**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

Направление: 09.03.02 Информационные системы и технологии

Профиль: Информационные системы в образовании

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**Поисковый веб-сервис для образовательного учебного курса**

Студент 4 курса

Группы 09-962

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (С.А. Морозов)

Научный руководитель:

канд. техн. наук, доцент

«\_\_\_»\_\_\_\_\_\_\_\_\_ 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (О.А. Невзорова)

Заведующий кафедрой

канд. физ.-мат. наук, доцент

«\_\_\_»\_\_\_\_\_\_\_\_\_ 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Ф.М. Гафаров)

Казань – 2023

Оглавление

[**Введение** 3](#_Toc138440396)

[**ГЛАВА 1. ТЕОРЕТИКО-АНАЛИТИЧЕСКАЯ ЧАСТЬ** 4](#_Toc138440397)

[**1.1 Используемые ресурсы и технологии** 4](#_Toc138440398)

[**1.1.1 Язык Python и библиотеки** 4](#_Toc138440399)

[**1.1.2.** **Язык разметки HTML** 17](#_Toc138440400)

[**1.1.3.** **База данных SQLite** 18](#_Toc138440401)

[**1.1.4.** **Электронный курс ЯКласс** 18](#_Toc138440402)

[**1.1.5.** **Распознавание именованных сущностей** 19](#_Toc138440403)

[**Глава 2. Основные модели и алгоритмы** 20](#_Toc138440404)

[**2.1. Подготовка данных для создания поискового индекса** 20](#_Toc138440405)

[**2.1.1 Структура данных на сайте ЯКласс** 20](#_Toc138440406)

[**2.1.2. Подготовка данных на основе учебника геометрии для 7-9 класса** 33](#_Toc138440407)

[**2.2 Базовые алгоритмы поискового сервиса** 34](#_Toc138440408)

[**2.2.1. Архитектура и основные функции поискового сервиса** 34](#_Toc138440409)

[**Заключение** 40](#_Toc138440410)

[**Список литературы** 40](#_Toc138440411)

[**Приложение** 41](#_Toc138440412)

# **Введение**

**Актуальность**. Данная работа посвящена созданию поискового веб-сервиса для образовательного учебного курса по геометрии для школьников средней школы. С развитием компьютерных технологий открываются новые возможности для модернизации процесса обучения, в том числе для разработки обучающих систем.

В последние годы актуально внедрение семантических технологий [1] в образовательный процесс. В основе таких технологий лежат онтологии.

В Казанском федеральном университете разработана новая образовательная математическая онтология OntoMathEdu [2]. Онтология OntoMathEd может быть использована в качестве исходного ресурса для обучающих систем для создания автоматических тестов и вопросов для тестирования уровня знаний учащихся, в качестве справочной базы математических данных, полученных из различных источников, а также в качестве лингвистического ресурса для автоматической обработки математических документов [3].

В составе обучающих систем важно иметь поисковый модуль, который выполняет роль справочной системы, позволяющей быстро найти нужный термин и его определение.

При построении поискового модуля необходимо решить задачу поиска и классификации определений терминов. Данная задача относится к классу задач распознавания именованных сущностей (NER, Named Entities Recognition) [4, 5].

**Целью** работы является создание поискового модуля для поиска определений в учебном курсе планиметрии, разрабатываемом в Казанском федеральном университете. При этом ставится задача извлечения определений по планиметрии из открытых учебных ресурсов для 7-9 классов. Также поставлена задача подготовки базы данных терминов и их определений для пополнения образовательной математической онтологии OntoMathEdu .

Объектом исследования являются определения и синтаксические правила для выделения терминов из предложений с определениями в области планиметрии. Методами исследования являются методы NLP, выполняющие морфологический, синтаксический и семантический анализ предложений на русском языке.

**Структура работы**: Работа состоит введения и двух глав. В первой главе даны теоретико-аналитические описания. В подпунктах этой главы последовательно описаны используемые ресурсы и технологии: язык Python, библиотеки *bs4, os, re, json, nltk, pymorphy3, import\_ipynb, tkinter, ast, xlsxwriter, sqlite3, requests, time*; описан язык разметки *HTML*, *XML,* база данных *SQLite.* Также описан электронный курс по геометрии *ЯКласс[6].* В последнем подпункте описана задача распознавания именованных сущностей. Во второй главе рассматриваются основные модели и алгоритмы и их реализация в поисковом модуле.

В заключение представлены результаты работы и даны перспективы дальнейших исследований.

# **ГЛАВА 1. ТЕОРЕТИКО-АНАЛИТИЧЕСКАЯ ЧАСТЬ**

## **1.1 Используемые ресурсы и технологии**

В работе используются следующие технологии: язык Python

и его библиотеки, язык разметки HTML, встроенная в Python база данных SQLite. В качестве ресурсов используется сайт ЯКласс [6] и учебник по геометрии [7].

### **1.1.1 Язык Python и библиотеки**

Библиотеки Python представляют собой наборы готовых классов, методов и функций, которые могут быть использованы для решения поставленных задач или при реализации определенных функций. Они помогают программисту упростить и ускорить процесс разработки, уменьшить сложность разработки, количество кода и ошибок в коде, увеличить читаемость, повысить поддерживаемость кода и в целом улучшить качество кода.

В выпускной квалификационной работе применялись следующие библиотеки функций: *bs4, os, re, json, nltk, pymorphy3, import\_ipynb, tkinter, ast, xlsxwriter, sqlite3, requests, time.* Ниже приведено описание этих библиотек и примеры их использования.

**Библиотека bs4 [8]**

Библиотека *bs4* имеет полное название *Beautiful Soup*. [*Beautiful Soup*](http://www.crummy.com/software/BeautifulSoup/) - это библиотека Python, позволяющая парсить данные из HTML- и XML-файлов и получать на выходе структурированные данные. Она представляет удобный интерфейс для извлечения любой информации, заключенной в HTML- или XML-тэги, а также для изменения этих файлов. С помощью данной библиотеки можно находить тэги с заданным именем и/или атрибутами.

Приведем пример использования библиотеки bs4. Данный код будет доставать все ссылки на темы уроков сайта ЯКласс (уроки на сайте группируются по темам и чтобы попасть на страницу урока надо перейти сначала на веб-страницу темы), исключая те ссылки, которые обеспечивают навигацию по сайту (боковые ссылки). На рисунке 1 представлен сайт.

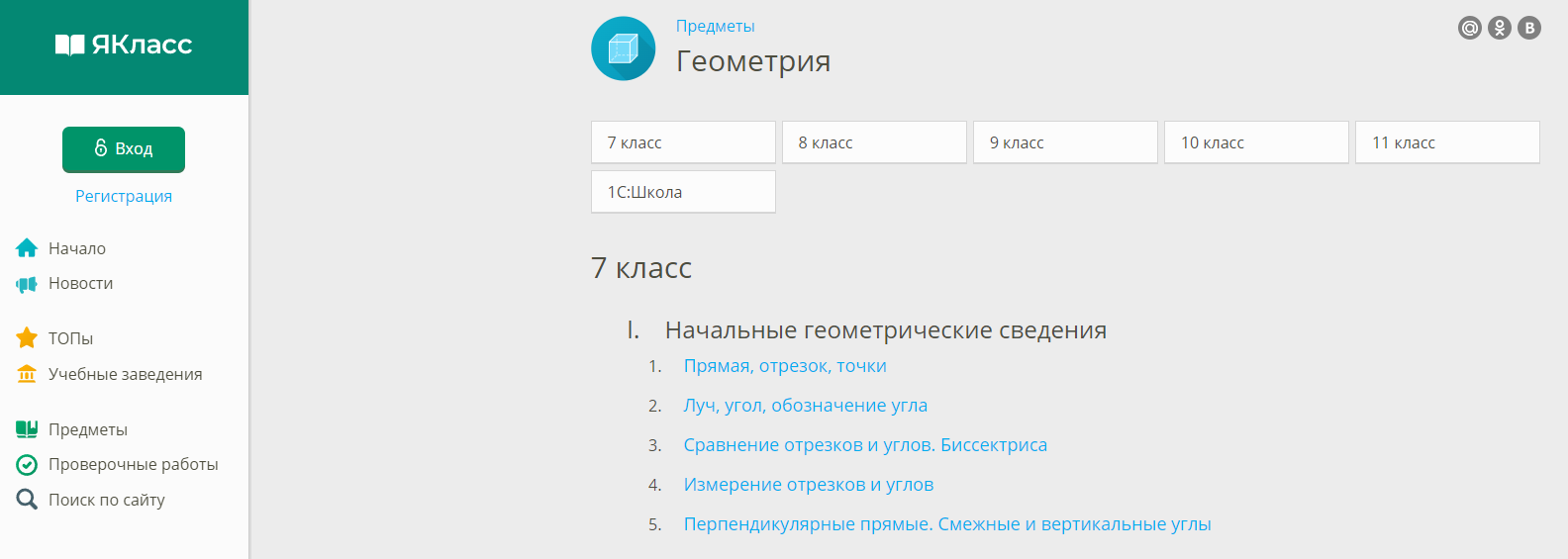


Рисунок 1 – Веб-страница сайта ЯКласс с ссылками на темы уроков по геометрии

Откроем инструменты разработчика (кнопка F12 для браузера Yandex).

HTML-разметку вышеуказанной веб-страницы приведем на рисунке 2.

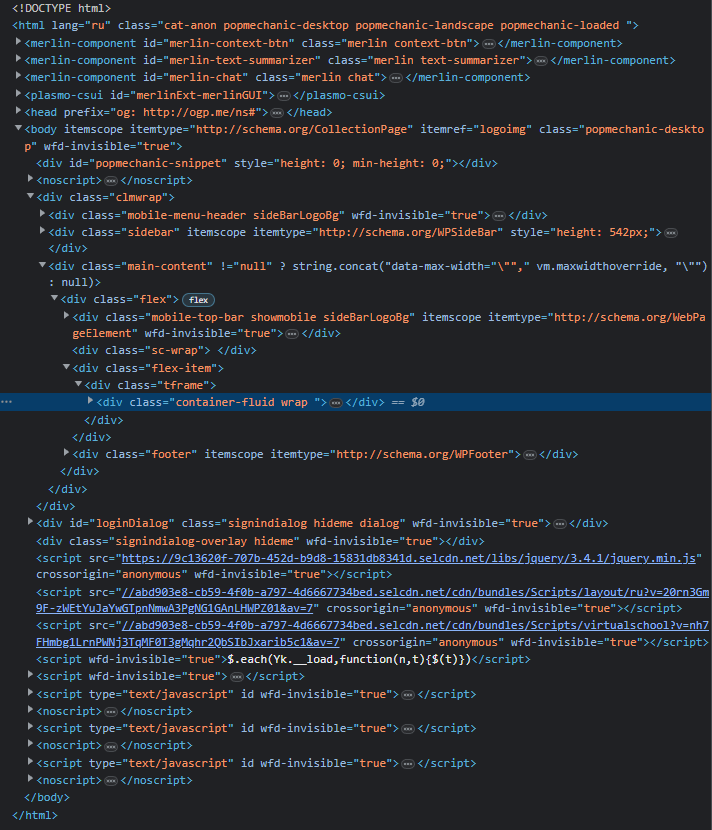


Рисунок 2 – HTML-разметка веб-страницы на Википедии по NER

Из всей разметки нас интересует только та часть, которая соответствует ссылкам на темы уроков на веб-странице. Опытным путем, используя инспектор (функция в инструментах разработчика, которая для каждого блока контента на сайте показывает какая HTML-разметка описывает его), определим, что эта часть задается тэгом *div* с атрибутом *class* равным ‘*container-fluid wrap’* (строка с нужным тэгом выделена синим цветом на рисунке 2).

