САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

Система коллаборативных рекомендаций туристических достопримечательностей

Дипломная работа студента 545 группы

Манаева Дмитрия Сергеевича

Научный руководитель	……	к.ф.-м.н., доцент каф. Информатики
	/ подпись /	Д. Ю. Бугайченко
Рецензент	……	ст.преп. каф. Системного Программирования
	/ подпись /	Н. А. Зонова
“Допустить к защите”	……	д.ф.-м.н., проф. А. Н. Терехов.
заведующий кафедрой,	/ подпись /	

Санкт-Петербург

2012

SAINT PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Software Engineering Chair

COLLABORATIVE RECOMMENDATIONS SYSTEM FOR TOURIST ATTRACTIONS

Graduate paper by

Dmitry Manayev

Supervisor	……	PhD. D. Y. Bugaichenko
Reviewer	……	Senior Lect. N. A. Zonova
"Approved by" Head of Department	……	Professor A. N. Terekhov

Saint Petersburg
2012

СОДЕРЖАНИЕ

Введение

Учитывая скорость роста количества динамической информации в интернете, пользователи начинают тонуть в море контента.

На помощь им приходят системы рекомендаций, упрощающие поиск нужной информации для миллионов пользователей, обеспечивая высококачественные предложения. Эти системы уже успешно справляются со своей задачей в таких областях как видео, музыка, книги и т.д.

Но когда доходит до туризма – эффектиность таких систем падает и основной проблемой при построении туристических рекомендации является недостаток статистики по месту не основного пребывания пользователя. При планировании путешествия пользователем нам известна только информация о интересных ему местах в его городе, но ничего не известно о его предпочтениях в городах, которые он хочет посетить. Таким образом невозможно построить рекомендации с помощью стандартных коллаборативных алгоритмов.

Рекомендации сегодня

Задача системы рекомендации - предлагать объекты с целью помочь пользователю в его выборе (или покупке) в большом объеме информации. Такие системы играют важную роль в комерции, сервисах подписки, фильтрации информации и т.д. Они предлагают индивидуальный подход к каждому пользователю, значительно увеличивая вероятность приобретения продукта, чем без рекомендации. Персональные рекомендации очень важны в областях, где разнообразие выбора обширно, интересы покупателя и благоприятная стоимость товара важны. Стандартные примеры областей использования - это сервисы, связанные с искусством (книги, фильмы, музыка), модой, едой и т.д.

С ростом интернет компаний, увеличивается количество коммерческих и арендных сервисов, использующих системы рекомендаций. Наиболее крупные участники электронной коммерции являются Amazon.com[1] (рекомендации товаров), Netflix[30] (рекомендации фильмов) и Last.fm[23] (рекомендации музыки).

Когда пользователь ищет продукт на Amazon.com, магазин будет предлагать ему дополнительные товары, основываясь на матрице содержащей информацию о уже купленных товарах другими пользователями, купивших только-что выбранный им.

Pandora Radio [32] берет на вход песню или имя музыканта и по ряду ключевых слов привязанных к музыканту или части песни проигрывает музыку с похожими характеристиками. Станции созданные на Pandora можно настроить через обратную связь с пользователями, подчеркивая или убавляя опредленные харкатеристики.

Netflix предлагает фильмы , которые пользователи скорее всего хотели бы посмотреть, учитывая их предыдущие предпочтения и особенности в отношении к просмотренному по сравнению с другими пользователями. А также учитывает характеристики фильма, такие как жанр и год.

Многим из нас знакома ситуация, когда собираясь в путешествие нам хочется узнать больше о тех местах, которые были бы нам наиболее интересны. Иметь возможность за короткое время ощутить себя «местным» в новом городе. Для этого чаще всего требуется перечитать несколько путеводителей, найти статьи на википедии, изучить сайты с предлагаемыми путевками и рекламу. Но мы не всегда найдем то что заинтересовало бы именно нас, так как все эти источники информации не способны предложить каждому человеку то, что хотел бы именно он сам. При этом придётся затратить большое количество времени. Сейчас пришло время создания технологий, которые могли бы помочь нам быстро найти нужное в огромном количестве информации во время

планирования своего путешествия.

На рынке уже существует несколько крупных и востребованных сервисов, предлагающих информацию и отзывы о местах: Yelp [42] (сервис поиск услуг на местном рынке), Facebook [7](самая крупная социальная сеть) и Google Places [12] (тоже сервис поиска услуг на местном рынке). Также сюда можно отнести мобильный сервис Foursquare [8] появившийся в 2010 году. Его основная идея – возможность зарегистрировать себя (check-in) в текущем месте пребывания, открыла вместе с интересом пользователей новые возможности в развитии области туризма.

Результаты работы

В рамках работы предложен новый подход к построению рекомендации туристических достопримечательностей, основанный на алгоритме «Random walk with Restart». Главная идея алгоритма - это «построение мостов» между местными и уже успевшими побывать в нескольких городах путешественниками. Система предлагает агрегацию различных подходов.

Предложенный инструмент работает с множеством предпочтений пользователя, представляющих собой список рейтингов мест с одним дополнительным отношением, привязкой места к определенному городу. На основе этих данных строятся рекомендации пользователям тех мест (достопримечательностей), которые они скорее всего хотели бы посетить.

В качестве эксперимента работы системы рекомендации была взята задача – порекомендовать релевантные места пользователям из одного города в другой. На основе метрики Precision and Recall проведена оценка результатов работы алгоритма.

Эксперименты продемонстрировали применимость

представленного алгоритма к задаче построения рекомендаций туристических достопримечательностей.

1 Анализ предметной области

Подходы к рекомендациям можно условно разделить на три группы: ориентированные на содержание (content-based), коллаборативные (collaborative) и социальные (social). Ориентированная на содержание фильтрация использует ряд дискретных характеристик объекта, для того чтобы порекомендовать дополнительные объекты с похожими свойствами. [10] В коллаборативном подходе строится модель из предыдущих предпочтений пользователей (объекты купленные или оценённые пользователем, возможно с числом, характеризующим «рейтинг» объекта) и похожих решений других пользователей. Используя эту модель, предсказываются объекты (или «рейтинги» к ним) в которых пользователь мог бы заинтересоваться. Социальный подход работает по тому же принципу, что и коллаборативный, но учитывает не только поведение пользователей, но и инофрмацию имеющуюся в социальной сети.

Разницу collaborative и content-based подходов можно продемонстрировать на примере двух популярных систем рекомендации музыки — Last.fm и Pandora Radio.

- Pandora использует информацию о песни и музыканте (подмножество из 400 атрибутов предоставленных в рамках проекта Music Genome) для того, чтобы настроить «станцию», которая проигрывает похожую музыку. Пользователи могут убавлять или подчеркивать важность атрибутов, фиксируя что им не нравится или нравится определенная песня.
- Last.fm строит список с предпочитаемыми песнями наблюдая, какие группы и отдельные музыкальные трэки пользователь слушает

регулярно и сравнивая их, со вкусами других пользователей. Last.fm будет играть трэки, которые не появляются в постоянной библиотеке пользователя, но часто играют у других пользователей с похожими интересами.

