Документация для алгоритма re:match v2.0

# Общее описание

Алгоритм re:match выдаёт рекомендации для пользователей библиотек на основе ранее прочтённых этим пользователем книг, анализе интересов и сопоставлении групп интересов с интересами других пользователей.

В основе лежит гипотеза:

1. рекомендации - есть книги пользователей, читающих то же, что и мы
2. ) множества книг вкладываются друг в друга; принадлежность книги более узкому множеству указывает на вероятность принадлежности более широкому

# Изменения с предыдущей версии

* Повышена производительность за счёт оптимизации кода
* Книги, взятые пользователем, рассматриваются не как единое цело, а как множество групп интересов, благодаря чему рекомендации стали разнообразнее

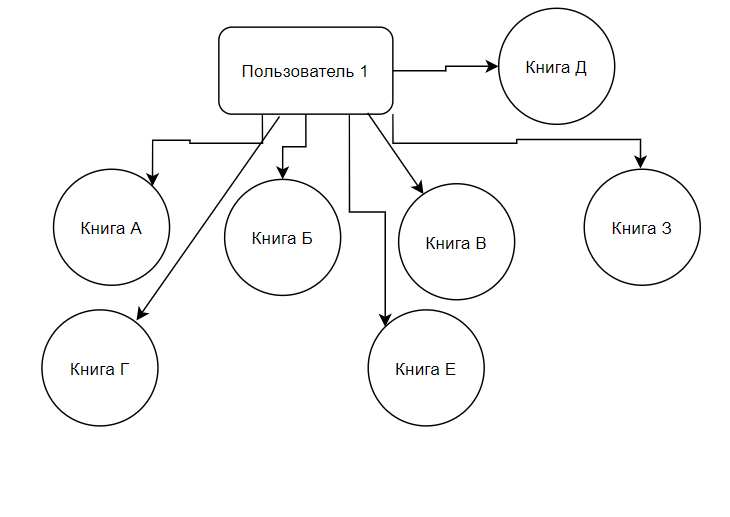
# Описание работы

## Выявление круга интересов

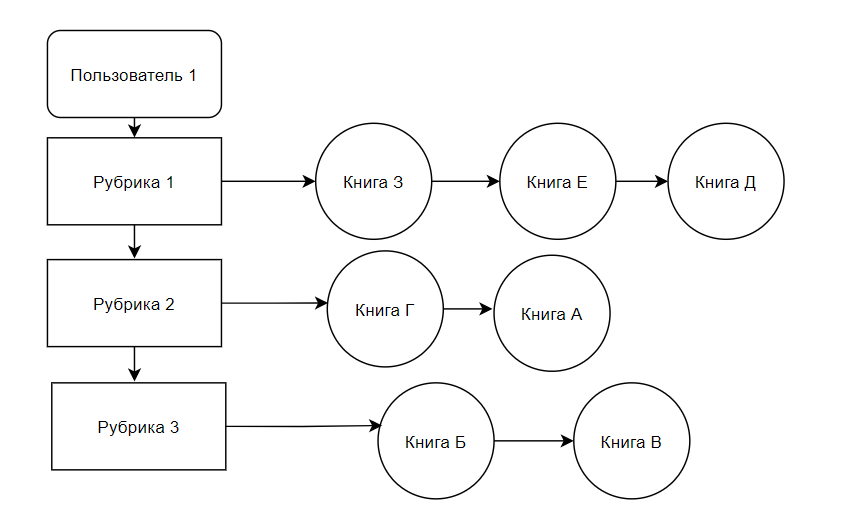
### Построение плоского множества интересов.

На данном этапе множество взятых пользователем книг преобразуется

В множество интересов.

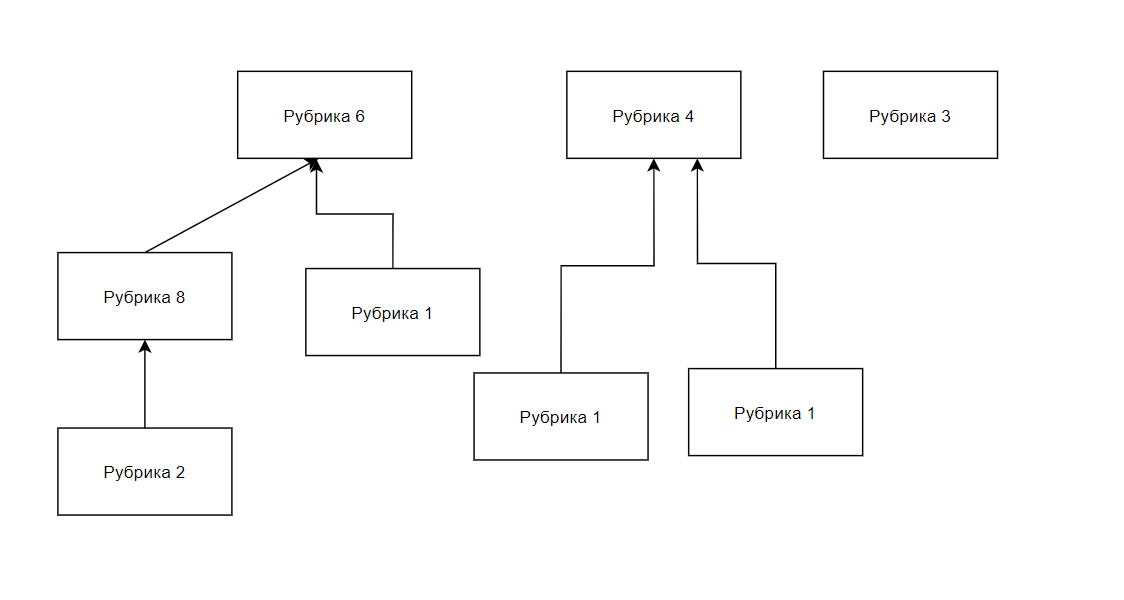
Было: 

