Системы рекомендаций: обзор современных подходов

А.Г. Гомзин, А.В. Коршунов
gomzin@ispras.ru, korshunov@ispras.ru

Аннотация. Статья представляет собой обзор основных алгоритмов, использующихся в системах рекомендаций. Рассмотрены методы коллаборативной фильтрации, методы, анализирующие содержимое объектов и методы, использующие базы знаний. Все рассматриваемые методы имеют свои недостатки. В статье рассмотрено, какими способами можно комбинировать методы построения рекомендаций, чтобы избавиться от этих недостатков.

Ключевые слова: система рекомендаций, коллаборативная фильтрация, базы знаний, гибридная система рекомендаций

1. Введение

Очень часто человек выбирает, какие новые фильмы посмотреть, какую новую музыку послушать, какие товары купить в интернет-магазине. Это совсем не простая задача, потому что он имеет в распоряжении лишь названия, краткие описания доступных фильмов или товаров, и, может быть, советы друзей и знакомых. Имея столь малое количество информации, очень трудно определить, понравится ли конкретный фильм, песня или товар данному человеку. Задача усложняется еще и тем, что объектов, из которых приходится выбирать, очень много. Поэтому в сети Интернет широко распространены системы рекомендаций, помогающие пользователю решить задачу выбора песен, фильмов, товаров.

Системы рекомендаций позволяют пользователю выбрать среди всех доступных объектов именно те, которые будут ему интересны. Эти системы обрабатывают информацию о различных объектах, а также о том, какие пользователи какие объекты купили, посмотрели, послушали и т.д. Примерами таких сервисов являются last.fm, hunch.com, youtube.com и другие. Имея эти данные и алгоритмы, которые будут рассмотрены в данной работе, можно быстро и качественно отфильтровать самые подходящие конкретному пользователю объекты.

Кроме того, есть системы, которые рекомендуют других пользователей, имеющих схожие интересы, или потенциальных знакомых. Как правило, это

социальные сети. Facebook (facebook.com), ВКонтакте (vk.com), last-fm (last.fm).

В данной работе будут рассмотрены основные подходы и алгоритмы, используемые в различных системах рекомендаций. Будут выявлены сильные и слабые стороны каждого подхода.

2. Основные понятия

Перед тем, как рассматривать алгоритмы, введем некоторые понятия.

Объект – это песня, фильм, товар, пользователь (в случае рекомендации друзей). Т.е. то, что потребляют пользователи системы рекомендаций. Это то, что им нужно рекомендовать.

Пользователь – это человек, зарегистрированный в системе, он может покупать, слушать, смотреть, оценивать объекты и пользоваться сервисом рекомендации.

Рекомендация – это объект или несколько объектов, которые система рекомендации выдает пользователю.

Основная задача системы рекомендации – имея данные об объектах и пользователях, получить список объектов, наиболее интересных для конкретного пользователя. В зависимости от того, какие данные используются для расчета рекомендаций, системы делятся на три больших класса:

- Методы коллаборативной фильтрации [1-4]
- Методы, анализирующие содержимое объектов
- Методы, основанные на знаниях

Методы коллаборативной фильтрации предполагают, что интересы пользователей представлены оценками, которые они дают объектам после просмотра, покупки и т.д. Основная идея данных методов заключается в сравнении между собой интересов различных пользователей или объектов на основе этих оценок. При этом никакой дополнительной информации о самих пользователях и объектах не используется.

Методы второго класса, наоборот, используют содержимое объектов для получения рекомендаций. Эти методы работают в тех случаях, когда содержимое объектов представлено в виде текстов. Они хорошо подходят для рекомендации книг. Также их можно использовать для сравнения названий, описаний и другой текстовой информации, доступной у фильмов, песен, товаров и т.д. [5]

Методы, основанные на знаниях, требуют от пользователя описать свои требования к нужным ему объектам. А затем ищут с использованием своей базы знаний объекты, удовлетворяющие поставленным требованиям.