Уточнив расположение ссылок на темы получаем, что все ссылки на темы содержатся в тэгах *h4*, внутри которых есть тэг *a* с атрибутом *href*, значение которого задает ссылку на тему урока, а содержимое тэга *a* – это название темы урока.

Код на рисунке 3 выделяет эти ссылки и названия и заносит их в специальную переменную *all\_lessons\_dict* – словарь с ключами-названиями и значениями ключей – ссылками, а потом сохраняет результат в файл с названием ‘*all\_lessons\_dict.json’*



Рисунок 3 – Код для извлечения ссылок на темы уроков из сайта ЯКласс

Опишем код на рисунке 3 подробнее. В первых трех строках мы достали содержимое HTML-страницы ‘[*https://www.yaklass.ru/p/geometria*](https://www.yaklass.ru/p/geometria)*’ и* записали в строковую переменную *src*. Далее, чтобы применять методы из библиотеки функций Beautiful Soup, мы должны были получить объект из этой библиотеки. Мы сделали это с помощью четвертой строки в коде. Мы используем конструктор класса *bs4.BeautifulSoup,* передавая в него строку *src* и значение *‘html.parser’*, которое означает, что перед нами html-разметка, результат записали в переменную soup.

Далее рассмотрим строку 5. Метод *find\_all*, примененный к переменной *soup* ищет все тэги в *soup*, которые удовлетворяют условиям, которые передаются в параметры функции. Если параметр строковый и он один, то вышеуказанный метод ищет все тэги с именем равный заданному параметру.

В переменную *h4* записались все тэги с именем *h4*, тип переменной - список.

Далее на 6 строке мы создали пустой словарь с именем *all\_lessons\_dict* для записи туда ссылок на темы по геометрии.

На 7 строке начинается цикл по списку *h4*.

В 8 строке достается текстовое содержимое тэга *a*, который содержится в тэге *h4*.

В 9 строке у тэга *a* достается значение атрибута *href*, добавляется справа к ссылке на главную страницу веб-сайта ЯКласс, потому что в атрибуте указаны относительные ссылки на темы по геометрии.

В 10 строке в словарь *all\_lessons\_dict* записывается пара ‘ключ-значение’: название темы и ссылка на тему.

В 12 строке мы создали json-файл ‘*all\_lessons\_dict.json’*.

А в последней 13 строке записали в этот файл словарь *all\_lessons\_dict*.

Результат выполнения кода представлен на рисунке 4.

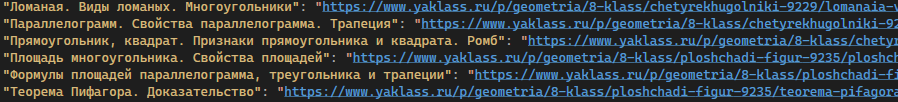


Рисунок 4 – Фрагмент файла ‘*all\_lessons\_dict.json’*

**Библиотека os**

Библиотека *os* предназначена для работы с папками и файлами на ЭВМ. Она содержится в служебных модулях языка Python, поэтому не требует установки. В настоящей ВКР используются следующие четыре метода библиотеки *os*:

* *os.chdir(<path>)*

Этот метод меняет рабочую директорию на ту, которая указана в пути *path*. Причем, строка *path* может иметь несколько значений: 1) полный путь к директории, например, *‘C:\Users\Sergey’* 2) относительный путь к директории, например, *‘YaKlass’* (если в системе нет соответсвующих папок, то метод вызовет исключение) 3) можно подняться на одну директорию выше с помощью передачи *path* *= ‘..’*

* *os.listdir()*

Этот метод возвращает список имен всех файлов, которые находятся в рабочей директории. Например, для получения всех имен файлов с наличием слова ‘simple’ в имени из рабочей директории следует выполнить код:

*simple\_paths = [path for path in os.listdir() if 'simple' in path]*

Все имена файлов, удовлетворяющие вышеуказанному условию, записываются в список *simple\_paths*

* *os.path.exists(<path>)*

Этот метод возвращает *True*, если существует путь path из рабочей директории. Иначе возвращает False.

* os.mkdirs(<folder\_name>)

Этот метод создает папку с именем folder\_name в рабочей директории.

Пример кода для последних первого, третьего и четвертого методов на рисунке 5:

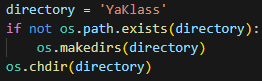


Рисунок 5 – Код, создающий директорию *‘YaKlass’*, в случае, если ее нет, и изменяющий рабочую директорию на *‘YaKlass’*

**Библиотека re**

Библиотека *re* предназначена для работы с регулярными выражениями на Python. Она содержится в служебных модулях языка Python, поэтому не требует установки. В настоящей ВКР используются следующие два метода библиотеки *re*:

* re.sub(<pattern>, <repl>, <string>)

Этот метод принимает три строки: pattern, repl, string и возращает строку string, в которой каждое вхождения шаблона pattern заменено на repl. Вместо pattern или repl может стоять регулярное выражение. Пример:

updated\_text = re.sub(“\s+”, ‘ ’, text)

Разберем pattern. *\s* - Соответствует пробельным символам Юникода (которые включают [*\t\n*])

Разберем repl. repl представляет собой строку, состоящую только из одного пробела.

Разберем re.sub(“\s+”, ‘ ’, text). Эта строчка кода заменяет каждый пробельный символ на пробел. Тем самым из переменной text удаляются все символы табуляции и переноса на новую строку, а вместо них ставятся символы пробела.

* re.finditer(<pattern>, <string>)

Этот метод возвращает итератор по всем неперекрывающимся совпадениям для шаблона регулярного выражения pattern в [строке](https://docs-python.ru/tutorial/osnovnye-vstroennye-tipy-python/tip-dannyh-str-tekstovye-stroki/) string.

Пример:

sents\_in\_brackets = [match[1] for match in re.finditer(r"\(([^()]+)\)", sent)]

Разберем pattern. Он начинается с ‘\(’. Так как скобки в регулярных зарезервированы для обозначения начала и конца групп, которые нужны, чтобы выделять части регулярного выражения, то чтобы сопоставить литералы ‘(’ и ‘)’ мы используем экранирование символов: ‘\(’ и ‘\)’.

Далее идут скобки ‘([^()]+)’. Внешние круглые скобки обозначают группу. Внутренние квадратные скобки нужны для обозначения того, какие символы входят или не входят в паттерн. Внутри квадратных скобок есть символ ‘^’, это означает, что в квадратных скобках указаны символы, которые НЕ присутствуют в паттерне. Этим символом может быть, например, буква ‘А’. После квадратных скобок идет символ ‘+’. Он относится к квадратным скобкам и означает. Что символ соответствующий содержимому квадратных скобок может использоваться от одного и более раз.

Оканчивается регулярное выражение на ‘\)’.

Подставив в переменную *sent* значение:

*‘Поэтому для построения высоты необходимо выполнить следующие действия: 1) провести прямую, содержащую одну из сторон треугольника ( в случае, если проводится высота из вершины острого угла в тупоугольном треугольнике ); 2) из вершины, лежащей напротив проведённой прямой, опустить перпендикуляр к ней ( перпендикуляр — это отрезок, проведённый из точки к прямой, составляющей с ней угол m ) — это и будет высота.’*

В переменную *sents\_in\_brackets* будет записан список:

[*' в случае, если проводится высота из вершины острого угла в тупоугольном треугольнике* ', ' *перпендикуляр — это отрезок, проведённый из точки к прямой, составляющей с ней угол m* '].

**Библиотека json**

Библиотека предназначена для удобной работы с json-файлами. Json-файлы могут содержать данные любого типа, включая числа, строки, объекты, массивы и т.д. В настоящей ВКР в json-файлы будут представлены в виде словарей Python, либо вложенных друг в друга словарей Python.

В настоящей ВКР используются следующие два метода библиотеки *json*:

* json.dump(<data>, <file>, <indent>, <ensure\_ascii>)

Этот метод принимает четыре параметра.

Пример:

with open('data\_file.json', 'w', encoding='utf-8') as file:

json.dump(data, file, indent = 4, ensure\_ascii = False)

На первой строке открывается файл 'data\_file.json' на запись.

На второй строке указывается, что нужно передать в этот файл данные, заключенные в переменной data, сохранить отступы внутри json-файла в 4 пробела (indent = 4), ensure\_ascii = False означает, что не ASCII символы будут записаны без экранирования.

* json.load(<file>)

Этот метод принимает файл, записанный в формате json и возвращает соответствующий ему объект Python.

Пример:

with open('data\_file.json', 'r', encoding='utf-8') as file:

data = json.load(file)

Если выше мы записали в файл 'data\_file.json' словарь, то при извлечении данных из json-файла получим словарь в переменную data.

**Библиотека nltk**

Библиотека nltk предназначена для создания программ для обработке естественного языка, NLP (Natural Language Processing).

В настоящей ВКР используются следующие два метода библиотеки *nltk*, а также список стоп-слов ‘stopwords’:

* sent\_tokenize(<text>)

Этот метод разделяет текст на предложения и выдает список предложений. Он так же различает сокращения в русском языке, такие как ‘т.е.’ или ‘и т.д.’ и не выделяет ошибочно их в отдельные предложения.

* word\_tokenize(<sent>)

Этот метод разделяет строковую переменную sent и токенизирует (разбивает) ее на слова и пунктуационные символы.

Пример:

sent = 'Грани — это многоугольники, из которых состоит многогранник.'

words = word\_tokenize(sent)

# В переменную words сохраняется список: ['Грани', '—', 'это', 'многоугольники', ',', 'из', 'которых', 'состоит', 'многогранник', '.']

