Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Рекомендации треков в социальных сетях

Магистерская диссертация

студента 661 группы

Александра Александровича Дзюба

Научный руководитель	.....	к.ф.-м.н., Д.Ю. Бугайченко
	/подпись/	
Рецензент	.....	к.ф.-м.н., доц. И.П. Соловьев
	/подпись/	
“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

Track recommendations in social networks

by

Aleksandr Aleksandrovich Dzyuba

Master's thesis

Supervisor	.....	PhD D.U. Bugaychenko
	/Signature/	
Reviewer	.....	PhD I.P. Soloviev
	/Signature/	
"Admitted to proof"	.....	Professor A.N. Terekhov
Head of Chair	/Signature/	

Saint Petersburg

2012

Содержание

1 Введение

Большинство социальных сетей позволяет пользователям публиковать различные объекты: видео, треки, документы. Некоторые, например Last.fm, предоставляют пользователю доступ к уже существующему контенту. В том и другом случае количество подобных объектов настолько велико, что пользователю сложно найти среди них интересующую его информацию путем обычного просмотра. Рекомендательные системы решают эту задачу, предсказывая предпочтительность объекта для конкретного пользователя, основывая свой прогноз на данных, указанных пользователем явно, или собранных из истории его взаимодействия с социальной сетью. От этого повышается привлекательность как самой сети, так и её содержимого.

Для составления рекомендаций обычно используются методы коллаборативной фильтрации [1]. Когда требуется предсказать, насколько понравится пользователю новый объект, для этого используются оценки похожих пользователей. Эти методы хорошо известны и применяются во многих проектах: Netflix, Amazon, Last.fm. За последние годы качество алгоритмов этого типа значительно увеличилось. Толчком для этого послужило соревнование на лучший алгоритм предсказания, проведенное Netflix [2, 3]. Тем не менее, до сих пор актуально решение проблемы «холодного старта» системы для новых пользователей и объектов, качество и скорость составления рекомендаций.

Как альтернатива коллаборативной фильтрации может быть использован алгоритм Random Walk with Restarts [4]. Он превосходит стандартные методы фильтрации по качеству, позволяя включать в рекомендацию разнородные знания из социальной сети. При этом данный метод универсален, он пригоден для работы с объектами любого типа: треки [4], фотографии [5]. К недостаткам алгоритма Random Walk with Restarts (RWR) можно отнести его вычислительную сложность.

В этой работе исследована применимость указанного алгоритма в крупных социальных сетях для рекомендации музыкальных треков. Предложены модификации оригинального алгоритма, позволяющие улучшить качество рекомендаций. В первую очередь, это способы включения в граф RWR различной информации из социальной сети, влияющей на скорость и качество работы алгоритма, а также способы предобработки этой информации.

Предложен способ составления рекомендации в режиме реального времени по запросу пользователя. Оригинальный алгоритм при работе использует структуры данных, которые приобретают колоссальные размеры в реальных социальных сетях. Представлен «облегченный» вариант алгоритма, описаны качество и время его работы.

Несмотря на то, что методы предложены для рекомендации музыкальных треков, большинство из них универсальны и могут быть использованы для других объектов социальных сетей.

2 Постановка задачи

- Реализовать рекомендательную систему на основе алгоритма Random Walk with Restarts [4].
- Использовать традиционные методы расчёта рекомендации для составления графа социальной сети.
- Оценить качество рекомендаций для разных способов построения графа, используя данные реальных социальных сетей.
- Предложить метод ускорения расчёта рекомендаций для применения в крупных социальных сетях.
- Оценить время и качество рекомендаций при использовании предложенного метода.

3 Обзор предметной области

В области разработки рекомендательных систем сложились традиционные подходы, обладающие рядом достоинств и недостатков. Так или иначе, каждый рекомендательный алгоритм стремится дать пользователю более релевантные рекомендации максимально быстро. В этом преуспел, среди прочих, алгоритм Random Walk with Restarts (3.2). Кроме того, качество рекомендации оценивается различными методами, которые рассмотрены в параграфе 3.3.

3.1 Подходы к рекомендациям

Рекомендательные системы предназначены для поиска объектов, которые понравятся пользователю или будут ему полезны. В типичных системах есть список пользователей $U = (u_1, u_2, \ldots, u_m)$ и предметов $I = (i_1, i_2, \ldots, i_n)$. В ходе взаимодействия с системой пользователи знакомятся с объектами, формируя *матрицу рейтингов* R, где $r_{u_a,i}$ – рейтинг предмета $i \in I$ у пользователя $u_a \in U$. Как правило, матрица рейтингов сильно разрежена, т.к. количество различных предметов в системе велико и уже известные u_a предметы $I_{u_a} \subseteq I$ составляют малую долю от общего количества. Задача рекомендательной системы обычно формулируется как вычисление *предсказания* и *рекомендации*. Пользователь, для которого они вычисляются, далее будет называться *активным*.

Предсказание – численное значение $P_{u_a,i}$, выражающее предсказанную предпочтительность предмета $i \notin I_{u_a}$ для активного пользователя u_a. Предсказанное значение лежит внутри заранее определенного интервала рейтинга, например от 1 до 5.

Рекомендация – список из N предметов $I_r \subset I$, наиболее предпочтительных для активного пользователя, причем в него входят лишь незнакомые пользователю элементы $I_r \cap I = \varnothing$. Задачи в такой постановке также называют *Top-N рекомендацией*.

Способы формирования матрицы рейтингов R бывают явные и неявные. Первые предполагают, что пользователь, ознакомившись с предметом, выставит ему оценку по имеющейся шкале (так работает, например, Netflix). В таких случаях R составлена из оценок в чистом виде. Однако этот подход применяется не везде. Часто для составления матрицы приходится анализировать историю взаимодействия пользователей с предметами, в этом случае имеет место неявное формирование рейтинга. В случае с треками, величина $r_{u_a,i}$ может вычисляться как количество прослушиваний.

3.1.1 Коллаборативная фильтрация

Идея *коллаборативной фильтрации* заключается в формировании списка рекомендованных объектов на основе мнений пользователей, ведущих себя похожим образом. Анализируя профили, рекомендательная система находит таких пользователей, после чего оценка

предпочтимости нового объекта рассчитывается с использованием их оценок. Существует два подхода к коллаборативной фильтрации: основанные на пользователях *(user-based)* и на предметах *(item-based)*. Первые оперируют схожестью пользователей, вторые – схожестью предметов. В описанной модели схожесть рассчитывается как коэффициент корреляции между многомерными векторами, векторами-рейтингами треков для user-based и векторами рейтингов трека у различных пользователей для item-based методов. Для этого может использоваться коэффициент корреляции Пирсона или косинус угла между векторами. Дополнительным плюсом для них является их нормированность, так как значения укладываются в $[0, 1]$. Для item-based методов хорошо зарекомендовал себя уточненный косинус угла [1], в котором из рейтингов $r_{u,i}$ вычитаются средние значения рейтинга $\bar{r_u}$ для данного пользователя, как показано в формуле (1). Это помогает учесть различные подходы к составлению рейтинга у пользователей, одни из которых могут оперировать лишь высокими оценками, а другие выставлять их лишь немногим предметам. В случае user-based подхода, это учитывается при расчете корреляции рейтингов разных пользователей.

$$sim(i,j) = \frac{\sum_{u \in U_{i,j}} (r_{u,i} - \bar{r_u})(r_{u,j} - \bar{r_u})}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r_u})^2} \sqrt{\sum_{u \in U_{i,j}} (r_{u,j} - \bar{r_u})^2}} \qquad (1)$$

Здесь $U_{i,j} \subseteq U$ – множество пользователей, оценивших как предмет i, так и предмет j.

