Введение

«Мы достигли точки критической массы, когда количество памяти, которую мы экстериоризировали в книги и базы данных (перечислил несколько источников), сейчас превышает количество памяти, хранящейся внутри совокупности наших биологических тел. Другими словами, «снаружи» памяти больше, чем существует внутри «всех нас». Мы перепрограммировали свою сущность».

Дуглас Коупленд, «Рабы Майкрософта».

За все время своего существования, интернет и смежные с ним комьюнити сагрегировали в себе немыслимые объемы информации. Для того чтобы проще ориентироваться в потоке этой информации разработчики придумывают разнообразные средства поиска и фильтрации информации.

Первым шагом в организации глобального информационного пространства стало создание поисковых систем. Первой такой системой стал *Wandex* в 1993 году. После него появился *Aliweb*, работающий и по сей день. Прошло около пятнадцати лет и поисковые системы настолько плотно вошли в жизнь, что в речи стали появляться глаголы, образованные от названий поисковых систем, их стали упоминать в литературе и кино.

Поисковые системы позволяют найти страницы (а в последнее время узкоспециализированную информацию: музыку, картинки, товары, магазины, научные статьи) по ключевым словам. Они помогают пользователю найти то, о существовании чего он уже знает. Ситуация сильно осложняется, когда пользователь не можете заранее определить ключевые слова для поиска, потому что не обладает конкретной информацией о том, что хочет найти.

К 90-м годам интернет достаточно наполнился всевозможным содержимым, чтобы пользователи озадачились новым вопросом – «я хочу найти то, что мне интересно, но не знаю что именно». К сожалению, поисковая

система не лучший советчик в этом вопросе. Как раз в этот момент начинают развиваться рекомендательные системы. Они берут за основу, имеющуюся у них, информацию о пользователе и предлагают ему то, что возможно его заинтересует. В 2000-х годах стали появляться первые прототипы таких сервисов и получили название рекомендательных систем (РС).

Такие системы собирают различную информацию о пользователе: его историю покупок, оценки, которые он выставлял фильмам, метки, которыми он помечал музыкальные композиции. На основе собранных данных об этом пользователе строится предположительное его мнение об остальных, еще не известных ему, объектах: книгах, фильмах, новых шоу или сайтов в интернете.

Такие системы – новый и только начинающий развиваться вид информационных фильтров.

В этой работе подробно рассматривается рекомендательная система музыкальных композиций. В настоящее время существуют аналоги, решающие похожую задачу, но не нацеленные на непрерывное прослушивание музыки пользователем.

Глава 1. Рекомендательные системы. Цели и задачи.

В данной главе рассматриваются виды рекомендательных систем, их цели и задачи.

1.1. Ухудшающийся отбор в культуре

В последнее время все чаще наблюдаются процессы «ухудшающего отбор в культуре». Возникновение таких процессов описал Джордж Акерлоф в 1970 году [1]. За объяснение этих процессов он получил Нобелевскую премию в 2001 году.

Суть их заключается в том, что потребитель обладает меньшей информацией о продукте по сравнению с продавцом. Акерлоф рассматривал вторичный рынок автомобильной промышленности, он продемонстрировал, что происходит, если на рынке отсутствует честное экспертное мнение. Продавать старые, отработавшие свое машины, под видом новых куда более выгодно, чем новые, а неопытный потребитель не в состоянии отличить приличный автомобиль от плохого и не готов платить справедливую цену за первый. В итоге рынок разрушается на нем остаются только плохие машины, которые Акерлоф называл «лимонками». По той же схеме развивается принцип ухудшающего отбора на рынках культурных продуктов, где автор обязан заплатить за товар, не зная, насколько он ему понравится, а продавец способен его обмануть, используя различные маркетинговые технологии. Из экономических соображений качественный товар требует больших затрат, поэтому производители предпочитают незначительные увеличения бюджета маркетинговых акций увеличениям затрат на производство.

1.2. Цели рекомендательных систем

Цель рекомендательные системы – решить проблему ухудшающего отбора в культуре. Иными словами, предоставить пользователю независимую

экспертную оценку о продукте, который заинтересует его, или составить предположительное мнение пользователя обо всех имеющихся продуктах.

По целям рекомендательные системы можно разбить на несколько классов (на основе статьи [2]).

1. **Аннотация в контексте.** Первоначальным сценарием рекомендательных систем была фильтрация всех сообщений на некотором форуме с целью принятия решение, какое из них читать. Эта задача требовала сохранения порядка и контекста сообщений, и соответствующим образом использовала предсказания для аннотирования сообщений в их контексте. В некоторых случаях «самые плохие» сообщения отфильтровывались.

2. **Найти хорошие объекты.** Это, пожалуй, является задачей большинства существующих рекомендательных систем. Требуется предоставить пользователю ранжированный список объектов с предполагаемыми оценками пользователя. Во многих коммерческих системах показываются только наиболее выигрышные объекты, а прогнозируемые значения оценок нет.

3. **Найти все хорошие объекты.** Такая задача может быть актуальна в рекомендательной системе по безе судебных дел. Для юриста, ищущего прецеденты, скорее важнее не пропустить ни одного случая, чем ранжировать их в нужном порядке. В первую очередь такие рекомендательные системы должны гарантировать низкую вероятность ошибочных заключений (минимизировать число нужных пользователю заключений, но кажущихся системе бесполезными).

4. **Последовательность рекомендаций.** С растущими объемами музыки, доступной для прослушивания в интернете, растет актуальность этой задачи. «*Здесь возникает проблема перехода от рекомендации одной песни к рекомендации целой последовательности музыкальных композиций, которую будет приятно прослушать. На данный момент неизвестно о каких-либо*

исследованиях в этом направлении или РС-системах, решающих такую задачу». [2]

5. **Только просматривание.** Обычно рекомендательная система оценивается на основе того, как хорошо она подбирает объекты для пользователя. «*В беседах с пользователями MovieLens, Amazon и некоторых других сайтов авторы статьи обнаружили, что многие из них пользуются РС-системой даже тогда, когда у них нет намерения приобрести вещь»* [2]. Некоторым пользователям приятно просматривать информацию о рекомендуемых объектах, я для других – это процесс познания.

1.3. Различные подходы к построению РС

1.3.1. Контентно-ориентированная фильтрация

Данный подход основан на измерении похожести объектов между собой, которыми оперирует массив данных. При взаимодействии с системой пользователь составляет тем или иным образом свое отношение к объектам (контенту), на основании которого система рекомендует ему ранее не встречавшиеся объекты, но похожие на те, которые понравились пользователю. Для простоты описания, рассмотрим пример.

Такой подход реализован в рекомендательной системе *PRES* (сокращение от *Personal Recommender System*, [3]). Система *PRES* специализируется на рекомендациях интересных (релевантных для выбранного пользователя) страниц внутри одного сайта. Разделяются понятия навигационных страниц и страниц-данных. (Страницы так же могут иметь смешанный тип).

Общая структура *PRES* выглядит следующим образом (рис. 1).

