**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление: 09.03.03 Прикладная информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

# Разработка рекомендательного сервиса для научных цифровых библиотек

Студент 4 курса

группы 11-502

«___»_____2019 г. Мустаев А.Р.

Научный руководитель

Доктор физ.–мат. наук, профессор

кафедры программной инженерии

«___»_____2019 г. Елизаров А.М.

Директор Высшей школы ИТИС

«___»_____2019 г. Хасьянов А.Ф.

Казань – 2019 г.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В настоящее время существует огромное количество научных цифровых библиотек, которые оперируют большими объемами данных. В основном эти данные представляют собой научные статьи. По мере увеличения количества научных статей в библиотеках задача ручного поиска полезных и интересных статей для пользователя становится все более долгой и трудоемкой. Но данную задачу в автоматическом режиме решают рекомендательные системы.

Рекомендательные системы – это комплексы алгоритмов, программы и сервисы, задача которых – предсказать, какие объекты (например: научные статьи) будут интересны пользователю в данный момент времени. В основе большинства рекомендательных систем лежит матрица рейтингов – матрица, на одной оси которой расположены пользователи, а на другой оси – объекты. На пересечении пользователей и объектов могут стоять оценки – степень заинтересованности конкретного пользователя в данном объекте.

Поскольку не все научные цифровые библиотеки предоставляют открытый доступ к своим базам данных, а для реализации рекомендательного сервиса для конкретной библиотеки необходимо иметь доступ к данным о пользователях и статьях, то в рамках данной работы было решено создать веб–приложение научной цифровой библиотеки с внедрением в него разработанного рекомендательного сервиса.

Таким образом, целью данной работы является разработка рекомендательного сервиса для научных цифровых библиотек.

Для достижения поставленной цели были определены следующие **основные задачи**:

1) исследование видов рекомендательных систем и принципов их построения;
2) изучение основных алгоритмов, применяемых в рекомендательных системах;

3) проектирование и реализация рекомендательного сервиса;
4) разработка веб–приложения научной цифровой библиотеки с внедрением в него реализованного рекомендательного сервиса.

**Объектом исследования** являются рекомендательные системы и их применение к научным цифровым библиотекам.

**Предметом исследования** является разработка рекомендательного сервиса, который формирует рекомендации на основе данных о пользователях и научных статьях в научной цифровой библиотеке.

# 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

В типичных рекомендательных системах есть множество пользователей $U = \{u_1, u_2, \ldots, u_n\}$, множество объектов $P = \{p_1, p_2, \ldots, p_m\}$ и матрица рейтингов $R$ размера $n \times m$ [1], где $i \in 1 \ldots n$, $j \in 1 \ldots m$ и $r_{u_i,p_j}$ – рейтинг пользователя $u_i$ объекту $p_j$. Значениями в матрице могут быть, например, числа от 1 (не понравилось) до 5 (очень понравилось). Обычно матрица рейтингов сильно разрежена, так как пользователь оценивает лишь небольшую часть товаров.

Матрицу рейтингов можно представить в виде таблицы:

| | Объект 1 | Объект 2 | Объект 3 | Объект 4 | Объект 5 |
|---|---|---|---|---|---|
| Джоэл | 4 | ? | 2 | 3 | ? |
| Лоуренс | 1 | ? | 1 | 5 | 4 |
| Джоди | 5 | ? | ? | ? | 4 |
| Леон | 3 | 2 | ? | 4 | 3 |

Таблица 1 – Матрица рейтингов

Обозначим через $\hat{r}_{u_i,p_j}$ наш прогноз относительно того, какую оценку пользователь $u_i$ поставит объекту $p_j$. Задача рекомендательной системы – наилучшим образом предсказать, какие оценки $r_{u_i,p_j}$ должны стоять на месте пропусков в матрице рейтингов, то есть рассчитать $\hat{r}_{u_i,p_j}$. Затем для каждого пользователя $u_i$ на основе спрогнозированных оценок $\hat{r}_{u_i,p_j}$ нам нужно сформировать список из $N$ объектов, которые наиболее точно удовлетворяют предпочтениям пользователя и которые еще не были им оценены. Список этих $N$ объектов обозначим через $N$–мерный вектор $(p_{i1}, p_{i2}, \ldots p_{iN})$. Таким образом, математически задача звучит так:

Дано:

$U = \{u_1, u_2, \ldots, u_n\}$ – множество пользователей;

$P = \{p_1, p_2, \ldots, p_m\}$ – множество объектов;

$R$ – матрица рейтингов размера $n \times m$, где на месте $r_{u_i,p_j}$ будет стоять некоторое число, если пользователь $u_i$ оценил объект $p_j$, и 0 в противном случае;

$N$ – требуемое число рекомендаций, которые хотим получить от системы (задачи в такой постановке также называют Top–N рекомендацией).

Требуется найти:

Для данного пользователя $u_i$ найти $N$–мерный вектор $(p_{i1}, p_{i2}, \ldots p_{iN})$, где объекты $p_{ik}$, $k \in N$, еще не оценены этим пользователем, то есть в матрице рейтингов $R$ стоит 0 на месте $p_{i,ik}$, а также чтобы эти объекты наиболее точно удовлетворяли предпочтениям пользователя, то есть прогнозные рейтинги $\hat{r}_{u_i,p_j}$ были наибольшими.

Способы формирования матрицы рейтингов $R$ бывают явные и неявные. Первые предполагают, что пользователь, ознакомившись с объектом, выставит ему оценку по имеющейся шкале (например, от 1 до 5). В таких случаях $R$ составлена из оценок в чистом виде. Однако данный способ применяют не всегда. Часто для составления матрицы приходится анализировать историю взаимодействия пользователей с объектами, в этом случае имеет место неявное формирование рейтинга. В случае, например, с музыкальными треками величина $r_{u_i,p_j}$ может вычисляться как количество прослушиваний [2].

### 1.1 Требования к данным для разных видов рекомендательных систем

Рекомендательные системы могут использовать явный или неявный сбор данных [1].

Явный сбор – пользователь заполняет анкеты, оценивает объекты, составляет список любимых продуктов. Также в некоторых видах

рекомендательных систем могут использоваться демографические данные, такие как: пол, возраст, регион, языки и т.д.

Неявный сбор – автоматическое протоколирование действий пользователя, наблюдение за тем, что он посмотрел, прокомментировал, купил. Например: если пользователь купил товар, то, значит, он ему нравится. Очевидный недостаток здесь – неопределенность: если пользователь посмотрел товар, то неизвестно, понравился он ему или нет; если пользователь не купил товар, то опять же неизвестно, чем было обусловлено такое решение.

Также можно совмещать явные и неявные способы [3]. К примеру, для сервиса, на котором публикуются статьи, есть смысл учитывать как явную оценку, проставленную пользователем, так и неявную – просмотр статьи (косвенная заинтересованность).

## 1.2 Проблема холодного старта

Холодный старт – это типичная ситуация, когда еще не накоплено достаточное количество данных для корректной работы рекомендательной системы [3]. В свою очередь проблема холодного старта делится на холодный старт для пользователей (что показывать новым пользователям) и холодный старт для объектов (кому рекомендовать вновь добавленные объекты).

## 1.3 Подходы к рекомендациям

В области разработки рекомендательных систем сложились традиционные подходы, обладающие рядом достоинств и недостатков.

Существует множество подходов к формированию рекомендаций, но в основном в рекомендательных системах используется один из трех базовых подходов [3]:

1) Коллаборативная фильтрация (collaborative filtering);
2) Контентная фильтрация (content–based filtering);

3) Гибридные рекомендательные системы (hybrid recommender system).

### 1.3.1 Коллаборативная фильтрация

Первый из рассматриваемых подходов, коллаборативная фильтрация, – это системы, в которых рекомендации пользователю рассчитывается на основе истории оценок как самого пользователя, так и других пользователей, ведущих себя похожим образом. Анализируя профили, рекомендательная система находит похожих пользователей, и оценка предпочтимости нового объекта рассчитывается с использованием их оценок [3]. Различают два варианта коллаборативной фильтрации: пользовательская фильтрация (user–based) и фильтрация по объектам (item–based) [4].

