Минобрнауки России

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Направление (специальность) Системная инженерия   
Кафедра Системы автоматизированного и поискового конструирования

Дисциплина Компьютерная Лингвистика

|  |  |  |
| --- | --- | --- |
|  | Утверждаю Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
|  | «\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г. |  |

##### **Задание**

##### **на курсовую работу (проект)**

Студент Володин Владимир Юрьевич

##### (фамилия, имя, отчество)

Группа ИВТ-364

1. Тема: Распарсить сайт с помощью краулера и занести данные в БД MongoDB. cоздать программный модyль для анализа новостей из БД. cоздать программный модуль для проведения с помощью Spark MlLib анализ модели word2vec на всем объеме новостных статей из БД.

Утверждена приказом от «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. № \_\_\_\_\_\_\_\_\_

2. Срок представления работы (проекта) к защите «\_\_» 2020 г.

3. Содержание расчетно-пояснительной записки: Разработаны программные модули, руководство системного администратора

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Перечень графического материала: 19 изображений показывающих работу проекта\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Руководитель работы (проекта)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коробкин Д.М.

подпись, дата инициалы и фамилия

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

Минобрнауки России

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

##### Кафедра Системы автоматизированного и поискового конструирования

##### **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе (проекту)**

по дисциплине Компьютерная Лингвистика

на тему \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент Володин Владимир Юрьевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##### (фамилия, имя, отчество)

Группа ИВТ-364

Руководитель работы (проекта) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коробкин Д.М.

(подпись и дата подписания) (инициалы и фамилия)

Члены комиссии:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и дата подписания) (инициалы и фамилия)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и дата подписания) (инициалы и фамилия)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и дата подписания) (инициалы и фамилия)

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата подписания) (инициалы и фамилия)

Волгоград 2020 г.

Оглавление

[1. Цель проекта 4](#_Toc53615562)

[2. Задача проекта 4](#_Toc53615563)

[3. Описание парсера 5](#_Toc53615564)

[3.2 Описание полей 6](#_Toc53615565)

[3.3 Формат хранения информации в базе данных MongoDB 6](#_Toc53615566)

[3.4 Формат выполнения действий 7](#_Toc53615567)

[3.5 Результаты выполнения программы: 8](#_Toc53615568)

[3.6 Запуск базы данных в операционной среде Linux 10](#_Toc53615569)

[4.1 Описание программного модуля выделения упоминаний персон 12](#_Toc53615570)

[Заключение 14](#_Toc53615571)

[Cписок использованной литературы 15](#_Toc53615572)

[Приложение А 16](#_Toc53615573)

[Программный код модуля 1 16](#_Toc53615574)

[Программный код модуля 2 17](#_Toc53615575)

### 

# Цель проекта

Необходимо разработать программу для парсинга новостной страницы [V1.ru](https://v1.ru/), программный модуль для анализа новостей из БД с помощью Томита-парсера и программный модуль для проведения с помощью Spark MlLib анализа модели word2vec на всем объеме новостных статей из БД.

# Задача проекта

Написать программу в интегрированной среде разработки Visual Studio 2019 на языке python, которая будет выполнять сбор данных и заносить их в базу данных. Написать программу в интегрированной среде разработки PyCharm 2019 Professional на языке python c помощью Томита-парсера, а также встроенных библиотек, которая будет выполнять выделение упоминаний в тексте значимых персон Волгоградской области и достопримечательностей и заносить их в базу данных, а также программный модуль, анализирующий модели word2vec на всех новостях из базы данных. Запуск программы осуществить с помощью операционной среды Linux.

# Описание парсера

В данной проекте разрабатывается парсинг новостной страницы [V1.ru](https://v1.ru/). Написанный на языке Python код позволяет собирать данные с сайта в определённом порядке с разными атрибутами и заносить их в базу данных.В проекте используется база данных MongoDB. Информация сохраняется в коллекцию данных.