Каждый подход имеет свои сильные и слабые стороны. В вышеуказанном примере, Last.fm нуждается в большом количестве информации изначально, чтобы построить рекомендации. Это пример «проблемы холодного старта» (cold start problem) и общий для всех систем коллаборативных рекомендации. В то время как Pandora необходимо мало информации для начала, она гораздо более ограничена по масштабу. К примеру, она может строить рекомендации, которые очень похожи на первоначальный источник.

На данный момент в основном применяются коллаборативные рекомендации, так как они требуют от пользователя минимума информации.

1.1 Коллаборативные рекомендации

Коллаборативные рекомендации основываются на анализе предпочтений пользователей в явном или неявном виде формируя список «рейтингов» для пользователей, а затем пытаясь «предсказать» рейтинги элементов пользователю еще неизвестных. Здесь есть два основных подхода: от пользователя к пользователю (user-user) и от объекта к объекту (item-item). Главное преимущество таких рекомендаций является то, что они не опираются на анализ содержимого элементов и, следовательно, они способны точно рекомендовать сложные элементы, такие как фильмы, на примере сервиса Netflix (рекомендации фильмов), не требуя «понимания» самого элемента.

Три основные проблемы этого направления[11]:

- “Холодный старт” (Cold Start): Эти системы часто требуют

большого количества существующих данных на пользователей, чтобы сделать точные рекомендации

- Масштабируемость (Scalability): Во многих областях, где используются эти системы рекомендации, есть миллионы пользователей и продуктов. Таким образом часто требуется большое количество вычислительной мощности для расчета рекомендаций.
- Разряженность (Sparsity): Количество единиц товара, продаваемых на крупных сайтах электронной коммерции велико. При этом наиболее активные пользователи оценят только небольшую часть общей базы данных. Из чего следует, что даже самые популярные элементы имеют очень мало рейтингов или в случае музыки, пользователей гораздо больше, чем музыкальных трэков.

Среди алгоритмов, используемых в коллаборативных рекомендациях, можно отметит: матричную факторизацию (техника аппроксимации матриц низкого ранга)[27][31][36], ближайшие соседи («neighborhood»), баесовские сети[4] и другие вероятностные модели.

1.1.1 “Пользователь - Пользователь”

В подходе “пользователь-пользователь” коллаборативные рекомендации пытаются смоделировать ситуацию из жизни, где пользователь просит друга дать совет. Прогноз выбора пользователя делается на основании предпочтений других пользователей. При этом ищутся пользователи с похожими профилями («neighbor based»), либо матрица рейтингов аппроксимируется через произведение двух других матриц, подгоняемых методами близкими к методам обучения нейронных сетей (например, градиентным спуском). Этот процесс называется «факторизация» [31].

Наиболее известные примеры использования направления user-user в

коллаборативных рекомендациях:

- Как уже подробно разбиралось, Last.fm рекомендует музыку, основываясь на сравнении музыкальных пристрастий похожих пользователей
- Facebook, MySpace, LinkedIn и другие социальные сети используют коллаборативную фильтрацию для построения рекомендации новых друзей, групп, и других социальных элементов путем проверки сети отношений между пользователем и его друзьями

1.1.2 “Обьект – обькт”

Основная идея — можно искать не только пользователей с похожими вкусами, но и элементы, которые часто нравятся или не нравятся вместе (люди, которые покупают x также купят y). Один из самых известных примеров использования этого направления - популярный алгоритм системы рекомендации на Amazon.com. Более подробную информацию о используемых алгоритмах и техниках для определения схожести элементов представлена здесь [19].

1.2 На основе содержания

Данный подход основан на идее сопоставления информации об интересах пользователя с описанием искомого элемента. Другими словами эти алгоритмы пытаются рекомендовать элементы похожие на те, которые понравились пользователю в прошлом (или изучает в настоящем времени). В частности, различные элементы-кандидаты сравниваются, с уже оцененными пользователем элементами и рекомендуются самые подходящие. Такой подход имеет свои корни в поиске информации и исследовании фильтрации информации.

В основном, эти методы строят профиль элемента (например, набор

дискретных атрибутов и свойств), характеризующий элемент в системе. Она создает ориентированный на содержание профиль пользователей на основе вектора весов свойств элемента. Веса обозначают важность каждого свойства для пользователя и могут быть вычислены из отдельных векторов оценок контента каждым пользователем, используя различные методы. Простые подходы используют среднее значение этих векторов в то время, как другие современные методы используют такие техники машинного обучения, как Байесовские Классификаторы, кластерный анализ, деревья принятия решений и искусственные нейронные сети для того, чтобы оценить вероятность того, что пользователю понравится фиксированный элемент. Для определения весов атрибутов используется прямая обратная связь с пользователем (обычно в форме кнопок «нравится» или «не нравится»).

Его больше всего критикуют мотивируя это тем что «пользователю приходится много о себе рассказывать. Еще одна проблема в том, что такой метод достаточно сложен в реализации — надо вытаскивать семантику из описания элементов и интересов пользователя, строить отношения и т. д. Современные работы в этой области так или иначе используют стандарт Friend Of A Friend [9] - «онтологии социальных сетей». Например рекомендации фильмов [11] и социальным сетям в целом [10].

Как уже подробно разбиралось, Pandora Radio популярный пример ориентированных на содержание систем рекомендаций. Он играет музыку с аналогичными характеристиками, что и песня, которую пользователь слушал первоначально. Есть также большое количество систем, нацеленных на предоставление рекомендации фильмов. Такие примеры, как Rotten Tomatoes[38], Internet Movie Database[16] и Jinni[17].

1.3 На основе социальных связей

Социальные рекомендации используют информацию социальной сети для построения предложений. При построении советов для ночных клубов в работе [6], были сделаны следующие выводы:

1. Такой подход дает более релевантные результаты при меньшей вычислительной сложности.
2. Отмечается важность психологического аспекта — личные рекомендации воспринимаются лучше, чем безличные.
3. Отмечается особо хорошая работа этого алгоритма в случае «холодного старта» - появления объекта, на который наткнулось еще немного пользователей.

При работе с музыкальными трэками в [18] предложили алгоритм «Random Walk with Restart». Тестировали его на данных Last.fm и утверждают, что использование данных графа социальной сети улучшает качество рекомендаций. Эта работа предлагает очень интересную агрегацию различных подходов, учитывая отношения в тройке «Пользователи-Трэки-Тэги». Пользователи могут дружить друг с другом (отношение «пользователь-пользователь»), могут слушать трэки (отношение «пользователь-трэк») и помечать их тэгами (отношения «пользователь-тэг» и «трэк-тэг»). Вся эта структура загружается в граф с весами, после чего начинается «Random walk with Restart». Алгоритм начинает работу с определенного узла (пользователя) и случайно блуждает по направленным дугам, при этом в любой момент времени может вернуться в начало. В конце определяется вероятность того, что пользователь окажется в каждом из узлов, эта вероятность и есть его интерес к другим пользователям и местам.