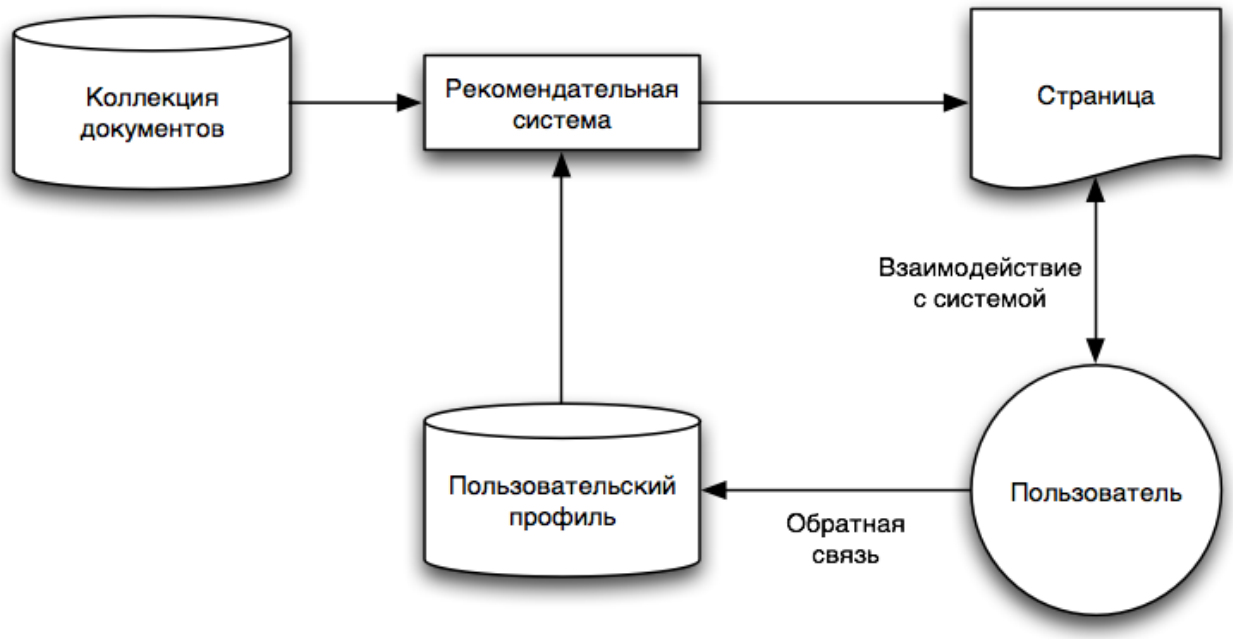

Рис. 1. Структура системы PRES

Рекомендательная система выбирает объекты, исходя из корреляции между его содержимым и профилем пользователя, в котором собраны его предпочтения, что полностью противоположно принципу коллаборативной фильтрации, который основывается на измерении похожести пользователей между собой.

В ходе работы с системой измеряется заинтересованность пользователя в том или ином документе, которые учитываются в его профиле. Далее система подбирает наиболее релевантные документы из имеющейся коллекции, основываясь на критериях, указанных в пользовательском профиле.

1.3.2. Коллаборативная фильтрация

В создании рекомендательных систем этот подход более популярен, чем подход с применением контентно-ориентированной фильтрации. Основная идея данного подхода заключается в составлении рекомендаций оценок для данного пользователя на основе мнения остальных. Оценки других пользователей могут быть получены из их профайлов явно или неявно, применяя различные способы.

Предположим, что имеется набор пользователей $U = \{u_1, u_2, u_2 \dots u_m\}$ и набор объектов, представляющих собой массив данных $I = \{i_1, i_2, i_2 \dots i_n\}$ (ими могут быть фильмы, песни или книги). Каждый пользователь имеет список элементов I_{u_k}, о которых он составил свое мнение. Могут использоваться как явные оценки (пользователь выставил рейтинг по заданной числовой шкале) или неявные (пользователь купил товар или прослушал песню определенное число раз, находился на странице определенное количество времени). Заметим, что множество $I_{u_k} \subseteq I$, а также может оказаться пустым. Также существует *выбранный пользователь* $u_a \in U$, для которого строится рекомендация. Рекомендация может быть в двух формах:

1. **Предсказание.** Число $P_{a,j}$, предсказывающее какую бы оценку поставил пользователь u_a объекту $i_j \notin I_{u_a}$. Предсказанная оценка должна принадлежать той же шкале, что и оценки, данные пользователем (например, от 1 до 10).

2. **Рекомендация.** Список из N элементов, $I_r \subseteq I$, которым бы пользователь дал высокую оценку. В зависимости от системы на эти элементы может накладываться дополнительное требование о том, должны ли они не содержаться (как в случае с купленными товарами или просмотренными фильмами) или нет (в случае с музыкальными композициями) в списке просмотренных объектов I_{u_k}.

Ниже представлен общий принцип работы рекомендательной системы на основе коллаборативной фильтрации (рис. 2).

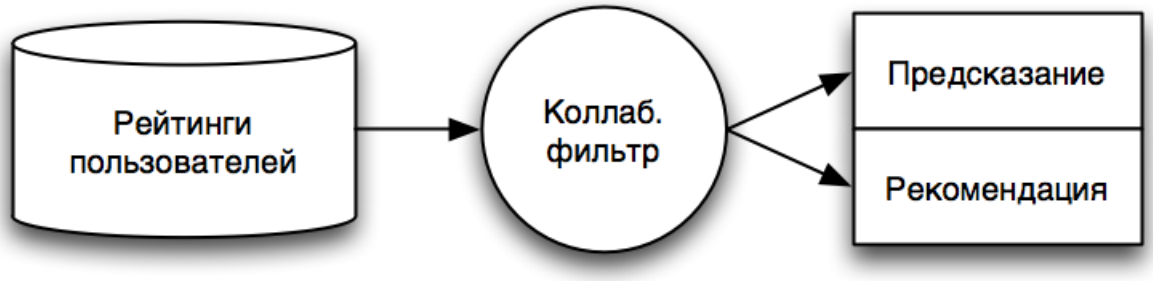

Рис. 2. Схема работы рекомендательной системы на основе принципа коллаборативной фильтрации

Существуют два вида коллаборативного алгоритма.

1. **Предметно-оринтированный**. Такие алгоритмы строят рекомендации на основе модели пользовательских оценок. С помощью вероятностного подхода эти алгоритмы рассматривают процесс построения рекомендации, как построение ожидаемого значения оценки. Процесс построения модели основан на алгоритмах самообучающихся систем, таких как алгоритмы Байесовских сетей, кластеризации, систем правил (rule-based systems). Байесовские сети помогают задать вероятностную модель для проблемы коллаборативной фильтрации. Кластерная модель понимает проблему коллаборативной фильтрации как проблему классификации пользователей и основывается на принципе разделения похожих пользователей на классы, вычисляет насколько вероятно, что определенный пользователь попадает в определенный класс, на основе чего считает вероятностный рейтинг пользователя данному объекту. Подход, основанный на правилах, применяет алгоритмы нахождения ассоциативных правил для поиска ассоциаций между товарами, купленными одновременно, а затем генерирует рекомендации товаров, основываясь на силе ассоциаций между ними [4].

