Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РЕАЛИЗАЦИЯ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ BIG DATA

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 01.04.02 — Прикладная математика и информатика
факультета КНиИТ
Романова Алексея Ивановича

Научный руководитель
доцент, к. ф.-м. н. _____ А. А. Кузнецов

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный мировой рынок достаточно жесток. Для того чтобы в нем выжить и занять свою нишу компаниям уже недостаточно просто предлагать свой продукт. Помимо отточенных бизнес-процессов им необходимо научиться предлагать свой продукт на более высоком уровне, с точки зрения как качества так и способа. Потребителя ежедневно “заваливают”огромными объемами информации о различного рода товарах и услугах, и он уже предпочитает игнорировать весь этот поток предложения, его внимание рассеивается и он уже ничего из этого не воспринимает. Чтобы суметь донести свой продукт до потребителя сейчас уже мало просто взять и сказать ему об этом. Необходимо научиться делать это так, чтобы конкретный потребитель услышал или увидел только ту информацию, которая будет интересна в первую очередь ему. Такой способ подачи информации имеет гораздо больше шансов достигнуть цели.

Если речь идет о сайте, то совершенно очевидно, что пользователю нужно предлагать только тот контент, который потенциально ему будет максимально интересен. В последние 3 десятилетия люди усердно пытаются разработать все новые методы определения и поиска такой релевантной информации. В зависимости от конкретной задачи применяются различные механизмы обработки пользовательских данных. Широкую популярность получили, так называемые рекомендательные системы. Поскольку объемы хранимых данных различными информационными системами с каждым годом существенно увеличивается, для их эффективной обработки были разработаны специальные подходы, которые легли в основу теории Big Data.

Цель работы: реализовать рекомендательную систему в параллельной среде с использованием парадигмы Map Reduce.

В данной работе ставятся следующие задачи:

- изучить теорию рекомендательных систем;
- изучить теорию Big Data;
- провести обзор существующих исследований в этих областях;
- спроектировать алгоритм рекомендаций решающий некоторые проблемы рекомендательных систем;
- исследовать спроектированный алгоритм;
- разработать гибридную рекомендательную систему на базе фреймворка Spark.

Программные средства, используемые в работе: IntelIiJ Idea 2018.1 Community edition, MySQL Workbeanch 6.3 CE, JDK 1.8.065, Spring Framework 1.5.3, Chrome Advanced REST client, Cloudera QuickStarts CDH 5.13.

1 Реализация гибридного алгоритма

1.1 Описание работы программы

В качестве прототипа было принято решение реализовать сервис RESTful API. Данные, необходимые для работы алгоритма рекомендаций, были собраны путем обхода списков аудиозаписей 165756 пользователей, зарегистрированных на сайте https://www.last.fm/. База данных содержит 15088806 записей, среди которых обнаружено 2829571 уникальных пользовательских треков. Для взаимодействия с сервисом использовался интерфейс программы Chrome Advanced REST client. В процессе работы при загрузке исходного набора данных в память возникали проблемы с переполнением кучи и стека. Для того чтобы обойти эту проблему, были изменены стандартные настройки виртуальной машины Java, путем установки следующий значений: `-Xmx8196`, `-Xss4096k` соответственно. Так же использовалась порционная загрузка данных в память.

Точкой входа в приложение при запуске служит класс `Application`.

Для того, чтобы получить доступ к возможностям сервиса, необходимо зарегистрировать нового пользователя в программе при помощи отправки `PUT` запроса `"/registerUser"`на сервер. Данные на сервер отправляются в формате `JSON`.

Для доступа к возможностям сервиса новый каждый пользователь в своем запросе должен передавать серверу `Access Token`. Токен доступа используется в системе для повышения безопасности. Время жизни токена 5 минут. Таким образом, для авторизации пользователя в системе, ему необходимо передав `GET` запрос: `"/getToken"`с параметрами авторизации, получить уникальный `Access Token`.

Генерация токена выполняется на основе учетных данных пользователя и текущей даты и времени. Класс `TokenGenerator` инкапсулирует в себе логику генерации токена.

Если данные авторизации введены некорректно, сервис вернет ошибку авторизации.