Стало:



### Выявление скрытых интересов методом наложения на граф онтологии рубрик

Схематический вид графа онтологии



Полный граф можно найти в разделе dataset/ontology.json

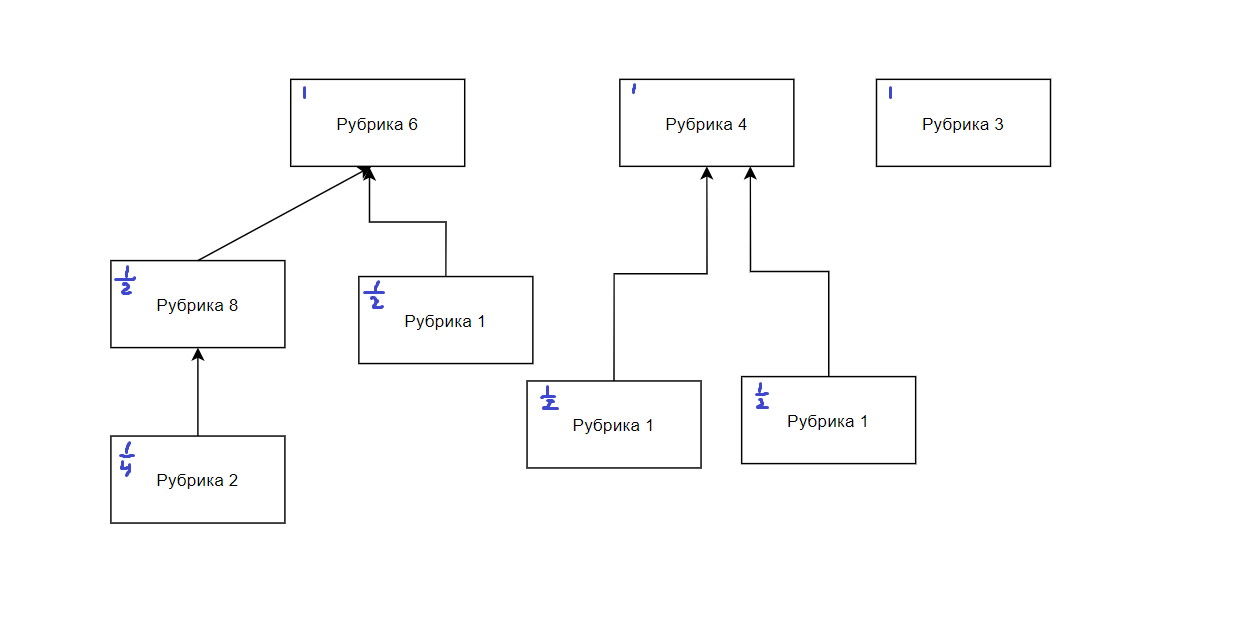
Метод наложения состоит из 3-х этапов.

1. Считаем собственный вес каждой рубрики

Собственный вес рубрик корневого уровня = 1. Вес вложенных:

Таким образом сумма собственных весов дочерних узлов даёт вес родителя.