2. **Пользовательски-ориентированные** алгоритмы используют всю базу пользователей для построения предположительных оценок. Такие системы, так или иначе, формируют множество пользователей, называемое *соседством*, для заданного пользователя. Обычно туда включается заранее определенное число пользователей, которые имеют схожие вкусы с выбранным. Это может выражаться в схожих оценках одинаковым объектам или в тенденции покупать одинаковые вещи. На основании этого множества строится список из объектов, которые наиболее понравятся пользователю (рис. 3).

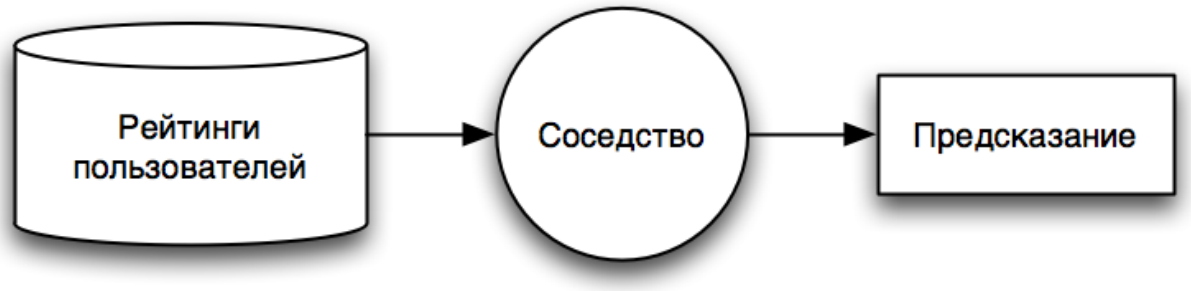

Рис. 3. Схема работы рекомендательной системы на основе коллаборативной фильтрации с использованием пользовательски-ориентированных алгоритмов.

Именно этот подход получил наибольшее распространение среди рекомендательных систем. Причем чаще всего используются пользовательски-ориентированные алгоритмы.

1.4. Музыкальные РС и специфические задачи

В данной работе будут рассматриваться рекомендательные системы, решающие задачу построения рекомендации последовательности объектов. С растущими объемами свободной для прослушивания музыки в интернете эта задача становится все более актуальной.

Предположим, что имеется музыкальная служба, которая позволяет пользователю прослушивание любой существующей на данный момент музыки. Отбросим проблемы хранения, доставки контента и авторских прав и сконцентрируемся на том, как сделать взаимодействие пользователя с такой службой максимально комфортным.

Пользователь может искать музыкальные записи по автору, композитору или названию, но слишком утомительно спрашивать каждый раз, когда хочется найти что-то новое, совета у друзей

В такой системе должен существовать режим «радио» (который будет использовать рекомендательную систему для построения последовательности

вещания) , который по желанию пользователя будет проигрывать ему новую и уже известную ему музыку в заданном соотношении. Пользователь должен иметь возможность пропустить музыкальную композицию, если она ему не нравится.

Допустим, что пользователь (назовем его Иван) регулярно пользуется музыкальным сервисом (далее МС). МС доступен для него в его плеере, на его компьютере и в его машине. На работе Иван слушает ритмичную и монотонную музыку одного стиля, которая помогает ему сконцентрироваться. В машине он предпочитает что-нибудь веселое и энергичное, это помогает ему взбодриться по пути на работу и поднимает настроение, когда он с нее возвращается. Когда Иван посещает спортзал, то он предпочитает жесткую, агрессивную и ритмическую музыку.

Из этого примера можно выделить три состояния Ивана, в каждом из которых он предпочтет разное поведение от рекомендательной системы МС.

Первой задачей этой работы является разработать метод построения рекомендательной системы, которая учтет наличие возможных эмоциональных состояний у самого Ивана.

Второй задачей является построение алгоритма, который сможет находить подобные состояния автоматически, без участия модераторов МС или Ивана.

Выводы по главе 1

1. Произведен обзор основных типов рекомендательных систем.
2. Представлен новый критерий фильтрации для рекомендации контента.

Глава 2. Построение музыкальной рекомендательной системы с учетом эмоциональных состояний

2.1. Описание предметной области

В рамках определенной задачи выделим основные сущности, которым далее дадим более четкие определения.

Под музыкальным сервисом будем понимать набор музыкальных записей. Не будем никак учитывать их жанровую принадлежность, авторство или информацию об альбомах. Тем не менее, каждая музыкальная композиция будет уникальна, и иметь собственный номер. Порядок композиций не имеет значения.

Под пользователем будем понимать потребителя услуг музыкального сервиса. В рамках описываемой модели будем предполагать, что пользователь слушает музыку непрерывно, каждую композицию он может дослушать до конца, либо пропустить и приступить к прослушиванию следующей. Оставим за пользователем возможность выражать свое мнение отдельно о каждой композиции двумя способами – с помощью кнопок «мне нравится» и «мне не нравится, следующая». Иначе говоря, пользователь может просто прослушать композицию, может отметить ее, как понравившуюся, может пропустить композицию или пропустить с указанием того, что она ему особенно не нравится.

Не все пользователи пользуются в жизни кнопкой перехода к следующей песни. Логично полагать, что такие пользователи имеют всего одно музыкальное состояние. Тем не менее, наличие такой кнопки говорит о ее востребованности среди пользователей, а объемы памяти компьютеров и портативных плееров, которые вмещают в себя сотни музыкальных

композиций, о том, что эта кнопка редко используется для навигации по списку воспроизведения.

На практике кто-то слушает музыку по альбомам, кто-то слушает разную музыку в разной обстановке, а кто-то предпочитает слушать музыку, соответствующую его настроению. Все это будем называть эмоциональным состоянием пользователя, вне зависимости от истинных его мотивов выбора той или иной группы музыкальных записей. Будем так же полагать, что такие группы существуют и их больше, чем одна.

Условимся, что кроме пользователей системы никто не должен участвует в поставке входных данных. Иначе говоря, исключим все внешние воздействия на рекомендательную систему. Подобными внешними воздействиями является, например, расстановка тегов в широко известном сервисе *Last.fm*. Такое вмешательство носит субъективный характер и может отрицательно сказаться на результатах рекомендаций.

Задача этой работы – построить рекомендательную систему, основываясь на принципе чистой коллаборативной фильтрации с использованием пользовательски-ориентированного алгоритма

2.2. Чисто коллаборативная музыкальная рекомендательная система.

Будем основываться на принципе построения систем чисто-коллаборативных рекомендаций, описанным Хеолокером в 1999 году [5].

Построение коллаборативной РС состоит из следующих шагов:

1. Выбор шкалы оценок.
2. Выбор критерия построения соседства.
3. Принцип нахождения наилучшей композиции, на основе данных соседей выбранного пользователя.

Выберем шкалу рейтинга. Обозначим множество всех пользователей за $U = \{u_1, u_2, \cdots, u_n\}$, а множество всех песен за $S = \{s_1, s_2, \cdots, s_m\}$. Рейтинг для музыкальных систем должен основываться на количестве прослушиваний и на количестве пропусков. Для каждого пользователя будем хранить в качестве первичной оценки им музыкальной композиции разницу между числом прослушиваний и пропусков каждого трека в матрице $K_{u,i}$, где u – номер пользователя, а i – номер трека.