3. Коллаборативная фильтрация

Коллаборативная фильтрация (Collaborative filtering) – это метод рекомендации, при котором анализируется только реакция пользователей на объекты: оценки, которые выставляют пользователи объектам. Оценки могут быть как явными (пользователь явно указывает, на сколько «звездочек» он оценивает объект), так и неявными (например, количество просмотров одного ролика). Чем больше оценок собирается, тем точнее получатся рекомендации. Получается, что пользователи помогают друг другу в фильтрации объектов. Поэтому такой метод называется также совместной фильтрацией.

3.1. Описание алгоритма

Пусть в системе есть пользователи и объекты. Пусть некоторые пользователи оценили некоторые объекты. И пусть оценка – это натуральное число от 1 до 5. Тогда все оценки можно изобразить в виде матрицы (см. рис. 1).

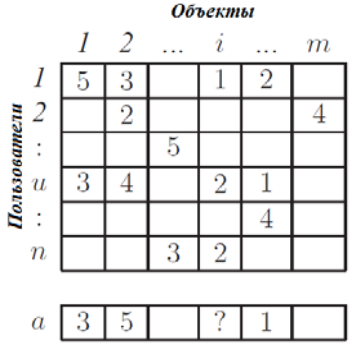

***Рис. 1** Матрица оценок.*

Пусть имеется пользователь *a*. Наша задача – предсказать, какую оценку поставил бы пользователь *a* объекту *i*. Будем рассматривать только пользователя *a* и тех пользователей, которые оценили объект *i*. Алгоритм включает в себя 3 шага:

1. Для каждого пользователя *u* вычислим, насколько его интересы совпадают с интересами пользователя *a*
2. После этого выберем множество пользователей, наиболее близких к *a*
3. Предскажем оценку на основе оценок объекта *i* "соседями" из предыдущего шага

Первый шаг. Каждому пользователю в матрице *R* соответствует одна строка. Поэтому будем вычислять близость векторов-строк пользователей.

Существует множество способов подсчета близости векторов [6,7]. Один из самых простых – посчитать косинус между этими векторами:

$$sim(u,a) = \frac{\sum_{i=1}^{m} r_{a,i} \cdot r_{u,i}}{\sqrt{\sum_{i=1}^{m} r_{a,i}} \cdot \sqrt{\sum_{i=1}^{m} r_{u,i}}} \quad (1)$$

Здесь *sim(u,a)* – мера близости (похожести) пользователей *a* и *u*. $r_{u,i}$ – значение матрицы *R*: *u* строка, *i* столбец. *sim(u,a)* принимает значения из отрезка *[0,1]*. Если пользователь не указал оценку для какого-то объекта, соответствующее значение матрицы равно 0.

Существует другая мера близости между векторами – коэффициент корреляции Пирсона:

$$sim(u,a) = \frac{\sum_{i \in I} (r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{\sqrt{\sum_{i \in I} (r_{a,i} - \overline{r_a})^2 \cdot \sum_{i \in I} (r_{u,i} - \overline{r_u})^2}} \quad (2)$$

Здесь *I* – множество объектов, которые оценил как пользователь *a*, так и пользователь *u*. $\overline{r_a}$ и $\overline{r_u}$ - это средние оценки пользователей *a* и *u*, соответственно.

В числителе подсчитывается произведение отклонений оценок двух пользователей от средних значений для одного объекта. Знаменатель необходим для того, чтобы данная величина принимала значения из отрезка *[-1,1]*. Чем сильнее совпадают интересы, тем ближе значение близости к *1*. Если коэффициент корреляции Пирсона отрицателен, то интересы пользователей противоположные.

(1) и (2) – самые распространенные меры близости, используемые в данном алгоритме. Коэффициент корреляции Пирсона (2) будет оставаться высоким, если все оценки сравниваемых пользователей отличаются на постоянную величину (один пользователь оценивает более строго, другой – ставит более высокие оценки). Косинус (1) в этом случае выдаст меньшее значение. Но, если оценки двух пользователей отличаются пропорционально, то косинус (1) будет больше, корреляция (2) – меньше. В обоих случаях оценки пользователей похожи. Поэтому, какую из этих метрик использовать – зависит от конкретной задачи.

Существуют и другие меры близости [6,7]. В данной работе они рассматриваться не будут.