**Библиотека pymorphy3**

В настоящей дипломной работе для решения задач NLP, помимо библиотеки *nltk*, используется библиотека *pymorphy3* (а не *pymorphy2*, потому что последняя не поддерживается Python 3 на котором написан код дипломной работы). *Pymorphy3* является морфологическим анализатором и повторяет все объекты, методы и функции из pymorphy2.

Для морфологического анализа в *pymorphy3* существует класс *MorphAnalyzer*. Экземпляры этого класса загружают в память словари для указанного языка (по умолчанию – это русский язык). Для морфологического анализа отдельного слова у этого класса есть метод *parse*, который возвращает один объект класса *pymorphy3.analyzer.Parse*, либо список объектов этого же класса. Одно и то же слово может иметь несколько морфологических разборов, потому что понять какой из них верный, рассматривая каждое слово изолированно от других слов в предложении, нельзя. Пример: слово ‘прямая’ может быть как прилагательным, так и существительным.

Пример:

morph = pymorphy3.MorphAnalyzer()

parsed = morph.parse('прямую')

В переменной parsed записался список из двух объектов класса *pymorphy3.analyzer.Parse*:

[Parse(word='прямую', tag=OpencorporaTag('ADJF,Qual femn,sing,accs'), normal\_form='прямой', score=0.7, methods\_stack=((DictionaryAnalyzer(), 'прямую', 1215, 10),)),

Parse(word='прямую', tag=OpencorporaTag('NOUN,inan,femn sing,accs'), normal\_form='прямая', score=0.3, methods\_stack=((DictionaryAnalyzer(), 'прямую', 117, 3),))].

Выделение грамматических характеристик слова выполняется обращениям к атрибутам класса *pymorphy3.analyzer.Parse*:

Обращение к атрибутам вышеуказанного класса показано на примере второго элемента списка *parsed\_2.*

parsed\_2 = parsed[1]

* parsed\_2.tag.POS

Возвращается часть речи, соответствующая слову ‘прямую’. В данном случае – ‘NOUN’.

* parsed\_2.tag.case

Возвращается падеж слова, если данное слово имеет падеж. В данном случае – ‘accs’ (винительный падеж).

* parsed\_2.normal\_form

Возвращается нормальную форму слова. В данном случае – ‘прямая’.

* parsed\_2.inflect({‘nomn’})

Возвращает слово, ставя его в нужный падеж и/или число.

**Библиотека import\_ipynb**

Библиотека предназначена для импорта функций, классов и переменных между блокнотами с кодом с расширением .*ipynb*.

Пример работы библиотеки:

Блокнот ‘A.ipynb’ содержит функцию, которую нужно импортировать в блокнот ‘B.ipynb’. Для этого эти два блокнота должны лежать в одной директории. В коде блокнота ‘B’ нужно импортировать библиотеку import\_ipynb:

import import\_ipynb.

Далее чтобы импортировать функцию a\_function используется код:

from A import a\_function.

Особенности работы библиотеки import\_ipynb:

При импорте функций из файла A.ipynb в файл B.ipynb выполняется весь код в первом файле, который не заключен в условие:

if \_\_name\_\_ == "\_\_main\_\_":

**Библиотека tkinter**

Эта библиотека предназначена для создания программ с графическим интерфейсом. Tkinter встроен в стандартную библиотеку языка Python, поэтому не требует установки.

Ниже представлен код для создания пустого окна:

import tkinter as tk

class Application(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

###

app = Application()

app.title('Название приложения')

app.geometry('100x200')

app.mainloop()

На первой строчке импортируется библиотека tkinter.

На второй создается класс, наследуемый от класса tk.Tk.

На третьей и четвертой строчке наследуются поля и функции из класса tk.Tk.

На пятой строчке стоят символы ‘#’. Вместо них пишутся виджеты библиотеки и функции, которые выполняются при взаимодействии с виджетами.

На шестой строчке создается экземпляр класса Application.

На седьмой строчке к созданному экземпляру класса применяется метод title для того, чтобы задать название программе.

На восьмой строчке задаются размеры окна приложения.

На последней строчке приложение запускается.

Опишем код, который создает виджеты:

* Message.

self.message = tk.Message(self, text =’Some text’, width = 500)

self.message.pack()

Виджет tk.Message используется для отображения текста, причем текст переносится на следующую строку при превышении заданной ширины (параметр width)

На первой строчке передается в конструктор класса tk.Message последовательно: родительский объект (self означает, что родительским объектом является объект класса Application, а виджет лежит внутри окошка графического приложения)

Вторая строчка означает, что для позиционирования виджета self.message используется метод pack().

Для изменения содержимого виджета существует метод config:

self.message.config(text = "NEW TEXT")

* Text.

self.text = tk.Text(self, width = 100, height = 3)

self.text.pack()

Виджет tk.Text используется для используются для ввода текста. tk.Text может содержать несколько строк текста.

На первой строчке передается в конструктор класса tk.Text последовательно: родительский объект, ширина и высота блока для ввода текста.

Можно получать текст из данного виджета с помощью метода get:

text\_from\_widget = self.text.get("1.0", tk.END)

Можно очистить текст в данном виджете при помощи метода delete:

text\_from\_widget = self.delete("1.0", tk.END)

* Button.

self.button = tk.Button(self, text = 'press button', width = 5, height = 2, command = self.button\_function)

self.button.pack(side = tk.RIGHT)

def button\_function(self):

print(‘Button Pressed’)

### **1.1.2. Язык разметки HTML**

Язык разметки HTML служит для  разметки документов для просмотра [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) в [браузере](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80). Поэтому любой веб-сайт написан на html-разметке. Язык HTML содержит набор парных тэгов (открывающихся и закрывающихся) и их атрибутов.

Пример тэга body:

<body>Некоторый текст внутри тэга</body>

Пример тэга с ссылкой:

<a href = “my-cite.ru”>Мой сайт</a>

В данном случае содержимым тэга ‘a’, отвечающим за создание ссылок, будет ‘Мой сайт’, а ссылка будет ‘my-cite.ru’.

### **1.1.3. База данных SQLite**

База данных SQLite поставляется вместе с языком Python. Для создания таблицы базы данных с автоинкрементированным полем id и двумя полями student\_name и age, причем, комбинация последних двух полей является уникальной в таблице, можно использовать запрос к базе данных:

CREATE TABLE IF NOT EXISTS new\_table(

id INTEGER PRIMARY KEY AUTOINCREMENT,

student\_name TEXT,

age INTEGER,

UNIQUE (student\_name, age));

### **1.1.4. Электронный курс ЯКласс**

Электронный курс ЯКласс – это цифровой образовательный ресурс для школ. Он состоит из частей для учителей, школьников и родителей (рисунок 6).

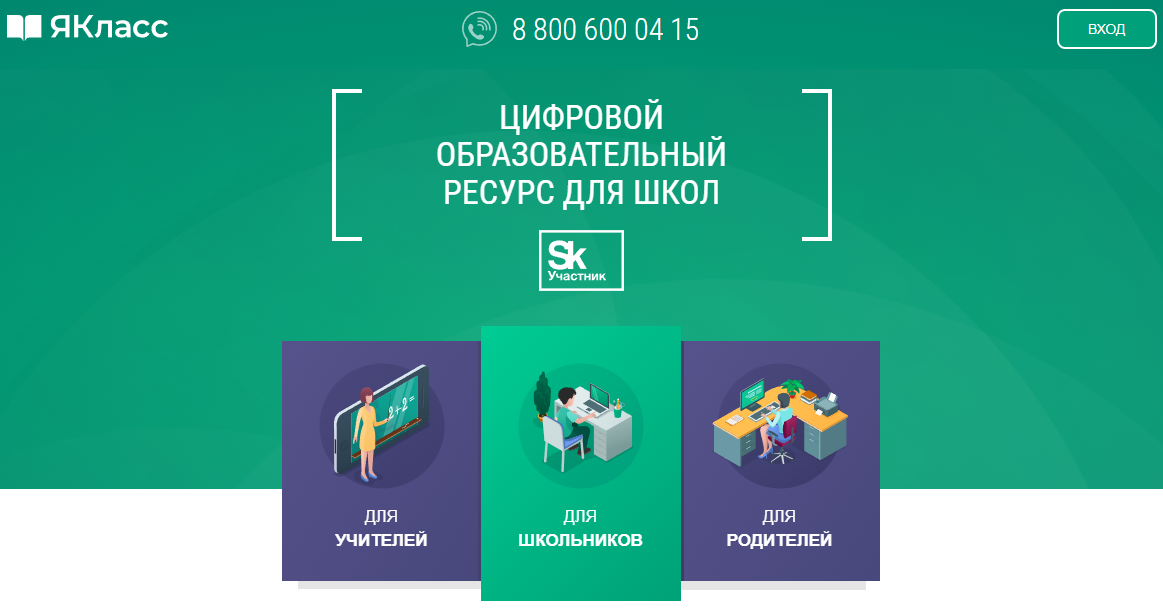
****

Рисунок 6 – Электронный образовательный ресурс ЯКласс

В части для школьников представлено множество теоретического и практического материала по предметам школьной программы, в том числе есть материалы по геометрии. Материал разделен на теоретическую и практическую части (рисунок 7)

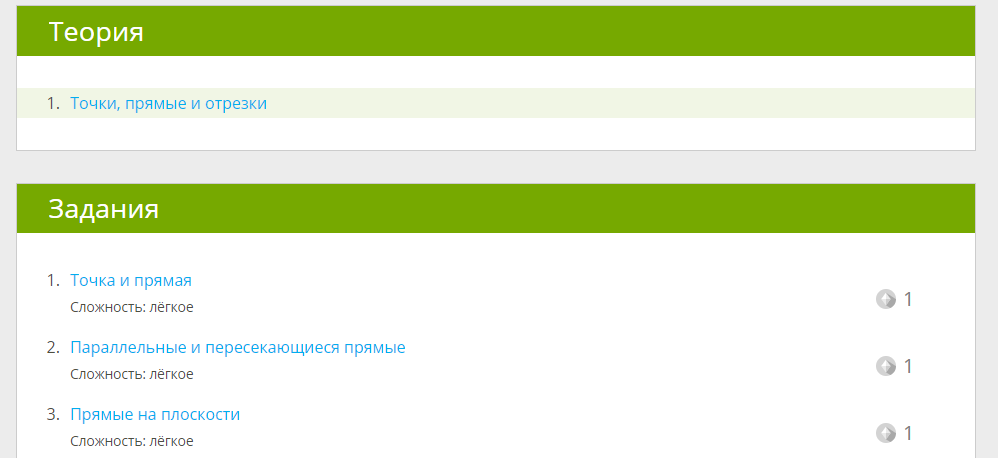


Рисунок 7 – Теоретическая и практическая части курса

### **1.1.5. Распознавание именованных сущностей**

Распознавание именованных сущностей – это подзадача обработки естественного языка (NLP, natural language processing), в которой целью является извлечение и классификация именованных объектов. Например, это могут быть адреса, имена, либо более сложные по структуре объекты, как определения или теоремы.