При этом первое знакомство пользователя с песней должно сильнее влиять на ее рейтинг, чем прослушивание этой песни в сотый раз. Иными словами с первыми прослушиваниями рейтинг композиции должен расти прямо пропорционально, но при увеличении числа прослушиваний рост должен замедляться.

Для этого подходит функция $f(k)$ следующего вида:

$$f(k) = \frac{2a}{1+e^{-\frac{x}{b}}} - a,$$

где k – разница между числом прослушиванием и числом пропусков, a – максимальный рейтинг композиции (достигается только при $k \to \infty$), b задает скорость роста функции при малых значениях k.

Кнопка пользователя о том, что ему нравится эта композиция, перманентно добавляет к рейтингу некоторую константу, а кнопка «не нравится» аналогично уменьшает рейтинг на ту же константу.

Если максимальным рейтингом считать число 100, а скорость роста при малых значениях p взять линейную, то при данном a, b должно быть равно примерно 50. График функции приведен на рис. 4

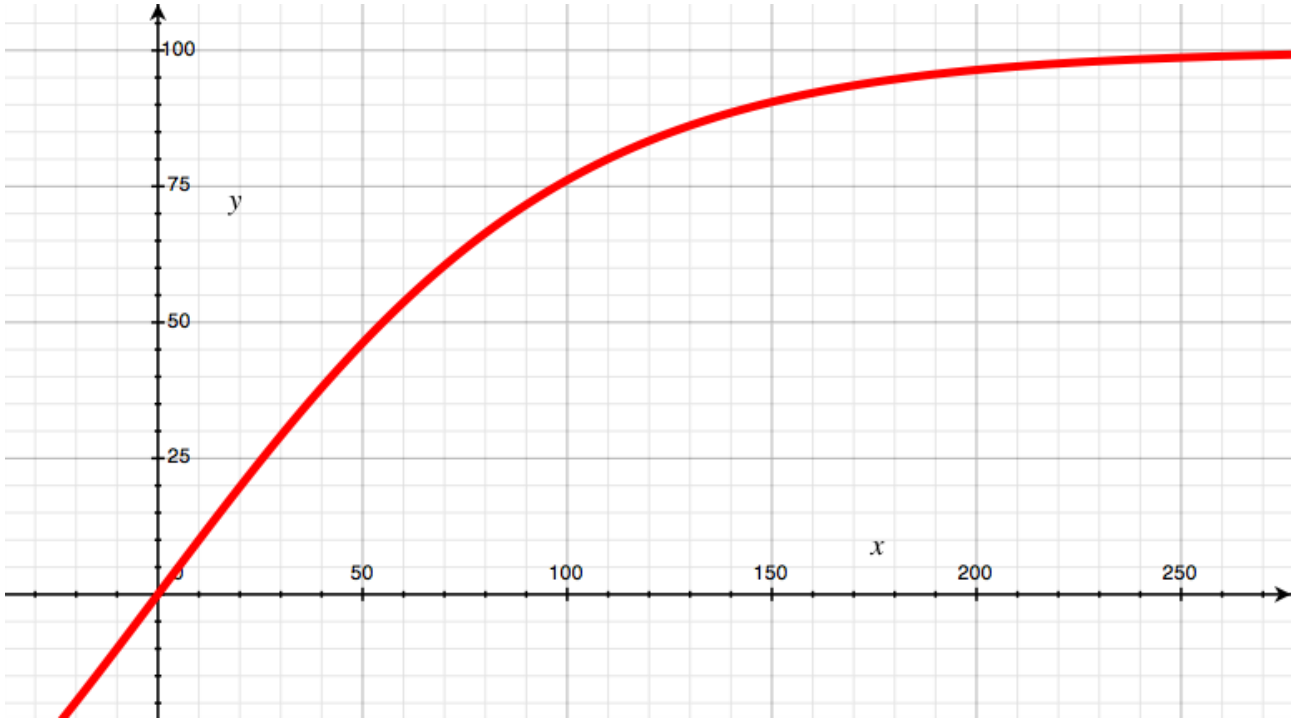

Рис 4. График функции $f(k)$.

Обозначим конечную оценку пользователем u трека i за $r_{a,i}$.

Тогда похожесть двух пользователей рассчитывается как коэффициент корреляции Пирсона их оценок трекам:

$$P_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i}-\overline{r_a})\times(r_{u,i}-\overline{r_u})}{\sqrt{\sum_{i=1}^{m}(r_{a,i}-\overline{r_a})^2\times(r_{u,i}-\overline{r_u})^2}},$$

где $\overline{r_a}$ – средний рейтинг пользователя a, а m – общее число треков.

Далее введем константу *размера соседства* N. Построим множество соседей для выбранного пользователя, используя матрицу $P_{a,u}$. Для выбранного пользователя a, отсортируем множество остальных пользователей по по убыванию значения $P_{a,u}$. Из полученного множества пользователей выберем первых N, или всех (если $N < users$). На основе рекомендаций Херлокера [5] будем считать $N = 30$.

Предположительные оценки строятся на основе средневзвешенного среднего по всем соседям пользователя, где в качестве веса берется схожесть пользователя со своим соседом:

$$p_{a,i} = \frac{\sum_{u=1}^{m}(r_{u,i} - \overline{r_a}) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}},$$

где $p_{a,i}$ – предположительное мнение пользователя u о треке i, а $P_{a,u}$ – похожесть выбранного пользователя a на своего соседа u. В своей работе Херлокер упоминает о том, что будет возникать много похожих пользователей, основываясь на небольшом пересечении оцененных ими объектах одинаково. Для разрабатываемой системы рекомендации музыки это не так актуально, потому что матрица оценок пользователей наполнена гораздо более плотно, чем те матрицы, которые рассматривал Херлокер (оценки фильмам и товарам).

Из всего множества песен выберем множество с положительными рекомендациями и обозначим за $E = \{e_1, e_2, \cdots, e_n\}$.

2.3. Подход с учетом эмоциональных состояний

2.3.1 Общий принцип

Вторая поставленная задача – учесть возможные эмоциональные состояния. Рассмотрим детальнее само понятие эмоциональных состояний. Это могут быть как настроения, так и определенные жизненные ситуации. Так или

иначе, это некие группы треков, которые пользователь слушает вместе. Основываясь на этом факте, кроме информации о рейтингах пользователями определенной композиции будем считать рейтинги композиций относительно друг друга. То есть если раньше считалось, насколько часто пользователь слушает определенный трек в принципе, то теперь будем дополнительно записывать, насколько часто пользователь слушает один трек после другого. Далее, основываясь на уже построенном соседстве, мы сможем предполагать, какие еще треки пользователь хотел бы слушать после того, который он слушает в данный момент.

Далее, имея эту информацию, кластеризуем все треки пользователя на классы, которые и будут являться эмоциональными состояниями. Для выбора эмоционального состояния в момент прослушивания будем использовать алгоритм Витерби [6].