Пользовательская фильтрация. Рейтинг объекта для данного пользователя складывается из проставленных оценок этому объекту похожими пользователями. Данный подход сначала ищет пользователей, похожих на данного (например, оценили одинаковые товары, совершили аналогичные покупки), а затем предлагает объекты с максимальным рейтингом среди объектов, которые выбирают похожие пользователи.

Фильтрация по объектам. Практически полностью повторяет принцип пользовательской фильтрации, кроме одного момента – рейтинг объекта складывается из рейтингов других схожих объектов, оцененных данным пользователем. То есть этот подход ищет близкие объекты, а не пользователей. В данном подходе, во–первых, строится матрица схожести объектов, а, во–вторых, определяется степень схожести из данной матрицы и на основании этой схожести предлагаются объекты, похожие на уже оцененные данным пользователем.

При выборе пользовательской фильтрации или фильтрации по объектам необходимо учитывать, кого больше – пользователей или объектов [5]. Если

больше пользователей, то предпочтительнее фильтрация по объектам, если наоборот, то пользовательская фильтрация. Суть этих алгоритмов – нахождение ближайших соседей [1]. Близость двух пользователей или объектов определяется метриками схожести [1].

Небольшое сравнение пользовательской фильтрации и фильтрации по объектам представлено в таблице [6]:

| | user–based | item–based |
|---|---|---|
| Актуальность рекомендаций | Пользовательские предпочтения со временем меняются. В результате система может генерировать множество неактуальных рекомендаций | Элементы в системе (например, фильмы) не изменяются со временем, поэтому рекомендации получатся более релевантными |

Таблица 2 – Сравнение пользовательской фильтрации и фильтрации по объектам

Преимущества подхода:

1) Является достаточно универсальным подходом, поэтому позволяет получать очень точные и уместные рекомендации [7].
2) Для работы данного метода не нужна детальная информация об объектах. Вместо этого используется как история оценок самого пользователя, так и других пользователей.

Недостатки подхода:

1) Холодный старт как для новых пользователей, так и для новых товаров.
2) Разреженность матрицы рейтингов. Объекты с небольшим числом оценок будут показываться реже даже с лучшими оценками.
3) Ресурсоемкость вычислений. Для того, чтобы делать предсказания, нам нужно держать в памяти все оценки всех пользователей.
4) Необходим большой объем данных для высокой точности предсказаний.

### 1.3.2 Контентная фильтрация

Одним из первых подходов к формированию персональных рекомендаций является контентная фильтрация [8]. Контентная фильтрация основывается на создании профиля пользователя и профиля объектов [8]. Профиль объекта формируется из его характеристик (тэги, жанры, слова и т.д.). Основное требование здесь: у каждого объекта должно быть описание или некоторые метаданные. Профиль пользователя может содержать различные демографические данные, а также явно указанные предпочтения (любимые жанры). Задача рекомендательных систем, основанных на контентной фильтрации, – сопоставить профиль пользователя и профиль объектов и найти множество объектов, которые наиболее близки к предпочтениям пользователя [3]. Кроме того, на основе профиля объектов могут быть найдены похожие объекты, которые пользователь оценил/просмотрел/прокомментировал в прошлом. Например, если пользователь посмотрел боевик, то ему автоматически будут рекомендовать боевики, а также фильмы похожих жанров, или, например, пользователь при регистрации указал музыкальные предпочтения, после чего ему можно рекомендовать похожие композиции. Контентная фильтрация полностью игнорирует оценки пользователей по поводу тех или иных объектов [9]. Выстраивая связи сугубо между самими

объектами, мы имеем возможность моментально, без сбора оценок и дополнительных персональных сведений, предложить человеку что–то похожее на ту позицию, которая его заинтересовала.

Преимущества подхода:

1) Почти полное отсутствие проблемы холодного старта для пользователей. Возможность заинтересовать нового пользователя предложениями с первых потребительских шагов. Для этого не нужно долго собирать данные о предпочтениях, а можно сразу включить пользователя в работу с сервисом.
2) Возможность рекомендовать пользователю те объекты, которые не оценили и обошли стороной другие пользователи.
3) Новые элементы можно рекомендовать сразу, как только у них появляются заполненные характеристики. Решение проблемы холодного старта для объектов.

Недостатки подхода:

1) Сильная зависимость от предметной области, полезность рекомендаций ограничена.
2) Профиль пользователей и объектов должен состоять из одинакового набора характеристик, чтобы их можно было сравнивать.
3) Узость рекомендаций. Рекомендательные системы на основе контента рекомендует только те объекты, чьи профили похожи на профили объектов, оцененных пользователем. Таким образом, сервис не может предложить пользователю что–то выходящее за рамки его известных интересов и тем самым не способна удивить

чем–то новым, а также может предлагать объекты, которые уже есть у пользователя, что в ряде случаев бесполезно.

### 1.3.3 Гибридные рекомендательные системы

Каждый из предыдущих подходов имеет свои преимущества и недостатки в зависимости от поставленной задачи. Достаточно очевидным решением всех этих проблем является объединение различных подходов для того, чтобы обеспечить большую точность рекомендаций [9].

Если, например, у нас есть данные об описании объектов, профиль пользователей и история его оценок, то мы можем улучшить рекомендательную систему путем объединения методов коллаборативной фильтрации и алгоритмов фильтрации по содержанию. Таким образом, в случае появления нового пользователя в системе, у которого нет истории оценок, мы сможем использовать рекомендации на основе алгоритмов фильтрации по содержанию, а в случае большого объема статистических данных строить более точный прогноз, используя методы коллаборативной фильтрации.

В гибридных рекомендательных системах встречаются следующие типы комбинирования [3]:

- Реализация по отдельности коллаборативных и контентных алгоритмов и объединение их предположений.
- Включение некоторых контентных правил в коллаборативную методику.
- Включение некоторых коллаборативных правил в контентную методику.
- Построение общей модели, включающей в себя правила обеих методик.

# 2. ПРОЕКТИРОВАНИЕ РЕКОМЕНДАТЕЛЬНОГО СЕРВИСА

После рассмотрения основных подходов, которые применяются в рекомендательных системах, было решено использовать гибридный подход: контентная фильтрация и коллаборативная фильтрация. Такой подход позволяет устранить недостатки, которые присущи контентной и коллаборативной фильтрациям по отдельности. Таким образом, решается проблема холодного старта, когда еще недостаточно накоплено данных о пользовательских оценках, а при достаточно большом объеме данных о пользовательских оценках коллаборативная фильтрация имеет более высокую точность предсказания по сравнению с контентной фильтрацией, что имеет большое значение при формировании рекомендаций в контексте научных цифровых библиотек.

Рассмотрим статистику российской научной цифровой библиотеки "eLIBRARY.ru":

Рис. 1 – Статистика "eLIBRARY.ru"

Из данной статистики видно, что для научных цифровых библиотек нецелесообразно использовать Item–based алгоритм для коллаборативной фильтрации, поскольку статей гораздо больше, чем пользователей. Таким образом, рекомендации будут формироваться с использованием алгоритма коллаборативной фильтрации по пользователям.

### 2.1 Алгоритм User–based коллаборативной фильтрации

Входными данными для алгоритмов коллаборативной фильтрации является матрица рейтингов $R$, далее для удобства будем называть ее user–item. При рассмотрении алгоритма значениями матрицы user–item будут числа от 1 (не понравилось) до 5 (очень понравилось). Таким образом, строки матрицы – вектор предпочтений для каждого пользователя, столбцы – вектор оценок пользователей для каждого объекта.

Данный алгоритм ищет похожих пользователей и на основании этих пользователей формирует рекомендации.

Алгоритм состоит из нескольких основных этапов:

1) формирование матрицы user–user, на обеих осях которой расположены пользователи, а значениями в ячейках матрицы выступают меры схожести между соответствующими пользователями;
2) на основании матрицы user–user ищутся k пользователей с наиболее высокой мерой схожести с данным пользователем $u_a$;
3) вычисление прогноза с использованием оценок соседей;
4) выбрать Top–N рекомендаций.