Оценив схожесть между пользователями или треками, можно сосчитать предпочтительность нового предмета, учитывая отклонения его оценок от уже оцененных пользователями элементов (для item-based) или его оценки похожими пользователями (для user-based).

Главными проблемами коллаборативной фильтрации являются размер и разреженность матрицы рейтингов, о которых уже было упомянуто, и проблема холодного старта – расчет рекомендации для новых пользователей и новых предметов. Методы фильтрации содержимого этих недостатков лишены. Тем не менее, тот факт, что учитывается история предыдущих оценок, делает коллаборативную фильтрацию крайне эффективной и популярной. В последнее время большинство исследований в области коллаборативной фильтрации были сосредоточены вокруг соревнования Netflix Prize [2], в рамках которого участники оптимизировали алгоритм предсказания рейтингов фильмов.

3.1.2 Фильтрация содержимого

Методы *фильтрации содержимого* основываются на характеристиках предмета и известной о нем информации. Эти данные сопоставляются с профилем пользователя, который может содержать различную демографическую информацию, а также явно указанные музыкальные вкусы. Кроме того, на основе этих характеристик могут быть найдены объекты, похожие на те, что понравились пользо-

вателю в прошлом. Эти методы используются для решения проблемы холодного старта в Last.fm [6]. Пользователь при регистрации указывает некоторые музыкальные предпочтения, после чего ему уже можно порекомендовать некоторые треки и альбомы.

Сложности данного подхода заключаются в составлении списка характеристик предмета. Например в подходе Music Genom Project [7], использующемся в интернет-радио Pandora[1], для треков выделено около 400 характеристик (генов), таких как пол вокалиста или уровень *distorsion*[2] электрогитары, вычисляемых опытными экспертами. Задание значений различных характеристик вручную чрезмерно трудоемко и порождает конфликты с правообладателями треков. Рекомендуемые предметы могут быть легальными, как в случае с Last.fm, а могут загружаться самими пользователями, и факт их ручного анализа переносит вину за использование пиратского контента на администраторов системы.

Кроме того, для сопоставления профилей предметов и пользователей последним приходится много рассказывать о себе, что создает неудобства, в то время как методы коллаборативной фильтрации могут работать совершенно скрыто.

[1] http://www.pandora.com

[2] Звуковой эффект, достигаемый искажением сигнала путём его ограничения по амплитуде.

3.1.3 Социальные подходы

В социальных сетях, кроме истории взаимодействия пользователей и предметов, также известны социальные связи пользователей. В ряде случаев они могут успешно заменить схожесть между пользователями в user-based методах коллаборативной фильтрации и других методах. Данные методы успешно используются в специализированных социальных сетях, где связь между пользователями означает не просто знакомство, но и некую схожесть в предметной области данной сети. Примером может служить рекомендация ночных клубов в посвященной им сети [8], в которой оценка схожести пользователей на основе социальных связей превзошла аналогичную оценку для коллаборативной фильтрации. При рекомендации треков для Last.fm использование социальных связей также даёт положительный эффект [4].

В случаях, когда социальная сеть не имеет определенной тематики (Facebook, Twitter), использовать социальные связи следует крайне осторожно, т.к. они обычно не означают схожести интересов. Однако можно воспользоваться косвенными оценками силы связей, таких как количество сообщений, общих фотографий или общими указанными интересами.

3.1.4 Известные рекомендательные системы и алгоритмы

Заметное влияние на развитие коллаборативной фильтрации оказал конкурс Netflix Prize, проведенный американской компанией Netflix, занимающейся прокатом DVD и интернет-трансляцией фильмов. Пользователи оценивают просмотренные фильмы по 5-бальной шкале, и, накопив значительное количество оценок, Netflix разработал рекомендательный алгоритм Cinematch, основанный на линейной регрессии.

Накопленные оценки были разделены на тренировочное и проверочное (скрытое) множества. Алгоритм Cinematch на проверочном множестве улучшал тривиальную оценку, рассчитанную как средняя оценка фильма, на 10%. Улучшение оценивалось согласно среднеквадратичному отклонению от реальной оценки. За улучшение алгоритма еще на 10% Netflix объявил награду в 1 млн. $. Соревнование началось 2 октября 2006 года и завершилось 26 июня 2009 года. Победителем стал алгоритм [9], учитывающий психологические факторы (характер, настроение пользователя) при выставлении рейтинга, рассчитываемые при помощи статистических методов. Также алгоритм агрегирует результаты нескольких предсказаний при помощи градиентного ускорения для деревьев принятия решений [9, 10].

Для уменьшения сложности коллаборативной фильтрации часто используются алгоритмы из семейства Slope One [11]. Они используют простую схему связи между оценками пользователей, уже оценив-

ших предмет, и рассчитываемой оценки для активного пользователя. Связь имеет вид: $f(x) = x + b$, где b – константа.

В классической схеме коллаборативной фильтрации для перехода от оценок одних пользователей к оценке активного пользователя (или от оценок активного пользователя других предметов) используется линейная регрессия в виде $f(x) = ax + b$. В ситуации, когда количество похожих пользователей или предметов велико, упрощение Slope One существенно уменьшает сложность вычислений при сохранении точности предсказания [11]. Алгоритм реализован на многих языках программирования и широко применяется в рекомендательных системах.

3.2 Алгоритм Random Walk with Restarts

В качестве главного объекта исследования выбран алгоритм Random Walk with Restarts [4]. Он позволяет давать рекомендации на основе разнородных данных социальной сети, таких как история прослушивания треков, социальные связи, схожесть музыкальных вкусов, теги, которыми пользователи помечают треки, и многое другое. Эти данные представлены в алгоритме в виде графа отношений между объектами. В простейшем случае такими объектами являются пользователи и треки. Они и связи между ними являются вершинами и дугами графа соответственно, веса дуг определяются силой связанности. Такая структура графа позволяет включать в него раз-

личные виды объектов: теги, исполнители, музыкальные жанры и списки воспроизведения, в зависимости от реализации сервиса прослушивания музыки.

Начиная в вершине x, RWR на каждом шаге выполняет случайный переход от текущей вершины к новой по инцидентным дугам, причем вероятность перехода прямо пропорциональна весу дуги. При этом каждый раз с фиксированной вероятностью a обход может вернуться в начальную точку x. Пусть $\mathbf{p}^{(t)}$ вектор-столбец, в котором $p_i^{(t)}$ означает вероятность того, что на шаге t RWR находится в вершине i. Пусть $\mathbf{q}$ вектор-столбец, в котором $q_x = 1$, а остальные элементы равны нулю, и $\mathbf{S}$ – нормализованная по столбцам матрица смежности графа. Тогда стационарное состояние алгоритма может быть получено применением (2) до сходимости:

$$\mathbf{p}^{(t+1)} = (1 - a)\mathbf{S}\mathbf{p}^{(t)} + a\mathbf{q} \tag{2}$$

Стабильное состояние даст нам вероятности оказаться в каждой вершине в долгосрочной перспективе. Таким образом $p_i^{(l)}$, где l – номер шаг после достижения сходимости, может быть рассмотрена как мера связанности вершин i и x. Если в качестве начальной вершины взять пользователя u, то её связанность с треками можно считать мерой предпочтимости этих треков. Матрица смежности графа $\mathbf{S}$ одинакова для всех пользователей и, единожды загрузив её в память, можно провести все необходимые расчеты, меняя вектора $\mathbf{p}$ и $\mathbf{q}$ для разных пользователей.