Заметим, что введение такого подхода не сильно изменяет сам процесс выбора следующей композиции для пользователя, но вводит дополнительный важный критерий фильтрации музыки.

2.3.2. Построение предположительных оценок

Аналогично матрице $K_{u,i}$ будем хранить матрицу $L_{u,i,j}$ – то, насколько часто пользователь u после трека s_i слушает трек s_j. При этом если пользователь после трека s_a пропустил треки s_b и s_c, то в матрице уменьшится число прослушиваний для элементов $L_{u,a,b}$ и $L_{u,a,c}$.

Эта матрица так же «уравнивается» функцией $f(l)$ потому как используются уже посчитанные корреляционные коэффициенты между пользователями. Матрицу со значениями функции $f(l)$ назовем $T_{u,i,j}$.

Допустим, что пользователь в данный момент прослушивает какую-либо композицию, и зафиксируем ее. Далее матрицу $T_{u,i,j}$ можно воспринимать как бывшую матрицу $R_{a,i}$, в которой вместо рейтингов композициям хранит

вектора, в которых указанны рейтинги остальных композиций, относительно зафиксированной. Это делается для удобства представления, чтобы не производить одни и те же действия для каждой песни. Заметим, что вычисления предположительных рейтингов относительно каждой песни ничем не отличается от рассматриваемого алгоритма в разд. 2.2.

Иначе говоря, $t_{a,i} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \end{bmatrix}$

Как и в предыдущем случае можно посчитать средневзвешенное среднее, в качестве весов, взяв корреляционные коэффициенты из предыдущей разд. 2.2.

Заметим, что при этом среднее будет равным

$$\overline{t_a} = \frac{\sum_{j=1}^{m} t_{a,j}}{m} = \begin{bmatrix} \frac{\sum_j t_{a,j,1}}{m} \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{r_{a,1}}{m} \\ \vdots \end{bmatrix} = \begin{bmatrix} \overline{t_{a,i}} \\ \vdots \end{bmatrix}.$$

Предположительные оценки о том, какой трек пользователь будет слушать после зафиксированного, назовем $q_{a,i}$. Вычисляться он будет по той же формуле, что и $r_{u,i}$.

$$q_{a,i,j} = \overline{t_{a,i}} + \frac{\sum_{u=1}^{n} \left(t_{u,i,j} - \overline{t_{u,i}}\right) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}}$$

Более наглядно переход от чисто-коллаборативной фильтрации, к фильтрации с учетом настроений представлен в таб. 1.

Таблица 1.

Было		Стало
$r_{u,i}$	$\rightarrow$	$t_{u,i,j}$

$\overline{r_a} = \sum_{i=1}^{n} r_{a,i}$	$\rightarrow$	$\overline{t_{u,i}} = \sum_{i=1}^{n} t_{a,i,j}$
$p_{a,i}$	$\rightarrow$	$q_{a,i,j}$

Теперь мы имеем таблицу для каждого пользователя, в которой предположительно указанно насколько часто пользователь будет слушать один трек после другого.

2.3.3. Алгоритм Витерби

Алгоритм Витерби [6, 7] – это алгоритм динамического программирования для нахождения наиболее вероятной последовательности скрытых событий (называемой **путем Витерби**) по имеющейся последовательности наблюдений.

Алгоритм основан на нескольких допущениях:

1. В каждый момент времени система существует в неком состоянии конечного автомата. Этих состояний конечное, сколь угодно большое число. В состояние может вести несколько путей, но лишь один из них наиболее вероятен и называется «выжившим путем». Это фундаментальное допущение алгоритма, потому как он будет перебирать все возможные пути о оставлять только наиболее вероятный. В этом случае алгоритм не должен следить за всеми путями, только за одним на каждое состояние.
2. Вторым ключевым предположением является то, что переходы из одного состояния в другое помечены инкрементальной метрикой, чаще всего числом.
3. Третьим предположением является то, что метрика событий накапливается вдоль пути в некотором смысле (чаще всего просто складывается).

Основная идея алгоритма в том, что при возникновении нового события (наблюдения), алгоритм, просчитывая новый набор метрик, соответствующий новому набору состояний с помощью мер переходов и метрик старых состояний, выбирает лучшее новое состояние.

После вычисления всех инкрементальных меток, запоминается только путь переходов с лучшим значением, а остальные забываются.

Рассмотрим пример. Допустим двое друзей Анна и Борис, которые живут далеко друг от друга, ежедневно созваниваются по телефону и обсуждают, как прошел их день. Борис обычно либо работает дома, либо гуляет по городу, либо занимается уборкой, причем это зависит напрямую от погоды. Анна знает, что в городе, где живет Борис, бывает всего два вида погоды: дождливая и солнечная. На основе информации о том, чем занимался Борис, она хочет угадать последовательность, в которой менялась погода в городе Бориса и предсказать ему погоду на следующий день.

Анна моделирует погоду дискретной скрытой марковской цепью, в которой есть два скрытых состояния: «дождливо» и «солнечно» и три возможных типа наблюдений: «работа», «прогулка» и «уборка». Анна знает, насколько вероятна та или иная погода в городе Бориса, а так же чем он предпочитает заниматься в зависимости от погоды, иными словами ей известны параметры скрытой марковской модели. Параметры модели, записанные на языке *Python*, представлены на листинге 1.

Листинг 1.

```
## скрытые состояния марковской модели
states = ('Rainy', 'Sunny')

## наблюдений марковской модели
observations = ('walk', 'work', 'clean')

## информация Анны на счет того насколько вероятна та или иная погода в городе
## Бориса
start_probability = {'Rainy': 0.6, 'Sunny': 0.4}
```

```
## Вероятности смены погоды или вероятности переходов скрытой марковской модели
## из одного состояния в другое
transition_probability = {
   'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
   'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6},
   }

## Информация Анны о том, насколько Борис любит заниматься теми или иными делами
## в зависимости от погоды. Вероятности проявления.
emission_probability = {
   'Rainy' : {'walk': 0.1, 'work': 0.4, 'clean': 0.5},
   'Sunny' : {'walk': 0.6, 'work': 0.3, 'clean': 0.1},
   }
```

Графически эта скрытая марковская модель может быть представлена на рис. 5.