Информация хранится в формате как представлено ниже:

{

                    'title' : title,

                    'url': url,

                    'text': text\_url,

                    'datetime' : datetimes,

                    'count\_comment': count\_comment,

                    'count\_view': count\_view,

'video': video

}

# 3.2 Описание полей

Title - поле, позволяющее узнать название новости.

URL - поле, позволяющее узнать, ссылка на новость.

Datetime - поле, позволяющее узнать, дата и время публикации новости.

Count\_comment - поле, позволяющее узнать количество комментариев под новостью, указанными пользователем.

Count\_view - поле, позволяющее узнать количество просмотров новости.

# 3.3 Формат хранения информации в базе данных MongoDB

Формат данных в БД, указанных после запуска парсера новостей:

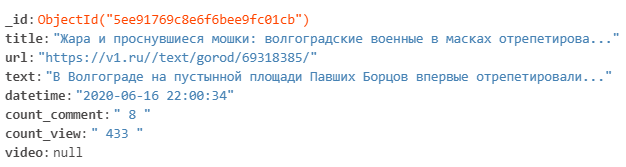


Рисунок 1- Формат данных в БД

# 3.4 Формат выполнения действий

Программа парсит страницу новостей v1.ru по определенным этапам:

1.    Распарсить страницу «Все новости»

2.    Переход на страницу список новостей

3.    Скрипт достает данные из тегов страницы

4. Вывод результаты в консоль

4.    Запись данных в БД.

Результат работы парсера:

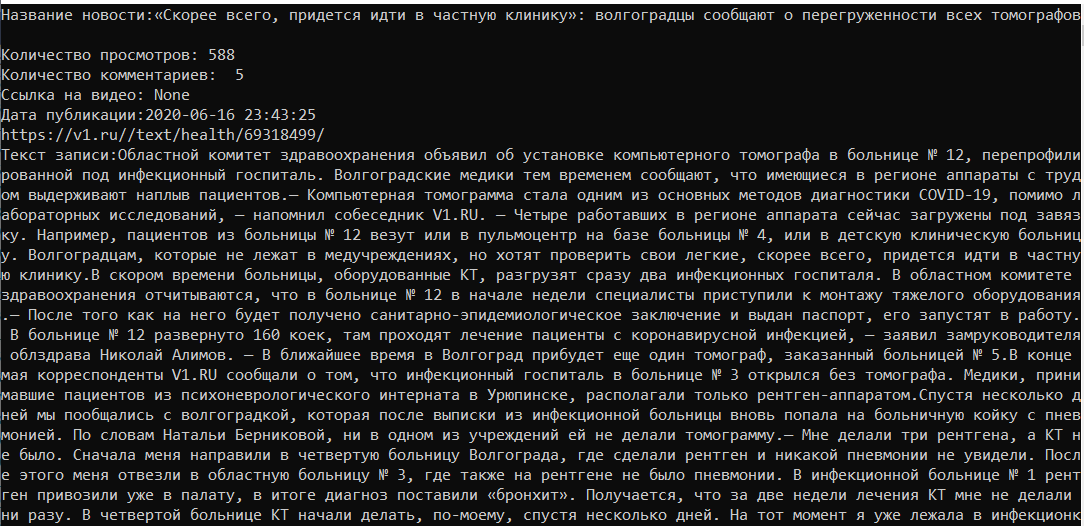


Рисунок 2 - результат выполнения парсера

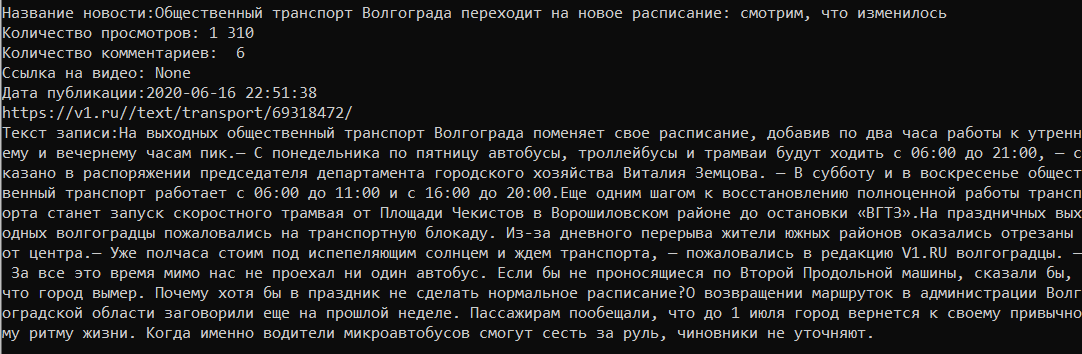


Рисунок 3 - результат выполнения парсера

Программа будет доставать информацию на сайте пока не кончатся страницы «Все новости».

# 3.5 Результаты выполнения программы:

Скриншоты таблиц заполненной парсером в базу данных MongoDB:

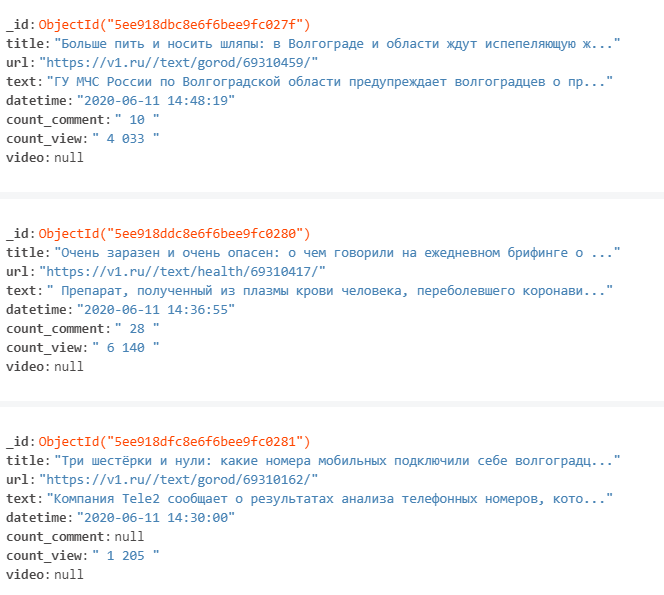


Рисунок 4 - хранение данных в БД

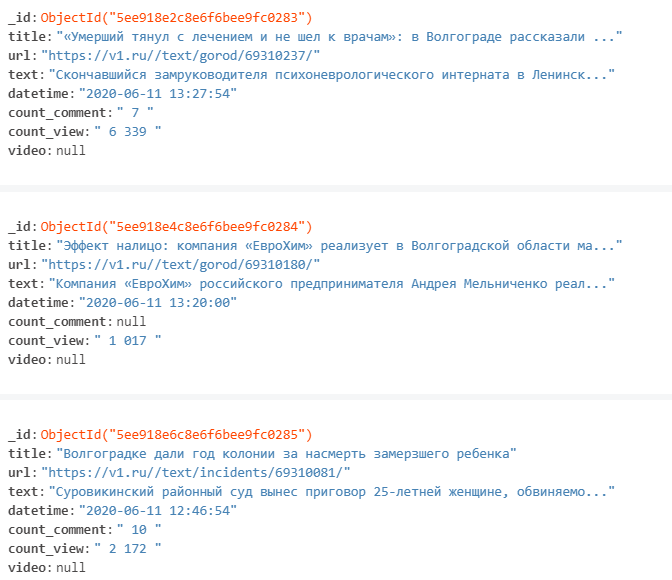


Рисунок 5 - хранение данных в БД



Рисунок 6 - хранение данных в БД

# 3.6 Запуск базы данных в операционной среде Linux

Просмотр существующих баз данных и активирование parserCLV1ru:

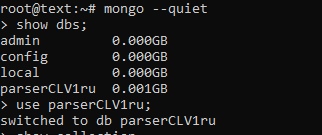


Рисунок 7 – активирование бд

Просмотр коллекций в активированной базы данных

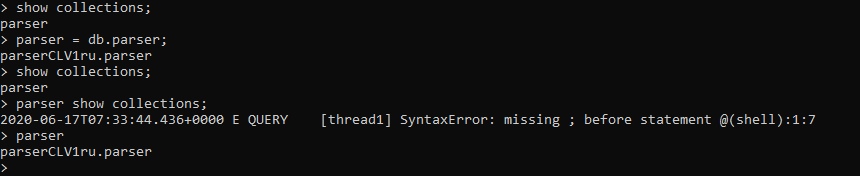


Рисунок 8 – открытие коллекций

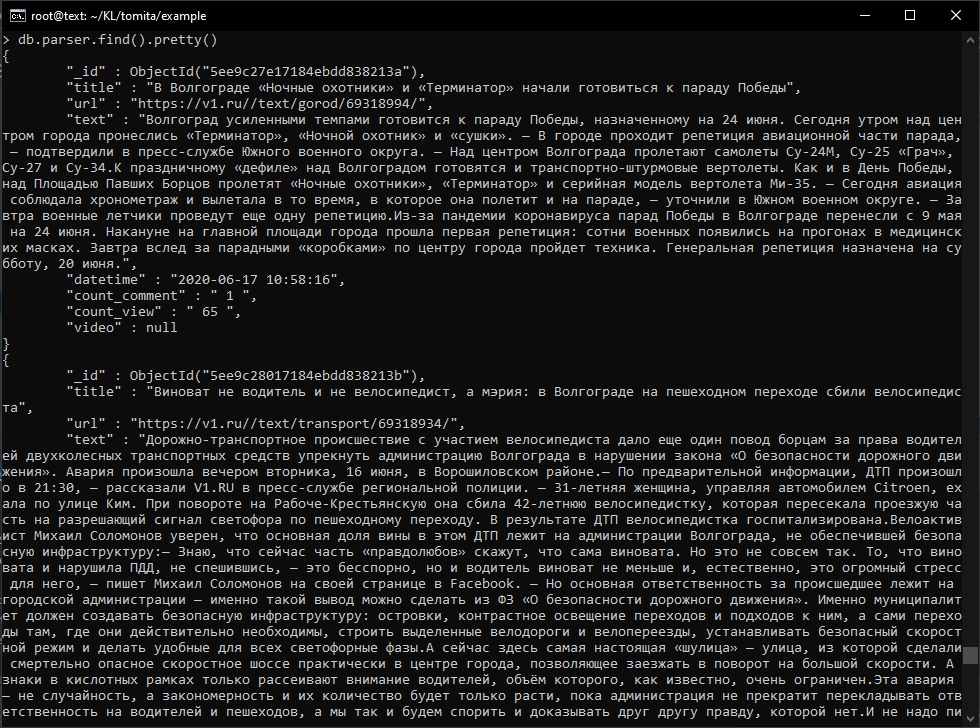


Рисунок 9 – информация о новостной ленте в БД MongoDB

# 4.1 Описание программного модуля выделения упоминаний персон

В проекте создается программный модуль для анализа новостей из БД. Выделение с помощью Томита-парсера упоминаний в тексте значимых персон Волгоградской области и достопримечательностей. Происходит фиксирование в БД предложения с их упоминанием для дальнейшего анализа тональности.

Также создается программный модуль для проведения с помощью Spark MlLib анализ модели word2vec на всем объеме новостных статей из БД. Для персон Волгоградской области и достопримечательностей определить контекстные синонимы и слова, с которыми они упоминались в тексте.

Для реализации программы, информация для программного модуля будет браться из базы данных MongoDB в формате json. Для этого необходимо конвертировать хранимую информацию в коллекциях базы данных в файл json.