1.4 Основные алгоритмы

1.4.1 K-ближайших соседей

Один из наиболее часто используемых алгоритмов в системах рекомендаций является k-ближайших соседей (k-nearest neighbors, k-NN) [20]. k-NN алгоритм - это метод для классификации объектов на основе свойств его ближайших соседей в пространстве признаков. В k-NN, объект классифицируется по большинству голосов своих соседей, с присвоением объекта к классу наиболее распространенных среди его k ближайших соседей (k-целое положительное число, обычно мала). Если k = 1, то объекту просто присваивается класс своего ближайшего соседа.

1.4.2 Меры сходства

В нашей работе используются три меры сходства, часто используемы при построении рекомендаций: корреляция Пирсона, сходство по косинус и метрика Евклида.

Корреляцией Пирсона является мера корреляции (линейная зависимость) между двумя переменными X и Y, имеющая значения в промежутке от +1 до -1 включительно. В социальной сети, окрестности конкретного пользователя с аналогичными вкусами или интересами может быть найдена путем вычисления коэффициента корреляции Пирсона. Собирая данные о предпочтениях топ-N ближайших соседей конкретного пользователя (взвешенные по схожести), могут быть построены для него рекомендации: [34]

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}},$$

где $\bar{x}$ и $\bar{y}$ средние значения X и Y соответственно.

Сходство по косинус является мерой сходства между двумя

векторами, которое получаем, измеряя косинус угла между ними. Косинус нуля это единица и ≤ 1 для любой другого угла, наименьшее значение косинуса это минус единица . Косинус угла между двумя векторами определяет, указывают ли эти вектора примерно в том же направлении. Формула в n-мерном пространстве:

$$\cos\theta = \frac{A \times B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Евклидова метрика при вычислении использует геометрическое расстояние d между двумя точками в многомерном пространстве, вычисляемое по формуле:

$$d = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2},$$

где $x = (x_1, \dots, x_n)$ и $y = (y_1, \dots, y_n)$. А рассчитывается по формуле $\frac{1}{1+d}$, так что полученные значения находятся в диапазоне (0,1].

1.4.3 Цепь Маркова

Цепь Маркова - это последовательность случайных событий с конечным или счетным числом исходов, характеризующаяся тем свойством, что при фиксированном настоящем будущее независимо от прошлого. [26]

Если цепь Маркова однородна во времени, так что процесс представлен одной независящей от времени матрицы p_{ij}, тогда вектор P называется стационарным распределением (или инвариантной мерой), если удовлетворяет сследующей фломуле:

$$P_j = \sum_{i \in S} P_i \, p_{ij}$$

, где p_{ij} – неотрицательно и их сумма в единице.

Неприводимая цепь Маркова имеет стационарное распределение,

если и только если все её состояния положительно возвратны. В этом случае P уникальна и связана с ожидаемым временем возврата:

$$P_j = \frac{C}{M_j}$$

, где C – постоянная нормализации и M_j – среднее время возвращения. Кроме того, если цепь и неприводимая и апериодическая, то для любых i и j:

$$\lim_{n\to\infty} p_{ij}^{(n)} = \frac{C}{M_j}$$

Нет никаких предположений о начальном распределения; цепь сходятся к стационарному распределению независимо от того, где она начинается. Такие P называются равновесным распределением цепи.

В данной работе цепь Маркова используется при построении графа “пользователи-места”, где P отрожает распределение вероятностей на дугах исходящих из каждого узла. Сумма вероятностей всегда равна 1. И при поиске интересов пользователя к местам и другим пользователям ищется вектор вероятностей того, что в бесконечный момент времени, он окажется в них.

1.4.3 Random Walk with Restart

За начальную точку в этом алгоритме берется текущий пользователь, после чего совершается переход по одной из дуг графа случайным образом (по дугам с большим весом вероятность перейти больше). При этом есть некоторая фиксированная вероятность a вернутся в начальную точку и начать снова.

Процесс блужданий в таком графе можно описать с помощь цепи Маркова с дискретным временем, для которой можно найти расспределение вероятностей, являющееся неподвжной точкой (стационарная распределение вероятностей). То есть для всех вершин

будет определена вероятность того, что пользователь до них доберется бесконечное число раз. Это расспределение и является прогнозом привлекательности объектов для пользователя. На рис. 1 изображен первый шаг алгоритма.

Этот подход наиболее интересен в основном благодаря тому, что удачно позволяет сочетать информацию из различных источников.

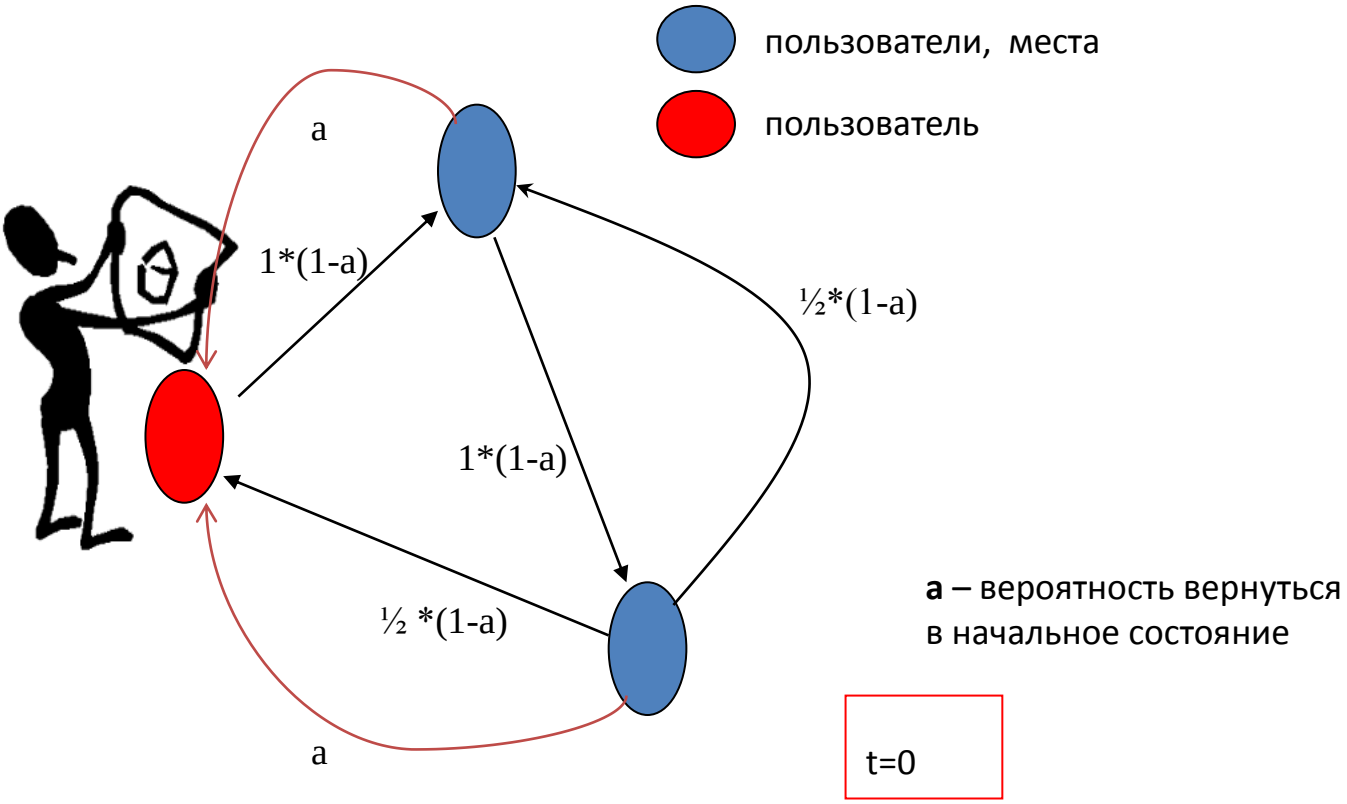

Рис. 1 «Random Walk with Restart»