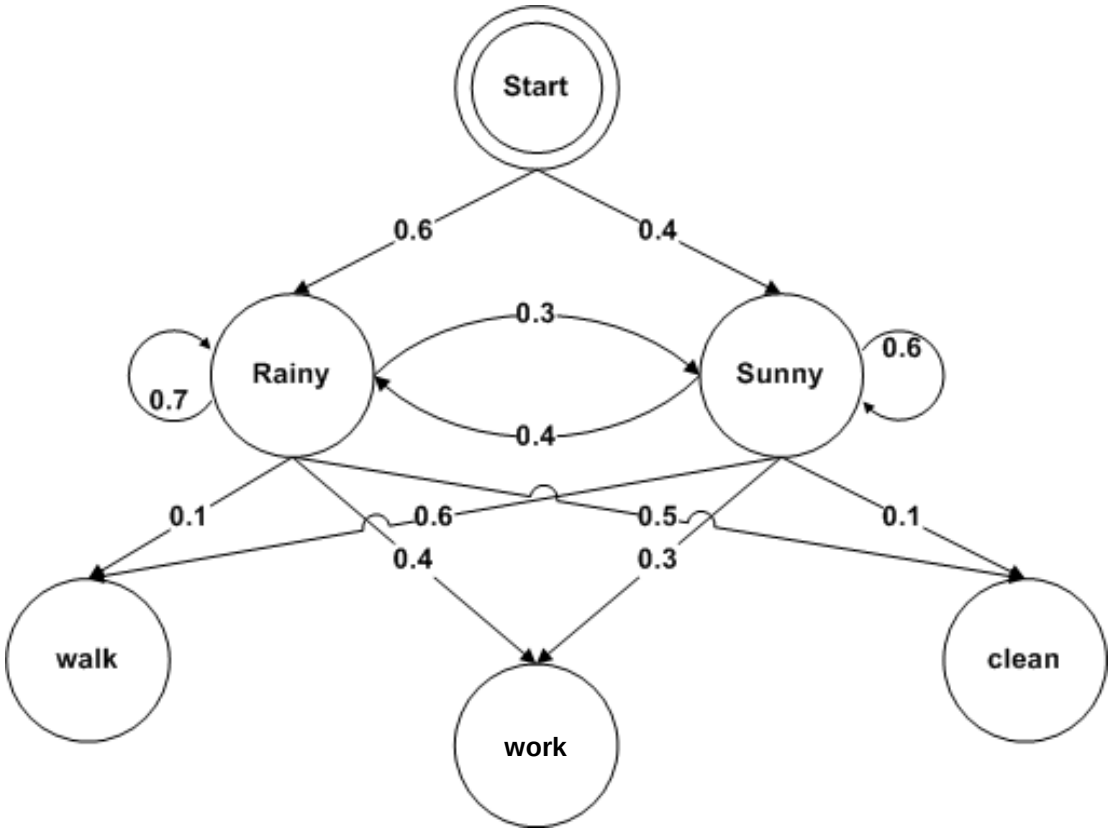

Рис. 5. Пример скрытой марковской модели для алгоритма Витерби

Допустим теперь, что Анна наблюдала за действиями Бориса на протяжении последних трех дней, и ей стало известно, что в первый он гулял по городу, во второй день работал, а на третий день убирался в комнате. Анна ставит перед собой два вопроса «какова вероятность такой последовательности действий для Бориса?» и «какова наиболее вероятная последовательность

смены погоды в городе, в котором живет Борис?» На первый вопрос Анне помогает ответить прямой алгоритм, а на второй – алгоритм Витерби. Оба этих алгоритма могут быть реализованы одной функцией:

```
def forward_viterbi(obs, states, start_p, trans_p, emit_p):
   T = {}
   for state in states:
       ##          prob.           V. path  V. prob.
       T[state] = (start_p[state], [state], start_p[state])
   for output in obs:
       U = {}
       for next_state in states:
           total = 0
           argmax = None
           valmax = 0
           for source_state in states:
               (prob, v_path, v_prob) = T[source_state]
               p = emit_p[source_state][output] * \
                  trans_p[source_state][next_state]
               prob *= p
               v_prob *= p
               total += prob
               if v_prob > valmax:
                   argmax = v_path + [next_state]
                   valmax = v_prob
           U[next_state] = (total, argmax, valmax)
       T = U
   ## применим sum/max в последним результатам:
   total = 0
   argmax = None
   valmax = 0
   for state in states:
       (prob, v_path, v_prob) = T[state]
       total += prob
       if v_prob > valmax:
           argmax = v_path
           valmax = v_prob
   return (total, argmax, valmax)
```

В качестве аргументов главная функция (forward_viterbi) принимает следующее: obs – последовательность наблюдений, то есть список [‘walk’, ‘work’, ‘clean’], states – набор возможных состояний, start_p – начальные вероятности каждого из состояний, trans_p – вероятности переходи из одного состояний в другое, emit_p – вероятности того или иного наблюдения в каждом состоянии.

Алгоритм работает с двумя отображениями: `T` и U. Каждое из них является отображением из множества состояний в множество наборов вида (prob, v_path, v_prob), где prob – вероятность всех путей из начального состояния до того, которому соответствует этот набор, v_path – путь витерби до состояния, которому соответствует набор, v_prob – вероятность этого пути. Отображение T соответствует некоторому моменту времени t, основной же цикл составляет отображение U, которое соответствует моменту t+1. По свойствам марковских моделей информация о каждой точке времени, предшествующей моменту t не требуется.

В начале своей работы алгоритм инициализирует T начальными вероятностями: общая вероятность попадания в состояние равна начальной вероятности этого состояния, путь Витерби равен самому состоянию, а вероятность этого пути соответствует начальной вероятности этого состояния.

В основном цикле алгоритм пробегает по всем имеющимся наблюдениям. Инвариант цикла – отображение T, которое хранит достоверную информацию о моменте времени, предшествующему текущему наблюдению. Полная вероятность следующего состояния вычисляется с помощью суммирования вероятностей всех путей, ведущих в него. Для каждого исходного состояния в T хранится полная вероятность всех путей, ведущих в это состояние. Эта вероятность умножается на вероятность проявления текущего наблюдения и на вероятность перехода их исходное состояние в рассматриваемое. В итоге полученная вероятность добавляется к полной (total). Путь витерби и его вероятность строятся аналогичным образом, за исключением того, что запоминаем только муть с максимальной вероятностью (valmax).

После завершения цикла по всем наблюдениям, алгоритм подсчитывает следующее наиболее вероятное состояние, основываясь уже только на вероятностях перехода между состояниями.

2.3.4. Квантовые автоматы

Составим модель рекомендательной системы, которая обладает описанными в разд. 2.1. критериями. Рекомендательная система должна иметь различное поведение, в зависимости от эмоционального состояния пользователя. Наиболее удобно в таком случае воспользоваться автоматным подходом [8].

Такой подход предлагает рассматривать объект (в нашем случае рекомендательную систему) как систему управления с входными (реакция пользователя на прослушиваемый трек) и выходными воздействиями (предложение рекомендательной системой нового трека).

Такая модель не полностью представляет пользователя: потому что в случае с эмоциональными состояниями нельзя точно сказать, в коком именно из них находится пользователь. Более уместно говорить о вероятности нахождения пользователя в том или ином состоянии. По аналогии с атомом и электроном, будем называть такой автомат квантовым.

Важно упомянуть, что так же квантовыми автоматами называются аналоги вероятностных автоматов в квантовых вычислениях. С ослаблением интереса науки к квантовым вычислениям, термин «квантовый автомат» практически не употребляется в этом значении.