Команда конвертирования данных: mongoexport -d parserCL -c parser-o C:/test/news.json

Создадим файлы для Томита-парсера, которые будут отвечать за нахождение Персон и Достопримечательностей Волгоградской области.

В них будут хранится словари, с помощью которых будут определяться ФИО персон и имена собственные достопримечательностей. Полученные предложения с упоминанием фиксируются в базе данных для дальнейшей работы третьим студентом.

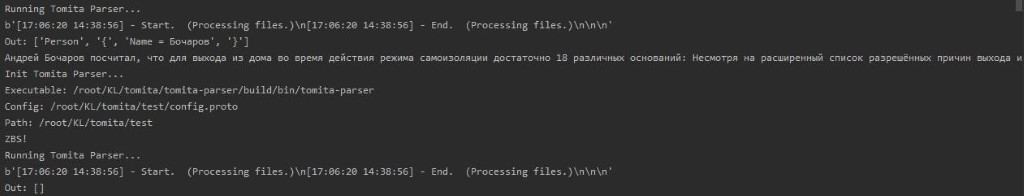


Рисунок 1. Пример выполнения томита-парсера

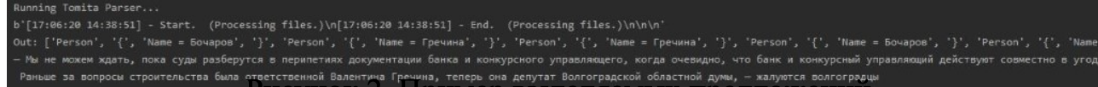


Рисунок 2. Пример выделяемых предложений Word2Vec

Модель word2vec была обучена на новостных статьях из базы данных, объем которой составлял примерно 1600 статей. Модель находится в папке

./news\_analyser/w2v/model/.

Запуск модуля осуществляется запуском программы

./news\_analyzer/w2v/main.py.

Модуль записывает контекстные синонимы в БД и осуществляет следующий вывод:

'Бочаров Андрей Иванович' бочаров

андрей губернатор попков куприков

' ' 'Мержоева Зина Османовна' зина

мержоева вицегубернатор эксвицегубернатор выстропов

# Заключение

В результате проделанной работы был разработан проект, осуществляющий парсинг данных со страницы новостей и занесение информации в базу данных MongoDB. Был разработан программный модуль, осуществляющий выделение упоминаний в тексте значимых персон Волгоградской области и достопримечательностей и занесение информации в базу данных MongoDB. Разработка осуществлялась с помощью операционной системы Linux, что позволило приобрести ряд навыков для написания проекта.

# Cписок использованной литературы

1. Установка и использование оболочки Linux **–** Режим доступа: <https://www.comss.ru/page.php?id=4897> (дата обращения 08.06.2020).
2. Основы работы в операционной системе Linux С.А.Немнюгин – Режим доступа: <http://www.cph.phys.spbu.ru/documents/Second/unix_SNv2.pdf> (дата обращения 09.09.2020)
3. Документация по **Beautiful Soup –** Режим доступа: <http://wiki.python.su/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D0%B8/BeautifulSoup> (дата обращения 10.06.2020)
4. Ознакомления с работой с **MongoDB**и**PyMongo – Режим доступа:** <https://api.mongodb.com/python/current/tutorial.html> (дата обращения 11.09.2020)
5. Установка и использование оболочки Linux – Режим доступа: <https://www.comss.ru/page.php?id=4897>(дата обращения 08.09.2020).
6. Наташа — библиотека для извлечения структурированной информации из текстов на русском языке – Режим доступа: <https://habr.com/ru/post/349864/> (дата обращения 12.09.2020)
7. Natasha - Документация <https://natasha.readthedocs.io/ru/latest/> (дата обращения 12.09.2020)

# Приложение А

# Программный код модуля 1

**import** requests

**from** bs4 **import** BeautifulSoup

**from** random **import** choice

**import** urllib.request

userparsers = open('userparser.txt').read().split('\n')

userparser = {'Userparser': choice(userparsers)}

# полчуение html страницы

**def** get\_html (url, userparser=**None**, proxies=**None**):

r=requests.get(url,headers=userparser,proxies=**None**)