1.5 Существующие технологии и решения

Как мы уже упомянули выше на данный момент уже существует несколько известных сервисов отзывов по местам. Каждый из них имеет свои сильные и слабые стороны:

- Google places – сервис рекомендации мест, использует социальный и ориентированный на содержание подходы. Социальный подход основывается на информации о активностях друзей в социальной сети Google+, которая слабо развита в

отношении интереса пользователей. Второй подход заключается в том, что к каждому месту привязан список тэгов [21], составляющие общее описание места. На основе предыдущих предпочтений строится вектор весов, который определяет наиболее интересные свойства в месте. Достаточно сложный и не всегда точный алгоритм, старающийся покрыть отношение пользователя вытаскивая семантику из места.[12]

- Yelp – самая крупная в мире база данных отзывов о местах, основная цель которого поиск на местном рынке услуг, например ресторанов или парикмахерских, с возможностью добавлять и просматривать рейтинги и обзоры этих услуг. Отсутствует механизм рекомендации [42]
- Facebook - основная социальная площадка в мире. Год назад появилась функциональность, подобная Foursquare, которая позволяет зарегистрировать(check-in) себя на месте. Также недавно появились функциональность для оставления отзыва о месте. Отсутствует механизм рекомендаций и мало информации о местах. [7]

Кроме вышеуказанных сервисов в последнее время заметен интерес к решению данной проблемы стартапами (компании созданные недавно, находятся в стадии развития или исследования перспективных рынков). Каждый из них пытается решить проблему по-своему. Но условно их можно разделить на две группы: сервисы использующие информацию в социальных сетях и сервисы предложений от местных. Здесь хочется отметить появившийся в мае в инкубаторе AngelPad [3] стартап SpotSetter [39], с идеей - строить рекомендации мест на основе информации о социальной активности друзей в Facebook, Twitter [41], Foursquare [8] и Instagram [15]. Он интересен тем , что ушел значительно дальше от своих конкурентов, научившись классифицировать запрос пользователя и находить нужный контент из социальных сетей по

городу и типу мест, определяя экспертов в той или иной области. Другие же продукты просят местных или друзей побывавших в этом городе помочь построить маршрут для пользователя. Данный проект пока находится в стадии разработки, но в ближайшее время планируется бета-версия.

Учитывая все вышесказанное можно твердо заявить о том, что проблема рекомендаций достопримечательностей не решена ни одним из вышеуказанных сервисов. А их развитие и появление новых проектов демонстрируют интерес пользователей и инвесторов к области туризма.

1.6 Используемые технологии

Технологии используемые в работе можно разделить на две задачи: разработка системы коллаборативных рекомендации и сбор тестовых данных для проведения экспериментов.

В качестве технологии для реализации системы была выбрана библиотека “Taste” [40], реализующая основные алгоритмы коллаборативных рекомендаций с примерами. Изначально он был отдельным проектом и мог работать только в автономном режиме на одном узле. Сейчас он является частью проекта с открытым кодом (open-source) Apache Foundation Mahout [25] и поддерживает, как автономную, так и масштабируюемую реализацию с поддержкой Hadoop [13] .

Mahout представляет собой набор библиотек для построения масштабируемых алгоритмов, решающих задачи машинного обучения на платформе Hadoop (свободно распространяемый набор утилит, библиотек и программный каркас для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов). Сейчас проект Mahout активно развивается, количество реализованных алгоритмов быстро увеличивается. Его основная часть алгоритмов реализована в рамках парадигмы MapReduce [28] (фреймворк для

вычисления некоторых наборов распределенных задач с использованием большого количества компьютеров (называемых «нодами»), образующих кластер) проекта Hadoop.

Для построения и экспериментирования системы нам достаточно стандартной реализаций “Taste” работающей на одном узле. Таким образом в будущем, для расширения проекта можно будет легко переписать систему для работы с Hadoop.

При решения задачи сбора тестовых данных были выбраны следующие технологии:

- База данных MSSQL Server 2008
- ADO.NET Entity Framework (EF)
- Язык программирования C#

2 Постановка задачи

Целью работы является построение системы рекомендаций способной помочь пользователям запланировать путешествие, предлагая наиболее релевантные им туристические достопримечательности (места). Конечная система должна находить пользователям достопримечательности, которые они скорее всего хотели бы посетить, учитывая их предыдущие предпочтения.

В рамках работы были выделены следующие основные подзадачи:

1. Проанализировать существующие решения и подходы к рекомендациям
2. Разработать алгоритм вычисления схожести между местами и пользователями, позволяющий идентифицировать сходство даже при условии недостатка статистики
3. Реализовать генерацию рекомендаций
4. Сравнить эффективность стандартных подходов с предложенным алгоритмом

3 Реализация системы

Изучив все существующие решения построения советов, было решено взять направление коллаборативных рекомендаций, т.к. они требуют от пользователя минимума информации. Но существующие в нем, подходы не способны дать хорошие предложения туристических достопримечательностей. Каждый из них имеет свои минусы.

Строя предложения пользователю из одного города в другом, требуется получить релевантные места только из другого города. Но как в подходе “пользователь – пользователь”, так и в подходе “место – место” имеется слишком мало пересечений между двумя разными локациями, то есть относительно мало пользователей побывавших в обоих городах и оценивших везде места. В результате в обоих случаях велика вероятность в списке релевантных мест для будущего путешественника получить много мест из его родного города (городов) и мало или совсем не получить релевантных мест из города будущего путешествия.

В итоге была выявлена одна общая проблема для коллаборативных рекомендаций - недостаток статистики по месту не основного пребывания пользователя. Она отражается в алгоритмах сравнения, которые надо изменить, наладить “мосты” между городами. Но этого не достаточно, также нужно построить цепочку связей, которая позволяла пользователю из одного города, найти похожие места и похожих местных в другом городе. Здесь при обзоре существующих решений построения предложений в различных областях наиболее интересной оказалась работа , предлагающая алгоритм “Random Walk with Restart”. Авторы построили граф знаний (социальный граф) для музыкального сервиса Last.fm. [18] Этот подход показался наиболее интересным в основном благодаря тому, что удачно позволяет сочетать информацию из различных источников. Он и был взят за основу при генерации

рекомендаций.

При разработке данного алгоритма можно выделить три основные этапа, это реализация “мостов”, построение матрицы переходов Марковской цепи и генерация рекомендации на основе “Random Walk with Restart”.