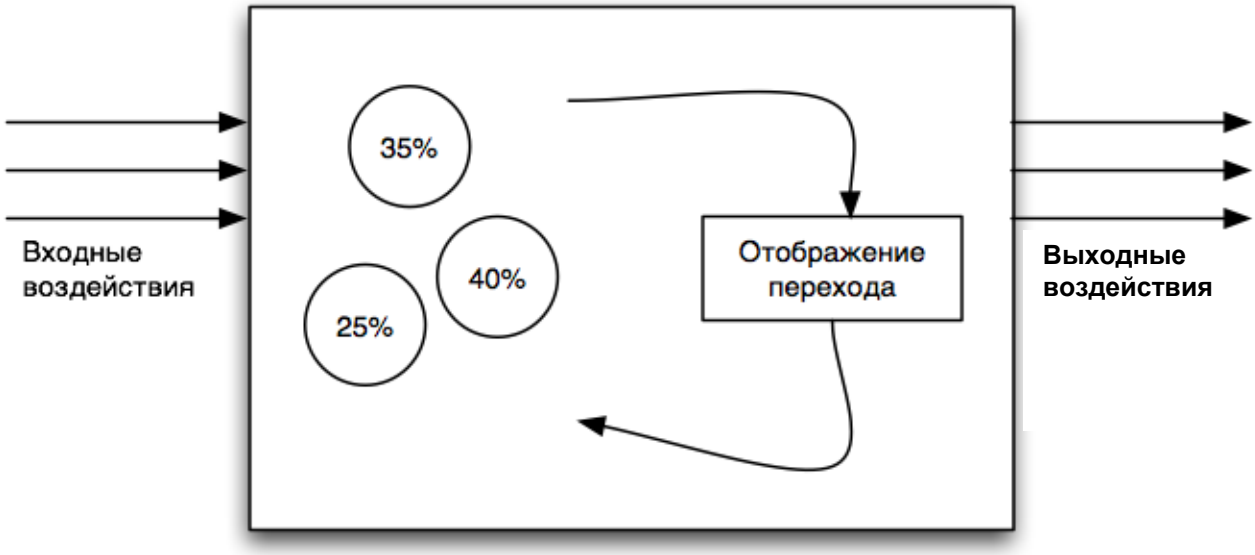

Рис. 6. Принцип работы квантового автомата.

Квантовый конечный автомат, как и обычный, имеет конечный набор возможных состояний. Вместо выделенного состояния задается набор вероятностей, описывающий текущее «распределение» системы, обозначаемый $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n) \in S$, где σ_i – вероятность того, что система находится в состоянии с номером i, причем $\sum_{i=0}^{n} \sigma_i = 1$. Роль функции перехода в таком случае играет отображение $(S, g) \to S$ (будем называть его отображением перехода), где g - входное воздействие. То есть, основываясь на текущем распределении вероятностей и входном воздействии будет «перераспределять» пользователя по состояниям.

2.4.5. Построение состояний квантового автомата

По предположению поставленной в работе задачи (разд. 2.1.), пользователь имеет эмоциональные состояния.

Для того чтобы построить состояния для такого автомата воспользуемся построенной в разд. 2.4.2 матрицей $q_{a,i,j}$. Тогда задача выделения состояний для квантового автомата сводится к задаче кластеризации всех рекомендуемых

песен алгоритмом коллаборативной фильтрации, где принципом схожести будет частота проигрывания одного трека после другого.

Алгоритмы кластеризации разбивают множество объектов на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер содержал схожие объекты, а объекты разных кластеров существенно различались. Похожесть объектов определяется функцией расстояния между объектами (в случае с разрабатываемой рекомендательной системой расстояние задается матрицей Q_a).

Для кластеризации выберем алгоритм «Выделения связных компонент», описанный в [9]. Его принцип заключается в том, что все элементы рассматриваются как вершины графа. Вводится понятие уровня трешхолда (F), которое будет характеризовать критическое значение расстояния между треками. В том случае, если расстояние меньше значения трешхолда, треки считаются связанными между собой, в противном случае связь между ними не учитывается. Кластером в этом алгоритме считается компонента связности в построенном графе.

Заметим, что рейтинги похожих друг на друга пользователей могут отличаться на порядок, поэтому все значения рейтингов других композиций относительно заданной нормируем в процентах.

Для кластеризации из матрицы $q_{a,i,j}$ построим новую матрицу следующего вида:

$$\mathrm{c}_{a,i,j} = \frac{q_{a,i,j}}{\sum_{j=1}^{m} q_{a,i,j}} \times 100\%$$

Уровень трешхолда (F) будет задавать число полученных кластеров. Если предположить, что в каждом состоянии пользователя равное число треков, то следует брать уровень трешхолда обратно-пропорциональный количеству желаемого количества кластеров.

Так или иначе, решая задачу о кластеризации необходимо выбрать либо приемлемый уровень того, насколько похожи элементы из одного кластера, либо задать число кластеров. Число кластеров может варьироваться в зависимости от пользователя, поэтому задавать его в качестве параметра алгоритма кластеризации представляется нецелесообразным.

Для подбора параметра трешхолда рекомендуется [10] построить гистограмму распределения попарных расстояний между треками $c_{a,i,j}$. *В задачах с выраженной кластерной структурой эта гистограмма имеет два чётких пика: зона небольших внутрикластерных расстояний и зона больших межкластерных расстояний. Параметр трешхолда задаётся как расстояние, соответствующее точке минимума между этими пиками* [10].

Чтобы достичь похожего результата алгоритмически, построим дикретизированную гистограмму. Разобьем весь спектр возможных значений $c_{a,i,j}$ на k частей. Для каждой части спектра найдем, сколько существует расстояний, содержащихся в ней, и обозначим как g_i. Заметим, что массив g и будет требуемой гистограммой. За один проход по массиву найдем два наибольших локальных максимума m_1 и m_2 (Под локальным макисмумумом будем понимать тот элемент массива, который больше обоих своих соседей). В качестве уровня трешхолда возьмем среднее между ними:

$$R = \frac{m_1 + m_2}{2}$$

Таким образом, множество треков, полученное с помощью алгоритма коллаборативной фильтрации, оказалось разбито на непересекающиеся подмножества-состояния $d_i, i = \overline{1, w}$.

2.4.6. Построение отображения перехода.

В качестве отображения перехода будем использовать внутренний цикл алгоритма Витерби (листинг 2).

Листинг 2.

```
for next_state in states:
   total = 0
   argmax = None
   valmax = 0
   for source_state in states:
       (prob, v_path, v_prob) = T[source_state]
       p = emit_p[source_state][output] * trans_p[source_state][next_state]
       prob *= p
       v_prob *= p
       total += prob
       if v_prob > valmax:
           argmax = v_path + [next_state]
           valmax = v_prob
   U[next_state] = (total, argmax, valmax)
```

Наблюдениями в смысле алгоритма Витерби будут являться входные воздействия автомата: прослушивание или пропуск пользователем композиции из определенного подмножества. Начальные данные для алгоритма задаются следующим образом:

`states` – D, состояния, найденные с помощью кластеризации множества советуемых пользователю песен.