Сложность распознавания именованных сущностей, например, математических определений, состоит в том, что термины предложений-определений встречаются в сильно изменяющихся и потенциально сложных синтаксических структурах.

В статье [3] рассмотрены существующие способы решения задач NER:

1. Метод основанный на правилах (Эти методы основаны на лексико-синтаксических паттернах или особенностях, которые создаются вручную или полуавтоматически усваиваются) [2]. Шаблоны представляют собой либо очень простые последовательности слов (например, «относится к», «определяется как», «является»), либо более сложные последовательности слов, частей речи и фрагментов)
2. Второй способ – это инженерный подход. В этом подходе используются модели статистического машинного обучения. Однако этот подход не может быть эффективно адаптирован к новым доменам, так как разработанные функции могут быть недоступны или менее эффективны в новых доменах.
3. Третье направление связано с подходом глубокого обучения, который недавно показал свою способность эффективно использовать вложения слов через несколько слоев нейронных сетей.

В ВКР использован первый подход из-за простоты и высокой точности этого метода.

Полученная база данных, состоящая из записей с полями «термин»-«определение»-«источник определения» может быть использована для пополнения онтологии OntoMathEdu.

# **Глава 2. Основные модели и алгоритмы**

## **2.1. Подготовка данных для создания поискового индекса**

### **2.1.1 Структура данных на сайте ЯКласс**

Для извлечения дефиниционных предложений сначала выполняется парсинг данных онлайн-уроков по геометрии для 7-9 классов с сайта Якласс. Проанализируем структуру сайта. Сначала мы заходим на страницу 'https://www.yaklass.ru/p/geometria', на которой находятся все темы по геометрии 7-11 класса (рис.8).

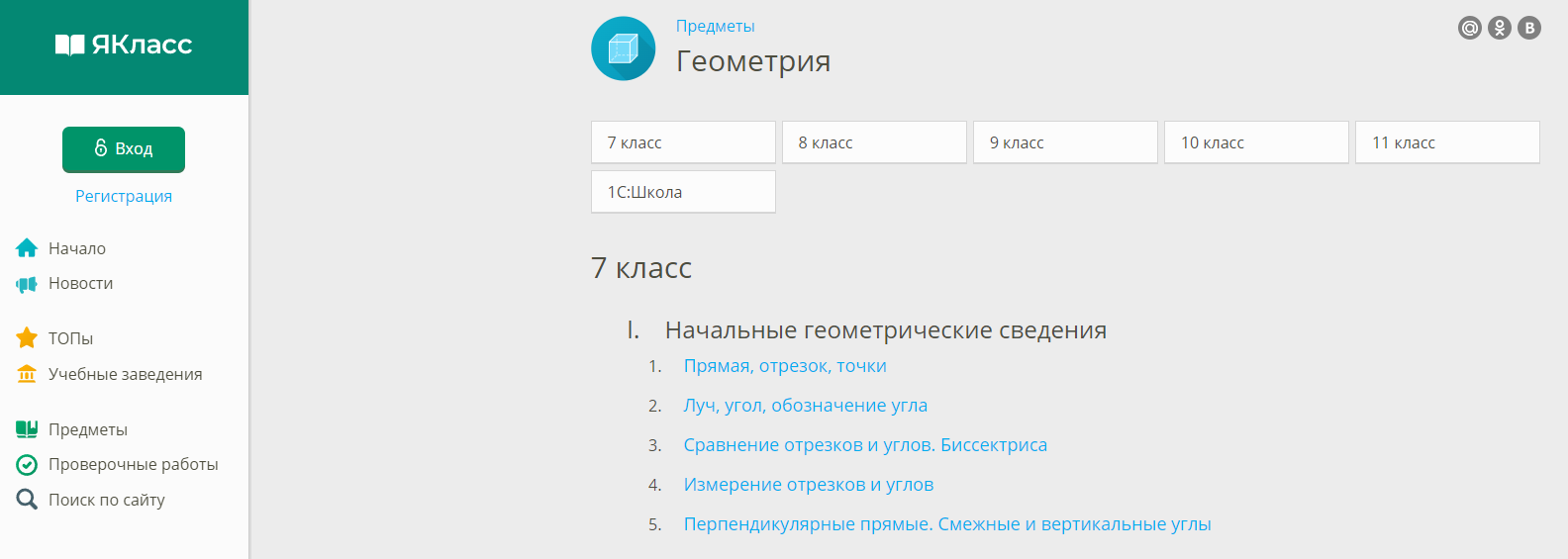


Рисунок 8 - Главная страница с темами по геометрии

Видим, что на сайте много ссылок, в том числе ненужных для решения задачи (ссылка на главную страницу, ссылка на новости и некоторые другие ссылки). Поэтому для анализа страницы сайта открываем «режим разработчика» и устанавливаем в каких html-тэгах находятся ссылки на темы по геометрии.

После анализа получаем, что все нужные нам ссылки находятся в тэгах <h4></h4>. К примеру, рассмотрим первую такую ссылку: <**h4**><**a** href="/p/geometria/7-klass/nachalnye-geometricheskie-svedeniia-14930/priamaia-otrezok-tochki-9703">Прямая, отрезок, точки</**a**></**h4**>.

С помощью библиотеки beautifulsoup4 попробуем достать все ссылки и их имена (код представлен на рисунке 9).



Рисунок 9 - Фрагмент кода для извлечения ссылок на темы

Немного прокомментируем код на рисунке 9. Сначала мы получаем текст запроса к сайту (переменная «src»). Далее получаем объект класса «bs4.BeautifulSoup» (переменная «soup»). Это нужно, чтобы применять методы библиотеки beautifulsoup4 для извлечения данных из html-страницы. Далее достаем все тэги <h4>, а в них достаем ссылку и содержимое в тэге <a>. Результат записываем в файл «all\_lessons\_dict.json». Часть содержимого этого json-файла выглядит так (рисунок 10):

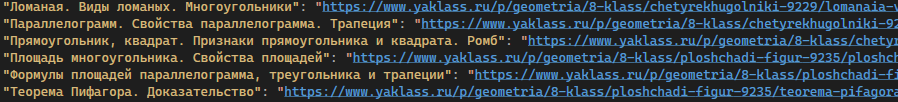


Рисунок 10 - Пример ссылок на темы

Переход к страницам с уроками по геометрии выполняется последовательным переходом с главной страницы, посвященной геометрии, на страницу темы урока, а потом со страницы темы на конкретную страницу урока. На рисунке 11 показана тема с ссылками на конкретные уроки.

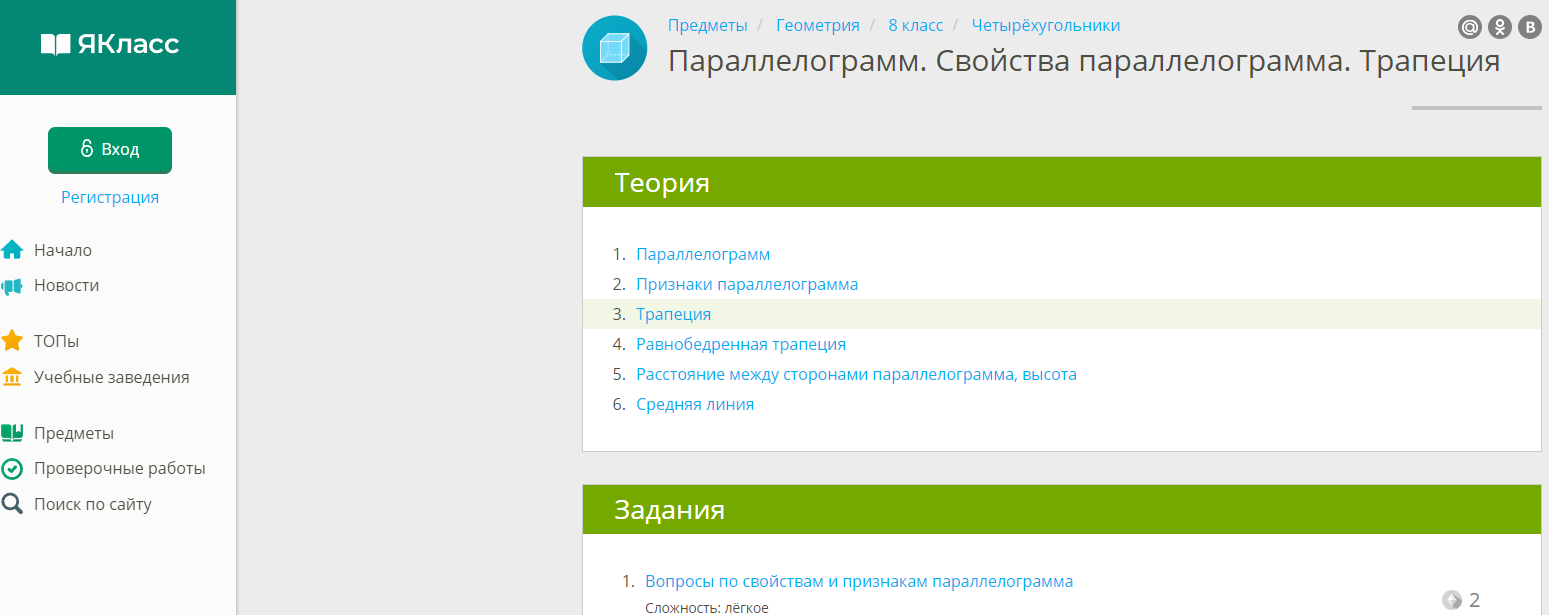


Рисунок 11. Веб-страница темы «Параллелограмм. Свойства параллелограмма. Трапеция»

В блоке с заголовком «Теория» имеется полный набор ссылок на уроке по геометрии. При переходе по ссылкам в этом блоке мы получаем веб-страницы с уроками.

Проанализируем с помощью инструментов разработчика эту веб-страницу. Блоку «Теория» соответствует тэг <section> с классом «TheoryBlock». В этом тэге содержатся тэги <a>, содержимое которых – название урока, а атрибут «href» является самой ссылкой на данный урок.