Прежде чем формировать матрицу user–user, необходимо произвести формальный контроль входных данных для уменьшения шума в рекомендациях

и увеличения их точности. Поэтому в матрице user–user используются данные о пользователях, оценивших не менее определенного значения (confidence value) объектов.

Далее в каждой ячейке матрицы user–user[a][b], где a, b $\in 1 \dots n$, необходимо проставить значение меры схожести между пользователем $u_a$, который соответствует строке матрицы, и пользователем $u_b$, который соответствует столбцу матрицы, с использованием одной из метрик подобия. Обозначим меру схожести между пользователями $u_a$ и $u_b$ через $sim(u_a, u_b)$, а вектора предпочтений пользователей $u_a$ и $u_b$ через $\vec{a}$ и $\vec{b}$ соответственно.

Наиболее популярные метрики подобия:

1) Корреляция Пирсона.
2) Корреляция Спирмена.
3) Косинусное расстояние.

В качестве меры схожести двух пользователей чаще всего используется косинусное расстояние [1]:

$$sim(u_a, u_b) = \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|} = \frac{\sum_{j=1}^{m} r_{u_a,p_j} r_{u_b,p_j}}{\sqrt{\sum_{j=1}^{m} r_{u_a,p_j}^2} \sqrt{\sum_{j=1}^{m} r_{u_b,p_j}^2}}$$

Поскольку все пользователи оценивают по–разному, существуют пользователи, которые дают только плохие оценки плохим объектам; другие ставят только хорошие оценки для всех объектов; третьи, которые оценивают исключительно хорошие объекты и т.д. Поэтому должно учитываться среднее отклонение пользователей для нормализации вклада оценок таких пользователей. Таким образом, в приведенной выше формуле из рейтингов $r_{u_a,p_j}$ вычитаются средние значения рейтингов $\bar{r}_{u_a}$ для данного пользователя:

$$sim(u_a, u_b) = \cos(\vec{a}, \vec{b}) = \frac{\sum_{j=1}^{m} (r_{u_a,p_j} - \bar{r}_{u_a})(r_{u_b,p_j} - \bar{r}_{u_b})}{\sqrt{\sum_{j=1}^{m} (r_{u_a,p_j} - \bar{r}_{u_a})^2} \sqrt{\sum_{j=1}^{m} (r_{u_b,p_j} - \bar{r}_{u_b})^2}}$$

Дополнительным плюсом косинусной меры является ее нормированность, поскольку значения укладываются в интервал $[0, 1]$.

Следующим этапом алгоритма является отбор k соседей с наиболее высоким значением меры схожести. То есть нужно взять строку в матрице user–user, соответствующую пользователю, для которого формируем рекомендации, отсортировать эти значения и выбрать k наибольших.

| | Ада | Виллем | Клэр | Вамос | Лирой |
|---|---|---|---|---|---|
| Ада | | | | | |
| Виллем | | | | | |
| Клэр | 0.9449 | 0.8944 | 0 | 0.8660 | 0.9049 |
| Вамос | | | | | |
| Лирой | | | | | |

Таблица 3 – Поиск k ближайших соседей для пользователя Клэр

После отбора k ближайших соседей можно рассчитать предполагаемую оценку пользователя $u_a$ объекту $p_x$. Искомая оценка будет вычисляться как сумма средней оценки данного пользователя и отклонений от средних оценок других пользователей для этого объекта с учетом коэффициента схожести:

$$\hat{r}_{u_a,p_x} = \bar{r}_{u_a} + \frac{\sum_{i \in kNN(u_a)} sim(u_a, u_i)(r_{u_i,p_x} - \bar{r}_{u_i})}{\sum_{i \in kNN(u_a)} |sim(u_a, u_i)|}$$

где $sim$ – выбранная нами мера схожести двух пользователей. После расчета прогнозируемого рейтинга следует выбрать N объектов с наибольшим рейтингом, исключая объекты, которые пользователь уже оценивал.

### 2.2 Алгоритм контентной фильтрации

До этого мы рассматривали персонализированную контентную фильтрацию. Здесь же мы будем рассматривать неперсонализированный

алгоритм контентной фильтрации, который выстраивает связи сугубо между документами, игнорируя пользовательские предпочтения.

Ниже перечислены основные этапы алгоритма:

1) формирование векторов, которые состоят из весов ключевых слов конкретного документа;
2) вектор целевого документа $\vec{x}$ сравнивается с векторами остальных документов с использованием косинусной меры схожести;
3) выбираются N самых похожих документов на целевой документ $p_x$ .

Сначала необходимо сформировать вектора документов, состоящие из весов ключевых слов. Но перед этим следует предобработать текст. Для достижения этого из основного содержания статьи удаляются все стоп–слова, не несущие смысловой нагрузки, и слова, короче трех символов. Каждое из оставшихся слов проходит процесс лемматизации для приведения слова к нормальной форме.

После предобработки основного содержания статьи можно переходить к формированию вектора документа. Для каждого слова рассчитывается их частота в документе, то есть мера TF:

$$tf_{w,p_x} = \frac{n_{w,p_x}}{n_{p_x}}$$

где в числителе – количество вхождений слова $w$ в документ $p_x$, а в знаменателе – общее число слов в документе $p_x$. Но если учитывать только частоту вхождения слова в документе, то в большинстве документов максимальный вес будет у самых распространенных слов, что, вероятнее всего, приведет к неправильному оцениванию сходства документов. Для избегания подобного применяется IDF – величина, обратная частоте вхождения слова в документе коллекции:

$$idf_{w,P} = \log \frac{N_P}{N_{w,P}}$$

где $N_P$ – количество документов в коллекции, а $N_{w,P}$ – количество документов в коллекции, в которых встречается слово $w$.

Таким образом, вес слова определяется как произведение TF на IDF. Но поскольку в современных научных цифровых библиотеках количество статей постоянно увеличивается, то и постоянный пересчет значения TF–IDF становится все более трудозатратным. Поэтому, исходя из соображений производительности, было решено оставить только меру TF, которая отображает частоту слова в документе.

После того как для каждого слова рассчитано значение TF, выбираются несколько самых наиболее встречаемых слов. Эти слова представляют собой вектор документа.

Далее вычисляется косинусная мера схожести для целевого документа $p_x$ и всех остальных документов по формуле:

$$sim(p_x, p_y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|} = \frac{\sum_{i=1}^{k} tf_{p_x,i} \cdot tf_{p_y,i}}{\sqrt{\sum_{i=1}^{k} tf_{p_x,i}^2} \sqrt{\sum_{i=1}^{k} tf_{p_y,i}^2}}$$

где k – общее количество ключевых слов.

В конце формируется список из N документов с наиболее высоким значением косинусной меры.

### 2.3 Формирование рекомендаций

Пусть некоторый пользователь $u_i$ просматривает статью $p_j$, тогда можно определить принцип, по которому будут строиться рекомендации для разных типов пользователей:

Пользователь, который оценил менее 9 статей – контентная фильтрация для статьи $p_j$.

Пользователь, который оценил 9 статей или более, – User–based алгоритм коллаборативной фильтрации для пользователя $u_i$.

Некоторые пользователи могут полагать, что ключевую роль в контексте научных статей играет содержимое самих статей, а не отношение пользователей к ним, а также не все научные цифровые библиотеки предполагают оценку пользователей. Поэтому при формировании рекомендаций на основе коллаборативной фильтрации будут также дополнительно формироваться рекомендации на основе контентной фильтрации, давая пользователю свободу в выборе рекомендательного подхода. Таким образом, принцип для пользователя, который оценил более 9 статей или более, будет следующим – User–based алгоритм коллаборативной фильтрации для пользователя $u_i$ и дополнительно контентная фильтрация для статьи $p_j$.