3.1 “Мосты”

Центральная идея данной работы это построение “мостов” между городами. Мы предполагаем, что уже существует группа людей, которая побывала в нескольких городах и везде оставила оценки местам, для того чтобы помочь другим людям. Эта информация очень важна и она представляет пересечение, за счет которого можно построить “мосты”. Для работы с ней необходимо преобразовать существующие метрики, вычисляющие схожесть между пользователями в случае подхода “пользователь – пользователь” и местами в подходе “объект – объект”, чтобы укрепить связи (отношения) через неё между двумя городами.

Следующие определения будут использоваться при описании процесса:

- U - множество пользователей
- P - множество мест
- $R = \{0, \dots, 5\}^{|U| \times |P|}$ - матрица рейтингов
- C - множество городов, где $C_i \in P$:

$$C_1 \cup \dots \cup C_n = P \text{ и } C_i \cap C_j = \emptyset \; \forall \; i, j : i \neq j$$

- $\overline{r_u}$ - средний рейтинг пользователя u
- $\overline{r_p}$ - средний рейтинг места p

При вычислении будут использоваться две основные метрики: корреляция Пирсона и метрики Евклида. Подробно разберем вычисление схожести между пользователями.

В стандартном алгоритме при сравнении двух пользователей u_x и u_y рассматриваются всё множество мест, которые оценили оба пользователя $P' \in P$. Вычисление схожесть происходит по следующим формулам:

- Корреляци Пирсона

$$sim(u_x, u_y) = \frac{\sum_{p \in P'}(r_{u_x p} - \overline{r_{u_x}})(r_{u_y p} - \overline{r_{u_y}})}{\sqrt{\sum_{p \in P'}(r_{u_x p} - \overline{r_{u_x}})^2}\sqrt{\sum_{p \in P'}(r_{u_y p} - \overline{r_{u_y}})^2}}$$

- Метрика Евклида $sim(u_x, u_y) = \frac{1}{1+d}$, где d это расстояние между двумя векторами рейтингов пользователей, вычисляемое по формуле:

$$d = \sqrt{\sum_{p \in P'}(r_{u_x p} - r_{r_{u_y p}})^2}$$

В таком случае расстояние между пользователями и путешественниками (пользователи посетившие несколько городов) достаточно велико, так как при сравнении будет обнаружено много мест, которые оценил только один из пользователей. Корреляция Пирсона и метрика Евклида будут давать плохую(маленькую) схожесть и за счет этого будет теряться связь с другим городом. Для решения этой проблемы будем брать только места из общих городов. Пользователь может иметь места из нескольких городов и это можно представить в форме определения: $D_u = \bigcup_{\{C \in \{C \mid \exists p \in C:\, r_{up} \neq 0\}\}} C$ - множество мест, в которых побывал пользователь u и соответственно изменятся формулы вычисления схожести:

- $d = \sqrt{\sum_{p \in D_{u_x} \cap D_{u_y}}(r_{u_x p} - r_{r_{u_y p}})^2}$ для метрики Евклида
- $sim(u_x, u_y) =$

$$\frac{\sum_{p\in D_{u_x}\cap D_{u_y}}(r_{u_xp}-\overline{r_{u_x}})(r_{u_yp}-\overline{r_{u_y}})}{\sqrt{\sum_{p\in D_{u_x}\cap D_{u_y}}(r_{u_xp}-\overline{r_{u_x}})^2}\sqrt{\sum_{p\in D_{u_x}\cap D_{u_y}}(r_{u_yp}-\overline{r_{u_y}})^2}}$$

для корреляции Пирсона

Таким образом уменьшится расстояние между путешественниками и местными, укрепятся отношения “пользователь – пользователь”, будет воздвигнут “мост”.

При стандартном вычислении схожести двух мест будут также задействованы, вышеуказанные метрики. Основные отличия будут заключаться, только в том что при сравнений двух мест берется список пользователей оценивших хотя бы одно место $U' \in U$и соответствующие формулы:

- Корреляция Пирсона

$$sim(p_x, p_y) = \frac{\sum_{u\in U'}(r_{up_x}-\overline{r_{p_x}})(r_{up_y}-\overline{r_{p_y}})}{\sqrt{\sum_{u\in U'}(r_{up_x}-\overline{r_{p_x}})^2}\sqrt{\sum_{u\in U'}(r_{up_y}-\overline{r_{p_y}})^2}}$$

- Расстояние Евклида $d = \sqrt{\sum_{u\in U'}(r_{up_x} - r_{up_y})^2}$

По этим формулам при вычислении схожести мест из разных городов, расстояние будет большим и метрики будут возвращать достаточно маленькие величины, что может отрицательно сказаться на поиске релевантных мест в другом городе. Для решения этой проблемы надо сравнивать места “по городам”, так что место может принадлежать нескольким городам, при наличии путешественника из другого города, оставившего отзыв о месте. Это можно представить в следующей форме $G_p = \bigcup_{\{C\in\{C\,|\exists u\in C:\, r_{up}\neq 0\}\}} C$, множество пользователей из разных городов, которые оценили это место. Следовательно схожесть мест p_x и p_y будут сравниваться по следующим формулам, где мы берем пользователей из общих городов:

- $d = \sqrt{\sum_{u \in G_{p_x} \cap G_{p_y}} (r_{up_x} - r_{r_{up_y}})^2}$ для метрики Евклида
- $sim(p_x, p_y) =$

$$\frac{\sum_{u \in G_{p_x} \cap G_{p_y}} (r_{up_x} - \overline{r_{p_x}})(r_{up_y} - \overline{r_{p_y}})}{\sqrt{\sum_{u \in G_{p_x} \cap G_{p_y}} (r_{up_x} - \overline{r_{p_x}})^2} \sqrt{\sum_{u \in G_{p_x} \cap G_{p_y}} (r_{up_y} - \overline{r_{p_y}})^2}}$$

для корреляции Пирсона

В итоге были построены и укреплены “мосты” между различными городами. Это увеличит покрытие алгоритма. Теперь используя эти связи можно построить цепочку, по которой для пользователя можно будет найти группу похожих путешественников и местных в городе планирования будущего путешествия.

3.2 Построение матрицы переходов Марковской цепи

На данном этапе строится матрицу отношений “Пользователи – Места”, где данными будут являться веса в ориентированном графе, отражающие схожесть между элементами графа. Для вычисления схожести будут использоваться, выше предложенные формулы, с построенными “мостами” между городами и использующие две основные метрики: корреляция Пирсона и метрика Евклида. Связи в графе разделены на три типа:

- “пользователь – пользователь” - двунаправленная связь, весом которой является схожесть между двумя пользователями.
- “место – место” - двунаправленная связь, весом которой является схожесть между двумя местами.
- “пользователь – место” – только связь в направлении места, весом которой является рейтинг пользователя, оставленный этому месту. Чаще всего в системах отзывов используется рейтинг из промежутка от 0 до 5. Он приводится к значениям от 0 до 1.

Обратная связь не нужна, так как основная цель найти релевантные места для пользователя и лишние циклы не нужны.