Код для извлечения из сайта ссылок на уроки показан на рисунке 12.

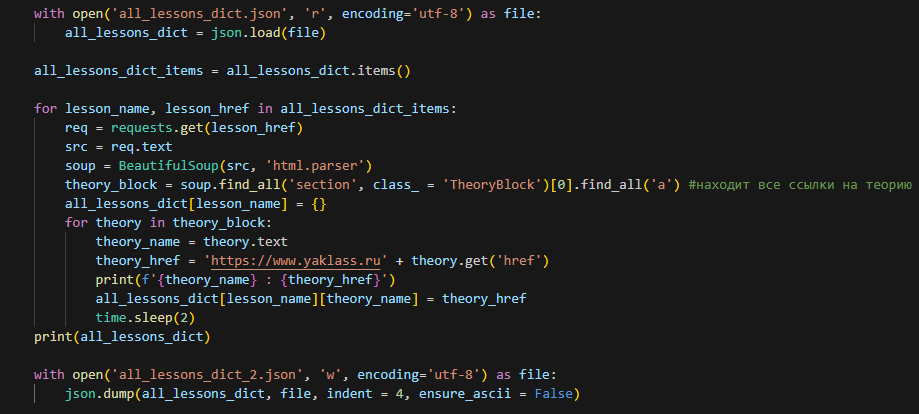


Рисунок 12 - Фрагмент кода для извлечения ссылок на уроки

Опишем процесс извлечения дефиниционных предложений, который включат пять этапов.

1. Сначала достаем в переменную «all\_lessons\_dict» содержимое файла «all\_lessons\_dict.json». Создаем новый словарь для записи туда ссылок на уроки. Потом проходимся по старому словарю. Для каждой ссылки на тему уроков находим содержимое блока «TheoryBlock», а именно ссылки на уроки (переменная «theory\_href»). Записываем ссылки на уроки и их названия как значение ключа с названием темы нового словаря. Результат заносится в файл «all\_lessons\_dict\_2.json».

На рисунке 13 показана часть полученного json файла.



Рисунок 13 – Файл с ссылками на веб-страницы с уроками

Таким образом строится словарь тем, включающий словари уроков, в котором есть все нужные нам ссылки на уроки.

1. Далее нам нужно сохранить html-страницы по указанным ссылкам.

Для этого создаем папку с именем «YaKlass». Проходимся по словарю словарей и «раскрываем» его (пусть имя темы будет «тема», а имя урока будет «урок», то название файла, в которого мы будем сохранять веб-страницу урока будет такое: «тема»\_\_«урок».html). А содержимое файла получаем с помощью функции «soup.find\_all(class\_ = ‘blockbody’» (потому что нас интересует только блок теория, которая задается тэгом с классом ‘blockbody’). Для наглядности приведем рисунок 14.

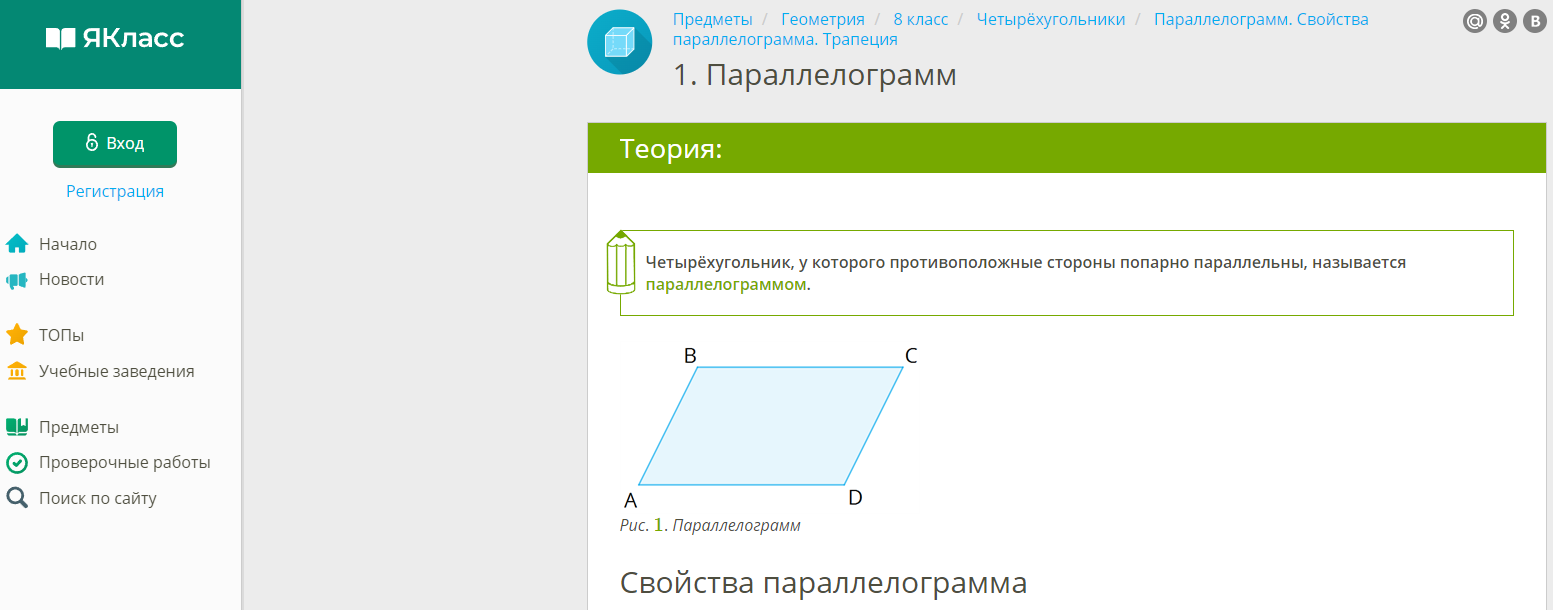


Рисунок 14 – Пример содержимого урока по геометрии

Код, который выплняет написанные выше действия, приведен на рисунке 15:



Рисунок 15 – Код для извлечения текста с теорией из веб-страницы урока

Строчка «time.sleep(4)» нужна для того, чтобы сайт не считал нас ботом, а считал обычным пользователем сайта, не использующим сторонние программы.

Примеры созданных файлов (рисунок 16):

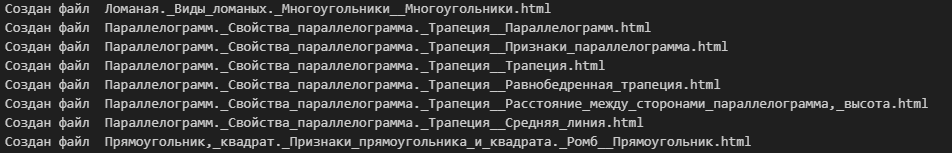


Рисунок 16 – Пример сохраненных файлов с теорией

В дальнейшем потребуется изменить имена html-файлов. В случае если длина имени файла превышает допустимую длину, мы сокращаем это имя файла, как например, «*Признаки\_параллельности\_двух\_прямых.\_Свойства\_параллельных\_прямых.\_Аксиома\_параллельных\_прямых\_\_Определение\_и\_доказательства\_признаков\_параллельности\_прямых\_в\_плоскости.html»*

на

«*Признаки\_параллельности\_двух\_прямых.\_Свойства\_параллельных\_прямых.\_Аксиома\_параллельных\_прямых\_\_Определение\_и\_доказательства\_признаков\_параллельности\_прямых.html»*.

1. Далее рассмотрим html-страницы уроков. Рассмотрим способы выделения дефиницонных предложений. Ниже представлен рисунок 17 с уроком «Перпендикулярные прямые»:

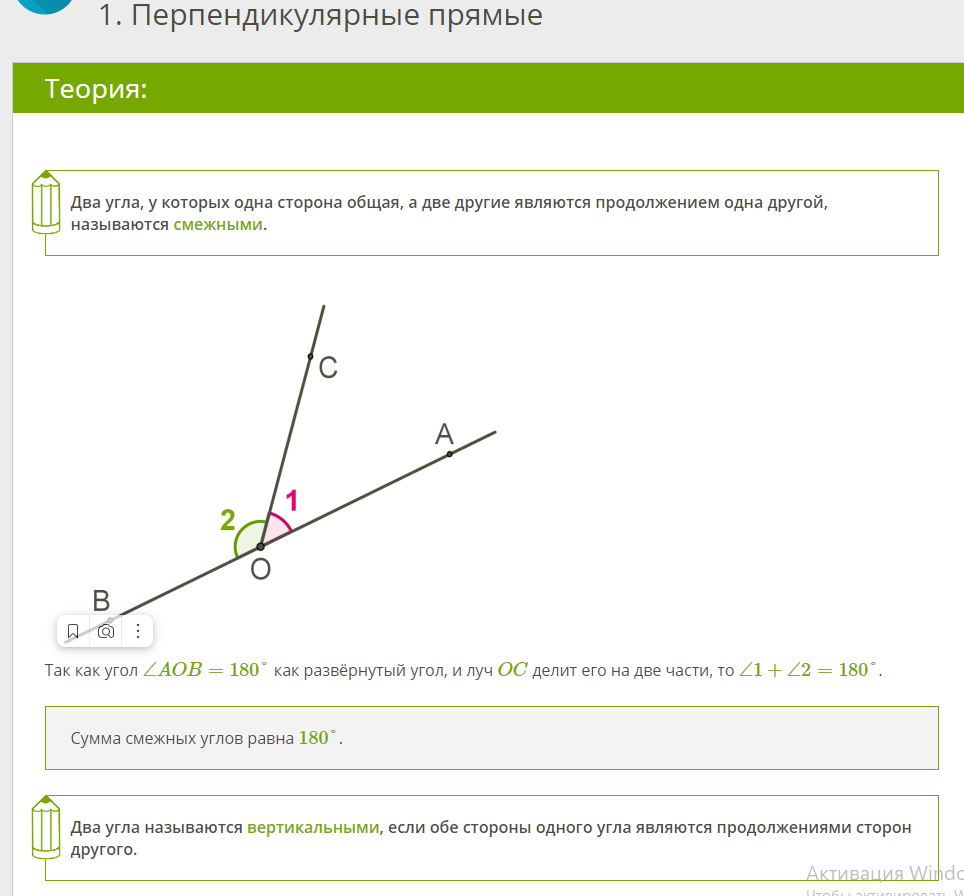


Рисунок 17 – Урок «Перпендикулярные прямые»

На веб-странице есть блоки, выделенные зеленой рамкой и иконкой карандаша. Это – определения, выделенные на самом сайте. Будем называть их рамочными определениями. В некоторых рамочных определениях есть слова, выделенные зеленым цветом. В некоторых – нет (см. рисунок 18).