Алгоритм Random Walk with Restarts находит применение в информационном поиске, анализе ссылок, классификации текстов и рекомендательных системах. Как было сказано ранее, граф социальной сети можно включать информацию из различных источников. Вершинами могут выступать пользователи, треки, теги и другие объекты сети. Алгоритмы расчёта весов соединяющих их дуг могут основываться на совершенно разных соображениях. В оригинальной версии алгоритма [4] лучшие результаты показал социальный граф, включающий пользователей **U**, треки **Tr**, теги **Tg**, и отношения **UU**, **UTr**, **UTg**, **TgTr**, а также обратные к ним **TrU** и другие (см. 1). Такой подход к построению графа дал лучшие результаты, нежели

UU	UTr	UTg
UTr	0	TgTr
UTg	TgTr	0

Рис. 1: Матрица смежности социального графа.

стандартный способ коллаборативной фильтрации, описанный в параграфе 3.1.1. Вес дуги такого графа, соединяющей пользователя и

трек, определялся пропорционально количеству прослушиваний этого трека пользователем (эту информацию обычно можно извлечь из профиля в социальной сети). Для взвешивания дуг отношения **UU** применялся следующий подход: ненулевой вес имели дуги лишь между пользователями, находящимися в социальной связи. Вес дуги при этом пропорционален среднему количеству прослушиваний пользователя. Таким образом наиболее авторитетными оказывались люди, слушающие больше музыки. Такой подход не всегда точно отображает силу социальной связи и общность музыкальных вкусов. В 5.1 описаны другие способы расчета весов и приведено сравнение различных подходов.

3.3 Оценка качества рекомендательных систем

Задача рекомендательной системы – оценить интерес пользователя к тому или иному объекту. Таким образом, наиболее достоверный способ оценки качества – это опрос пользователей. По понятным причинам он может осуществляться далеко не всегда, а на ранних этапах построения и тестирования системы невозможен. Поэтому популярный подход к оценке и тестированию заключается разделении истории взаимодействия пользователей с треками на две части – *тренировочную* и *проверочную* [2, 4, 12]. Тренировочная используется для «обучения» системы, в случае с алгоритмом RWR – расчёт весов дуг в графе, а проверочная – для оценки качества с помощью

различных метрик. Пропорции такого разделения требуют исследования для каждого тестового набора данных, так как от них может зависеть, какой метод покажет себя наиболее качественно [12].

При этом оценка пользователями трека может быть явной – выставление рейтинга по шкале (Netflix), но в большинстве социальных музыкальных сервисов такая возможность отсутствует. В таком случае за оценку предпочтимости трека берут количество его прослушиваний, а проверочное множество считается множеством релевантных треков. В задачах предсказания рейтинга элементов у пользователей обычно используется средняя величина ошибки по модулю или среднеквадратичное отклонение от реального рейтинга, известного для всех элементов проверочного множества [2, 12]. Для задач типа Top-N рекомендации требуются другие методы оценки, такие как отношение точности и полноты или *Half-life utility*.

3.3.1 Точность и полнота

Важную роль играют метрики оценки списка Top-N рекомендации, в котором для каждого элемента известна его релевантность. Традиционно в информационном поиске принято использовать *точность* и *полноту* [4, 12].

Точность (*Precision*) – доля релевантных элементов в найденном.

Полнота (*Recall*) – доля найденных релевантных элементов в их общем количестве.

Для отображения распределения релевантных элементов внутри списка принято использовать Recall-Precision кривые. Для их расчета находится точность для для фиксированных значений полноты (например $\{0.1, 0.2, \ldots, 1\}$), и строится интерполирующая функция. Такие функции для разных прогнозов удобно сравнивать с помощью метрики *Average Precision*, определяемой как:

$$AveP = \int_0^1 p(r)d(r), \tag{3}$$

где p – точность, r – полнота, или дискретным аналогом этого интеграла. Для нескольких рекомендаций (для разных пользователей) используется средняя величина Average Precision:

$$MAP = \frac{\sum_{q=1}^{N} AveP(q)}{N} \tag{4}$$

Однако данная метрика не учитывает такой важный фактор, как порядковый номер релевантного элемента. В данном случае любая перестановка первых элементов, на которых достигается значение r равное 0.1, дает одинаковую интерполирующую зависимости точности от полноты, в то время как релевантные элементы, расположенные в начале списка более важны чем аналогичные им в конце, так как пользователь начинает знакомится с ними в порядке выдачи. Для учета этого фактора пригодна метрика *Half-life utility*.

3.4 Half-life utility

Для оценки практической применимости системы больше подходит метрика Half-life Utility [12], имитирующая радиоактивный распад[3]. Чем дальше релевантный элемент от начала списка рекомендованных элементов, тем меньше его вес. И вес элементов, стоящих друг от друга на расстоянии *периода полураспада* c, отличается в 2 раза.

$$HLU_a = \frac{\sum_j r_{a,j}}{2^{(j-1)(c-1)}}, \tag{5}$$

где a – активный пользователь и $r_{a,j}$ – рейтинг элемента j для .

Это хорошо имитирует ситуацию получения рекомендаций в социальной сети, когда у пользователя нет возможности ознакомится с большим количеством рекомендованных треков. В данной работе для сравнения рекомендаций использовалась преимущественно эта метрика.

3.5 Выводы

Рекомендательные системы предлагают нам неизвестный контент, выбирая из массы доступных предметов наиболее интересные для каждого пользователя. Для этого они используют алгоритмы коллаборативной фильтрации и фильтрации содержимого, каждые из

[3]По закон радиоактивного распада число испусканий частиц за интервал времени пропорционально числу имеющихся частиц. Такой процесс характеризуется периодом полураспада (half-life), за который количество нераспавшихся частиц уменьшается в два раза.

которых наделены своими достоинствами и недостатками. Ключевыми факторами в данном случае являются качество и сложность расчета рекомендаций, от которых зависят интерес пользователя к сервису и объем продаж, если рекомендуемый контент впоследствии продается.

Алгоритм Random Walk with Restarts, примененный для рекомендации треков в социальных сетях, превосходит по качеству методы коллаборативной фильтрации. Он достигает этого за счет успешного сочетания в графе социальной сети информации из различных источников. Этот алгоритм работает в условиях, в которых алгоритмам фильтрации содержимого не хватит данных для качественной работы.

Размер графа социальной сети, с которым работает алгоритм, ставит под сомнение его применимость в крупных сетях. В данной работе предложены методы улучшения качества Random Walk with Restarts, а также исследованы способы его упрощения.

4 Особенности реализации

В данном разделе описаны детали реализации рекомендательных систем и их тестирования. Работа алгоритма Random Walk with Restarts требует большого количества настроек, а также методов взаимодействия с набором данных и оценки качества рассчитанной рекомендации. Для выполнения этих задач была реализована Java-платформа, описанная в 4.2. Кроме того, дана характеристика набора данных, на котором тестировались рекомендательные системы, а также сторонних библиотек, оказавшихся полезными при реализации алгоритмов.

4.1 Тестовый набор данных

Для тестирования алгоритма рекомендации и оценки влияния различных модификаций на его качество был использован набор данных[4], подготовленный в работе [4]. Он включает в себя историю 29 млн. прослушиваний треков в социальной сети Last.fm [6] и состоит из 3148 пользователей, 30520 треков и 12565 тегов, которыми помечены треки. В этот набор данных были включены лишь пользователи, сильно вовлеченные в социальную сеть и наиболее популярные у них треки и теги. Социальный граф для алгоритма Random Walk with Restarts строится с помощью выборки из подобного набора. В

[4]http://www.dcs.gla.ac.uk/~jj/data/lastfm dataset.htm

нем треки, теги и пользователи представлены в виде вершин, а отношения между ними – в виде дуг. Характеристики данного графа и включенных в него отношений приведены в таблице 1.