# 3. РЕАЛИЗАЦИЯ

Для начала рассмотрим страницу одной из статей российской научной цифровой библиотеки "КиберЛенинка":

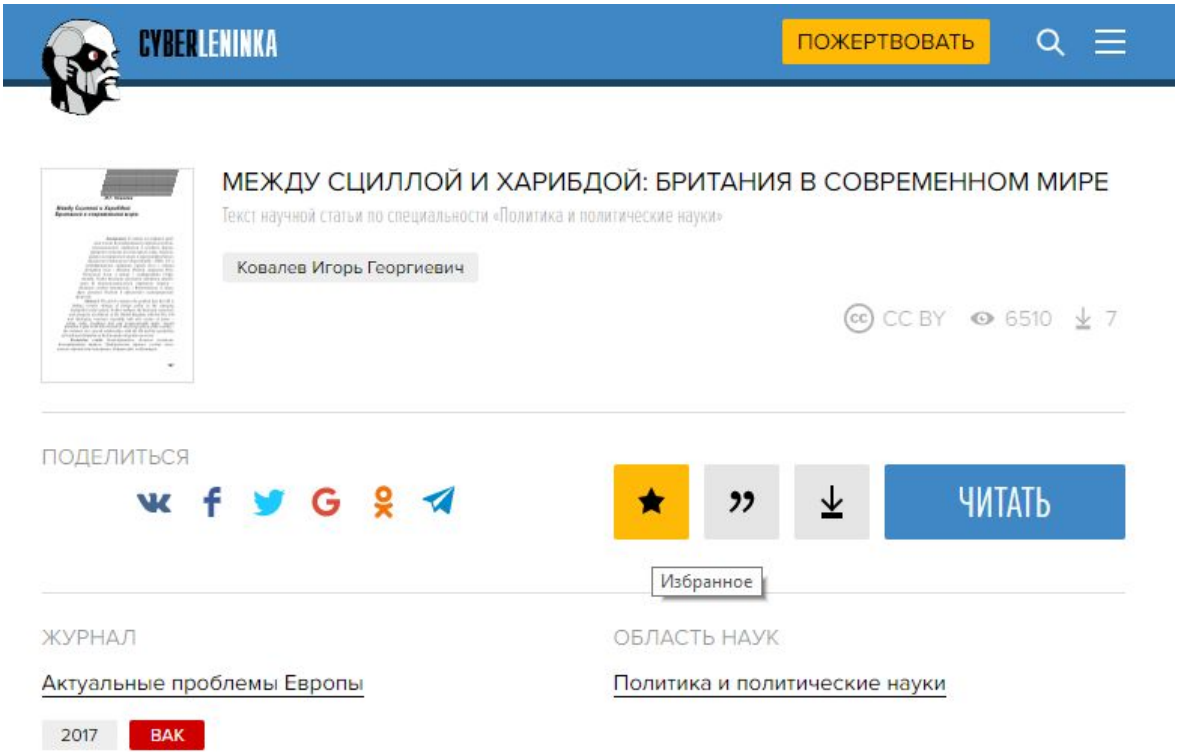


Рис. 2 – "КиберЛенинка"

У каждой статьи есть кнопка "Избранное". Это является примером бинарной оценки, то есть если пользователь добавил статью в "Избранное", значит, он считает ее полезной или интересной, а если оценки нет, то это не значит, что она ему не понравилась, возможно, пользователь просто не увидел данную статью.

В разработанном сервисе при формировании матрицы user–item используются бинарные оценки: "Понравилось" или "Нет оценки". Следовательно, при расчете меры схожести между пользователями нужно использовать бинарный косинусный коэффициент – коэффициент Отиаи (Otsuka–Ochiai coefficient). Таким образом, формула расчета меры схожести между пользователями будет следующая:

$$sim(u_a, u_b) = \frac{|N_{u_a} \cap N_{u_b}|}{\sqrt{N_{u_a} \cdot N_{u_b}}}$$

где $|N_{u_a} \cap N_{u_b}|$ – количество объектов, оцененных пользователями $u_a$ и $u_b$ и равно $\sum_{j=1}^{m} r_{u_a,p_j} r_{u_b,p_j}$. Часто используется мера Отиаи, возведенная в квадрат:

$$sim(u_a, u_b) = \frac{|N_{u_a} \cap N_{u_b}|^2}{N_{u_a} \cdot N_{u_b}}$$

Поскольку при бинарных оценках вычислить среднее значение рейтингов пользователей невозможно, то формула, вычисляющая прогнозируемую оценку пользователя $u_a$ объекту $p_x$, будет иметь следующий вид:

$$\hat{r}_{u_a,p_x} = \frac{\sum_{i \in kNN(u_a)} sim(u_a, u_i) \cdot r_{u_i,p_x}}{\sum_{i \in kNN(u_a)} sim(u_a, u_i)}$$

### 3.1 Используемые технологии

Разработанный сервис был написан на языке Java с использованием фреймворка Spring Boot для быстрой настройки приложения и простой реализации серверной части.

Описание моделей было реализовано с помощью Java библиотеки Lombok, которая позволяет генерировать шаблонный код (например: геттеры, сеттеры и т.д.).

В качестве базы данных была использована PostgreSQL, так как она является оптимальным и, в то же время, бесплатным вариантом реляционной базы данных.

Для взаимодействия сервиса с базой данных используется библиотека Spring–jdbc, которая позволяет писать запросы к базе данных любой сложности.

Для извлечения контента статьи с ресурсов научных цифровых библиотек используется библиотека Jsoup.

Лемматизация слов производилась с помощью библиотеки Apache OpenNLP, с использованием стеммера Портера.

Для реализации веб–интерфейса был использован фреймворк Vaadin, который предлагает сервер–ориентированную архитектуру и позволяет писать клиентскую часть на языке Java.

Таким образом, все клиент–серверное приложение написано на языке Java.

### 3.2 Архитектура системы

Архитектура системы представляет собой веб–интерфейс взаимодействия с пользователем, серверную часть и базу данных.

Константы, обозначенные в системе:

1) KEYWORDS_LIMIT = 15 – количество ключевых слов, из которых составляется вектор документа.
2) K = 20 – количество ближайших соседей, на основании которых будут строиться прогнозы.
3) CV = 9 – (confidence value) количество оценок, которое должно быть у пользователя, чтобы для него формировались рекомендации на основе пользовательской коллаборативной фильтрации.
4) N = 10 – количество рекомендуемых документов.

### 3.2.1 Описание базы данных

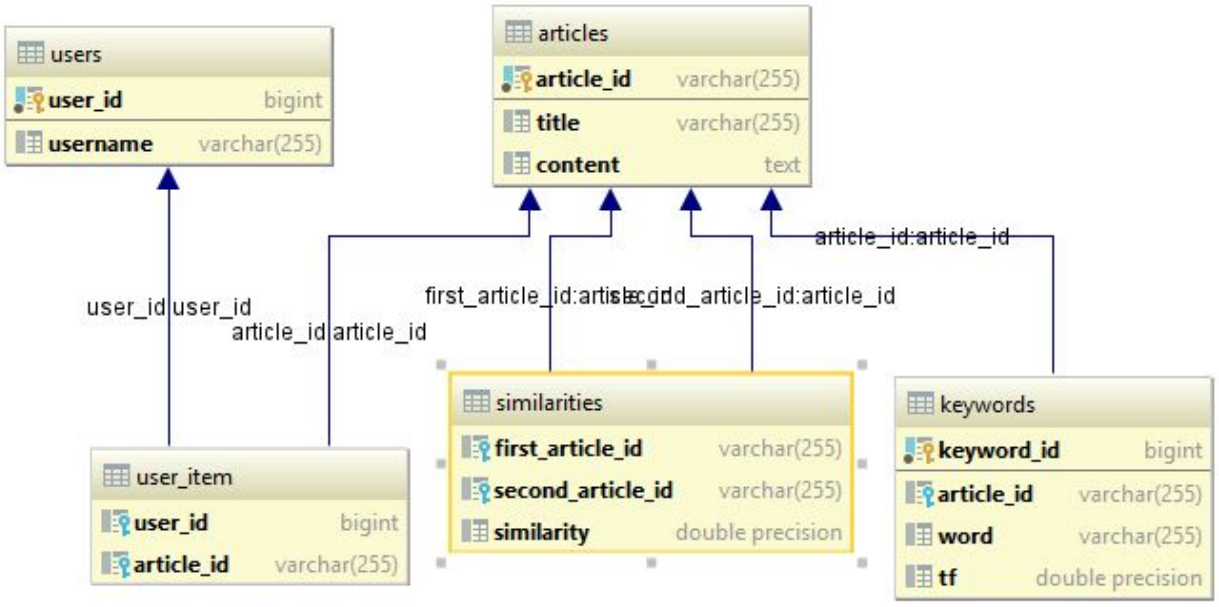


Рис. 3 – Диаграмма базы данных

Таблицы users и articles содержат информацию о пользователях и статьях.