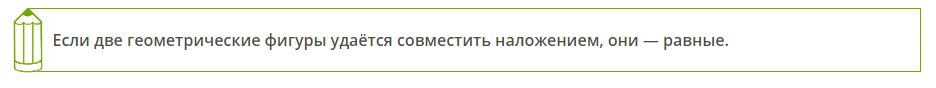


Рисунок 18 – Рамочное определение

Но мы не можем полагаться на выделенные зеленым цветом слова, потому что даже в случае, когда они есть в тексте определения, термин может состоять из нескольких слов, а зеленым выделено лишь одно слово. К примеру, на рисунке 17 имеем 2 дефиниционных предложения: в первом термин будет «смежные углы», при выделенном зеленым словом «смежными», а во втором термин будет «вертикальные углы», при выделенном зеленым словом «вертикальными». Поэтому нам надо придумать другой способ извлечения терминов из рамочных определений.

Заметим, что рамочные определения всегда задаются тэгом <div class="gxst-definition"></div>

Также есть определения с математическими формулами, что создает дополнительную сложность для их анализа программой или экспертом, а также дополнительную сложность при хранении и выдаче этих данных.

Так как мы в работе применяется как автоматическое выделение терминов из дефиниционных предложений, так и ручное или полуавтоматическое, то, для облегчения работе с математическими формулами применяется замена формулы на символы l или m в зависимости от контекста.

Изучив структуру таких формул на всем сайте разделим их на 2 типа: первый тип содержит тэги <math></math>, а его содержимое пишется на MathML (язык математической разметки), второй тип – это встраиваемые математические формулы вида \(формула\).

Приведем пример формулы первого типа:

1. Формула в формате MathML:

<math displaystyle="true" linebreakstyle="after" xmlns="http://www.w3.org/1998/Math/MathML">

<mrow>

<mi>

k

</mi>

<mo>

≠

</mo>

<mn>

0

</mn>

</mrow>

</math>

Отображается ученику как: .

1. Приведем пример формулы второго типа:

\(k &gt; 0\).

Отображается ученику как: .

1. Далее проведем дополнительную обработку для очистки служебной информации.

Сначала для каждого файла «урок1.html» создадим его упрощенную версию «урок1\_new.txt». Для этого мы удалим из исходного файла тэги <meta>, тэги идущие внизу всего текста урока (рисунок 19), удалим комментарии, поставим точки в конце некоторых предложений, например, после текста каждого заголовка (рисунок 20). Особенности оформления заголовков на странице вызывают некоторые сложности для автоматической обработки.



Рисунок 19 – Нижний колонтитул урока

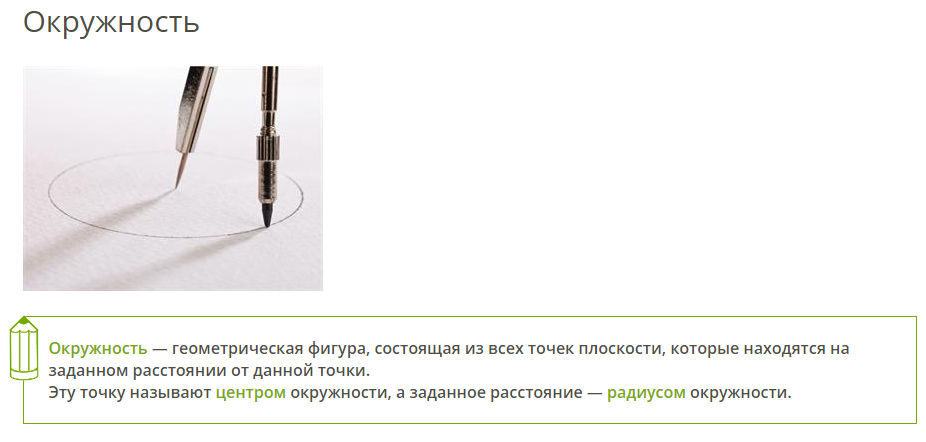


Рисунок 20 - Рамочное определение с тремя терминами

Последние действия нам нужны, чтобы при токенизации текста по предложениям (разбиение текста на предложения), не склеилось в одно предложение два предложения, как в примере: «Окружность» и «Окружность – геометрическая фигура, состоящая из всех точек плоскости, которые находятся на заданном расстоянии от данной точки».

Ниже на рисунке 21 приведем код для создания файлов с «суффиксом» «new».

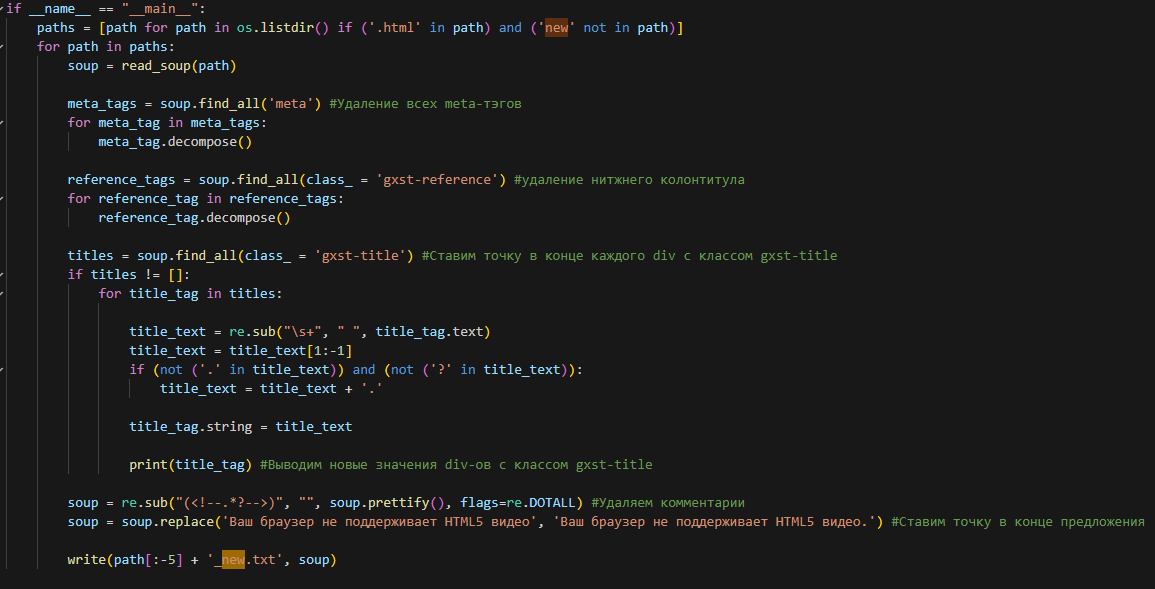


Рисунок 21 – Код создания очищенных от служебной информации файлов с теорией

Следующим этапом является создание отдельных двух файлов для хранения математических формул (один файл для формул первого типа, второй файл для формул второго типа), а также третьего файла, в котором все формулы будут заменены на символы (символ ‘m’ для формул первого типа, символ ‘l’ для формул второго типа).

Первый файл получается от исходного добавлением к имени файла (без той части, которая идет после точки) суффикса «\_math». Второй файл получается от исходного добавлением суффикса «\_slash\_math». Третий файл получается от исходного добавлением к имени файла «\_simple». На рисунке 22 показан код, который создает эти файлы.

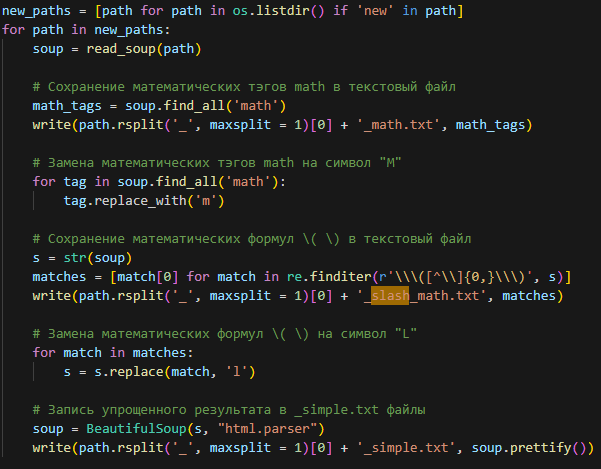


Рисунок 22 – Код для выделения из теории математических формул

1. Выделение рамочных определений.

Рассмотрим часть содержимого файла «Подобные\_треугольники.\_Пропорциональные\_отрезки\_\_Пропорциональные\_отрезки\_simple.txt» на рисунке 23:

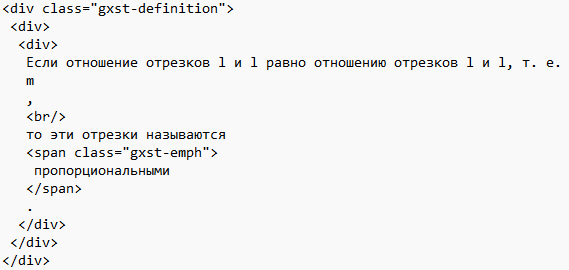


Рисунок 23 – Фрагмент html-страницы, определяющий блок с рамочными определениями

На рисунке 23 представлено рамочное определение, причем математические формулы заменены на буквы ‘l’ и ‘m’ соответственно. Это упрощает анализ определений.

Так же для извлечения текста определения из блока «div» с классом «gxst-definition» создана функция «text\_from\_soup» (рисунок 24).

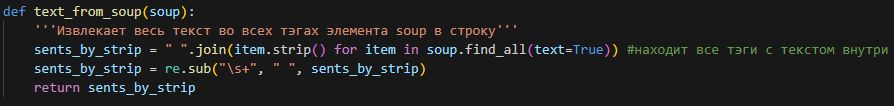


Рисунок 24 – Функция для извлечения текста из объекта *bs4.BeautifulSoup*

Применив эту функцию к фрагменту файла на рисунке 23, получим строку вида:

«Если отношение отрезков l и l равно отношению отрезков l и l, т. е. m ,

то эти отрезки называются пропорциональными .»

Далее извлекаем все рамочные определения. Для этого написан код, который находит все тэги <div class=”gxst-definition”>, а потом применяет к нему функцию «text\_from\_soup», а затем применяется к полученной строке функция **sent\_tokenize** из библиотеки **nltk** (рисунок 25).