Второй шаг. Теперь нужно выбрать множество *K* наиболее похожих на *a* пользователей. Можно выбрать всех пользователей. Но так как пользователей с непохожими или несильно пересекающимися интересами довольно много, то они будут отрицательно влиять на точность предсказания оценки для объекта *i*. Кроме того, количество пользователей влияет на объем вычислений на третьем шаге.

Одно из возможных решений – установить порог меры близости, вычисленной на первом шаге: (1), (2). Пользователи с мерой близости, превышающей порог, войдут во множество *K*. Остальные – нет. Но чаще всего выбирается целая константа *k*. Затем все пользователи сортируются по убыванию меры близости. И во множество *K* входят *k* пользователей, наиболее близких к *a*.

Третий шаг. Имея множество *K* близких пользователей нужно вычислить оценку, которую поставил бы пользователь *a* объекту *i*. Напомним, что рассматриваются только те пользователи, которые оценили объект *i*.

Нужная оценка вычисляется по формуле:

$$p_{a,i} = \overline{r_a} + \frac{\sum_{u \in K} (r_{u,i} - \overline{r_u}) \times sim(a,u)}{\sum_{u \in K} | sim(a,u) |} \quad (3)$$

$p_{a,i}$ – это предсказываемая оценка пользователя *a* для объекта *i*. За основу берется его средняя оценка $\overline{r_a}$, а затем прибавляется среднее отклонение оценки других пользователей из множества *K* для объекта *i* от их средней оценки. Чем ближе пользователь *u* к пользователю *a* (согласно мере близости *sim(a,u)*, вычисленной на первом шаге), тем сильнее его вклад в предсказание оценки.

Таким образом, описанный алгоритм предсказывает оценки для объектов, которые текущий пользователь еще не оценил. Для того чтобы сделать рекомендацию для данного пользователя, достаточно предсказать оценки для всех неоцененных объектов и выбрать объекты с наибольшей предсказанной оценкой.

3.2. Альтернативные алгоритмы и улучшения

Описанный выше алгоритм основан на сравнения между собой пользователей. Он называется User-based Collaborative filtering. Существует также другой подход: вместо пользователей сравнивать объекты (Item-based Collaborative filtering). В этом случае алгоритм почти такой же, включает 3 шага:

1. Для каждого объекта *j* вычислим, насколько он похож на объект *i*, для которого предсказывается оценка.
2. Выберем множество объектов, наиболее близких к *i*
3. Предскажем оценку на основе оценок выбранных на втором шаге объектов пользователем *a*.

Сравнение объектов на первом шаге происходит также как и в (1) и (2)

$$sim(i,j) = \frac{\sum_{u \in U} (r_{u,i} - \overline{r_i}) \cdot (r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U} (r_{u,i} - \overline{r_i})^2 \cdot \sum_{u \in U} (r_{u,j} - \overline{r_j})^2}} \quad (4)$$

Здесь суммирование ведется по множеству *U* пользователей, оценивших объекты *i* и *j*. $\overline{r_i}$ – средняя оценка объекта *i*.

Второй шаг аналогичен второму шагу алгоритма, основанного на сравнении пользователей. Основное отличие в том, что *K* – множество близких к *i* ***объектов***. На третьем шаге считается взвешенная средняя оценка похожих объектов, полученная пользователем *a*:

$$p_{a,i} = \frac{\sum_{j \in K} r_{u,j} \times sim(i,j)}{\sum_{j \in K} | sim(i,j) |} \quad (5)$$

В алгоритме, основанном на сравнении объектов, использовалась более сложная формула (3). Это связано с тем, что средняя оценка объекта показывает, насколько хорошим является объект, а средняя оценка пользователя показывает, насколько строго пользователь оценивает объекты. Формула (3) вычисляет оценку с учетом того, что строгость оценки разных пользователей, вообще говоря, различна.

Алгоритм, основанный на сравнении объектов, имеет несколько преимуществ. Так как интересы пользователей часто меняются, а средняя оценка объекта остается постоянной, то при сравнении объектов достаточно посчитать меру близости каждого с каждым и сохранить в памяти. В частом обновлении этих данных нет необходимости. Причем эти меры близости можно посчитать в оффлайне, т.е. когда в системе нет активных пользователей.

