Федеральное государственное автономное образовательное учреждение высшего профессионального образования

Национальный исследовательский университет

Высшая школа экономики

НИУ ВШЭ-Нижний Новгород

Курсовая работа

На тему

**Разработка рекомендательной системы фильмов на основе анализа пользовательских оценок**

Студент группы 13ПИ

Цыпляев Александр Сергеевич

Оценка за работу\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель:

Манагина Т.А., преподаватель

Нижний Новгород

2015

Оглавление

[Часть 1.Введение 3](#_Toc420053984)

[Часть 2. Теория и практика рекомендательной системы. 5](#_Toc420053985)

[**Глава 1. Теория рекомендательных систем и алгоритмы рекомендаций.** 5](#_Toc420053986)

[**История появления** 5](#_Toc420053987)

[**Типы рекомендательных систем** 5](#_Toc420053988)

[**Алгоритм User/User:** 7](#_Toc420053989)

[**Глава 2. Практическая реализация рекомендательной системы** 9](#_Toc420053990)

[**Требования** 9](#_Toc420053991)

[**Межпроцессное взаимодействие** 9](#_Toc420053992)

[**Логика работы сервера и описание важнейших функций** 9](#_Toc420053993)

[**Логика работы клиента и описание важнейших функций** 13](#_Toc420053994)

[**Диаграммы классов и взаимодействий.** 16](#_Toc420053995)

[Часть 3. Выводы, направления дальнейшего развития 21](#_Toc420053996)

[Список литературы 22](#_Toc420053997)

# Часть 1.Введение

**Рекомендательные системы** - это определенный вид программ, которые, исходя из некоторого набора данных о пользователе, выдают информацию об объектах (в зависимости от вида системы это могут быть как фильмы, так и музыка, книги, товары обихода, сайты и др.), которые могут быть интересны пользователю.

**Цель работы**:

Разработка приложения – рекомендательной системы, которая на основании введенных пользователем оценок предлагает пользователю определенный набор фильмов.

**Задачи работы:**

1. Изучение теории рекомендательных систем
2. Изучение основных алгоритмов рекомендательных систем
3. Разработка приложения на языке С++ на основе изученных алгоритмов

**Актуальность:**

Актуальность данной темы не вызывает сомнений, так как в современном мире пользователи интернета сталкиваются с системами рекомендаций практически каждодневно. Очевидно, что множество сайтов пользуется данными программами, чтобы предложить пользователю подходящий контент, будь то музыка, фильмы, сайты или книги. Несомненно, множество известных интернет-магазинов, таких как Ozon или Amazon, интернет-поисковики (Google, Yandex, etc), непосредственно сами рекомендательные сайты (Last.fm, Imhonet) работают с использованием данных систем. Поэтому тема о системах рекомендаций не оставляет сомнений о своей актуальности.

В процессе изучения данной темы были использованы несколько ресурсов для подготовки. Среди них и книжные издания (Programming collective intelligence), так и ресурсы интернета, содержащие множество полезных статей на данную тему (Habrahabr и Wikipedia – самые яркие примеры). Полный список используемых материалов находится

Было изучено множество алгоритмов, и выбраны самые оптимальные из них. Описание выбранных алгоритмов находится в главах 2(теоретической) и 3(практической) работы.

# Часть 2. Теория и практика рекомендательной системы.

## **Глава 1. Теория рекомендательных систем и алгоритмы рекомендаций.**

### **История появления**

Рекомендательные системы впервые появились в интернете около 20 лет назад. Однако настоящий подъем в этой области случился в 2006-2011, когда компания Netflix организовала соревнование Netflix prize. Netflix давала в прокат не цифровые копии, а рассылала VHS-кассеты и DVD. Для них было очень важно повысить качество рекомендаций. Чем лучше Netflix рекомендует своим пользователям фильмы, тем больше фильмов они берут в прокат. В 2006 году они запустили соревнование Netflix Prize. Они выложили в открытый доступ собранные данные: около 100 миллионов оценок по пятибалльной шкале с указанием ID проставивших их пользователей. Участники соревнования должны были как можно лучше предугадывать, какую оценку поставит определенному фильму тот или иной пользователь. Качество предсказания измерялось при помощи метрики RMSE (среднеквадратичное отклонение). У Netflix уже был алгоритм, который предсказывал оценки пользователей с качеством 0.9514 по метрике RMSE. Задача была улучшить предсказание хотя бы на 10% — до 0.8563. Победителю был обещан приз в $1000000. Соревнование длилось примерно три года. За первый год качество улучшили на 7%, дальше все немного замедлилось. Но в конце две команды с разницей в 20 минут прислали свои решения, каждое из которых проходило порог в 10%. Данное соревнование подтолкнуло развитие рекомендательных систем и способствовало значительному улучшению качества рекомендаций и, как следствие, их распространению.