**return** r.text

#Получение всех страници

**def** get\_total\_pages(html):

soup = BeautifulSoup(html,'lxml')

pages = soup.find('div', class\_='G1acv' ).find\_all('a')[-1].get('href')

result='https://v1.ru/'+ pages

**return** result

**def** get\_page\_data(html):

soup = BeautifulSoup(html, 'lxml')

items = soup.find('div', class\_='BXer').find\_all('article', class\_='MJaz5')

**for** c, item **in** enumerate(items, 1):

#time.sleep(1)

**try**:

title = item.find('div', class\_='MJaz7').find('a').get('title')

url = item.find('div', class\_='MJaz7').find('a').get('href')

datetimes = item.find('div', class\_='MJlr').find('time').get('datetime')

count\_view = item.find('div', class\_='LVaxt').find('span').get\_text()

#проверка на наличие комментариев

**if** item.find('a', class\_='LVcp LVbn') != **None**:

count\_comment = item.find('a', class\_='LVcp LVbn').find('span', class\_='LVcl').get\_text()

**else**:

count\_comment = **None**

#проверка на наличие видео

#iFrames=[] # qucik bs4 example

#for iframe in soup("iframe"):

#iFrames.append(soup.iframe.extract())

**if** BeautifulSoup(get\_html('https:/'+'/v1.ru/'+ url), 'lxml').find('div', class\_='ERsn') != **None**:

video = BeautifulSoup(get\_html('https:/'+'/v1.ru/'+ url), 'lxml').find('iframe')

**else**:

video = **None**

#достаем полностью текст

text\_url=''

**for** i **in** BeautifulSoup(get\_html('https:/'+'/v1.ru/'+ url), 'lxml').find\_all('div', class\_='LPawp'):

**for** j **in** i.find\_all('p'):

text\_url+=j.get\_text()

i = {

'title' : title,

'url': url,

'text': text\_url,

'datetime' : datetimes,

'count\_comment': count\_comment,

'count\_view': count\_view

}

print("Ссылка на видео:",video)

print("Количество просмотров:" + count\_view)

print("Количество комментариев:", count\_comment)

print("Текст записи:"+ text\_url)

print("Название новости:"+title)

print("Дата публикации:"+ datetimes)

print('https:/'+'/v1.ru/'+ url)

**except** Exception **as** e:

#Выводит ошибку

print(e)

print('EXCEPT {} \n'.format(c,e))

**continue**

url = '<https://v1.ru/>'

base\_url = '<https://v1.ru/text/>?'

page\_part = 'page='

total\_pages = 1

**for** i **in** range(1, int(total\_pages)+1):

url\_gen = base\_url + page\_part + str(i)

html = get\_html(url\_gen, userparser)

get\_page\_data(html)

# Программный код модуля 2

import os

import subprocess

import re

import xml.etree.ElementTree as ElementTree

from pymongo import MongoClient

client = MongoClient()

db = client.parserCLV1ru

parser = db.parser

tomita = db.tomita

class TomitaParser(object):

def \_\_init\_\_(self, executable, config, debug=True, validate=False):

self.debug\_mode = debug

self.debug("Init Tomita Parser...")

self.executable = os.path.expanduser(executable)

if not os.path.exists(self.executable):

raise Exception("Tomita executable not found at: %s" % self.executable)

self.debug("Executable: %s" % self.executable)

self.config = os.path.expanduser(config) if not os.path.exists(self.config):

raise Exception("Config file not found at: %s" % self.config) self.debug("Config: %s" % self.config)

self.path = self.config[:self.config.rfind("/")] self.debug("Path: %s" % self.path)

self.debug("ZBS!")

if validate: self.validate\_config()

def validate\_config(self): is\_xml = False

with open(self.config, "r") as f: for line in f.readlines():

line = line.strip()