В современных системах рекомендаций постоянно растет число пользователей и объектов, поэтому наступает момент, когда необходимо использовать распределенную архитектуру системы. При этом возникает проблема хранения данных: было бы неплохо похожих пользователей или похожие объекты хранить рядом. Для ее решения можно применить алгоритмы кластеризации для пользователей или объектов, используя меру близости между ними: (1), (2), (4). Разбиение пользователей или объектов на группы дает еще одно преимущество: так как в кластере находятся похожие объекты, то нужный объект достаточно на первом шаге сравнивать с объектами из своего кластера. Это значительно сокращает количество вычислений.

Описанные выше методы коллаборативной фильтрации называются методами, основанными на работе с памятью, так как необходимо хранить матрицу *R*. Есть и другая группа алгоритмов – методы, основанные на моделях. Идея заключается в том, что имея матрицу *R* можно построить модель, которая будет меньше по объему занимаемой памяти. Используя полученную модель, можно будет вычислять рекомендации. Примерами таких моделей являются Байесовский алгоритм [2], при котором исходная задача сводится к задаче классификации; Марковские модели [8], которые позволяют учитывать не только похожесть объектов, но и их ценность; а также другие модели [3].

4. Анализ содержимого

Коллаборативная фильтрация использует данные обратной связи пользователя (оценки, отзывы). Однако, не всегда доступно необходимое количество оценок. Кроме того, существует проблема "холодного старта": система не знает, что рекомендовать новому пользователю и кому рекомендовать новый объект.

В случаях, когда коллаборативная фильтрация не дает результата, приходится использовать информацию об объектах. Например, был снят фильм – продолжение какого-то фильма. Он добавляется в систему. Возникает вопрос, кому его рекомендовать. Коллаборативная фильтрация ответ не даст из-за проблемы «холодного старта». Но есть смысл рекомендовать новый фильм тем, кто посмотрел первую часть фильма.

4.1. Представление данных об объектах

Каждый объект в любой системе рекомендаций имеет название, а также и другую текстовую информацию, которую можно использовать. В табл. 1 представлен пример доступной информации о книгах на английском языке в интернет-магазине.

Название	Жанр	Автор	Цена	Ключевые слова
The Night of the Gun	Memoir	David Carr	29.90	press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	45.90	American fiction, murder, neo-Nazism

***Табл. 1** Информация о книгах в интернет-магазине.*

Пользователи системы имеют свой профиль. Там они указывают информацию о себе и своих интересах: любимые жанры, любимые авторы и т.д. Имея эти данные и информацию о книгах из таблицы 1 можно сделать выводы, интересна ли конкретная книга пользователю. Но, в этом случае мера похожести интересов пользователя и книги принимает два возможных значения: 1 – если жанр книги есть в профиле пользователя, 0 – в противном случае.

Для более точных рекомендаций используется другой подход: зная информацию о книгах, купленных человеком, ему можно посоветовать похожие книги. Похожесть двух книг измеряется с помощью сравнения

данных из табл. 1. Например, если $keywords(B_i)$ – множество ключевых слов книги B_i, то сравнить ее с книгой B_j можно следующим образом:

$$dice(B_1, B_j) = \frac{|\, keywords(B_i) \bigcap keywords(B_j) \,|}{|\, keywords(B_i) \,| + |\, keywords(B_j) \,|} \quad (6)$$

Формула (6) называется коэффициентом Дайса. Он принимает значения из отрезка *[0,1]*. Он показывает, насколько близки два множества.

Но множество ключевых слов почти никогда не доступно явно. Их можно получить, анализируя непосредственно содержание объектов. Естественно, что извлекать ключевые слова можно только из книг, статей и других текстовых объектов. Для выделения ключевых слов используются различные алгоритмы: с использованием Википедии [9], TF-IDF и др.

4.2. TF-IDF

Пусть имеется набор текстов (документов). Все тексты разбиты на слова. Коэффициент TF-IDF показывает, насколько слово важно в тексте. Основная идея в том, что слово считается более важным, если оно часто встречается в одном тексте и редко в других.

