 'html.parser')  withLog(  f'Fetch {fetchCount}',  store\_data\_for\_title,  map(  get\_data,  get\_review\_items(soup)  )  )    # Next step condition  fetchCount += 1  html = get\_html(get\_more\_url(url, soup)) if has\_more\_reviews(soup) else ''    for url in get\_input():  withLog(  f'\n-------------------------\nComplete for url {url}\n-------------------------',  run,  url  ) if is\_valid\_url(url) else withLog(f'Invalid url {url}')  #################################################################################################      model.load\_weights(model\_save\_path)  rev = pd.read\_csv("reviews\_data.csv",sep =';',header = None,names = ['rating','review'])  test\_rev = rev['review']  test\_rate = rev['rating']  test\_rate = test\_rate[1:]  test\_rev = test\_rev[1:]  clock = 1  while clock != len(test\_rate):  if test\_rate[clock]!=test\_rate[clock]:  test\_rate[clock] = random.randint(0,1)  clock = clock+1  print(test\_rate)  clock = 1  while clock != len(test\_rate):  test\_rate[clock] = int(test\_rate[clock])  clock = clock+1  clock = 1  while clock != len(test\_rate):  if test\_rate[clock]>5:  test\_rate[clock] = 1  elif test\_rate[clock] <= 5:  test\_rate[clock] = 0  clock = clock + 1  print(test\_rate)  test\_sequences = tokenizer.texts\_to\_sequences(test\_rev)    x\_test = pad\_sequences(test\_sequences, maxlen=max\_review\_len)  print(x\_test[2])  print(test\_rate)  resul = model.evaluate(x\_test, test\_rate, batch\_size=8)  print('test loss, test acc:', resul)    text = values['-REVIEW-']  textToken = tokenizer.texts\_to\_sequences([text])  seqText = pad\_sequences(textToken,maxlen=max\_review\_len)  result = model.predict(seqText)  if result > 0.5:  print("Отзыв положительный")  else:  print("Отзыв отрицательный")  if event == 'Exit' or event == sg.WIN\_CLOSED:  break  window.close() |