Таблица user_item отображает список избранных документов пользователя.

В таблице keywords находится информация о всех ключевых словах конкретной статьи.

Таблица similarities описывает меру схожести между двумя статьями.

### 3.2.2 Описание Веб–интерфейса

Веб–интерфейс состоит из нескольких основных компонентов:

1) Главная страница.
2) Компонент загрузки данных.
3) Страница поиска статьи.
4) Страница персональных рекомендаций.

Главная страница.


Рис. 4 – Главная страница

На главной странице можно осуществить переход к другим компонентам, которые предоставляют основной функционал:

Компонент загрузки данных.

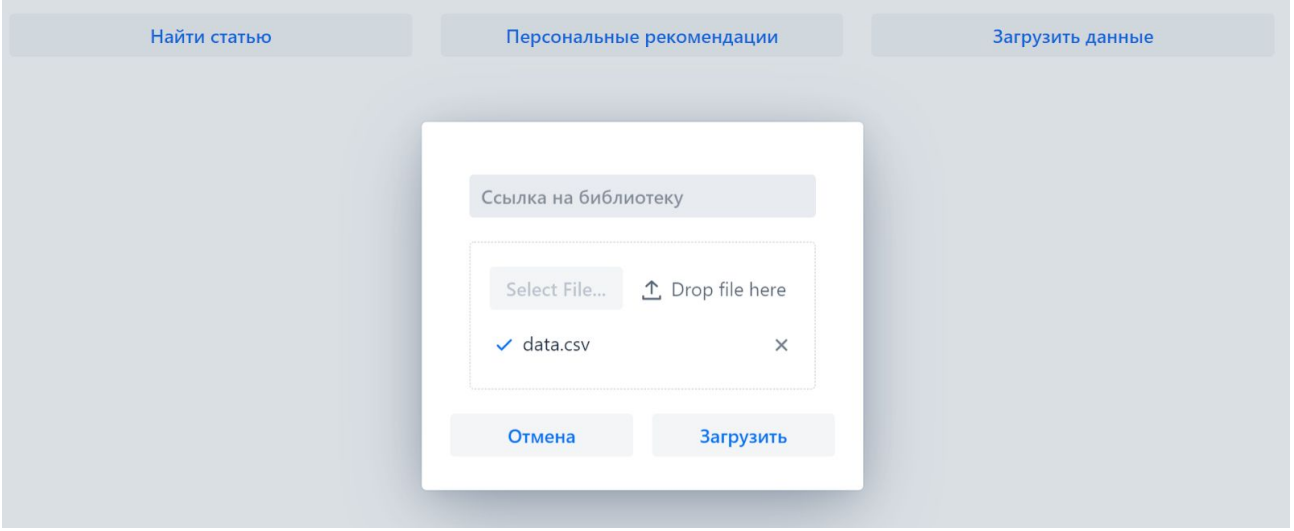


Рис. 5 – Компонент загрузки данных

Прежде чем формировать рекомендации, необходимо загрузить данные о пользователях и статьях.

В данном компоненте можно указать ссылку на цифровую библиотеку и выбрать файл, который содержит информацию о пользовательских оценках и имеет следующий вид:

| username | article_url |
|---|---|
| Akson87 | 455624 |
| … | … |

Таблица 4 – Входные данные

где article_url – ссылка на статью в библиотеке (например: ссылка на статью "https://habr.com/ru/post/455624/", тогда article_url будет равно "455624").

Или же, если цифровая библиотека не подразумевает пользовательские оценки, загружаемый файл может состоять только из ссылок на статьи.

После нажатия на кнопку "Загрузить" сервис начинает обрабатывать данные следующим образом: для каждой статьи на основе меры TF выбираются ключевые слова в количестве, равном KEYWORDS_LIMIT, которые представляют собой вектор документа; из базы данных достаются все ключевые слова и вычисляется значения косинусной меры между вновь добавляемой статьей и уже добавленными статьями; затем эти значения сортируются и в таблицу similarities добавляется информация только о N статьях с наибольшими значениями косинусной меры; в конце ключевые слова добавляемой статьи записываются в таблицу keywords.

Если в файле указаны пользователи, то эти данные записываются в таблицу user_item.

Страница поиска статьи.

После обработки загруженных данных можно перейти к компоненту поиска статьи.

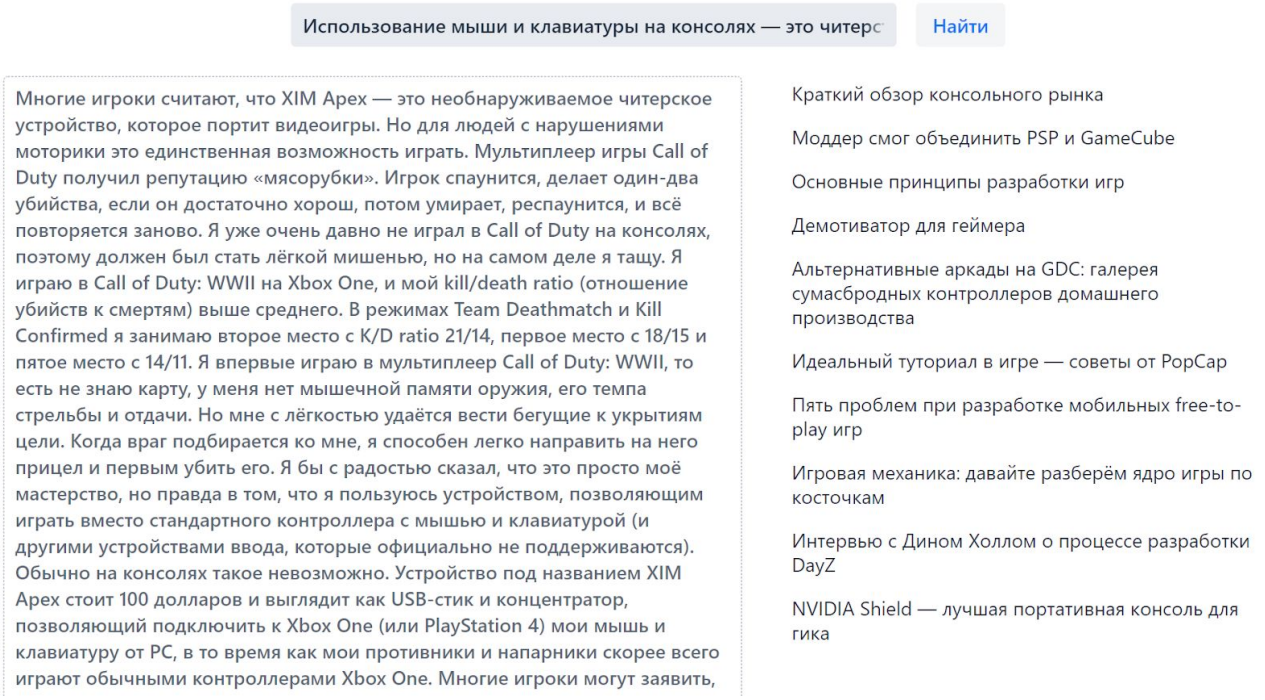

Рис. 6 – Страница поиска статьи

Здесь можно ввести название статьи, которую мы загрузили, и в ответ на запрос выводится содержание искомой статьи, а также список из N статей, похожих на данную (из таблицы similarities сразу достаются нужные статьи).

Страница персональных рекомендаций.