Как только пользователь получил токен доступа к данным, он может отправить `POST` запрос `"/getAudio"`для получения списка рекомендованных треков. Заголовок передаваемого запроса должен содержать, полученный ранее `Access Token`. В теле запроса пользователь должен указать список треков в формате `JSON` .

Результатом выполнения данного запроса станет список рекомендаций, состоящий из 100 треков. Данный метод контроллера для построения списка рекомендаций обращается к классу `MusicRecomendManager`. Вся логика алгоритма рекомендаций музыки инкапсулирована в этом классе.

Если в заголовке передаваемого запроса будет содержаться невалидный `Access Token`, сервер данный запрос обрабатывать не будет и вернет сообщение об ошибке авторизации.

С ростом объема данных, работа сервиса станет неэффективной, объема оперативной памяти одной вычислительной машины зачастую оказывается недостаточным, поэтому при работе с большими объемами данных прибегают к кластерным вычислениям. Делегирование вычислений от одного узла к нескольким вычислительным машинам позволит существенно сократить время ответа сервиса, а так же уменьшить нагрузку на оперативную память и позволит обрабатывать сколь угодно большие объемы данных.

1.2 Реализация параллельной версии алгоритма в среде Spark

Основной задачей, поставленной в данной работе является реализовать алгоритм рекомендательной системы в параллельной среде с использованием парадигмы Map Reduce. Для выполнения данной задачи был выбран фреймворк Spark. Ниже представлены основные шаги работы алгоритма.

1) Read tracks to RDD to format (UserID, TrackID, ArtistID, TrackRating).
2) Read artists to RDD to format (UserID, ArtistID, ArtistRating).
3) Fill control user tracks to shared HashMap (TrackID -> Rating).
4) Fill control user artists to shared HashMap (ArtistID -> Rating).
5) Fill similar users tracks to shared HashMap (SimilarUserID -> List<TrackInfo>).
6) Transformation RDD from (UserID, TrackID, ArtistID, TrackRating) to (UserID, (TrackID, TrackRating)).
7) Group RDD by userID.
8) Calculate rating of similarity users (RatingOfSimilarity, UserID).
9) Sort users by rating of similarity.
10) Fill similar users all tracks list to shared HashMap (SimilarUserID -> List<TrackInfo>
11) For each user get sorted users tracks by rating of evaluation this track by control user.
12) Prepare and out recommended tracks.

Стоит отметить, что данный алгоритм ложится в модель параллельных вычислений на базе Spark. Вычислительное ядро Spark оптимизирует процесс вычисления за счет обработки ориентированного ациклического графа.

Граф вычислений предложенного алгоритма представлен на рисунке 1

1.3 Оценка точности рекомендательной системы

Для оценки точности алгоритма использовалось стандартная метрика проверки релевантности результирующего набора для рекомендательных систем. Данная метрика основывается на выборке 50 треков контрольного пользователя для работы алгоритма и оценки количества попаданий множества рекомендованных треков в оставшееся множество, состоящее из 50 треков, которые не вошли во входной набор данных алгоритма. Каждый алгоритм запускался дважды. При первом запуске в обучающее множество попадали только треки с четными номерами, при повторном запуске выбирались треки с нечетными номерами. Таким образом обучающее множество для каждого конкретного пользователя делилось на два не пересекающихся подмножества. Контрольный алгоритм сравнивался со следующими алгоритмами: Taste (CF - Apache Machout), ALS (Apache Spark MLib), Алгоритм рандомных рекомендаций. Результаты работы алгоритмов для первого обучающего набора представлены в таблице 1.