В итоге получается матрица смежности размером равному квадрату суммы всех мест и пользователей, которая представляет собой отношения в графе “пользователи – места”. Значения в ней это числа от 0 до 1. Теперь этот граф следует нормализовать, привести нашу матрицу к стохастическому виду(это матрица, чьи строки или колонки дают в сумме единицу) по строкам, чтобы получить матрицу переходных вероятностей для пользователей и мест. В этой матрице в каждой строке каждое значение указывает вероятность перехода пользователя или места по графу на другого пользователя (схожесть двух пользователей) или место (схожесть пользователя и места).

Построив один раз эту матрицу, можно генерировать рекомендации для каждого пользователя.

3.3 Генерация рекомендации

Здесь стартует алгоритм Random walk with Restart. Начиная с определенного пользователя происходит случайный переход по выше построенной матрице переходных вероятностей от узла к узлу каждый один шаг. Так же в каждый момент времени с вероятностей а можно вернуться в начальную точку (пользователя). В этом алгоритме данная вероятность будет мала, так как нам не важно возвращаться назад и необходимо укрепить другие связи и отношения, чтобы узнать интерес пользователя к местам из города его не постоянного пребывания.

Элементы формулы алгоритма генерации:

- пусть $p^{(t)}$ это вектор, где каждое значение $p_i^{(t)}$ отражает вероятность перехода в элемент i на шаге t.
- q – это вектор нулей с единицей в элементе, с которого начинается алгоритм, т.е. $q_x = 1$.

- матрица S представляет собой транспонированную матрицу перехода смежности, где каждый элемент S_{ij} – дает вероятность того, что j является следующим состоянием после i.

Вектор вероятностей для каждого пользователя высчитывается по следующей формуле до тех пор пока не получим устойчивый, стационарный вектор:

$$p^{(t+1)} = (1 - a)Sp^{(t)} + aq$$

Таким образом в конечном состоянии l, где вектор сходится к самому себе с некой погрешностью, $p_i^{(l)}$ возвращает релевантность x (наш пользователь) и элемента i (пользователь или место).

В данном случае для генерации предложений в системе коллаборативных рекомендаций вычислялся вектор предпочтений до момента сходимости с значением погрешности, которое искалась по следующая формуле:

$$\varepsilon = \frac{\sum_{i=1}^{n}(p_i^{(t+1)} - p_i^{(t)})^2}{\sum_{i=1}^{n}(p_i^{(t+1)})^2},$$

где n - количество всех элементов графа. Величины 0,01 для ε оказалось достаточно при решении задачи, алгоритму в основном хватало 3 шага, чтобы пройти по графу и найти релевантные места.

На основе данной формулы для каждого пользователя x можно построить предложения.

3.4 Система на основе Mahout "Taste"

В качестве технологии для реализации вышеуказанного алгоритма была выбрана библиотека "Taste" на Java, предназначенная для работы с коллаборативными рекомендациями. В ней уже были реализованы известные метрики для вычисления схожести и были предоставлен интерфейс для работы с данными, как с базы данных, так и с файловой

системы. Также интересна была она тем, что может работать как автономно на одном узле, так и на кластере, благодаря реализации поддержки Hadoop, что может помочь при работе с большими данными.

При реализации алгоритма были решены следующие задачи:

- обновлена модель данных, к привычной информации о рейтинге добавилась информация о городе места.
- для построения “мостов” между городами, был обновлен существующий алгоритм вычисления схожести пользователей в классе AbstractSimilarity и реализован интерфейс для разделения мест по группам (“по городам”)
- реализован алгоритм построения матрицы переходов Марковской цепи и алгоритм “Random Walk with Restart”
- обновлен механизм, чтения данных, для работы с идентификатором города

3.4.1 Загрузка и работа с данными

Для работы с данными был выбран файловый формат хранения и чтения данных. Стандартные коллаборативные рекомендации имеют вид: идентификатор пользователя, идентификатор объекта и оценка, если есть (нет значит используются логические предпочтения). Соответственно реализован механизм чтения в библиотеке Mahout, “Taste” – каждая строка это рекомендация с данными разделенными через запятую.

В нашем же подходе, данной информации не достаточно, необходимо дополнительно значение, идентификатор города. Для этого была переписана реализация чтения информации из файла. При этом был добавлен новый параметр и обновилась модель промежуточного хранения данных рекомендации.

Результатом стал новый класс PlaceRatingsDataModel,

унаследованный от стандартного FileDataModel, где были переопределены методы чтения файла и добавлена новая логика для решения задач:

- возвращать список мест по определенному идентификатору города для решения задачи сравнения мест по наличию в них местного этого города
- определять среди пользователей путешественников и их города для расчета схожести пользователей, т.к. мы сравниваем пользователей только по местам их общих городов.

4 Эксперименты

В качестве эксперимента работы системы рекомендации была взята задача – порекомендовать релевантные места в Нью-Йорке пользователям из Лондона, учитывая их предыдущие предпочтения и наличие пользователей, побывавших в обих городах. Для оценки платформы использовалась метрика “Точность и охват” (Precision and Recall), демонстрирующая релевантность возвращаемых системой рекомендаций. Результаты, полученные с помощью построенного алгоритма, были проанализированы и сопоставлены с известными коллаборативными подходами построения рекомендации.

Процесс вычисления, анализа алгоритмов проходил на машине с процессором Intel Core i5 и использованием 2 гигабайт оперативной памяти.

4.1 Тестовый набор

На данный момент не существует тестовых наборов для анализа и тестирования работы алгоритмов, решающих задачу построения релевантных рекомендаций по местам. Так что в качестве тестовых

данных было решено взять данные из популярного сервиса Google Places. Для этого использовалось существующее API для поиска мест, на выбранном расстоянии от определенной точки. На основе страниц мест с рекомендациями, возвращаемых по веб-адресам мест был реализован специальный механизм для вытягивания информации о рейтингах, местах и пользователях.

Реализованная система может по названию, центральным координатам города и расстоянию поиска доставать данные из сервиса Google places, строя четырех-реберный граф и сохранять их в базу данных. Написана она на C#, с поддержкой Entity Framework и базой данных MSSQL Server для хранения полученных тестовых данных.

Для экспериментирования систем рекомендации было выбраны Нью-Йорк и Лондон. В итоге для анализа использовались следующие данные:

- около 102300 оценок пользователей
- около 39000 пользователей
- около 5000 мест
- около 370 пользователей побывали в обоих городах (и там и там оставили свои оценки)
- строились рекомендации из Лондона в Нью-Йорк

В качестве тестового набор было взято 100 путешественников, имеющие минимум по 3 рейтинга в Лондоне и не меньше 4 в Нью-Йорке. Для каждого пользователя полностью удалялась информация о Нью-Йорке и каждая система должна была построить ему рекомендации в этот город. На основе удаленной информации оценивалась релевантность возвращаемых системами результатов.

Важно! Тестовые данные после использования, по правилам работы с сервисом Google были удалены.

4.2 Оценка релевантности результатов

Для оценки результатов использовалась метрика “Точность и охват” (Precision and Recall)[35][14], наиболее популярная в распознавании и поиске информации. Точность - это доля полученных экземпляров, являющихся актуальными и отзыв это доля релевантных экземпляров из всех полученных. И точность и отзыв основываются на понимании и вычислении релевантности.