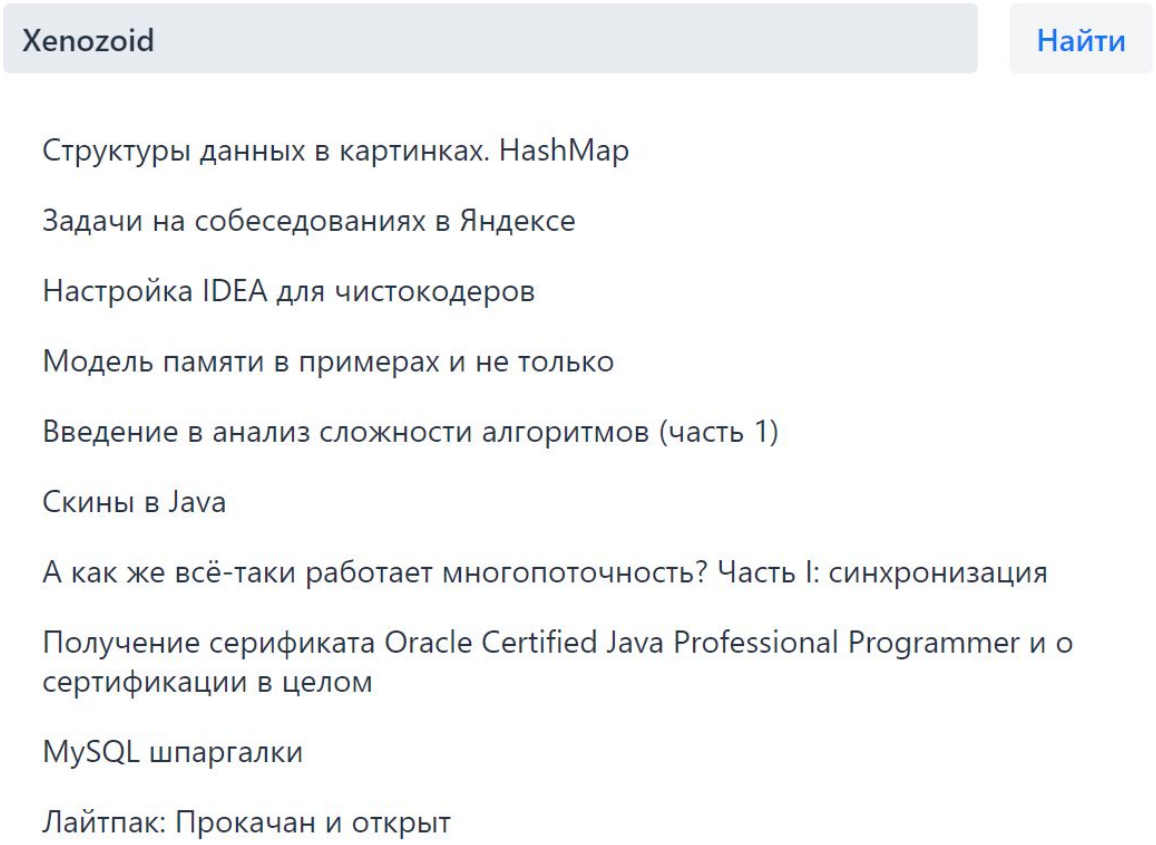


Рис. 7 – Страница персональных рекомендаций для пользователя "Xenozoid"

Если были предоставлены данные о пользовательских оценках, то в компоненте "Персональные рекомендации" может быть отображен список из N рекомендаций, который формируется на основе K ближайших соседей для конкретного пользователя, если у данного пользователя количество оценок больше или равно CV.

# 4. ТЕСТИРОВАНИЕ НА РЕАЛЬНЫХ ДАННЫХ

## 4.1 Тестовый набор данных

В реализованном сервисе используются бинарные оценки пользователей. Такая модель применима к избранному на сайте "Хабр". Алгоритм сбора данных: для случайных постов собирались пользователи, которые оставили комментарий. У каждого пользователя есть закладка "Избранное", которую можно распарсить и получить данные о статьях, которые понравились пользователю. Таким образом, тестовый набор данных содержит информацию о 19747 пользователях, 20826 статьях и 492694 пользовательских оценках.

## 4.2 Результаты тестов

Для тестирования контентной фильтрации была выбрана случайная статья:

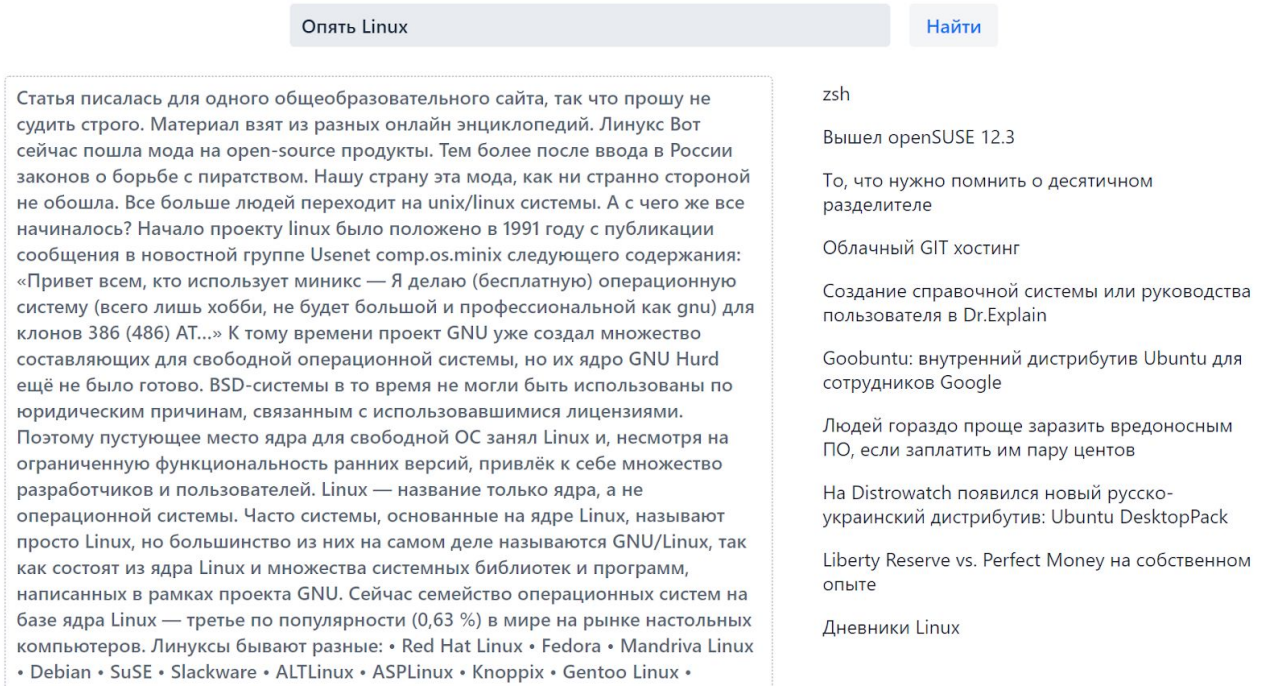


Рис. 8 – Список похожих статей на статью "Опять Linux"

Из полученных результатов видно, что контентная фильтрация сильно промахивается, что логично, поскольку была использована только мера TF, учитывающая только самые популярные слова в статье.

Для тестирования коллаборативной фильтрации случайным образом были выбраны два пользователя с количеством оценок более или равным значению CV.