### **Типы рекомендательных систем**

Существует несколько типов рекомендательных систем, которые используют различные подходы при рекомендации пользователю объектов.

#### Content based systems (системы, основанные на содержимом):

1) В данных системах пользователю рекомендуются объекты, похожие на те, которые этот пользователь уже купил или посмотрел.

2) Объекты сравниваются на основе содержимого этих объектов.

3) Сильная зависимость от предметной области, вследствие чего полезность данных рекомендаций ограничена. Например, если пользователь интересуется определенным типом товаров, ему будут предлагать товары, похожие на этот, в то время как другие товары останутся нерекомендованными.

Основные шаги в данной системе:

1) Проанализировать контент предметов и составить набор его критериев.

2) Узнать какие критерии нравится пользователю

3) Сопоставить эти данные и получить рекомендации.

Критерии объединяют пользователей и предметы в единой системе координат, если точка пользователя и предмета рядом, то вероятно предмет понравится пользователю.   
Одним из самых простых видов этой системы является жанровая система.

#### Collaborative filtering (коллаборативная фильтрация):

1) Основой рекомендации служит история оценок пользователей к определенным товарам.

2) Такой подход более универсален, вследствие чего рекомендации более разнообразны.

3) Имеет место быть проблема холодного старта, когда новые товары остаются невостребованными (и наоборот, новым пользователям предлагается весьма ограниченный набор наиболее распространенных объектов)

Здесь существует множество алгоритмов, но наиболее популярных 3:

1) User/User

2) Item/Item

3) SVD

Исходя из названий, можно понять, что алгоритм User/User находит наиболее похожих пользователей и на данной основе формирует возможную оценку для конкретного пользователя, а алгоритм Item/Item – напротив, ищет товары, наиболее схожие с теми, которые пользователю понравились.

Алгоритм данных подходов можно коротко описать так:

1) Найти, насколько другие пользователи (продукты) в базе данных похожи на данного пользователя (продукт).

2) По оценкам других пользователей (продуктов) предсказать, какую оценку даст данный пользователь данному продукту, учитывая с большим весом тех пользователей (продукты), которые больше похожи на данного пользователя (продукт).

Алгоритм SVD работает на другой основе. В качестве исходных данных для этого алгоритма имеется матрица пользователей и объектов.

Основой алгоритма является сингулярное разложение матрицы *R*:

image

Если *R* – матрица большого размера image, но малого ранга *f* (в частности, разреженные матрицы часто бывают малого ранга), её можно разложить в произведение матрицы image и матрицы image, тем самым сократив число параметров, с  *NM* до (*N*+*M*)*f*

В случае рекомендательных систем получается, что представляется каждый пользователя вектором из *f* факторов image и представляется каждый продукт вектором из *f* факторов image, а потом, чтобы предсказать рейтинг пользователя *i* товару *j*, берём их скалярное произведение image.

Можно сказать, что вектор факторов пользователя показывает, насколько пользователю нравится или не нравится тот или иной фактор, а вектор факторов продукта показывает, насколько тот или иной фактор в продукте выражен.

SVD очень широко употребляется в машинном обучении, где для выражения погрешностей используют различные математические метрики, например, уже упомянутое среднеквадратичное отклонение.

#### Гибридные системы:

Также существуют системы, объединяющие предыдущие типы и использующие различные алгоритмы. В данных системах основой является удельный вес определенного алгоритма, с помощью которого формируется итоговая оценка для предметов, которые рекомендуются данному пользователю.

В своей работе я реализовал коллаборативную систему рекомендации, основанную на алгоритме User/User.

### **Алгоритм User/User:**

Пусть имеется система, включающая пользователей, фильмы и оценки определенного пользователя к фильмам.

Система представляет собой комбинации (U,F,M), где U –пользователь (зачастую целое число), F – фильм (название, либо идентификатор), M – оценка пользователя фильму по числовой шкале (в зависимости от выбора, пятибалльная или десятибалльная).