`observations` – для каждого найденного состояния строится два наблюдения, когда пользователь прослушал трек из этого состояния и когда пользователь пропустил его.

$$observation = \{o_{d1,skip}, o_{d1,play}, o_{d2,skip}, o_{d2,plays}, \dots\}$$

`start_probability` – отношения количества прослушиваний треком из выбранного состояния к общему количеству прослушиваний.

$$start\_probability[d_l] = \frac{\sum_{s_i \in d_l} k_{u,i}}{\sum_{i=1}^{m} k_{u,i}}$$

`transition_probability` – Общая вероятность перехода их одного выделенного состояния в другое (сумма по всем элементам состояния вероятностей прехода в другое, деленная на их число).

$$transition\_probability[d_k \to d_l] = \frac{\sum_{\substack{s_i \in d_k \\ s_j \in d_l}} c_{u,i,j}}{\sum_{i=1}^{m} k_{u,i}}$$

`emission_probability` – Для каждого состояния d_i вероятность того, что пользователь прослушал трек в этом состоянии равна вероятности перехода из этого состояния в себя, вероятности того, пользователь прослушал трек в других состояниях равно нулю.

$$emission\_probability\left[d_i \to o_{di,play}\right] = transition\_probability[d_i \to d_i]$$

$$emission\_probability\left[d_i \to o_{dj,play}\right]_{i \neq j} = 0$$

Для всех наблюдений, связанных с пропуском трека, суммарная вероятность перехода в соответствующие состояния из состояния d_i в другие делится пропорционально вероятности перехода между состояниями.

$$emission\ \ probability\left[d_i \to o_{dj,skip}\right] = \frac{1 - transition\_probability[d_i \to d_i]}{transition\_probability[d_k \to d_l]}$$

2.4. Дальнейшая работа.

Дальнейшая работа над описанным методом возможна в следующих направлениях:

- Кластеризация. Выбранный метод кластеризации основывается на том, что каждый трек принадлежит только одному кластеру, хотя на самом деле это не так. Один и тот же трек может быть прослушиваем пользователем в разных эмоциональных состояниях.
- Масштабируемость. Так как предложенный метод является новым, причем требующий больше времени для работы, чем метод, основанный на принципе коллаборативной фильтрации, для него следует рассмотреть вопросы масштабируемости.

Выводы по главе 2.

1. С помощью алгоритма Витерби построена система рекомендации музыки, основанная на квантовых автоматах.

2. Построенная система автоматически строит эмоциональные состояния пользователя, основываясь на прослушанных им треках.

3. Система способна угадывать состояние пользователя по мере прослушивания музыки.

Глава 3. Практическое применение

3.1. Проблема «холодного» старта

Рекомендательные системы, построенный на принципе коллаборативной фильтрации, как правило, дают плохие результаты на малых наборах данных [11].

Иными словами, рекомендательная система не может рекомендовать элементы, о которой у нее не имеется данных. Более того, такая система не сможет дать полезных рекомендаций, если в данных о прослушиваниях мало пользователей с одинаковыми прослушанными элементами.

Такие системы должны собирать много информации о пользователях, прежде чем они начнут давать полезные рекомендации.

3.2. Yota-музыка

Сервис «Yota-музыка» компании «Скартел» специализируется на предоставлении музыкального контента своим пользователям для прослушивания через интернет-браузер или через специальный мобильный телефон HTC MAX 4G, разработанный совместно компаниями «Скартел» и «HTC».

В конце 2008-го года сервис предоставлял своим пользователям около 100 000 композиций для прослушивания, а в 2009-м планируется увеличения этого объема до двух миллионов [12]. Описанный в данной работе будет применен в построении рекомендательной системы для данного сервиса. Пользователи получат возможность слушать персональное радио, вещание которого будет подбираться для каждого пользователя индивидуально, соответствовать его музыкальным вкусам и текущему настроению.

3.3. Исследования

Построен прототип системы и проведены исследования на малых объемах данных. Для этого было взято 300 пользователей портала *Last.fm*, для которых были получены данные о 37 000 треках, 48 000 оценок и 50000 временных оценок. Для этого для каждого пользователя было взято 50 его любимых треков и 250 последних прослушанных.

На основе этих данных для отдельно выбранного пользователя удалось построить множество его соседей, и множество его эмоциональных состояний.

В зависимости от пользователя число состояний варьируется от одного до нескольких сотен. Связанно это с малыми объемами данных, Для некоторых пользователей информации о последних 250 прослушиваниях было недостаточно и в них было мало повторяющихся треков. Для пользователей с большим числом повторяющихся треков, число состояний сокращалось примерно до 10.

Было произведено сравнение с системой, построенной на принципе чисто-коллаборативной фильтрации. Для отдельного выбранного пользователя, при намеренных пропусках всех треков, не входящих в выбранное настроение, система на основе квантового автомата находила это состояние, значительно уменьшая, таким образом, число пропусков. Число пропусков в системе, на основе чисто-коллаборативной фильтрации, не зависело от преднамеренного выбора состояния и времени.

Выводы по главе 3

1. Требуется более детальное исследование построенного алгоритма на больших объемах данных для более точных оценок.
2. Возможно улучшение качества работы алгоритма на начальном этапе за счет добавления мнимых пользователей с большой базой оценок.

Выводы по работе

1. В данной работе был проведен обзор существующих методов построения рекомендательных систем, рассмотрен наиболее подходящий метод для построения музыкальной рекомендательной системы.
2. Предложен новый критерий фильтрации, повышающий качество рекомендаций.
3. Предложен новый метод построения рекомендательной системы, на основе квантовых автоматов, с использованием принципа коллаборативной фильтрации и алгоритма Витерби.
4. На отдельных примерах метод показывает положительные результаты, поэтому требует дополнительных исследований на больших объемах данных.

Список источников

[1] *George A. Akerlof.* The Market for "Lemons": Quality Uncertainty and the Market Mechanism // The Quarterly Journal of Economics, v.84, August 1970, p.488-500.

[2] *J. Herlocker, J. Konstan, L. Terveen, and J. Riedl.* Evaluating collaborative filtering recommender systems // ACM Translations on Information Systems, Vol. 22(1), 2004

[3] Robin van M. and Maarten van S. Using Content-Based Filtering for Recommendation // NetlinQ Group, Gerard Brandtstraat 26-28, 1054 JK, Amsterdam, The Netherlands.

[4] *Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl.* ItemBased Collaborative Filtering Recommendation Algorithms. (GroupLens Research Group/Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455)

[5] *Herlocker, J., Konstan, J., Borchers, A., Riedl, J.*. An Algorithmic Framework for Performing Collaborative Filtering. // Proceedings of the 1999 Conference on Research and Development in Information Retrieval. Aug. 1999.

[6] *Forney. G. D.* The Viterbi algorithm. // Proceedings of the IEEE 61(3):268–278, March 1973

[7] *Поликарпова Н., Шалыто А. А.* Автоматное программирование // Издательство: Питер, 2009 г. ISBN 978-5-388-00692-9

[8] *Воронцов К.* Лекции по алгоритмам кластеризации и шкалирования // http://www.ccas.ru/voron/download/Clustering.pdf

[9] *Лагутин М. Б.* Наглядная математическая статистика. // М.: П-центр, 2003.

[10] *Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, David M. Pennock* Methods and Metrics for Cold-Start Recommendations. // Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002): 253–260, New York City, New York: ACM. ISBN 1-58113-561-0.

[11] Обзор GSM/WiMAX-коммуникатора HTC MAX 4G // http://www.mobile-review.com/pda/review/htc-touch-4g-2.shtml