Характеристика	**UU**	**UTr**	**UTg**	**TgTr**
Мощность	5616	727168	33259	115351
Плотность	$1.2 * 10^{-3}$	$7.6 * 10^{-3}$	$8.4 * 10^{-4}$	$3 * 10^{-4}$
Максимальная степень вершин в графе				
Прямое отношение	26	465	922	5
Обратное отношение	26	2619	433	9172

Таблица 1: Связи в наборе данных Last.fm.

Кроме того, в наборе данных есть информация о количестве прослушиваний пользователем каждого трека в Last.fm. Она используется для взвешивания дуг типа User-Track, а в оригинальной работе – еще и User-User. Хотя, как показали проведенные эксперименты, есть и более эффективные способы измерения близости пользователей.

Для оценки качества рекомендаций история прослушиваний была разделена на тренировочную и проверочную части в соотношении 2 к 3 по размеру (см. раздел 3.3).

4.2 Платформа тестирования рекомендаций

Эксперименты по исследованию рекомендательных систем состоят из нескольких этапов, каждый из которых может осуществляться с помощью различных подходов, алгоритмов, оперировать разными данными. Можно выделить следующие классы задач, возникающие при проведении экспериментов:

- взаимодействие с набором данных, как правило хранящемся в базе данных;
- выполнение алгоритма RWR, расчет Top-N рекомендации;
- оценка полученных результатов.

Для настройки и тестирования рекомендательных систем, основанных на алгоритме Random Walk with Restarts была создана специальная Java-платформа, дающая возможность гибкого конфигурирования этих процессов.

4.2.1 Работа с данными

Самым приемлемым вариантом хранения тестового набора является база данных (в данной работе это БД под управлением Microsoft SQL Server). Кроме того, он может храниться в виде .csv-файлов, но этот подход менее удобен. Методы, общие для разнородных наборов данных описывает интерфейс Dataset (см. рис. 2). За взаимодействие с БД с помощью JDBC-драйвера отвечают наследники класса

DatabaseDataset. Конкретные расчеты и выборки, связанные с набором данных Last.fm (4.1), реализованы в LastfmDataset.

Для базовой версии алгоритма Random Walk with Restarts, описанной в 3.2, от этих классов требуется, в первую очередь, составление матриц смежности социального графа сети и выборка объектов, связанных с конкретным пользователей (вектор рестарта, начальное приближение алгоритма). Такого рода объекты хранятся в структурах данных PreferencesVector, используемых не только для хранения предпочтений пользователя, но и для отображения связанных с ним элементов (связанным инцидентным объектам соответствуют веса дуг, остальным – нули). Фрагменты матрицы смежности графа представлены наследниками класса Submatrix, выборка которых осуществляется в методах DatabaseDataset. Общая схема взаимодействия и зависимости классов показана на рис. 2.

4.2.2 Оценка результатов

Оценка рекомендации для одного пользователя, как правило, не дает представления о качестве алгоритма, поэтому классы, реализующие интерфейс EvaluationMetric, считают средние значения метрики для списка пользователей. В данной работе использовались метрики Half-life Utility и Recall-Precision-кривые (по которым считалась Average Precision (3)). Кроме того, для оценки качества симуляции Random Walk with Restarts оказалась удобна метрика Precision-N,

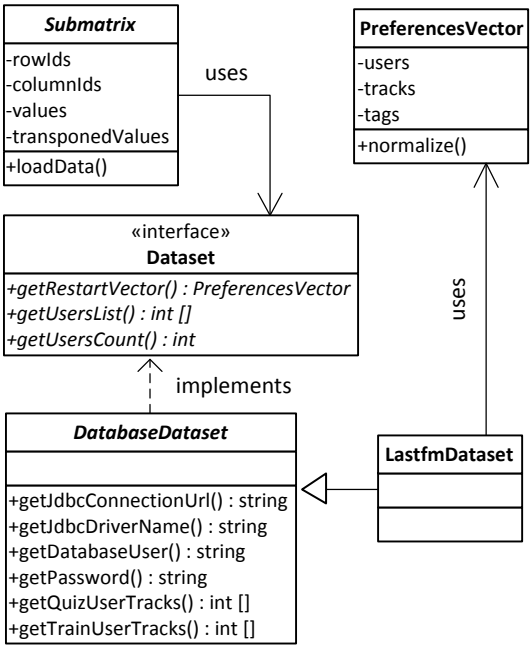

Рис. 2: Диаграмма классов для оценки качества рекомендации.

считающая точность Top-N рекомендации.

Принцип разделения истории прослушиваний на тренировочное и проверочное множество реализован в классе QuizSetVerification. Его методы используются для определения релевантных треков при расчете метрик, а также подготовки списка Top-N рекомендации (рис. 3).

4.2.3 Выполнение алгоритма рекомендации

Содержательная часть расчетов алгоритма заключается в умножении матрицы смежности социального графа на вектор предпо-

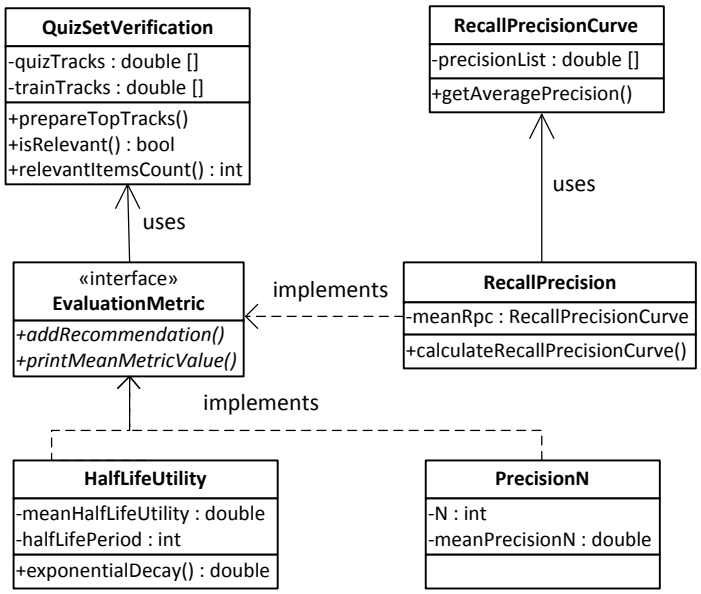

Рис. 3: Диаграмма классов для оценки качества рекомендации.

чтений пользователя, предварительно выбранных из набора данных (4.2.1). Каждая подматрица смежности (рис. 1) является наследником класса Submatrix (рис. 4). Вместе они формируют матрицу целого графа – интерфейс SocialGraph. Классы, реализующие этот интерфейс – это различные комбинации отношений, которые можно в него включить:

- **UTr, TrU** – UserTracks,
- **UTr, TrU, UU** – UserTracksSimilarUsers,
- **UTr, TrU, UU, TrTr** – CompleteUserTrackRelation.

Численные характеристики каждого из этих отношений хранятся в объектах типа Submatrix. Там же реализовано умножение подматрицы на часть вектора предпочтений – PreferencesVector. Работа алгоритма представлена в классе RandomWalkWithRestarts. Сначала формируется SocialGraph и начальный вектор предпочтений, для этого происходит обращение к набору данных с помощью соответствующих классов. После чего полученные данные перемножаются по формуле (2) до сходимости. На основе итогового вектора предпочтений строится рекомендация, которая оценивается по выбранным метрикам.