Для пользователя с именем "XvonabuR", который оценил следующие статьи:

| | title |
|---|---|
| 1 | Поражение роботов: взлеты и падения высокочастотного трейдинга (Часть 2) |
| 2 | How-to: Торговля фьючерсами на фондовом рынке |
| 3 | Тренды e-commerce: Зарубежные игроки приходят в Россию |
| 4 | Физики о наклонной беговой дорожке |
| 5 | Биржевой софт: Инструменты для создания торговых роботов |
| 6 | Гаджеты для онлайн-трейдинга |
| 7 | Почем опиум для народа? Как устроен FOREX и нужен ли он. (Часть II) |
| 8 | Шоу Звук #7 — Подкаст об аудиотехнике, комплектующих, форматах и технологиях |
| 9 | Пять UX-приемов для создания биржевого софта |
| 10 | 4 способа превратить «Нет» в «Да» |
| 11 | Как устроен Forex и нужен ли он |
| 12 | PHDays IV: Открыта регистрация на онлайн-конкурсы «Конкурентная разведка» и Hash Runner |
| 13 | Инструментарий фондового рынка: что такое фьючерсы и как они работают |
| 14 | Бизнес и Большие данные: лаборатория FABERNOVEL |
| 15 | Инструментарий фондового рынка: что такое опционы, и как они работают |
| 16 | Поражение роботов: взлеты и падения высокочастотного трейдинга (Часть 1) |
| 17 | Метрика #25 — Подкаст о технологиях, продуктах и сервисах из мира ИТ |
| 18 | Тенденции и перспективы алгоритмической торговли в России |
| 19 | Быстрый старт на фондовом рынке: 10 шагов |
| 20 | Новое на PHDays IV: как взломать Gmail, шпионить через телевизор, подслушать любой телеф… |
| 21 | How-to: как выбрать язык программирования для создания торгового робота |
| 22 | Как именно на нас работают социальные медиа |
| 23 | Электронная коммерция в России: Взгляд с Запада |
| 24 | Будущее API в 2014 году |

Рис. 9 – Статьи, которые оценил "XvonabuR"

Список рекомендаций выглядит следующим образом.

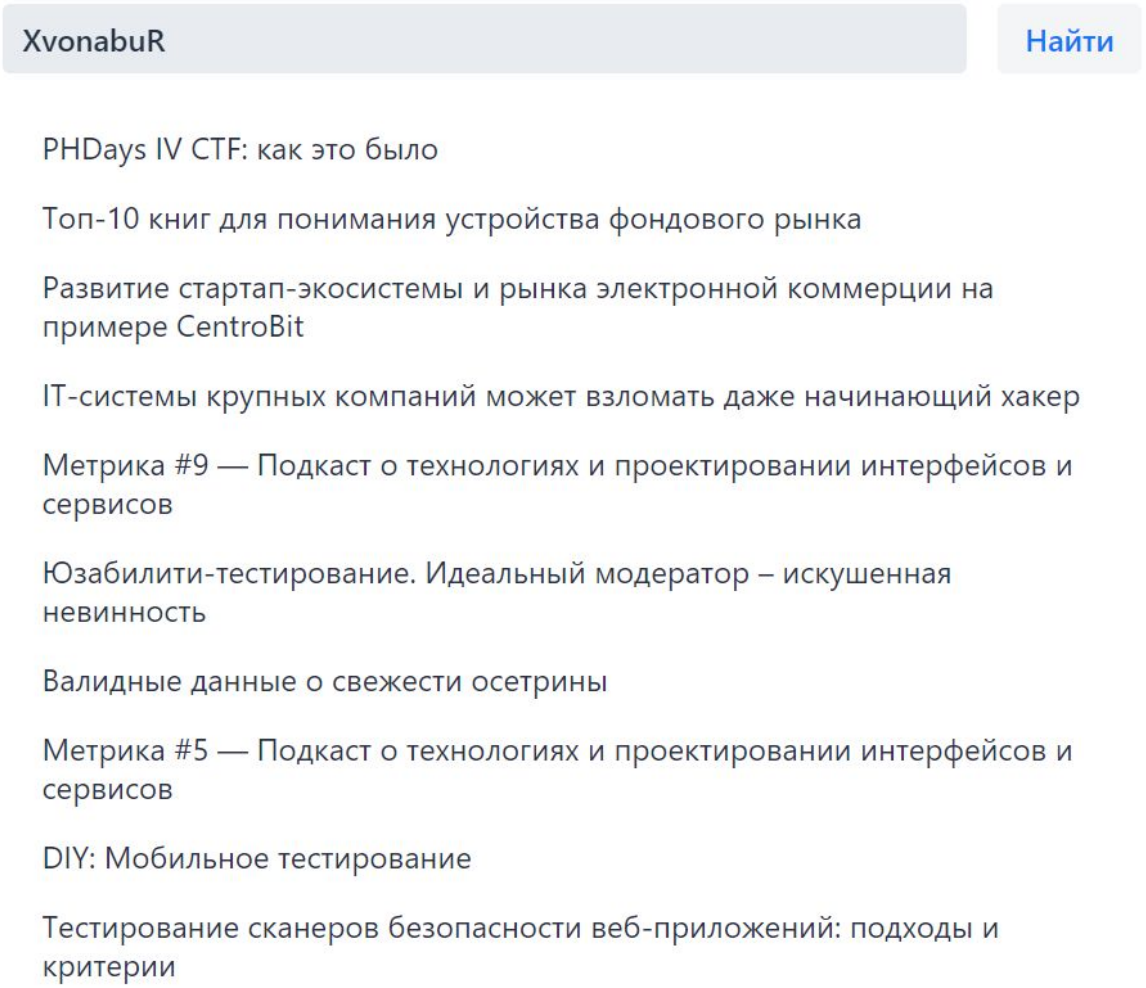


Рис. 10 – Рекомендации для "XvonabuR"

Видно, что данный пользователь интересуется темами "коммерция" и "торговля", и рекомендательный сервис в автоматическом режиме порекомендовал ему статьи, которые нравятся похожим пользователям.

Для пользователя с именем "SadJoker", который оценил следующие статьи:

| | title |
|---|---|
| 1 | Рецепт нагрузочного тестирования на JMeter |
| 2 | Tips & tricks for MySQL Developers. Работа с SQL |
| 3 | JDK concurrent package |
| 4 | Реализация Singleton в JAVA |
| 5 | Почему стоит изучить Clojure? |
| 6 | Книги для тимлидов и руководителей проектов |
| 7 | Gradle и решение задач автоматизации |
| 8 | Concurrency: 6 способов жить с shared state |
| 9 | У вас здесь ошибка… или о практике инспекций кода в мобильной разработке |
| 10 | Мой отчет по годовому изучению платформы Java EE |
| 11 | Делаем релизы с помощью Maven в Java |
| 12 | Введение в JMeter |
| 13 | Модель памяти в примерах и не только |
| 14 | Впечатления от работы с Play! Framework 2.1 + Java |
| 15 | Учимся проектировать на основе предметной области (DDD: Domain Driven Design) |
| 16 | Десять смертных грехов в оценке трудоёмкости разработки программного обеспечения |
| 17 | Захабренный договор на разработку сайта, дизайна, софта. Версия 1.1 |
| 18 | Введение в анализ сложности алгоритмов (часть 4) |
| 19 | Улучшаем опыт взаимодействия с формами |
| 20 | Вся правда о Chrome (и не только) Web Inspector. Часть 1 |

Рис. 11 – Статьи, которые оценил "SadJoker"

Список рекомендаций выглядит следующим образом.