Для построения рекомендации необходимо получить некий коэффициент корреляции пользователя с другими пользователями, основываясь на его оценках (коэффициенты корреляции описаны далее).

Далее, пользуясь данными коэффициентами, выстраиваем линейку наиболее похожих на данного пользователя пользователей. Следующим шагом, определяем фильмы, которые не смотрел данный пользователь, и оценку к ним на основе долей корреляции и оценок других пользователей.

Как можно понять, основой данного алгоритма является понятие и использование коэффициента корреляции.

Корреляция или корреляционная зависимость - это статистическая взаимосвязь двух или более случайных величин (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми). При этом изменения значений одной или нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин.

Существует множество корреляционных коэффициентов, в данной работе был описан коэффициент линейной корреляции Пирсона, но основным используемым коэффициентом было Евклидово расстояние (основано на геометрической дистанции двух величин). Реализация алгоритма вычисления данных коэффициентов описана в практической части работы.

## **Глава 2. Практическая реализация рекомендательной системы**

Для реализации алгоритмов рекомендательной системы была выбрана операционная система Windows, язык программирования C++ и среда разработки Microsoft Visual Studio.

### **Требования**

Прежде чем говорить о непосредственной реализации системы рекомендаций на практике, следует упомянуть о требованиях к данной системе. Одним из основных требований было требование к клиент-серверной концепции приложения. Таким образом, необходимо было имитировать ситуацию, когда один процесс передает информацию другому, тот ее анализирует и возвращает некий набор данных на ее основании.

В результате, была смоделирована ситуация, когда клиент и сервер являются отдельными автономными процессами одной системы.

### **Межпроцессное взаимодействие**

Существует множество различных средств межпроцессного взаимодействия, причем как в среде Windows, так и в среде Unix.

В данной работе основным средством межпроцессного взаимодействия между клиентом и сервером является регулятор поочередного доступа к ресурсу, используемому как клиентом, так и сервером. В роли данного регулятора выступает объект Mutex, который позволяет ограничить доступ к ресурсу одному процессу в то время, когда ресурс использует другой процесс.

В качестве данного ресурса используется простой файл, куда записывает свои данные и клиент, и сервер.

### **Логика работы сервера и описание важнейших функций**

Сервер реализован как консольное приложение в среде VS2013.

Так как предполагается, что сервер единственный, то приложение-сервер реализовано с помощью одного класса Server, который использует внутренние методы для своей работы.

Основой сервера является система, описанная в алгоритме User/User. Данную систему можно рассматривать как базу данных, хотя ее реализация отлична от стандартной БД. После запуска приложение выгружает информацию о пользователях, фильмах и оценках в объект std::map, что позволяет работать с основными данными непосредственно в оперативной памяти, получив к ним доступ через выгруженный в память объект std::map. Данная логика реализована в связи с отсутствием требований по занимаемой памяти, а также в интересах быстродействия программы.

Выгрузка базы в память реализована в функции Strver::getbase()

while (!myfile.eof())

{

getline(myfile, line);

while ((pos = line.find(delimiter)) != std::string::npos)

{

token = line.substr(0, pos);

tokens.push\_back(token);

line.erase(0, pos + delimiter.length());

}

while (std::stoi(tokens[0]) == current && (!myfile.eof())){

base.insert(std::pair<std::string, double>(tokens[1], std::stod(tokens[2])));

tokens.clear();

getline(myfile, line)

while ((pos = line.find(delimiter)) != std::string::npos)

{

token = line.substr(0, pos);

tokens.push\_back(token);

line.erase(0, pos + delimiter.length());

}

if (std::stoi(tokens[0]) != current)

{

finals.emplace(current, base);

base.clear();

base.insert(std::pair<std::string, double>(tokens[1], std::stod(tokens[2]))); current=std::stoi(tokens[0]);

break;

}

}

}

Выше представлена основная часть функции getbase. При помощи данной части кода сервер читает файл с базой данных и выгружает его в объект std::map.

Также, сервер создает объект Mutex, который ответственен за регулирование доступа к ресурсам, используемым и клиентом, и сервером.

System::Threading::Mutex^ m = gcnew System::Threading::Mutex(false, "MyMutex");

Захватив Mutex после того, как клиент отправил свои данные, сервер добавляет информацию о выставленных пользователем оценках в базу данных. Далее, сервер считает коэффициенты корреляции данного пользователя с другими, находит положительную корреляцию и на основе рейтинга коэффициентов корреляции строит предположения о фильмах, которые могут понравиться тому пользователю, который освободил объект Mutex.