Таблица 1 – Таблица точности алгоритмов для первого набора данных

Login	**Hybrid**	**ALS**	**Taste**	**Random**
nickbyrnedesign	4	4	15	3
h0bbel	11	13	1	16
Ishkur88	13	2	35	5
zachhale	8	2	1	9
shaungwalker	8	12	15	3
eletido	12	26	44	2
atompoet	4	42	17	12
XoBeautifulSin	4	7	13	7
SuperTroels	12	0,5	10	5
vsaket	3	3	15	6
Avg.	7.9	11.15	16.6	6.8
Result count	50	50000	5000	5000
Avg. %	15.8	0.0223	0.332	0.136

Результаты работы алгоритмов для второго обучающего набора пред-

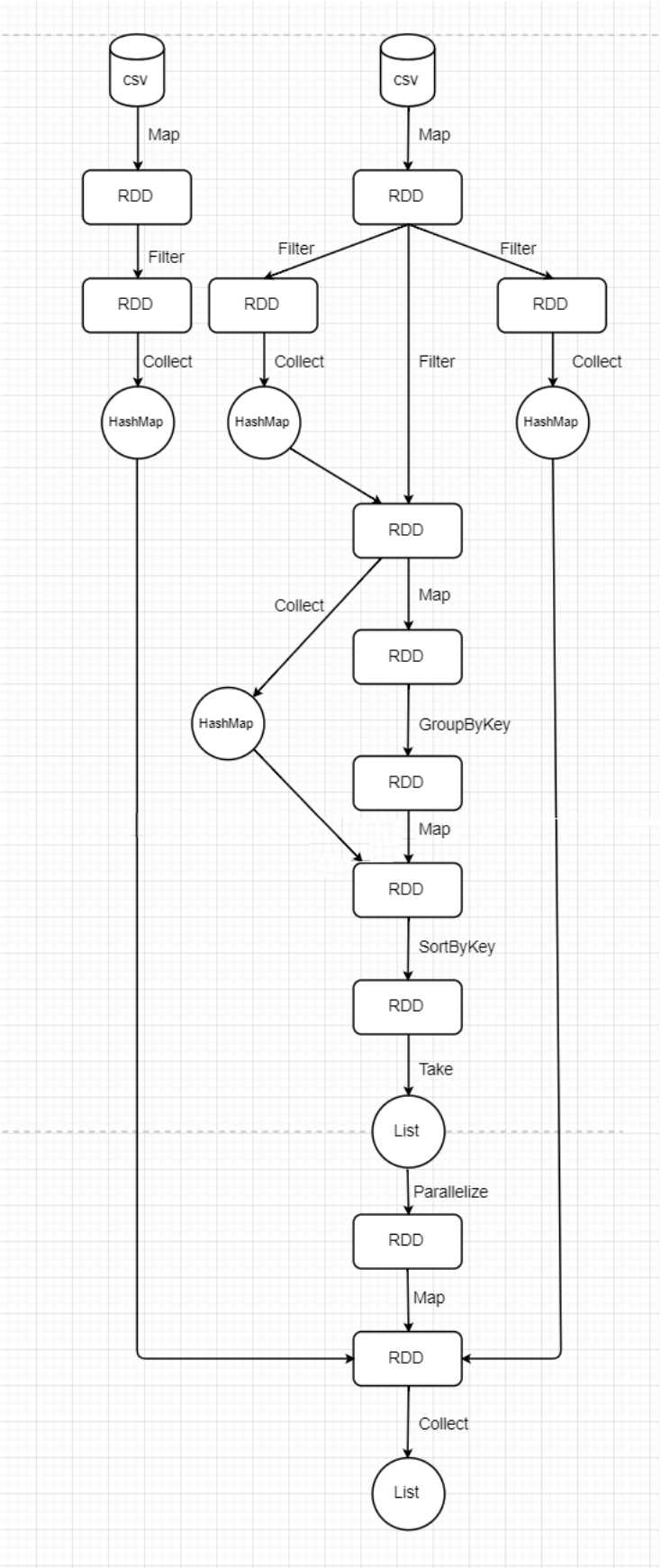

Рисунок 1 – Граф вычислений алгоритма Hybrid

ставлены в таблице 2.

Таблица 2 – Таблица точности алгоритмов для второго набора данных

Login	**Hybrid**	**ALS**	**Taste**	**Random**
nickbyrnedesign	6	4	10	8
h0bbel	5	8	2	12
Ishkur88	15	2	21	2
zachhale	7	5	2	4
shaungwalker	7	13	25	2
eletido	5	22	39	3
atompoet	5	40	4	9
XoBeautifulSin	7	11	14	9
SuperTroels	8	1	4	3
vsaket	8	3	9	10
Avg.	7.3	10.9	13	6.2
Result count	50	50000	5000	5000
Avg. %	14.6	0.0218	0.26	0.124

Гистограмма, отражающая данные этих таблиц представлена на рисунке 2.