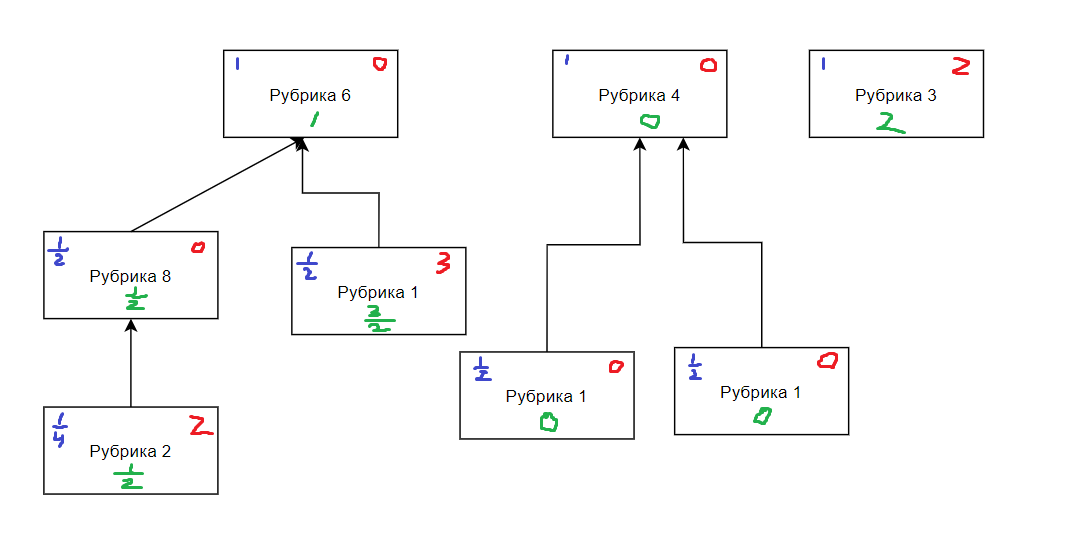
Пример:

**

Далее “утяжеляем” узлы по следующей формуле:

Пример:  
 Красным – кол-во книг.

Зелёным – итоговый вес.



Считаем итоговую квоту по рекомендациям из каждой рубрики.

Для этого выписываем рубрики и сортируем в порядке убывания веса.

Далее вычисляем квоту по рекомендациям из каждой рубрики исходя из таблицы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Место в рейтинге рубрик | 1 | 2 | 3 | 4 | 5 |
| Кол-во ненулевых рубрик |  |  |  |  |  |  |
| 1 |  | 5 |  |  |  |  |
| 2 |  | 3 | 2 |  |  |  |
| 3 |  | 2 | 2 | 1 |  |  |
| 4 |  | 2 | 1 | 1 | 1 |  |
| 5+ |  | 1 | 1 | 1 | 1 | 1 |

В нашем случае рубрики 1, 2, 3, 6, 8 получат по 1 квоте в рекомендациях.

**Итог: В рекомендации проходят рубрики 1, 2, 3, 6, 8 с соответствующими квотами**

## Расчёт итоговых рекомендаций

Расчёт итоговых рекомендаций происходит на основе поиска смежных интересов. Т.е смотрим, что ещё читают в данной рубрике пользователи, читающие схожие с нами книги.

Продолжим рассматривать предыдущий пример:

Как мы выяснили, в п 3.1 в рекомендации должны попасть книги из рубрик 1, 2, 3, 6, 8

Разберём на примере рубрики 6. Как видно, пользователь в этой рубрике ещё ничего не читал, поэтому сделаем оценку на основании всех книг, прочтённых пользователем:

{А, Б, В, Г, Д, Е, З} – набор книг пользователя.

Смотрим, какие пользователи читали книгу из набора, и что ещё они читали вместе с данной книгой по рубрике 6.

|  |  |
| --- | --- |
| **Книга А** | Книги, взятые пользователем в рубрике 6 |
| Пользователь 1 | К1, К2, К3 |
| Пользователь 2 | К4, К2, К1 |
| Пользователь 3 | К5, К6, К1 |
| … |  |
| **Кинга Б** |  |
| Пользователь 1 | К1, К2, К3 |
| … |  |

Теперь считаем самые частые книги. В нашем случае это: К1( 4раза) – её и берём в рекомендацию

**Итог: по рубрике 6 советуем пользователю книгу К1**

Для остальных рубрик проделываем тоже самое.

## Итог

Комбинация методов 3.1 и 3.2 даёт разнообразные рекомендации, учитывающие интересы пользователя.

# Развёртывание

Примечание: для работы системы требуется от 8 ГБ ОЗУ

Инструкция по установке Relib для linux ubuntu

1. Установить git: sudo apt install git
2. Установить python3: sudo apt install python3
3. Установить pip: sudo apt install python3-pip
4. Установить django: pip3 install django
5. Скачать relib с github: git clone https://github.com/morethanbot/relib\_django
6. Перейти в директорию проекта: cd relib\_django
7. Применить миграции: python3 manage.py migrate
8. Запустить сервер: python3 manage.py runserver 0:80

Прямой доступ к api через get-запрос:

http://<server address>/getUserInfo/?user\_id=<id>

где:

<server address> - адрес веб-сервера

<id> - id пользователя для которого нужно получить рекомендации

пример запроса: http://84.201.172.116/getUserInfo/?user\_id=311713

пример ответа в формате json:

Изображение выглядит как текст

Автоматически созданное описание