TF (term frequency) -- это нормализованная частота слова в тексте:

$$TF(x, D) = \frac{freq(x, D)}{\max_{y \in D} freq(y, D)} \quad (7)$$

Здесь *freq(x,D)* – количество слов *x* в документе *D*. *TF* принимает значения из отрезка *[0,1]*.

IDF (inverse document frequency) – обратная частота документов.

$$IDF(x) = \frac{N}{n(x)} \quad (8)$$

Здесь *N* – количество документов в наборе, *n(x)* – количество документов, в которых встречается слово *x*.

Коэффициент *TF-IDF* вычисляется как произведение *TF* на *IDF*.

Для сравнения двух текстов, их можно представить в виде векторов в евклидовом многомерном пространстве. Каждому измерению соответствует слово, а значение каждой компонент вектора равно *TF-IDF*. А затем посчитать косинус между этими векторами.

Для уменьшения вычислительных затрат, затрат памяти и шума в вычислениях используются различные эвристики:

1. **Стоп-слова**. Такие слова, как предлоги, артикли, междометия встречаются в текстах очень часто, хотя не несут никакой полезной информации. Поэтому эти слова заносится в список

стоп-слов и не рассматриваются при построении векторной модели.

2. **Стемминг**. Известно, что одно и то же слово может принимать множество форм (разные падежи, времена и т.д.). Поэтому существуют алгоритмы, позволяющие превести слово к нормальной форме, чтобы отождествлять слова
3. **Уменьшение количества компонент вектора**. Различных слов в документе очень много. Но можно уменьшить их количество, оставив только с наибольшим весом *TF-IDF*.

Более подробно методы сравнения текстов рассмотрены в [10].

5. Методы, основанные на знаниях

Методы коллаборативной фильтрации и анализа содержимого хорошо работают, когда человек довольно часто покупает товары или слушает музыку. Но если это происходит редко (например, при покупке дома, автомобиля, компьютера), эти методы не помогут, потому что система не имеет нужного количества информации о покупаемых пользователем товарах.

Для решения такой задачи используются методы, основанные на знаниях [11]. Они делятся на две группы:

- Использование жестких ограничений
- Выбор близких объектов

Идея обоих методов следующая: пользователи формулируют свои требования к товару, система пытается найти нужный товар.

В первом случае ищутся и рекомендуются только те объекты, которые точно соответствуют всем требованиям пользователя. Во втором случае ищутся объекты с характеристиками, близкими к требованиям (с использованием некоторых мер близости).

5.1. Знания об объектах и запросы пользователей

Информация об объекте обычно представляется в виде набора параметров. Например, для фотокамеры это размер дисплея, зум, количество мегапикселей и т.д. Пусть в магазине имеются следующие фотокамеры (см. табл. 2).

номер	цена	Мега-пиксели	опт. зум	размер экрана	видео	звук	водостойкость
1	148	8.0	4x	2.5	нет	нет	да
2	182	8.0	5x	2.7	да	да	нет
3	189	8.0	10x	2.5	да	да	нет
4	196	10.0	12x	2.7	да	нет	да
5	151	7.1	3x	3.0	да	да	нет
6	199	9.0	3x	3.0	да	да	нет
7	259	10.0	3x	3.0	да	да	нет
8	279	9.1	10x	3.0	да	да	да

Табл. 2 *Фотокамеры в интернет-магазине.*

Задача системы – выбрать камеры, удовлетворяющие запросам пользователя. Но не всегда эти запросы выражаются в виде множества значений параметров, например, "камера ценой не более $200, не менее 8 мегапикселей, с поддержкой видео". Достаточно часто пользователя интересуют некоторые функциональные особенности. Тогда его запросы принимают вид: "камера для спортивной съемки", или "камера для печати больших фотографий". Поэтому выделяются два типа свойств: свойства товара и пользовательские свойства.

В нашем примере с камерами свойствами товара являются:

цена(0...1000), мегапиксели(3.0...12.0), опт-зум(4x-12x), дисплей(2.5...3.0), видео(да,нет), звук(да,нет), водостойкость(да,нет).