4.3 Расчет схожести векторов-рейтингов

Как видно из описания набора данных 4.1, в нём отсутствует информация о связях между треками. Социальные связи между пользователями в наличии, однако нет данных о силе этой связи и о характере – люди с разными музыкальными вкусами могут быть связаны на основании личного знакомства. Поэтому возникла идея добавить в граф алгоритма Random Walk with Restart недостающую связи типа User-User и Track-Track, которую можно рассчитать по истории прослушиваний с помощью коллаборативной фильтрации 3.1.1, где задача поиска схожести между векторами-рейтингами является одной из базовых.

Таблица 1 позволяет оценить, насколько трудоемко решение дан-

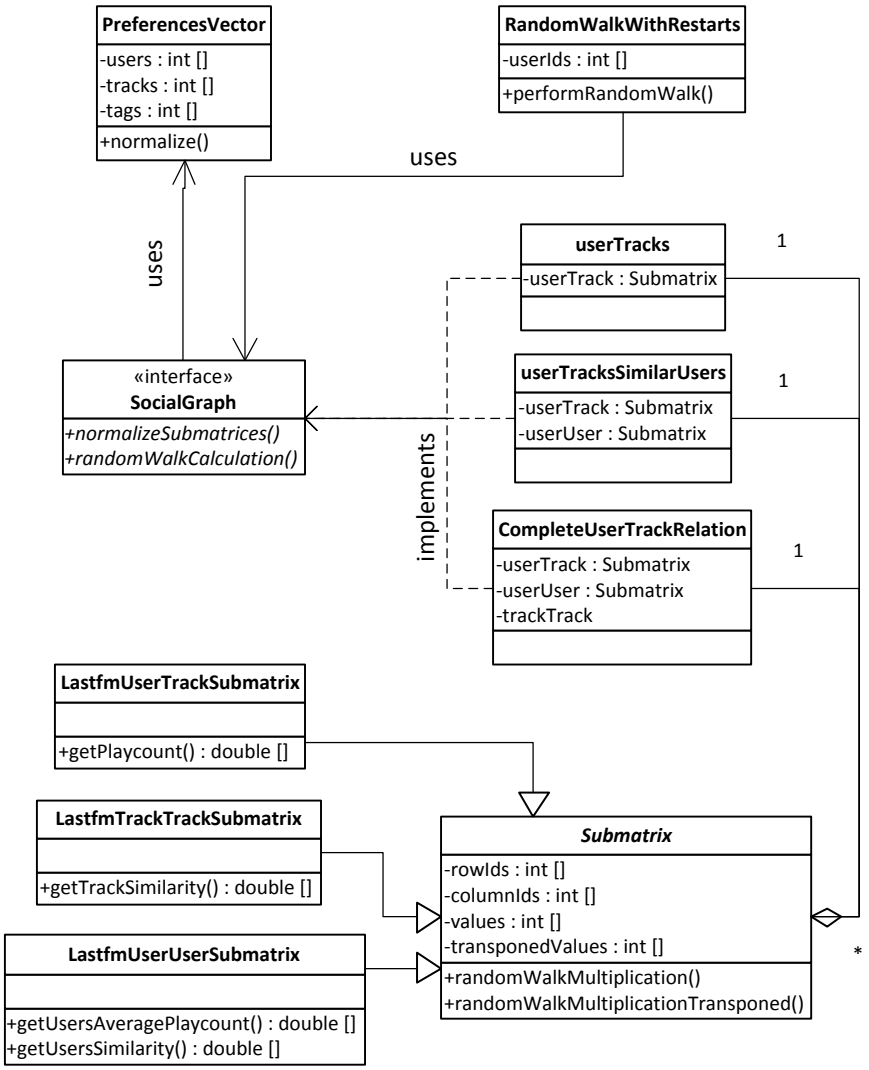

Рис. 4: Диаграмма классов алгоритма рекомендации.

ной задачи. Когда речь идет о рейтингах треков у пользователей, то мы имеем дело с 3148 векторами (соответственно почти 5 млн. пар этих векторов) размерности 30520 и схожесть между каждой парой рассчитывается по формуле, похожей на (1). В крупных социальных сетях, где количество пользователей и треков может исчисляться

миллионами, необходимо максимально ускорить этот процесс. Как один из методов борьбы с большой размерностью векторов может служить *локально-чувствительное хеширование* [13]. Также можно ограничиться лишь теми парами векторов, в которых количество совместно-ненулевых координат больше заданного числа.

4.3.1 Библиотека Apache Mahout

Расчет item-based и user-based схожести векторов реализован в библиотеке Apache Mahout[5]. Она предоставляет свободный доступ к распределенным алгоритмам машинного обучения: кластеризации, сингулярного разложения, классификации, но особую ценность для данной работы представляют алгоритмы коллаборативной фильтрации и рекомендации. Их количество постоянно растет и все они хорошо масштабируемы, так как реализованы в рамках проекта Apache Hadoop[6] с использованием подхода *MapReduce* [14].

Библиотека Mahout предоставляет возможность создавать распределенные рекомендательные системы, реализующие различные виды коллаборативной фильтрации, как user-based так и item-based. Кроме фреймворка для создания рекомендательных систем, библиотека содержит готовые MapReduce задания, полезные в их построении. Одно из таких заданий, ItemSimilarityJob[7], рассчитывает схо-

[5]http://mahout.apache.org/

[6]http://hadoop.apache.org/

[7]org.apache.mahout.cf.taste.hadoop.similarity.item.ItemSimilarityJob

жесть между предметами на основе пользовательских оценок. На вход оно принимает .csv-файл предпочтений, каждая строка которого оформлена в виде

ID пользователя, ID предмета, рейтинг;

и выдает аналогичный файл в форме

ID предмета, ID предмета, схожесть;

Кроме того, параметры ItemSimilarityJob позволяют сократить количество векторов, для которых будет считаться схожесть, а также размер возвращаемого множества кортежей.

- *maxSimilaritiesPerItem* задает предел количества схожих предметов.
- *maxPrefsPerUse, minPrefsPerUser* ограничивают число предпочтительных предметов, рассматриваемых для каждого пользователя.
- *booleanData* позволяет игнорировать величину рейтинга, рассматривая только факт наличия связи между пользователем и предметом.
- *threshold* задает минимальный допустимый порог схожести в возвращаемых результатах.

Особо интересен параметр, связанный с метрикой расчета схожести векторов. В Mahout на данный момент присутствуют следующие метрики:

- количество пользователей, оценивших вместе оба предмета,
- евклидово расстояние,
- коэффициент корреляции Пирсона,
- косинус угла между векторами и другие.

С помощью ItemSimilarityJob на данных тренировочного множества прослушиваний была рассчитана косинусная схожесть между различными треками и различными пользователями. Эти величины впоследствии были использованы как веса дуг социального графа.

5 Результаты экспериментов

Для улучшения качества рекомендаций, рассчитываемых с помощью Random Walk with Restarts, а также времени их расчета, был проведен ряд экспериментов, нацеленных на выявление лучших конфигураций алгоритма. Для улучшения качества оригинального подхода были проверены различные способы составления социального графа, на основе которого строятся рекомендации.

Предпринято несколько попыток упростить алгоритм, чтобы избежать чрезмерного расхода вычислительных ресурсов в крупных социальных сетях. В параграфе 5.2 исследован метод, берущий в рассмотрение только небольшие подграфы социального графа, что позволяет строить их сразу после запроса рекомендации. Кроме того, рассмотрена симуляция случаймного обхода графа, призванная заменить точный расчет предпочтений (параграф 5.3).

5.1 Построение социального графа

Если ограничиться рассмотрением социального графа из треков и пользователей (теги присутствуют не в каждой социальной сети), то, согласно результатам [4], лучшее качество показывает алгоритм, описанный в 3.2. Как было сказано ранее, способы взвешивания дуг между пользователями в оригинальном подходе не всегда соответствуют степени их общности:

- **UU** – среднее количество прослушиваний треков для пользователя,

- **UTr** – количество прослушиваний трека пользователем.