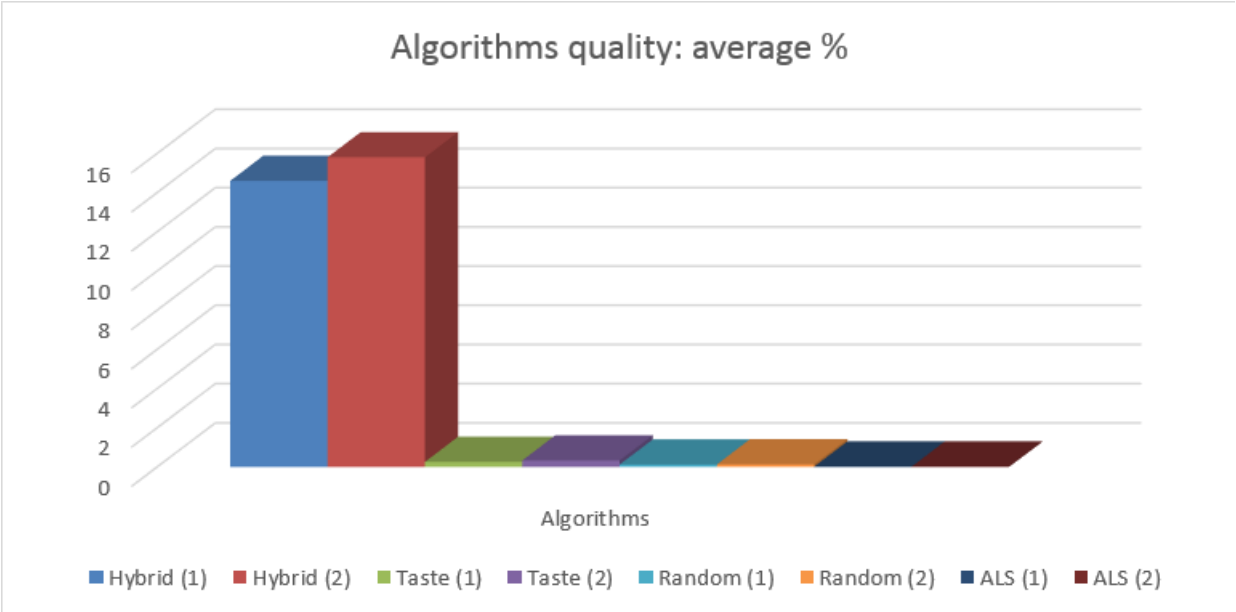

Рисунок 2 – Гистограмма точности алгоритмов

Так же проводились замеры времени работы алгоритмов в секундах. Результаты тестового замера для первого набора данных представлены в таблице 3.

Результаты тестового замера для второго набора данных представлены в таблице 4.

Таблица 3 – Таблица времени работы алгоритмов для первого набора данных

Login	**Hybrid**	**ALS**	**Taste**	**Random**
nickbyrnedesign	50	24	57	43
h0bbel	56	29	316	40
Ishkur88	49	30	36	39
zachhale	53	31	154	38
shaungwalker	54	31	51	39
eletido	50	31	40	39
atompoet	54	31	61	40
XoBeautifulSin	53	32	58	42
SuperTroels	52	25	47	40
vsaket	53	30	70	42
Avg.	52.4	29.4	89	40.2

Таблица 4 – Таблица времени работы алгоритмов для второго набора данных

Login	**Hybrid**	**ALS**	**Taste**	**Random**
nickbyrnedesign	50	24	56	42
h0bbel	63	30	236	43
Ishkur88	50	30	42	42
zachhale	58	29	221	42
shaungwalker	51	30	34	40
eletido	50	31	37	39
atompoet	59	31	135	41
XoBeautifulSin	54	31	54	41
SuperTroels	50	28	76	38
vsaket	58	30	91	40
Avg.	54.3	29.4	98.2	40.8

Гистограмма, отражающая данные этих таблиц представлена на рисунке 3.