Пользовательские свойства:

максимальная_цена(0...1000), использование(цифровое, печать_больших_фото, печать_маленьких_фото), фотографии (спортивные, пейзажи, портреты, макросъемка).

Т.е. в своем запросе пользователь может указать максимальную цену, какого размера он будет печатать фотографии, что он будет фотографировать (пейзажи, портреты).

Помимо знаний о товарах (см. табл. 2) в базе знаний присутствуют ограничения. Ограничения могут представлять собой разрешенные значения пользовательских свойств, например:

использование=печать_больших_фото → максимальная_цена>200, т.е. система не позволяет пользователю указать минимальную цену меньше 200, если он хочет печатать большие фотографии.

Другой вид ограничений позволяет преобразовать свойства пользователя в свойства объектов:

использование=печать_больших_фото → мегапиксели>5.0, т.е. если пользователь хочет большие фотографии, то не следует ему рекомендовать камеры, меньше чем с 5 мегапикселями.

Итак, задача системы – выбрать объекты, удовлетворяющие запросам пользователя, с помощью знаний об объектах и ограничениях.

5.2. Использование жестких ограничений

Идея метода проста: найти объекты в базе данных, полностью удовлетворяющие требованиям пользователя. Но, поиск объектов не является основной проблемой. Главной задачей является получение требований пользователя. Нужно определить, как взаимодействовать с пользователем, какие вопросы и в какой последовательности задавать, какие вопросы можно не задавать. Таким образом, основная задача – получить систему, удобную для пользователя.

Для того чтобы ограничить количество задаваемых пользователю вопросов, используются значения параметров по умолчанию. Они делятся на 3 группы:

- **Статические умолчания.** Пример: *использование = печать_больших_фото.* Так как большинство пользователей требуют возможности печати больших фотографий, то можно установить такое значение параметра *использование* по умолчанию
- **Зависимые умолчания.** Пример: *использование = печать_маленьких_фото → максимальная_цена = 300.* Т.е. если пользователь указал, что будут печататься маленькие фотографии, то его можно не спрашивать о максимальной цене и указать ее равной 300
- **Производные умолчания.** Если первые два типа умолчаний определяются заранее, то производные умолчания вычисляются с использованием логов взаимодействия системы с пользователями

Пользователи указывают не все требуемые параметры, а только те, которые считают наиболее существенными. Логи взаимодействия предыдущих пользователей с системой используются также для предсказывания, какой следующий параметр будет интересен пользователю. Табл. 3 содержит информацию о том, в какой последовательности пользователи устанавливали атрибуты.

№	1 атрибут	2 атрибут	3 атрибут	4 атрибут	…
1	*цена*	*оптический зум*	*мегапиксели*	*видео*	…
2	*цена*	*оптический зум*	*мегапиксели*	*видео*	…
3	*цена*	*мегапиксели*	*оптический зум*	*размер дисплея*	…
4	*мегапиксели*	*цена*	*оптический зум*	*размер дисплея*	…
5	*мегапиксели*	*цена*	*размер дисплея*	*оптический зум*	…

Табл. 3 *Последовательность ответов пользователей на вопросы системы.*

Параметры предсказываются с учетом их популярности. Первый предсказываемый параметр – самый популярный первый параметр среди предыдущих пользователей (в нашем примере – *цена*). Если пользователь указал в качестве первого параметра *цену*, а в качестве второго – *оптический зум*, то ищется самый популярный третий атрибут в строках с номерами 1 и 2 (см. табл. 3).

Если пользователь указал слишком жесткие требования, то может оказаться, что искомых объектов нет в системе. В этом случае система последовательно и автоматически ослабляет ограничения пользователя на объекты. Методы выбора минимального множества ослабляемых параметров рассмотрены в [10]. В данной работе они рассматриваться не будут.

5.3. Выбор близких объектов

В отличие от методов, использующих жесткие ограничения, методы данного класса рекомендуют объекты, наиболее близкие к требованиям пользователей. В этом случае возможна рекомендация объектов, которые не полностью удовлетворяют требованиям.