За расчет корреляции с различными пользователями, составление рейтинга корреляций и определение наиболее схожих с данным пользователем отвечает функция Server::topMatches. В процессе работы данной функции используются объекты std::vector (добавление и сортировка коэффициентов корреляции) и std::map (формирование объекта, включающего пользователей с наиболее высоким коэффициентом корреляции относительно данного пользователя).

Как было сказано выше в работе, необходимо описание функций, отвечающих за расчет коэффициентов, так как именно на них строится алгоритм получения рекомендаций.

Основной функцией для получения коэффициента является функция Server::sim\_distance().

static int count = 1;

vector<string> si;

double sum = 0.0;

for (auto item = prefs[person1].begin(); item != prefs[person1].end(); ++item)

{

for (auto item2 = prefs[person2].begin(); item2 != prefs[person2].end(); ++item2)

{

if (item2->first.compare(item->first) == 0)

{

si.push\_back(item2->first);

sum += pow(item->second - item2->second, 2);

break;

}

}

}

if (si.size() <= 2)

{

return 0;

}

return 1 / sqrt((1 + sum));

Выше представлен код функции sim\_distance. Как можно заметить, функция, проходя по проставленным баллам, находит совпадения в фильмах и добавляет в переменную sum возведенную в квадрат разность оценок пользователей. Соответственно, чем более одинаковые оценки ставят пользователи фильмам, тем больше получится итоговый коэффициент. Также, необходимо упомянуть, что коэффициент будет считаться равным нулю, если пользователи имеют менее трех общих фильмов, которые они оценили. Это сделано для предотвращения высоких коэффициентов с пользователями, которые имеют слишком мало общего с данным пользователем.

Код вычисления альтернативного коэффициента (линейный коэффициент корреляции Пирсона) представлен в функции Server::sim\_pearson()

За непосредственное формирование списка рекомендации и отправку ее клиенту отвечает функция Server::GetRecommendations

for (auto i = prefs.begin(); i != prefs.end(); ++i)

{

double sim = sim\_distance(prefs, person, i->first);

for (auto y = i->second.begin(); y != i->second.end(); ++y)

{

auto item = prefs[person].find(y->first);

if (item == prefs[person].end() && y->second >= 4.0)

{

if (totals.find(y->first) == totals.end())

{

totals.emplace(y->first, 0.0);

simSums.emplace(y->first, 0.0);

counter += 1;

if (counter >= 10)

flag = true;

}

totals[y->first] += y->second \* sim;

simSums[y->first] += sim;

if (flag == true)

break;

}

}

if (flag == true)

break;

}

Выше представлен основной цикл функции GetRecommendations. Цикл итерируется по фильмам, которые пользователь не смотрел и добавляет их в список рекомендации, если оценка к фильму больше или равна четырем. Данное правило работоспособно в силу того, что первыми в списке идут комбинации людей, фильмов и оценок, где люди отсортированы в порядке уменьшения коэффициентов корреляции с ними.

В конце работы функция считает предполагаемую оценку фильму пользователя.

for (auto z = totals.begin(); z != totals.end(); ++z)

{

rankings.emplace(z->first, z->second / simSums[z->first]);

}

Далее, функция записывает рекомендации в файл и отдает объект Mutex, который захватывает клиент и отображает рекомендации пользователю.

Следующим шагом, сервер ожидает освобождения объекта Mutex, чтобы обработать новые данные и выдать уже новые рекомендации.

Следует упомянуть, что использование всех этих функций скрыто в классе сервера, а для того, чтобы запустить сервер, нужно только создать его экземпляр и вызвать функцию Server::execute(), которая отвечает за работу сервера и влечет за собой другие функции.

### **Логика работы клиента и описание важнейших функций**

Клиент представляет собой приложение WinForms, что было обговорено в требованиях. Имея довольно простой интерфейс, клиент обеспечивает возможность доступа к приложению по логину/паролю, запоминает прошлые рекомендации и начинает работу с предложения оценить набор фильмов, чтобы получить новый.