Рисунок 25 – Код для извлечения рамочных определений

### **2.1.2. Подготовка данных на основе учебника геометрии для 7-9 класса**

В дополнение к учебным материалам сайта ЯКласс в данной ВКР использовался учебник “Геометрия” для 7-9 классов Атанасяна. Он был дан в распознанном системами распознавания изображений виде. Предстояла задача сформировать из текстового варианта этого учебника сформировать файл, содержащий xml-разметку, в которой были бы выделены термины и определения.

Первоначально в данном файле были размечены уроки тэгами paragraph и вложенными в них тэгами section. Далее для каждого тэга paragraph выделили дочерние тэги p\_id, p\_name, обозначающие номер и имя параграфа, соответствующие номеру и названию параграфа в учебнике.

Вручную, опираясь на учебник в формате pdf были выделены тэги definition с атрибутами name (в него записывается термин) для определений, тэги theorem с атрибутами name (в него записывается название теоремы) для теорем, дочерними тэгу theorem являются тэги t\_text (текст теоремы), proof (доказательство теоремы), иногда присутствует тэг comment (комментарий к теореме). Также в xml-разметку выделены картинки из учебника и задачи на построение. Оставшийся неразмеченный вышеуказанными тэгами текст оборачивается либо в тэги text, либо как задачи exercise.

Пример xml-разметки приведен на рисунке 26.

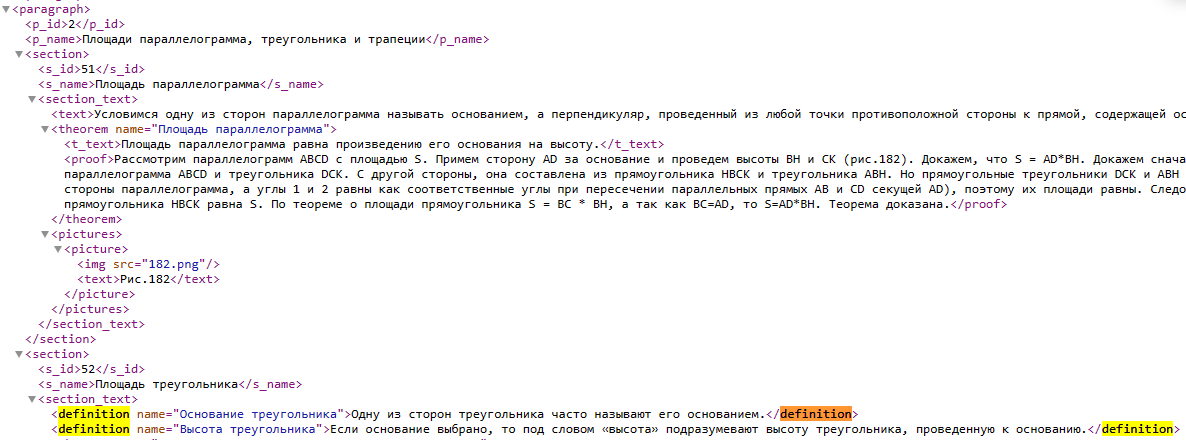


Рисунок 26 – Разметка учебника по геометрии Атанасяна

## **2.2 Базовые алгоритмы поискового сервиса**

### **2.2.1. Архитектура и основные функции поискового сервиса**

Программный модуль, реализующий поисковой сервис, содержит два источника данных: сайт ЯКласс и, распознанный системой распознавания символов, учебник по геометрии 7-9 классов Атанасяна. С сайта приложение YaKlass.ipynb распарсивает ссылки на уроки по геометрии, а также сохраняет их в удобном виде (с расширением html). Следующее приложение Definition\_parser.ipynb открывает сохраненные файлы и пытается их анализировать: упрощает представление данных об уроках, строит таблицу Excel для анализа синтаксической структуры определений. Следующее приложение Terms\_parser.ipynb дает эксперту возможность полуавтоматически выделить термины из определений, а также задать начальные и конечные границы определения, осуществляет автоматическое выделение терминов для простых случаев и добавляет определения с терминами в обоих случаях в БД definitions. Приложение Xml\_terms\_parser.ipynb читает все размеченные вручную определения из учебника по геометрии и добавляет результат в БД. Программа Search\_System.ipynb принимает от учащегося запрос к базе данных и выдает список определений при наличии в БД.

Архитектура приложения показана на рисунке 27.

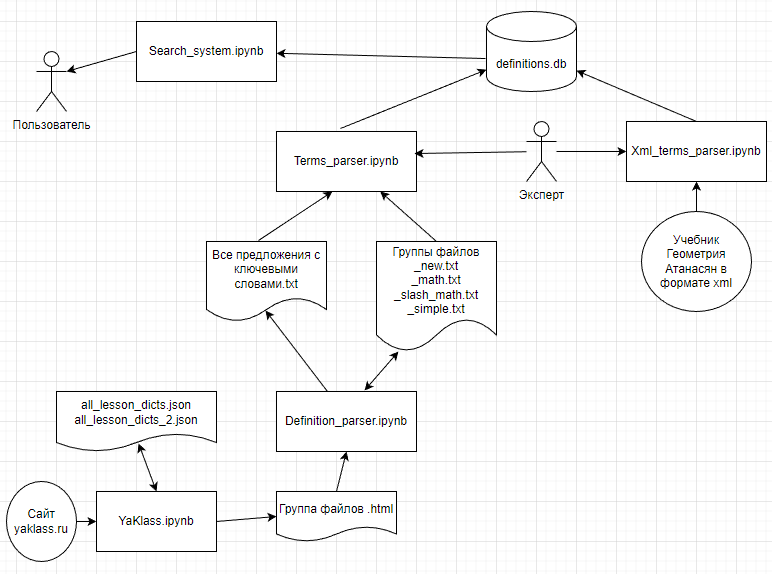


Рисунок 27 – Архитектура приложения

#### **2.2.2.1 Построение моделей шаблонов извлекаемых терминов из определений**

Для построения моделей шаблонов терминов нужно представить потенциальные дефиниционные предложения в удобно для анализа виде. Шаблон можно выделить, если известны синтаксические характеристики слов, такие как часть речи, падеж. Для визуализации синтаксических характеристик в данной работе используется Excel. Каждое предложение, разбивается на слова и знаки пунктуации и записывается поэлементно в каждый столбец первого ряда. В столбце под каждым элементом (словом или знаком) пишутся часть речи и падеж, если они есть у данного элемента.

На рисунке 28 представлен фрагмент кода для создания вышеуказанного Excel-файла.

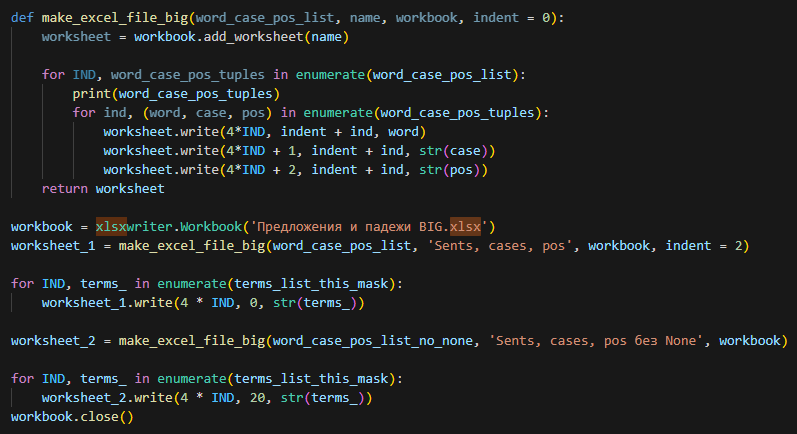


Рисунок 28 – Фрагмент кода для создания Excel-файла

После создания Excel-файла мы проанализировали его содержимое и выбрали цвет для некоторых ячеек для удобства анализа. Результат на рисунке 29.

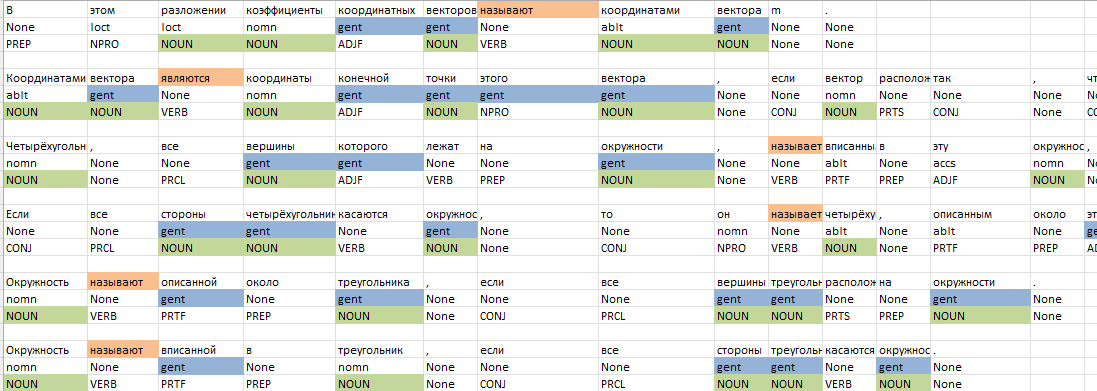


Рисунок 29 – Фрагмент Excel-файлы с выделенными ячейками

Выявленные шаблоны:

1. Определение, содержащее подстроку ’ – это ’
2. Предложение делится на две части по запятой. Слово ‘если’ в левой части предложения, также в левой части находится существительное (например, прямая). Ключевое слово находится в правой части предложения. Слева от ключевого слова стоит местоимение. Справа от ключевого слова стоит прилагательное (например, секущая). Термин состоит из прилагательного и существительного (например, секущая прямая)

#### **2.2.2.2. Автоматическое извлечение терминов из определений**

Процесс автоматического извлечения терминов из определений, содержащих подстроку ’ – это ’. Для начала мы достаем все определения с данной подстрокой из текста уроков. Получается 33 определения. Из которых 10 штук содержат математические формулы. Исключаем их. Также исключаем предложения со словом ‘например’ (например, картина и её фотография — это подобные фигуры), потому что это не определение. Осталось 22 определения.

Теперь делим предложение две части: та, что находится до ’ – это ’ и та, что находится после. После считаем количество слов (без символов пунктуации) в левой и правой частях и отбираем ту, которая содержит наименьшее количество слов.

Если предложение содержит скобки, в которых есть подстрока ’ – это ’, то рассматриваем содержимое скобок отдельно.

После этого в меньшей по количестве слов части предложения находится термин. Надо лишь удалить следующие слова “и будет”, “другими словами”.

Результат работы автоматического выделения терминов представлен на рисунке 30.