Рассмотрим пример программы распознавания собак на сцене. Она определила, что 7 из 9 животных на сцене собаки. Если 4 из опознаний верны , а 3 другие из 7 на самом деле коты, тогда точность программы будет 4/7, а отзыв будет 4/9. Другой пример - поисковая система возвращает 30 страниц, из которых только 20 были похожими и ошибочно находит еще 40 дополнительных похожих страниц, тогда точность будет 20/30 = 2/3 и отзыв 20/60 = 1/3.

Точность (Precision) и отзыв (Recall) можно представить в виде следующих формул соответственно:

$$P = \frac{N_{rs}}{N_s} \text{ и } R = \frac{N_{rs}}{N_r},$$

где N_{rs} - это количество релевантных среди выбранных данных, N_s - это количество всех выбранных данных и N_r – это количество всех доступных релевантных данных.

На основе данной метрики было произведено сравнение, описываемого в работе алгоритма с стандартными коллаборативными алгоритмы “пользователь- пользователь” и “объект – объект”, используя Евклидову и корреляцию Пирсона, метрики схожести. Так как данных было мало и пересечение (количество путешественников) состовляло всего 1% от всех пользователей, было решено запускать алгоритмы для каждого из 100 пользоватей по-отдельности, удаляя полностью информацию о Нью-Йорке.

Алгоритм тестирования подходов рекомендаций выглядит

следующим образом:

1. Для пользователя u удаляются рейтинги из Нью-Йорка:

$$R'_u = \{r_{up} | \, p \in C(\text{Нью} - \text{Йорк})\}$$

Из них будут нужны при вычислении метрики “Точность и охват” только релевантные:

$$D_u = \{r_{up} | \, r_{up} \geq 3 \cap r_{up} \in R'_u\}$$

2. Запускается алгоритм вычисления рекомендаций пользователю u
3. На выходе получаем в порядке убывания список рекомендации по всем местам, из которых берутся только места из Нью-Йорка:

$$\text{F} = \{\{r'_{up} | \, p \in C(\text{Нью} - \text{Йорк})\}\}$$

4. Вычисляем метрику “Точность и охват” (Precision and Recall), для этого пробегаем по списку F и находим:

k_1, где recall = 0,1,

k_2, где recall = 0,2,

...

k_{10}, где recall = 1 ,

k_i – это количество элемнтов выбранных из списка при поиске релевантных элементов из D_u и считаем для всех precision. В итоге получаем для пользователя вектор из 10 чисел. При этом если нет какого-нибудь k_i, то дальше не ищем, так как больше не будет найдено релевантных.

5. Востанавливаем удаленные рейтинги пользователя u, берем следующего пользователя и идем к шаг 1.

В конце для каждого алгоритма вычислялась средняя величина “Precision and Recall” по всем 100 векторам пользователям. На рисунке 2 отражены результаты работы каждого из алгоритмов с различными мерками схожести. На рисунке идеальной точкой является точка (1,1) и худшая (1,0). Чем график ближе к оси охват (recall), тем больше среди возвращаемых данных “мусора”. Если график не доходит до 1 по охвату и

заканчивается раньше - это значит, что для > 30% пользователей не было найдено больше релевантных мест.

В итоге получили, что наш алгоритм (RWR) лучше всего справляется с задачей поиска релевантных туристических достопримечательностей, а лучшей метрикой для определения схожести оказалась Евклидова метрика. Как видно из графика, предложенный алгоритм возвращает все релевантные данные первыми. Такой высокий показатель достигается потому, что в топ попадают достаточно известные места, о которых пользователь бывает скорее всего в курсе. На данный момент не существует метрик, которые оценивают новизну рекомендации. Но в плане новизны лучше рассмотреть алгоритм Пирсона.

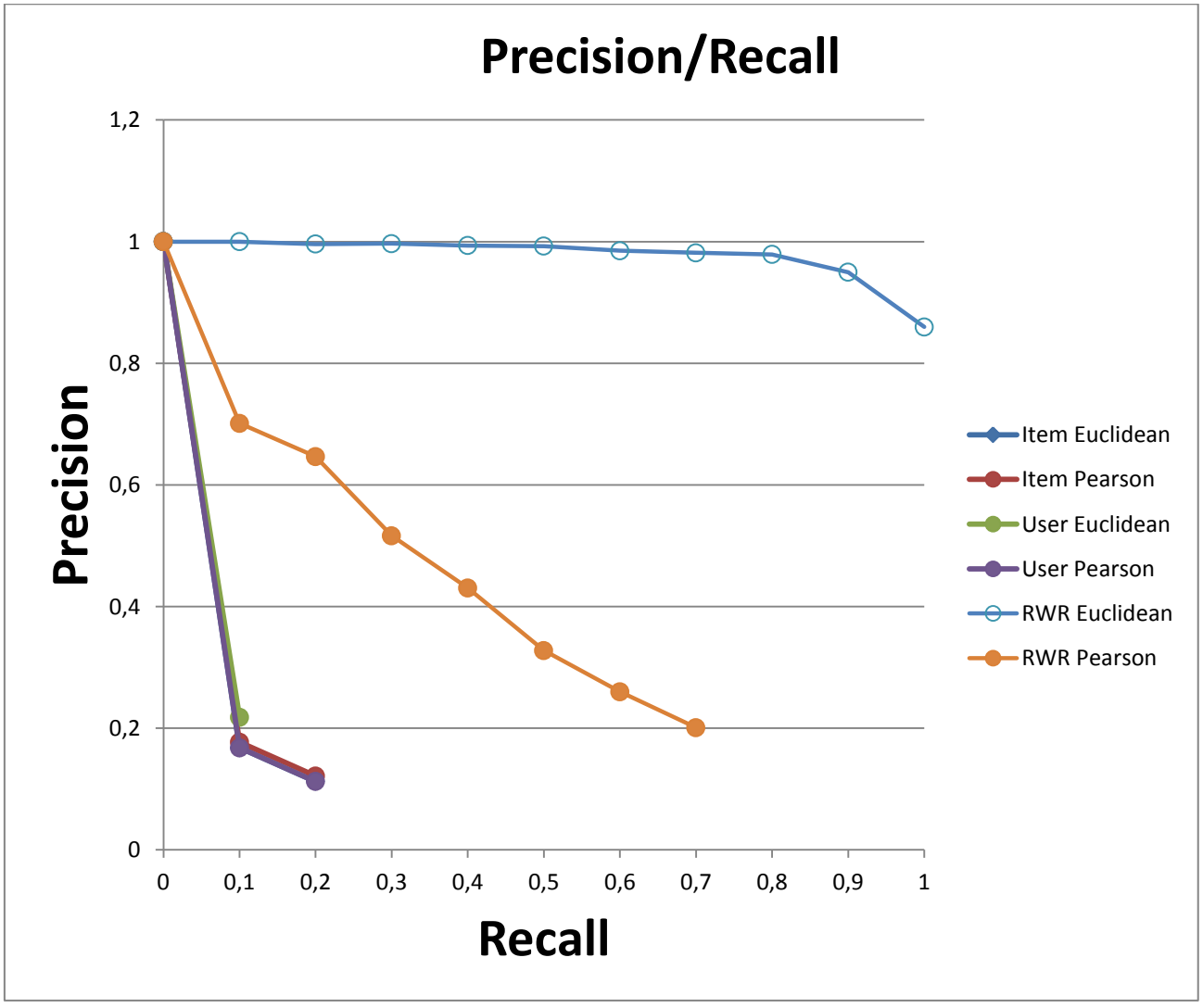

Рисунок 2 “Оценка результатов работы алгоритмов рекомедации с помощью метрики Precision Recall.”