Если расчет для **UTr** достаточно точно отображает силу описываемой связи, так как чаще слушаемому треку соответствует большая вероятность перехода в Random Walk, то в случае с **UU** приближение слишком грубое. Авторы оригинального подхода обосновывают такое решение тем, что соответствующая величина достаточно велика, чтобы не быть подавленной количеством прослушиваний при нормировании. Если рассматривать веса дуг как вероятность перехода к соответствующим смежным вершинам, то необходимо, чтобы первая гёльдерова норма $\|x\|_1 = \sum_i x_i$ столбца матрицы смежности **S** была равна 1. При таком расчете веса отношений **UTr** и **UU** оказываются величинами одного порядка.

В качестве альтернативы данному подходу, было предложено заменить социальные связи на схожесть векторов-рейтингов для пользователей (см. 4.3). При этом возникает необходимость нормировать не весь столбец **S**, а по частям, чтобы большие значения в матрице **UTr** не подавляли схожесть пользователей ($|cos(u_i, u_j)| \leq 1$). Часть столбца, соответствующая смежным пользователям, нормируется отдельно от части, соответствующей трекам. Эксперименты показали, что не имеет значения, будет ли норма части столбца зависеть от количества элементов в нём или окажется хорошо подо-

бранной константой. Кроме того, раздельное нормирование позволяет интуитивно понятно задавать влияние того или иного отношения на рекомендацию. Похожим образом можно посчитать и схожесть между треками **TrTr**, которой недостает в оригинальном социальном графе.

Кроме того, как видно в формуле (2), для имитации рестарта используется вектор $\mathbf{q}$, в котором вершинам, связанным с активным пользователем u_a дугой, соответствуют единицы[8], а остальным элементам нули. То есть Random Walk с одинаковой вероятностью начинается в одном из смежных элементов. Модификация алгоритма, в котором обход возвращается в вершину графа, соответствующую самому u_a, позволяет увеличить качество рекомендации. Для этого в векторе $\mathbf{q}$ нужно заменить единицы на соответствующие веса дуг от u_a к каждому объекту. Это равносильно рестарту и одному шагу случайного обхода.

5.1.1 Сравнение качества рекомендации

Из методов построения графа, описанных в предыдущем параграфе, можно выделить базовый метод и два дополненных: с добавлением музыкальной схожести пользователей (US) и схожести треков (TS). В табл. 2 приведены значения наиболее интересных скалярных метрик оценки качества рекомендаций, посчитанных на тестовом на-

[8]После составления вектора его норма приводится к 1

боре данных. Лучшие результаты показал метод $US + TS$, работающий на графе со всеми возможными отношениями: **UTr**, **TrU**, **TrTr**, **UU**, в котором применена модификация вектора рестарта.

Стоит заметить, что время выполнения умножения разреженной матрицы на вектор пропорционально количеству ненулевых элементов ней. Дополнительное отношение **TrTr**, таким образом, увеличивает время расчета рекомендаций. В системе, где это непозволительно, можно использовать метод US, работающий без схожести треков. Он позволит сэкономить не только время расчета рекомендаций, но и избавит от необходимости рассчитывать эту схожесть. Правда, при этом будет наблюдаться несколько худшее качество рекомендации.

Метрика	**Базовый метод**	**US**	**US+TS**
Half-life(10) utility	2.675	3.925	4.150
Half-life(5) utility	1.574	2.339	2.500
Average precision	0.047	0.055	0.058

Таблица 2: Среднее значение метрик качества на наборе данных 4.1

Распределение релевантных элементов внутри результата Top-N рекомендации хорошо описывают Recall-Precision кривые (см. 3.3.1). На рис. 5 приведены кривые для трех описанных методов. Вместо точности для нулевой полноты, в данном случае выступает точность на первом найденном релевантном треке.

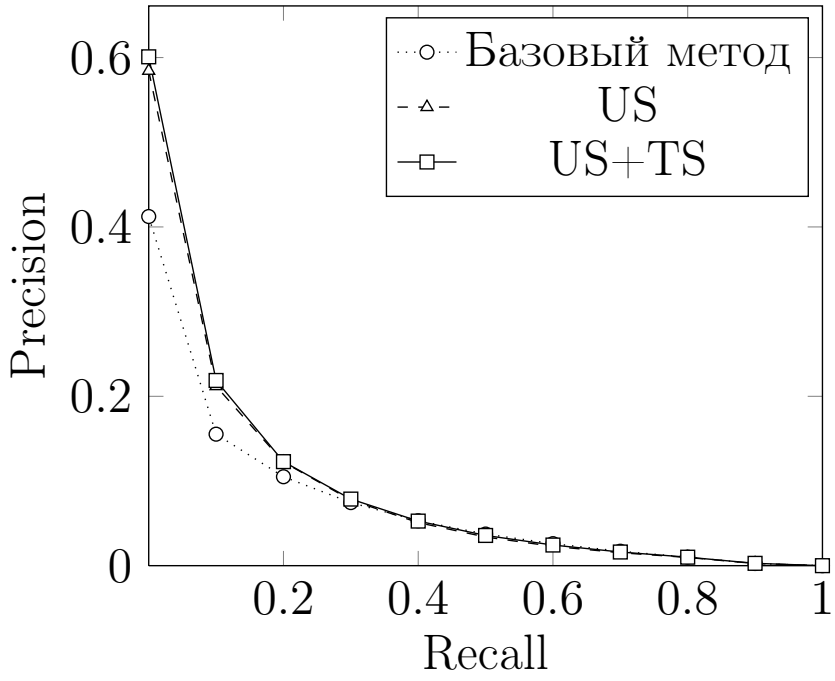

Рис. 5: Recall-Precision кривые результатов рекомендации.

Как видно на рисунке, методы US и TS дают лучшие значения скалярных метрик за счет того, что первые 20% релевантных треков располагаются ближе к началу списка рекомендаций. При этом показатели двух модифицированных методов практически неотличимы. Во всех трех случаях алгоритм сходится за 3 шага.

5.2 Персональные подграфы

Для расчета рекомендаций методом Random Walk with Restarts требуется умножение векторов предпочтений на матрицу смежности графа социальной сети, который одинаков для всех пользователей.

Таким образом, для работы рекомендательной системы в режиме реального времени необходимо хранить в оперативной памяти эту матрицу и, составляя по запросу пользователя начальное приближение предпочтений, умножать его на матрицу. Время умножения пропорционально количеству ненулевых элементов в матрице, поэтому большой размер графа может сделать эти расчеты слишком медленными для использования на практике. Кроме того, матрицу графа нужно уместить в памяти, что тоже не гарантировано в крупных социальных сетях.

Несмотря на сильную разреженность матрицы, которую можно наблюдать в таблице 1, количество пользователей и треков в социальных сетях может достигать десятков и сотен миллионов. Графы таких социальных сетей могут быть огромны по размеру, поэтому задача упрощения предложенного алгоритма является ключевой в вопросе его применимости на практике. В качестве одного из таких методов предложено построение рекомендаций на основе персональных подграфов, уникальных для каждого пользователя. В такой граф включается лишь наиболее близкие активному пользователю вершины. Данный подход не только экономит время вычислений, но и легко позволяет включать граф только что прослушанные композиции. Это может сделать рекомендацию более актуальной.

5.2.1 Алгоритм построения

Для построения персонального подграфа пользователя u_a предложен следующий алгоритм.