Из таблицы видно, что точность рекомендаций предложенного алгоритма выше чем у конкурентов на несколько порядков. Так же интересно заметить, что точность рекомендаций на исходном наборе данных у алгоритма Taste выше чем у ALS. В качестве альтернативы предложенным алгоритмам использовался алгоритм, основанный на случайных рекомендациях. По приведенным выше метрикам у данного алгоритма процент попадания оказался выше чем у алгоритма ALS. Однако самым быстрым из рассматриваемых алгоритмов оказался ALS. Контрольный алгоритм на заданном наборе данных

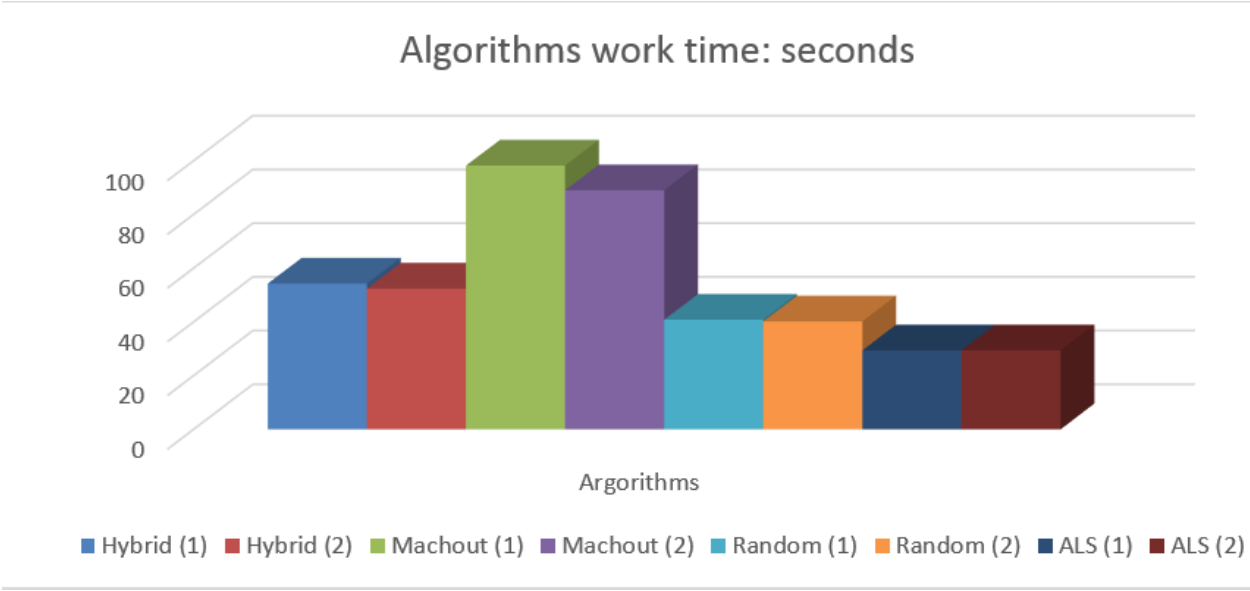

Рисунок 3 – Гистограмма времени работы алгоритмов

работает медленнее алгоритма ALS на 22-24 секунды, при этом он демонстрирует более высокую скорость в сравнении с алгоритмом Taste.

ЗАКЛЮЧЕНИЕ

В ходе данного исследования были рассмотрены технологии Data mining и Big Data, а так же основные подходы к проектированию рекомендательных систем. В работе описываются наиболее популярные алгоритмы рекомендательных систем, их сильные и слабые стороны, а так же проблемы, с которыми сталкиваются разработчики, при использовании этих алгоритмов. В качестве альтернативы существующим алгоритмам в работе предлагается гибридный алгоритм, позволяющий решить такие проблемы как: холодный старт и утеря новых объектов. Для проведения исследований алгоритмов был подготовлен датасет из реальных пользовательских данных популярного интернет ресурса. Предложенный алгоритм был реализован при помощи средств языка Java, MySql, а также фрейморков Spark и Spring. В результате сравнительных тестов с конкурентами контрольный алгоритм продемонстрировал гораздо более высокую точность, а так же конкурентную эффективность.