Для проверки результатов был исследован один из 100 путешественников, который оценил 17 мест в Нью-Йорке и 13 из них отметил релевантными для себя. Среди мест в основном были отели, кафе и рестораны, которые оказались достаточно популярными (в среднем 90 оценок на место, а в сумме всего 1626 оценок). Судя по этим оценкам такие высокие показатели, полученные при проверки алгоритма на наш взгляд можно объяснить следующими причинами:

1. Наличие популярных мест, для которых высока концентрация рейтингов путешественников, и которые попадают в рекомендации. Большинство из мест выбранного пользователя, также посетили и другие путешественники и отметили, что данные места им нравятся.
2. Наличие связей между местами (рекомендации). Путешественники, оставившие оценки в разных ресторанах, имели общую связь через один или несколько отелей. В таком случае могла повлиять реклама мест в отеле (скидочные программы на ресторан от отеля).
3. Однородность аудитории - пользователи Android любящие регистрировать себя в местах (check-in) с мобильных телефонов. Google Places, как и другие сервисы Google используют в основном на этой платформе.

Заключение

Результаты работы

В рамках работы была разработана система коллаборативных рекомендаций туристических достопримечательностей на основе алгоритма «Random Walk with Restart».

Все задачи, перечисленные в разделе 1.4 были успешно выполнены. Основные результаты:

- Проанализированы существующие решения и подходы к рекомендациям
- Разработан алгоритм вычисления схожести между пользователями и местами, позволяющий идентифицировать сходство даже при условии недостатка статистики
- Реализован инструмент генерации рекомендации
- Проведено сравнение с стандартными коллаборативными алгоритмами на собранных тестовых данных, результаты которого показали достаточно высокий результат работы разработанного алгоритма.

Дальнейшие исследования

Планируется дальнейший анализ и усовершенствование алгоритма построения туристических рекомендации:

- Проведение экспериментов на большем количестве городов и оптимизации работы системы
- Расширение графа отношений «Пользователи — Места», добавляя теги и категории характеризующие места, что повысит релевантность возвращаемых данных
- Поддержка в алгоритме информации о регистрациях (check-in) пользователей на местах
- Поиск решения проблемы холодного старта (отсутствие информации о пользователе)

Естественным и крайне важным продолжением работы является добавление поддержки Hadoop для возможности масштабировать текущую систему, совершать вычисления на кластерах состоящих из нескольких узлов.

Также на основе данной системы, планируется начать реализацию системы построения туристических маршрутов наиболее релевантных пользователю.

Список литературы

1. Amazon, July, 1994. http://www.amazon.com/
2. Anil R., Dunning T., Owen S. Mahout in Action. Manning Publications Co. 2012. P. 11 – 115
3. AngelPad, August, 2010. http://angelpad.org/about/
4. Bayesian Network, http://en.wikipedia.org/wiki/Bayesian_network
5. Collaborative Filtering, http://en.wikipedia.org/wiki/Collaborative_filtering
6. Ehmig C., Groh G., Recommendations in taste related domains: collaborative filtering vs. social filtering // GROUP, 2007. P. 127-136
7. Facebook, February, 2004. http://www.facebook.com/
8. Foursquare, March, 2009. https://ru.foursquare.com/about/
9. Friend Of A Friend (FOAF), http://www.foaf-project.org
10. Ghita S., Nejdl W., Paiu R. Semantically Rich Recommendations in Social Networks for Sharing, Exchanging and Ranking Semantic Context // International Semantic Web Conference, 2005. P. 293-307
11. Golbeck J., Hendler J. Movie Recommendations using trust in web-based social networks, 2006.
12. Google Places, http://support.google.com/hotpot/bin/static.py?hl=en&page=guide.cs&guide=29942
13. Hadoop, 2005. http://hadoop.apache.org/
14. Herlocker J., Konstan J. , Terveen L. Evaluating collaborative filtering recommender systems. // ACM Trans. Inf. Syst. (TOIS) 22(1), 2004. P. 5-53
15. Instagram, October, 2010. http://instagr.am/
16. IMDb, October, 1990. http://www.imdb.com/
17. Jinni, 2008. http://www.jinni.com/
18. Jose J., Konstas I., Stathopoulos V., On social networks and collaborative recommendation // SIGIR, 2009. P. 195-202

19. Karypis G., Konstan J., Sarwar B. Item-based collaborative filtering recommendation algorithms // WWW, 2001. P. 285-295
20. Karypis G., Konstan J., Sarwar B., Analysis of recommendation algorithms for e-commerce. // ACM Conference on Electronic Commerce, 2000. P. 158-167
21. Kim H., Kim J., Ryu J. TripTip: a trip planning service with tag-based recommendation. // CHI Extended Abstracts, 2009. P. 3467-3472
22. Koren Y. The BellKor Solution to the Netflix Grand Prize, 2010. http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
23. Last.fm, 2002. http://www.last.fm/
24. Lee S., Park S., Yang J. Discovery of Hidden Similarity on Collaborative Filtering to Overcome Sparsity Problem. // Discovery Science, 2004. P. 396-402
25. Mahout, http://mahout.apache.org/
26. Markov Chain, http://en.wikipedia.org/wiki/Markov_chain
27. Markovsky I. Low Rank Approximation: Algorithms, Implementation, Applications. // Springer, 2012
28. MapReduce, http://en.wikipedia.org/wiki/MapReduce
29. Mooney R., Roy L. Content-based book recommendation using learning for text categorization // ACM DL, 2000. P. 195-204
30. Netflix, August, 1997. http://www.netflix.com/
31. Németh B., Pilászy I., Takács G. Matrix factorization and neighbor based algorithms for the netflix prize problem // RecSys, 2008. P. 267-274
32. Pandora Radio, January, 2000. http://www.pandora.com/
33. Papagelis M., Plexousakis D. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. // Eng. Appl. Of AI (EAAI), 2005. P. 781-789
34. Pearson Correlation, http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

35. Precision and Recall, http://en.wikipedia.org/wiki/Precision_and_recall
36. Rennie J., Srebro N. Fast maximum margin matrix factorization for collaborative prediction. // ICML, 2005. P. 713-719
37. Recommender System, http://en.wikipedia.org/wiki/Recommender_system
38. Rotten Tomatoes, 1998. http://www.rottentomatoes.com/
39. SpotSetter, May, 2012. http://www.spotsetter.com/
40. "Taste" https://cwiki.apache.org/confluence/display/MAHOUT/Recommender+Documentation
41. Twitter, March, 2006. http://twitter.com/
42. Yelp, 2004. http://www.yelp.com/