$U = Tr = V_u = V_{tr} = V = \varnothing$ – множества вершин, $E = \varnothing$ – множество дуг.

1. В U добавляется n_u пользователей, наиболее похожих на u_a
2. В Tr добавляется n_{tr} треков, наиболее предпочтительных для u_a
3. $V_u := U, V_{tr} := Tr$
4. $\forall u \in U$, в V_{tr} добавляется m_u его любимых треков
5. $\forall t \in Tr$ в V_{tr} добавляется m_{tr} его любимых треков
6. $V := V_u \cup V_{tr}$
7. E формируется из дуг, у которых обе инцидентные вершины принадлежат V

Получившийся граф $G = (V, E)$ включает лишь треки, связанные с наиболее похожими на u_a пользователями и его любимыми композициями. На каждом шаге выборки остаются лишь объекты, наиболее сильно связанные с уже имеющимися. В данном случае размеры выборок задаются константами, так как это позволяет сделать

размеры получившегося графа предсказуемыми. Если же оперировать пороговыми значениями схожести или квантилями выборочных распределений, то контролировать размер подграфа будет сложнее.

5.2.2 Зависимость качества рекомендации от размеров персонального подграфа

При подобной схеме построения, размеры подграфа зависят от плотности дуг в графе и размеров выборок инцидентных вершин (в описанном алгоритме это константы n и m). Для исследования качества алгоритма, основанного на персональных подграфах, была проведена серия экспериментов. Результаты можно видеть на рисунке 6. В качестве основной метрики была выбрана Half-life utility, так как именно она лучше всего отображает успех рекомендации и корректно работает со списками небольшого размера (существенный вклад вносят лишь элементы, близкие к началу списка). На рисунке показаны средние значения метрики для разных $n = n_u = n_{tr}$ и $m = m_u = m_{tr}$.

Как видно на поверхности 6, качество рекомендации растет в направлении увеличения m и уменьшения n. Максимум, равный 2.58, достигается при $n = 4, m = 35$. Такое значение Half-life utility сопоставимо с качеством не модифицированного алгоритма рекомендации на полном графе (табл. 2), при этом в расчет берется существенно меньший граф.

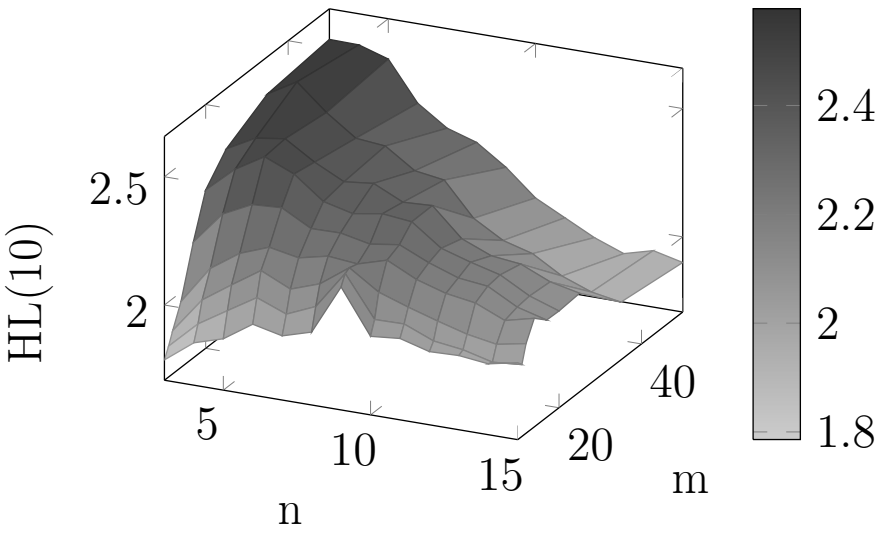

Рис. 6: Half-life(10) utility для разных размеров выборок.

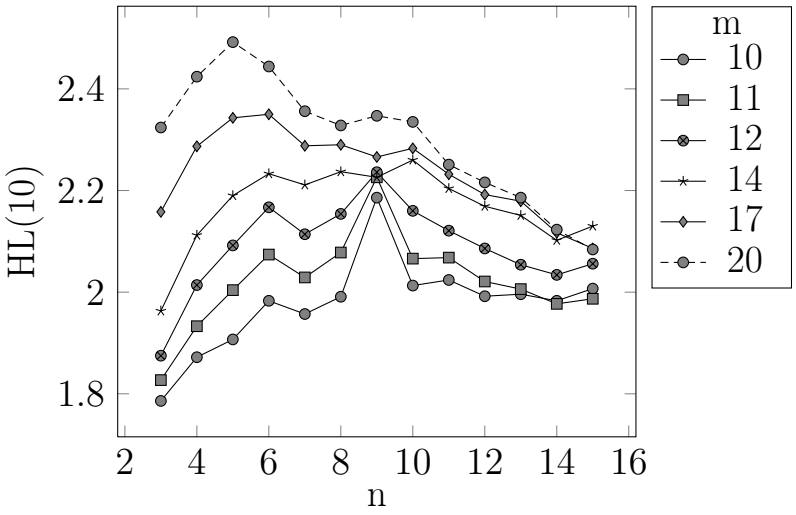

Рис. 7: Half-life(10) utility для фиксированных $m = 10 \dots 20$.

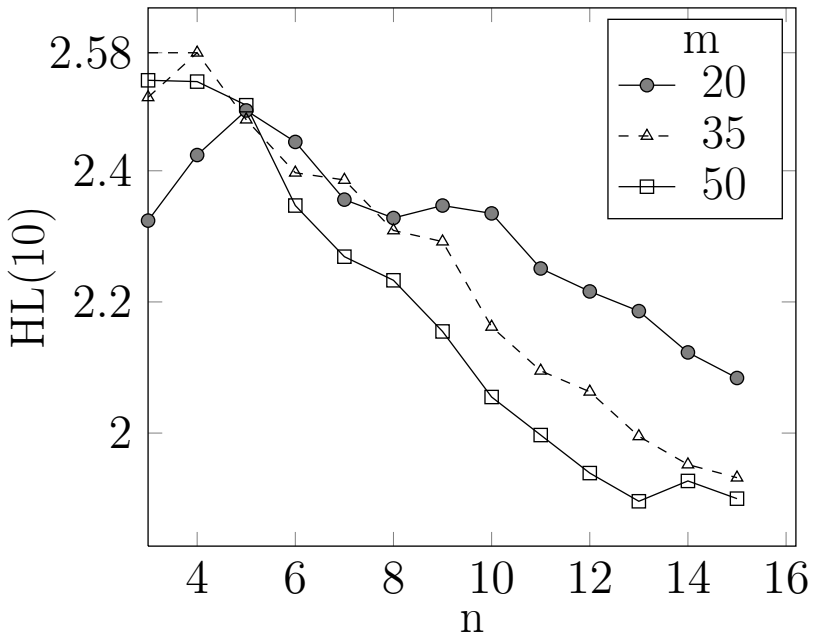

Рис. 8: Half-life(10) utility для фиксированных $m = 20, 35, 50$.

Поиск максимума можно производить, исследуя поведение функций качества $H_m(n)$, являющихся сечениями поверхности плоскостью с одинаковым m. На рис. 7 можно видеть, как меняются эти функции при увеличении m от 10 до 20. Имеет место незначительный рост значений функции, график при этом поворачивается по часовой стрелке. Однако эти изменения становятся слабее с увеличением m и максимум достигается для $m = 35$, а дальнейшее увеличение не приводит к улучшению качества. Это демонстрирует график $H_{50}(n)$ на рис. 8, отличающийся от $H_{20}(n)$ и $H_{35}(n)$ лишь углом поворота. Отсутствие увеличения качества при $m > 50$ подтверждается экспериментами. Для произвольного набора данных поиск оптимального

соотношения размеров выборок следует производить аналогично, исследуя динамику роста качества при изменении m и n.