Рис.1 Мимималистичный интерфейс

На рисунке 1 изображен минимальный интерфейс приложения. Для того, чтобы получить доступ к приложению, необходимо ввести логин и пароль. При этом система, при правильном сочетании логина и пароля, отобразит интерфейс для данного пользователя именно с тем набором фильмов, что были рекомендованы ему для просмотра. Если же пользователь новый, то есть его логин не совпадает ни с одним из существующих, то ему будет представлен интерфейс с стандартным набором фильмов, которые необходимо оценить.

Необходимость первичного набора фильмов связана с проблемой холодного старта в коллаборативных системах. Новому пользователю можно представить только наиболее распространенные фильмы, так как новый пользователь не имеет оценок, и, следовательно, не может иметь корреляции с другими пользователями.



Рис 2.Интерфейс оценивания фильмов

Следует описать процедуру регистрации и непосредственно работу клиента.

Процедура регистрации довольно проста – пользователь вводит логин и пароль, и получает доступ к системе в зависимости от того, правилен ли пароль и нов ли логин. Проверка осуществляется таким способом: в специальной папке проверяется существование файла, названного по имени пользователя, если такой файл имеется, то проверяется содержимое этого файла. Если первая строка файла совпадает с полем пароля, преобразованным по алгоритму хеширования md5. При совпадении, из файла читаются последние рекомендованные пользователю файлы. Если пароли не совпадает, пользователю предлагают ввести другую комбинацию логина и пароля. Если же данного файла не оказывается в папке, он создается, туда записывается пароль, зашифрованный алгоритмом md5, и идентификатор пользователя, полученный с помощью случайного выбора из большого спектра значений, а также начальный набор рекомендаций в виде фильм-оценка. Изначальные оценки стандартного набора полагаются равными единице.

Далее, клиент захватывает объект Mutex, созданный сервером. Пользователь, выставивший оценки, нажимает кнопку получения рекомендаций. Клиент отправляет данные в разделяемый файл, откуда после освобождения объекта Mutex их заберет сервер. Сервер совершит предписанную ему работу и вернет новый набор данных, которые клиент, получив, отобразит пользователю для оценивания.

if(!(line.compare(md5(msclr::interop::marshal\_as<std::string>(this->textBox12->Text)))))

Выше представлено сравнение хешированного поля пароля с 1 строкой файла, одноименного имени пользователя.

this->textBox1->Text=msclr::interop::marshal\_as<System::String ^>(tokens[1]);

this->comboBox1->Text=msclr::interop::marshal\_as<System::String ^>(tokens[2]);

this->textBox1->Visible=true;

this->comboBox1->Visible=true;

Фрагмент кода, отвечающего за отображение рекомендаций.

std::ofstream myfile(msclr::interop::marshal\_as<std::string>(this->textBox11->Text));

myfile<<md5(msclr::interop::marshal\_as<std::string>(this->textBox12->Text))<<"\n";

myfile<<UserId<<"\n";

myfile.close();

this->textBox1->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->comboBox1->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox1->Visible=true;

this->comboBox1->Visible=true;

this->textBox2->Text=msclr::interop::marshal\_as<System::String ^>("4");

this->comboBox2->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox2->Visible=true;

this->comboBox2->Visible=true;

this->textBox3->Text=msclr::interop::marshal\_as<System::String ^>("8");

this->comboBox3->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox3->Visible=true;

this->comboBox3->Visible=true;

this->textBox4->Text=msclr::interop::marshal\_as<System::String ^>("9");

this->comboBox4->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox4->Visible=true;

this->comboBox4->Visible=true;

this->textBox5->Text=msclr::interop::marshal\_as<System::String ^>("11");

this->comboBox5->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox5->Visible=true;

this->comboBox5->Visible=true;

this->textBox6->Text=msclr::interop::marshal\_as<System::String ^>("12");

this->comboBox6->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox6->Visible=true;

this->comboBox6->Visible=true;

this->textBox7->Text=msclr::interop::marshal\_as<System::String ^>("15");

this->comboBox7->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox7->Visible=true;

this->comboBox7->Visible=true;

this->textBox8->Text=msclr::interop::marshal\_as<System::String ^>("22");

this->comboBox8->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox8->Visible=true;

this->comboBox8->Visible=true;

this->textBox9->Text=msclr::interop::marshal\_as<System::String ^>("81");

this->comboBox9->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox9->Visible=true;

this->comboBox9->Visible=true;

this->textBox10->Text=msclr::interop::marshal\_as<System::String ^>("100");

this->comboBox10->Text=msclr::interop::marshal\_as<System::String ^>("1");

this->textBox10->Visible=true;

this->comboBox10->Visible=true;

this->GetRec->Visible=true;

this->button1->Visible=false;

this->textBox11->Visible=false;

this->textBox12->Visible=false;

this->label1->Visible=true;

Выше представлен фрагмент кода, отвечающий за отображение стандартного интерфейса и создание файла с информацией о пользователе.