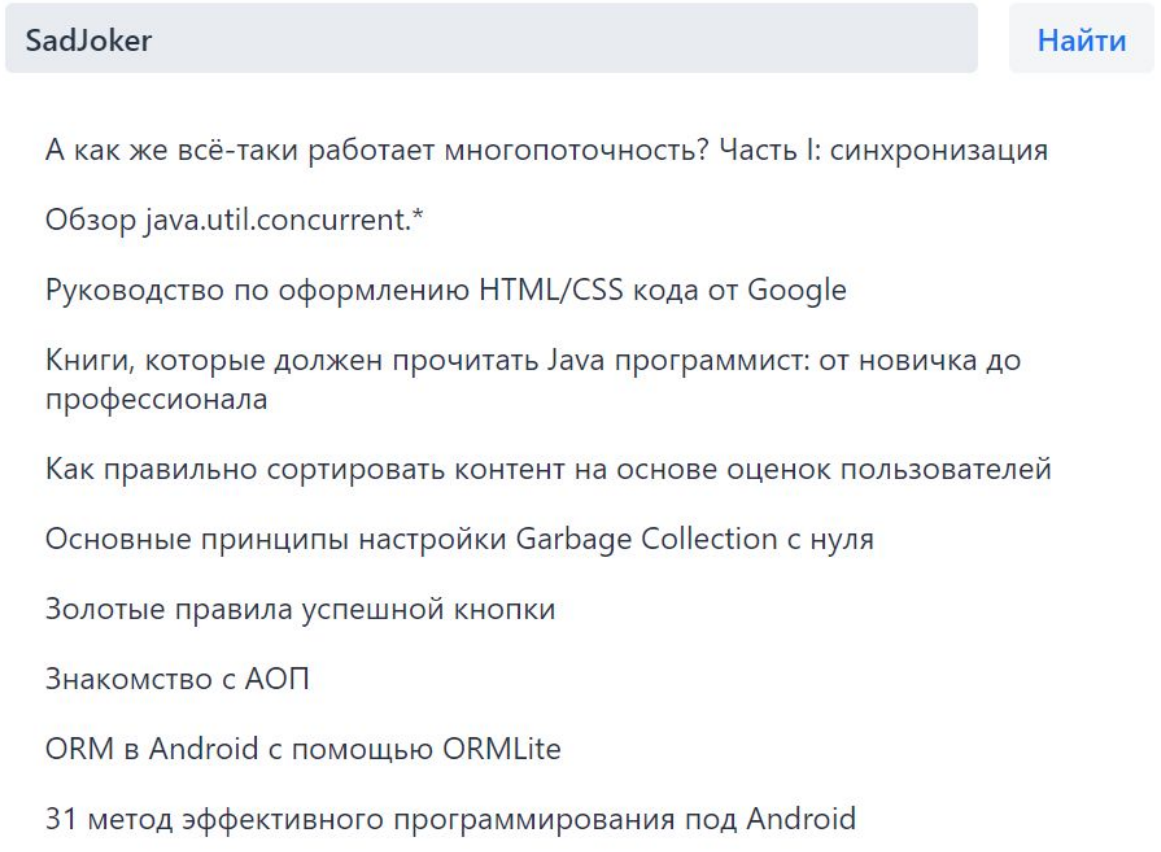

Рис. 12 – Рекомендации для "SadJoker"

Данный пользователь интересуется темой "Java" и, соответственно, ему рекомендуются статьи, которые оценили пользователи, интересующиеся данной темой.

### 4.3 Тестирование производительности

При том же наборе тестовых данных (20826 статей, 309061 ключевых слов, 207660 записей в таблице similarities, 19747 пользователей и 492694 пользовательских оценок) были сделаны следующие замеры производительности.

| Действие | Время (мс) |
|---|---|
| Поиск статьи и формирование списка похожих статей | 197 |
| Добавление новой статьи к уже имеющимся | 6847 |
| Формирование матрицы user–user, с учетом того, что в эту матрицу попали 9385 пользователей (количество пользователей, у которых количество оценок более или равно CV) | 168453 |
| Формирование списка рекомендаций для пользователя | 144 |

Таблица 5 – Результаты теста производительности

На основании результатов тестов производительности можно сделать вывод, что формирование матрицы user–user является очень трудоемким процессом. Поэтому в целях оптимизации было решено формировать матрицу не при каждом запросе пользователя, а один раз в несколько часов и хранить эту матрицу в памяти. При таком подходе необходимо сделать следующее допущение: предпочтения пользователей не сильно меняются в течение этого

промежутка времени. Но если пользователей миллион, то хранить такую матрицу в памяти не является оптимальным решением. В реализованном сервисе эти данные хранятся в виде пар ключ–значение, где ключом выступает пользователь, а значением – список из K ближайших соседей для данного пользователя. Таким образом, в памяти расположена структура размеров K × n (где n – количество пользователей, которые оценили более или равно CV объектов), а не матрица размеров n × n. При запросе пользователя на получение персональных рекомендаций данные извлекаются из памяти и формирование рекомендаций осуществляется за константное время.

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы было сделано следующее:

1) Исследованы виды рекомендательных систем и принципы их построения.
2) Изучены основные алгоритмы, применяющиеся в контентной и коллаборативной фильтрациях.
3) Спроектирован и реализован рекомендательный сервис на основе гибридного подхода, который включает в себя контентную и коллаборативную фильтрации.
4) Разработано веб–приложение с внедрением в него реализованного рекомендательного сервиса для научных цифровых библиотек.

Поставленная цель – разработка рекомендательного сервиса для научных цифровых библиотек – была достигнута.

Данная работа размещена на gitlab – http://gititis.kpfu.ru/Mustaev/rs.

Разработанный сервис может быть использован как цифровыми научными библиотеками для формирования персональных рекомендаций пользователям и выдачи списка похожих статей на основе данных внутри базы конкретной библиотеки, так и любыми другими пользователями для выдачи только списка похожих статей на ту, которую он просматривает в данный момент, поскольку обычно научные библиотеки не предоставляют свободный доступ к своим базам данных.

В дальнейшем может быть сделано следующее:

1) Автоматический поиск и добавление статей при вводе ссылки на научную цифровую библиотеку.
2) Повышенная интеграция сервиса с цифровыми библиотеками.
3) Улучшение точности контентной фильтрации.

# ГЛОССАРИЙ

- API – Application Programming Interface, набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.
- IDF (inverse document frequency – обратная частота документа) – инверсия частоты, с которой некоторое слово встречается в документах коллекции.
- Jsoup – это библиотека для Java с открытым исходным кодом, предназначенная для анализа, извлечения и управления данными, хранящимися в документах HTML.
- Lombok – проект по добавлению дополнительной функциональности в Java с помощью изменения исходного кода перед компиляцией.
- PostgreSQL – свободная объектно–реляционная система управления базами данных.
- Spring Boot – проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web–приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.
- TF (term frequency – частота слова) – отношение числа вхождения некоторого слова к общему количеству слов документа.
- TF–IDF (TF – term frequency, IDF – inverse document frequency) – статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса.
- Лемматизация – процесс приведения словоформы к лемме – её нормальной (словарной) форме.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Анатомия рекомендательных систем. Часть первая [Электронный ресурс] / crazyhatter. – 2018. – Режим доступа: https://habr.com/ru/company/lanit/blog/420499/ (дата обращения: 25.02.2019).
2. About The Music Genome Project [Electronic resource] // Pandora Internet Radio / Oakland: owner Pandora Media, Inc. – 2017. – URL: http://pandora.com/about/mgp (дата обращения: 14.03.2019).
3. Recommender Systems Handbook [Text] / Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. – N.Y.: Springer, 2011. – 842 p.
4. Рекомендательные системы: user–based и item–based [Электронный ресурс] / snikolenko. – 2012. – Режим доступа: https://habr.com/ru/company/surfingbird/blog/139518/ (дата обращения: 19.03.2019).
5. Рекомендательные системы: You can (not) advise [Электронный ресурс] / overmes. – 2013. – Режим доступа: https://habr.com/ru/post/176549/ (дата обращения: 28.03.2019).
6. Linden G., Smith B., York J., Amazon.com recommendations: item–to–item collaborative filtering [Text] / Linden G., Smith B., York J. // Internet Computing – IEEE 7 2003 – P. 76–80.
7. Using collaborative filtering to weave an information Tapestry [Text] / Goldberg D., Nichols D., Oki B. M., Terry D. // Special issue on information filtering. – 1992. – Vol. 35, Issue 12. – P. 61–70.
8. Melville P., Mooney R.J., Nagarajan R. Content–boosted collaborative filtering for improved recommendations / Melville P., Mooney R.J., Nagarajan R. // in Proceedings of the National Conference on Artificial Intelligence. – 2002. – P. 187–192.

9. Рекомендательные системы. Часть 1. Введение в подходы и алгоритмы [Электронный ресурс] / М. Джонс. – 2014. – Режим доступа: https://www.ibm.com/developerworks/ru/library/os–recommender1/index.html (дата обращения: 22.02.2019).
10. Заболеева–Зотова, А.В. Латентный семантический анализ: новые решения в Internet [Текст] / А.В. Заболеева–Зотова. – Москва: Информационные технологии, 2001. – 22 с.
11. Якобсон А. Унифицированный процесс разработки программного обеспечения: пер. с англ. В. Горбунков [Текст] / А. Якобсон, Г. Буч, Дж. Рамбо – Спб.: Питер, 2002. – 496 с.