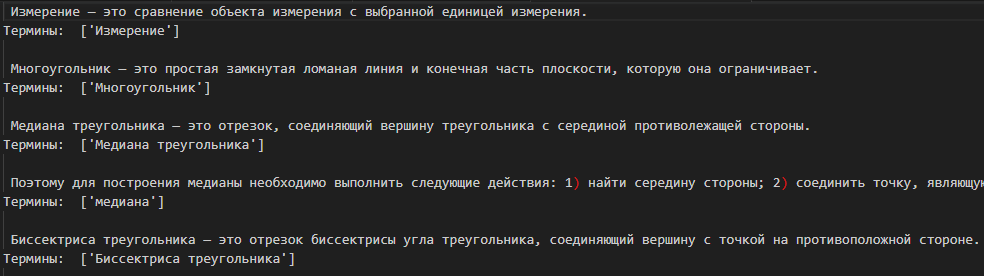


Рисунок 30 – Автоматически выделенные термины

#### **2.2.2.3. Полуавтоматическое извлечение терминов**

Для терминов, которые не выделяются автоматически, разработан полуавтоматический способ.

Для этой задачи разработан графический интерфейс, который позволяет вручную выделять термины из определений, а также выделять границы этих определений.

В разработанном приложении «Система для извлечения терминов» имеются два поля для выделения терминов, кнопка сохранения терминов в базу данных (подсвечивается зеленым при нажатии), кнопки для перехода к следующему или предыдущему определениям, кнопки «вверх» и «вниз» для того, чтобы расширить определение до нескольких предложений. Последнее необходимо, так как некоторые определения зависят от предыдущих предложений или распространяются на последующие.

Пример такой ситуации приведен на рисунке 31.

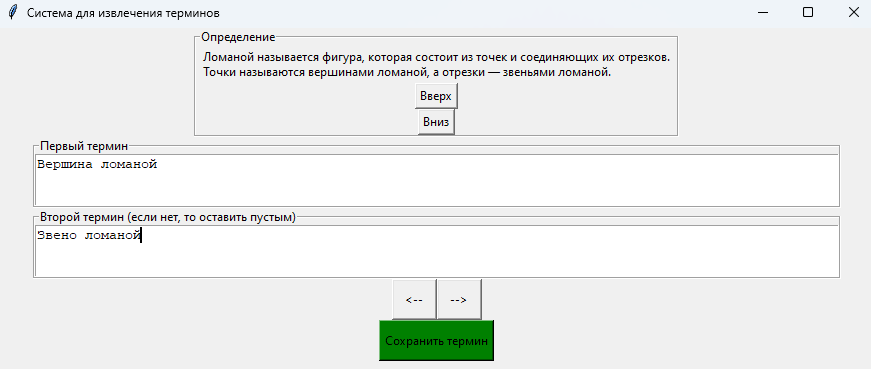


Рисунок 31 – Полуавтоматическое выделение терминов

#### **2.2.2.4. База данных для поискового индекса.**

Опишем базу данных, которая используется в данной ВКР.

Она имеет поля:

* termid – ключевое автоинкрементированное поле
* firstterm – первый термин определения
* secondterm – первый термин определения (если присутствует)
* definition – сам текст определения (может состоять из нескольких предложений)
* document – исходный документ определения (либо “YaKlass.ru”, либо “Геометрия 7-9 Атанасян”)

Фрагмент SQL-таблицы представлен на рисунке 32.

****

Рисунок 32 – SQL-таблица «definitions»

#### **2.2.2.5. Построение поисковых запросов к базе данных.**

Так как термин, по которому осуществляется поиск в базе данных может быть как в поле firstterm, так и в поле secondterm, то нужно формировать SQL-запрос следующим образом.

*SELECT definition, document FROM terms WHERE firstterm = '?' OR secondterm = '?';*

В приложении есть поле для ввода, если термин есть в базе данных, то приложение выводит все известные определения, пример работы поисковой системы на рисунке 33.

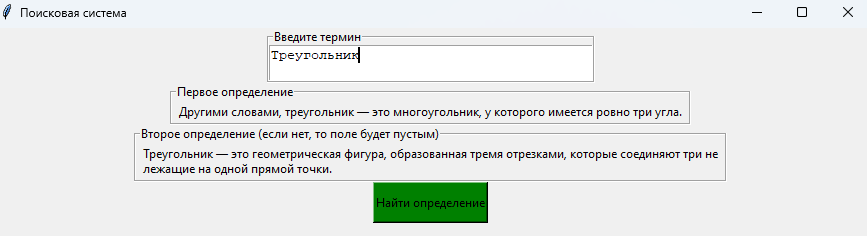


Рисунок 33 – Пример работы поискового модуля

# **Заключение**

В ходе выполнения настоящей ВКР была создана коллекция математических учебных текстов и преобразована к удобному для анализа формату. Разработан инструмент (Excel-файл) для выявления синтаксических шаблонов математических определений. Реализовано автоматическое и полуавтоматическое выделение терминов с помощью специального модуля с графическим интерфейсом. Разработана БД для хранения выделенных объектов. Разработан поисковый индекс и интерфейс поисковой системы по математическим терминам.

# **Список литературы**

**[1]** Семантическая аналитика: новые возможности работы с информацией : [Электронный ресурс] / Бизнес Семантика - URL: <http://www.business-semantic.ru/files/SemanticAnalytics.pdf> (дата обращения 12.05.2023)

**[2]** OntoMathEdu: Towards an Educational Mathematical Ontology : [статья] A. Kirillovich1, O. Nevzorova, M. Falileeva, E. Lipachev, L. Shakirova // Workshop Papers at 12th Conference on Intelligent Computer Mathematics CICM 2019 (дата обращения 08.05.2023)

**[3]** Системы образовательных проекций, уровней и пререквизитов математической онтологии OntoMathEdu : [статья] / М. В. Фалилеева, А. В. Кириллович, О. А. Невзорова, Л. Р. Шакирова, Е. К. Липачёв, А. Э. Дюпина.// Электронные библиотеки. 2021 Т. 24 № 4 15.07.21 С. 505–530. (дата обращения 03.05.2023)

**[4]** СИСТЕМНЫЙ БЛОКЪ. NER : [сайт] - 2018-2023. - Дата обновления: 22.01.2020. - URL: https://sysblok.ru/glossary/named-entity-recognition-ner/ (дата обращения 07.04.2023)

**[5]** NER [Электронный ресурс]: Википедия. Свободная энциклопедия. - URL: https://en.wikipedia.org/wiki/Named-entity\_recognition (дата обращения 07.04.2023).

**[6]** ЯКласс : [сайт] - Москва, Инновационный центр Сколково, 2014-2023. - URL: https://www.yaklass.ru/p/geometria (дата обращения 05.04.2023). - Текст: электронный.

**[7]** https://market.yandex.ru/product--atanasian-l-s-butuzov-v-f-kadomtsev-s-b-i-dr-geometriia-7-9-klass-uchebnik-2023/1811684056?clid=703&sku=101922318688&cpa=1

*Атанасян Л. С, Бутузов В. Ф, Кадомцев С. Б. и др. Геометрия. 7-9 класс. Учебник ( 2023 )*

# **Приложение**

stops = set(stopwords.words('russian'))

stops.add('.')

stops.add(',')

stops.add('Другими')

stops.add('словами')

def text\_from\_soup(soup):

'''Извлекает весь текст во всех тэгах элемента soup в строку'''

sents\_by\_strip = " ".join(item.strip() for item in soup.find\_all(text=True)) #находит все тэги с текстом внутри

sents\_by\_strip = re.sub("\s+", " ", sents\_by\_strip)

return sents\_by\_strip

def term\_from\_sent(sent, sep = '— это '):

sent = sent.replace('.', '')

left\_side\_sent = sent.split(sep)[0]

right\_side\_sent = sent.split(sep)[1]

term = (left\_side\_sent, right\_side\_sent) \

[len(word\_tokenize(left\_side\_sent.replace('.', '').replace(',', ''))) > \

len(word\_tokenize(right\_side\_sent.replace('.', '').replace(',', ''))) \

] #выделяет наименьшую по количеству слов часть предложения, которая разбивается '— это'

return term.strip()

def score(word):

return morph.parse(word)[0].score

def merging\_broken\_splitted\_words(words):

'''объединяет слова, например, "Т" и "реугольник" объединятся в "Треугольник"'''

if ('l' in words) or ('m' in words):

return None

new\_words = []

words\_was\_joined = False

for ind, word in enumerate(words):

if ind == len(words) - 1:

if words\_was\_joined == False:

new\_words.append(word)

break

if len(word) == 1:

new\_word = words[ind] + words[ind + 1]

if score(new\_word) > score(words[ind + 1]):

new\_words.append(new\_word)

words\_was\_joined = True

else:

new\_words.append(word)

elif words\_was\_joined == False:

new\_words.append(word)

else:

words\_was\_joined = False

return new\_words

def concat(terms):

terms = [merging\_broken\_splitted\_words(word\_tokenize(term)) for term in terms]

if None in terms:

return None

filtered\_words = [' '.join([word for word in term if word not in stops]) for term in terms]

return filtered\_words

if \_\_name\_\_ == "\_\_main\_\_":

extracted\_terms = 0

new\_paths = [path for path in os.listdir() if 'simple' in path]

for path in new\_paths:

soup = read\_soup(path)

sents\_by\_strip = text\_from\_soup(soup)

sents\_by\_strip\_2 = [sent + '.' for sent in sents\_by\_strip.split('.')]

for sent in sents\_by\_strip\_2:

if ('— это ' in sent) and ('l' not in sent) and ('Например' not in sent):

print(sent)

terms = [] # список терминов, извлеченных из определения

if '(' in sent and ')' in sent:

sents\_in\_brackets = [match[1] for match in re.finditer(r"\(([^()]+)\)", sent)]

for sub\_sent in sents\_in\_brackets: # так как может быть несколько пар скобок ()

if '— это ' in sub\_sent:

terms.append(term\_from\_sent(sub\_sent))

sent = re.sub(r"\(([^()]+)\)", "", sent)

terms.append(term\_from\_sent(sent))

if len(terms) > 1:

#print(sent)

#print(terms)

extracted\_terms += 2

pass

else:

extracted\_terms += 1

print(terms)

finally\_terms = concat(terms)

finally\_terms = [finally\_terms\_.capitalize() for finally\_terms\_ in finally\_terms]

print('Термины: ', finally\_terms)

termin = (None, finally\_terms[0], (finally\_terms[-1], None) [len(finally\_terms) == 1],

sent, 'YaKlass.ru')

print(termin)

print()

cur.execute("INSERT INTO terms VALUES(?, ?, ?, ?, ?);", termin)

conn.commit()

print(extracted\_terms)