Стоит сказать, что для работы с регистрацией и получением рекомендаций создан класс, предоставляющий интерфейс обработки данных, взаимодействующий с основным классом формы клиента. Поэтому в основном классе формы клиента вызываются методы класса-помощника Handler. Их взаимодействие показано в разделе UML-диаграмм.

### **Диаграммы классов и взаимодействий.**

Ниже представлены диаграммы классов и взаимодействий для клиента и сервера, которые призваны обеспечить максимальную прозрачность работы клиента и сервера.

Рис3. Диаграмма классов сервера

Как видно из диаграммы, сервер имеет 1 класс, в котором описаны методы, о которых было написано ранее. Сервер имеет единственную реализацию – ConcretServer, через которую и обеспечивается работа сервера.



Рис4. Диаграмма последовательностей сервера

Как можно заметить, клиент, в роли которого тут выступает функция main(), создает единственный экземпляр сервера ConcretServer. После этого вызывается функция execute(), которая создает последовательность вызовов getrecommendations()->topmaches()->sim\_distance().



Рис5. Диаграмма классов клиента

Как следует из описания клиента и диаграммы, в клиенте существуют 2 класса, один из которых содержит в себе экземпляр другого.

Класс формы вызывает перечисленные методы для отображения компонентов и изменения их значений, в то время как класс-помощник Handler помогает проверять необходимые значения, осуществляет получение данных от сервера и передачу данных серверу при помощи перечисленных методов (создание пользователя, получение рекомендаций, проверка существования пользователя, отправка данных серверу)



Рис6. Диаграмма последовательностей клиента

Как видно из диаграммы, класс Handler позволяет предельно упростить интерфейс вызовов. После создания нового экземпляра следует функция нажатия кнопки. Класс-помощник проверяет наличие пользователя и создает его, если пользователя нет, а также получает нужный набор рекомендаций.

После этого обрабатывается функция кнопки для получения рекомендации. Здесь класс-помощник отвечает за отправку данных серверу и возвращение их обратно, а Form отображает полученные данные, обеспечивая тем самым необходимый функционал приложения.

# Часть 3. Выводы, направления дальнейшего развития

Была проделана работа по реализации системы рекомендаций, которая позволяет пользователю получить набор фильмов для просмотра.

В процессе разработки были реализованы несколько концепций, наиболее заметной из них является клиент-серверная концепция. Были изучены множественные источники, позволяющие получить обширное представление о системах рекомендации и представление об их ценности в наши дни.

Направление систем рекомендаций достаточно перспективно в наше время. Если говорить о данной работе, то направлений развития для нее достаточно много. Во-первых, возможно преодолеть имитированные условия, созданные специально для данной работы (клиент и сервер на 1 компьютере, взаимодействие посредством файлов), использовать более совершенные средства коммуникации (возможно сделать клиент и сервер удаленными и использовать семейство протоколов TCP/IP для взаимодействия). Возможно перенести клиент на браузер, чтобы вся работа велась непосредственно с браузера пользователя, а сервер будет удаленным.

Также существует возможность применять системы рекомендаций на практике, например, в интернет-магазинах, поисковиках, онлайн-кинотеатрах.

Таким образом, можно сделать вывод, что направлений для развития множество, а сама область рекомендательных систем в будущем будет только развиваться и распространяться все больше и больше.

# Список литературы

1)

<https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D0%BE%D0%BC%D0%B5%D0%BD%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0>

2) <https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%84%D0%B8%D0%BB%D1%8C%D1%82%D1%80%D0%B0%D1%86%D0%B8%D1%8F>

3)

<http://habrahabr.ru/company/yandex/blog/241455/>

4)

<http://habrahabr.ru/company/surfingbird/blog/139022/>

5)

<http://habrahabr.ru/post/176549/>

6)

<http://habrahabr.ru/company/surfingbird/blog/139863/>

7)

<http://habrahabr.ru/company/surfingbird/blog/139518/>

8)

<https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D1%8F>

9)

Toby Segaran, “Programming Collective Intalligence”,2007г