5.3 Симуляция алгоритма

Как альтернатива построению персонального подграфа, для упрощения расчетов была рассмотрена симуляция случайного обхода графа, призванная заменить полный расчет предпочтительности в виде умножения матрицы на вектор. Для этого на каждом шаге генерировалось случайное число, равномерно распределенное на $[0, 1]$. По нему определялось, в какую из смежных вершин обход пойдет на следующем шаге (либо с вероятностью a вернется в начальную вершину). Вероятность пойти из вершины i в j при этом равна $(1 - a) * S(i, j)$ – соответствующему элементу матрицы смежности социального графа (2), умноженному на вероятность избежать рестарта. Во время такого обхода ведется подсчет количества посещений вершин-треков, которое в итоге и используется для оценки предпочтительности.

При большом количестве шагов обхода k, эта предпочтительность должна сравняться с результатами точного расчета, вопрос лишь в количестве шагов, необходимых для этого. Эксперименты показали, что при $k = 1000, 10000, 100000$ сходимость не достигается. Роль случая даже при ста тысячах шагов все равно очень велика. Это можно объяснить количеством посещений наиболее популярных вершин

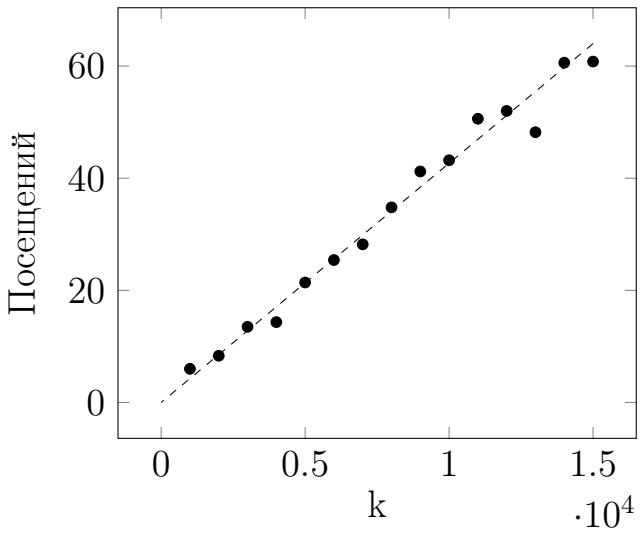

Рис. 9: Количество посещений наиболее популярных вершин.

(рис. 9).

При точном расчете Top-50 рекомендации различия между предпочтимостью двух соседних элементов в списке в среднем[9] равны 2%. Поэтому даже при 100000 шагах обхода, когда наиболее посещаемый трек обходится около 400 раз, эта двухпроцентная разница в предпочтимости не проявляется. Учитывая, что на каждом шаге необходимо делать выбор из несколько сотен смежных вершин, метод симуляции случаного обхода плохо применим на практике.

[9]Для набора данных Last.fm.

5.4 Выводы

В данной работе исследованы различные аспекты применения алгоритма Random Walk with Restarts [4] для рекомендаций треков в социальных сетях, касающиеся качества и сложности расчетов. Данный алгоритм рассчитывает предпочтительность треков, имитируя случайный обход социального графа, состоящего из пользователей социальной сети, треков, содержащихся в ней и, иногда, других сущностей, включенных в сеть. В параграфе 5.1 рассмотрены различные подходы к составлению социального графа и описаны методы, позволяющие существенно улучшить качество рекомендаций по сравнению с оригинальной версией алгоритма.

Количество объектов в рассматриваемом графе велико, операции с ним требуют значительных вычислительных ресурсов. Поэтому в данной работе предложены и исследованы подходы упрощения алгоритма. В частности, это метод построения персональных подграфов для каждого пользователя, позволяющий оперировать значительно меньшим графом, не зависящим от размеров социальной сети. В такой граф включаются лишь наиболее близкие и авторитетные объекты. Рекомендации, основанные на графе, построенном по алгоритму 5.2.1, позволяют достичь качества, сопоставимого с качеством оригинального алгоритма на целом графе, однако уступают в этом модифицированным методам. Тем не менее, такие графы можно конструировать по запросу пользователя в режиме реального

времени, что невозможно для всего графа сети.

В качестве второго метода упрощения была рассмотрена симуляция описанного алгоритма, призванная заменить точный расчет предпочтений. Однако проведенные эксперименты показывают, что она не может сойтись к точному результату за приемлемое время (см. параграф 5.3), что делает данный подход менее предпочтительным, чем рекомендация на персональных подграфах.

6 Заключение

В работе исследованы подходы по созданию рекомендательных систем на основе алгоритма Random Walk with Restarts [4]. В частности:

- реализована Java-платформа для конфигурирования и тестирования рекомендательных систем;
- предложена модификация алгоритма Random Walk with Restarts, использующая традиционные способы коллаборативной фильтрации и дающая более качественные рекомендации;
- предложен способ упрощения оригинального алгоритма, подходящий для применения в крупных социальных сетях, проведено исследование качества предложенного метода;
- приведены результаты симуляции алгоритма Random Walk with Restarts;
- описанные методы протестированы на наборе данных социальной сети Last.fm.

Список литературы

[1] Item-based collaborative filtering recommendation algorithms / B. Sarwar, G. Karypis, J. Konstan, J. Reidl // Proceedings of the 10th international conference on World Wide Web / ACM. — 2001. — P. 285–295.

[2] Netflix prize — http://www.netflixprize.com/.

[3] Matrix factorization and neighbor based algorithms for the netflix prize problem / G. Takács, I. Pilászy, B. Németh, D. Tikk // Proceedings of the 2008 ACM conference on Recommender systems / ACM. — 2008. — P. 267–274.

[4] Konstas I., Stathopoulos V., Jose J. On social networks and collaborative recommendation // Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval / ACM. — 2009. — P. 195–202.

[5] Cantador I., Konstas I., Jose J. Categorising social tags to improve folksonomy-based recommendations // Web Semantics: Science, Services and Agents on the World Wide Web. — 2011. — Vol. 9, no. 1. — P. 1–15.

[6] Last.fm (Россия) — http://www.lastfm.ru/.

[7] Castelluccio M. The music genome project // Strategic Finance. — 2006. — Vol. 88, no. 6. — P. 57–58.

[8] Groh G., Ehmig C. Recommendations in taste related domains: collaborative filtering vs. social filtering // Proceedings of the 2007 international ACM conference on Supporting group work / Citeseer. — 2007. — P. 127–136.

[9] Koren Y. The bellkor solution to the netflix grand prize // Netflix prize documentation. — 2009.

[10] Friedman J. Greedy function approximation: a gradient boosting machine // Annals of Statistics. — 2001. — P. 1189–1232.

[11] Lemire D., Maclachlan A. Slope one predictors for online rating-based collaborative filtering // Society for Industrial Mathematics. — 2005. — Vol. 5. — P. 471–480.

[12] Evaluating collaborative filtering recommender systems / J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl // ACM Transactions on Information Systems (TOIS). — 2004. — Vol. 22, no. 1. — P. 5–53.

[13] Slaney M., Casey M. Locality-sensitive hashing for finding nearest neighbors [lecture notes] // Signal Processing Magazine, IEEE. — 2008. — Vol. 25, no. 2. — P. 128–131.

[14] Dean J., Ghemawat S. Mapreduce: simplified data processing on

large clusters // Commun. ACM. — 2008. — Vol. 51, no. 1. — P. 107–113.