Мера близости между требованиями и свойствами вычисляется по формуле:

$$similarity(p, REQ) = \frac{\sum_{r \in REQ} w_r \cdot sim(p, r)}{\sum_{r \in REQ} w_r} \quad (9)$$

Здесь p – объект, REQ – множество требований пользователя. r – требование. w_i – вес данного требования, $sim(p,r)$ – мера близости объекта p к требованию r.

Численные свойства для пользователя делятся на три группы:

- «больше - лучше» (обозначаются MIB). Например, чем больше мегапикселей в камере, тем она считается лучше
- «меньше - лучше» (обозначаются LIB). Например, чем цена ниже, тем лучше для пользователя
- «ближе - лучше». Например, пользователю нужен монитор с определенной диагональю, больше или меньше – хуже.

Поэтому для каждого типа мера близости определяется следующим образом:

«Больше – лучше»:

$$sim(p,r) = \frac{\varphi_r(p) - \min(r)}{\max(r) - \min(r)} \quad (10)$$

«Меньше – лучше»:

$$sim(p,r) = \frac{\max(r) - \varphi_r(p)}{\max(r) - \min(r)} \quad (11)$$

«Ближе – лучше»:

$$sim(p,r) = 1 - \frac{|\varphi_r(p) - r|}{\max(r) - \min(r)} \quad (12)$$

Здесь *sim(p,r)* – мера близости объекта *p* к требованию *r*. *min(r), max(r)* – минимальные и максимальные значения свойства, соответственно. *r* в (12) – точное значение требования пользователя. $\varphi_r(p)$ – значение соответствующего свойства у объекта *p*.

6. Гибридные методы

Итак, выше были рассмотрены 3 основных вида систем рекомендаций. Каждый из них имеет свои плюсы и минусы [12]:

Метод	Плюсы	Минусы
Коллаборативная фильтрация	• достаточно иметь только данные о рейтингах	• проблема «холодного старта» (при добавлении новых пользователей и объектов) • нужно много данных о разных пользователях
Анализ содержимого	• достаточно иметь профиль текущего пользователя и содержимое объектов	• проблема «холодного старта» (только для новых пользователей) • работают преимущественно только с текстовыми данными
Основанный на знаниях	• не обязательно иметь много данных о покупках	• необходимость использования базы знаний

Табл. 4*. Положительные и отрицательные стороны методов рекомендаций.*

Крупные системы рекомендаций (СР), которые используют различные данные об объектах, пользователях, связях между ними, представляют собой сложную гибридную систему. Она является объединением различных подходов и алгоритмов: коллаборативной фильтрации, анализа содержимого, анализа социальных связей и связей между объектами, и т.д.

Выделяется три типа гибридных систем [9]:

- С монолитной организацией
- С параллельной организацией
- С конвейерной организацией

Схемы организации гибридных СР представлены на рисунках (2-4):

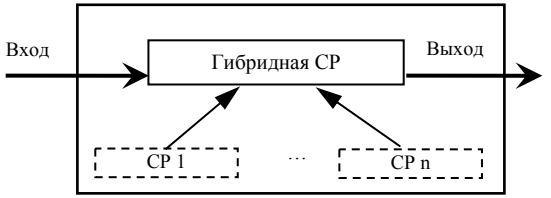

Рис. 2. *Монолитная организация гибридной СР.*

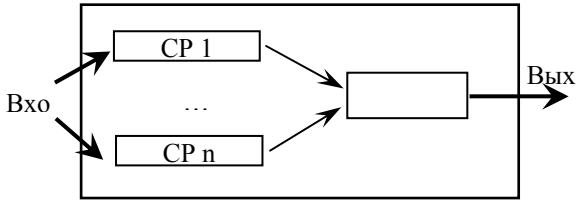

Рис. 3. *Параллельная организация гибридной СР.*

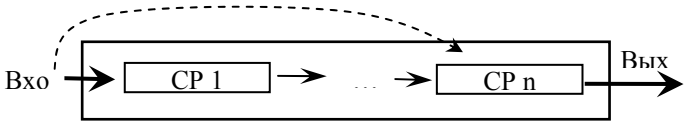

Рис. 4. *Конвейерная организация гибридной СР.*

6.1. Монолитная организация

В отличие от параллельной и конвейерной организаций гибридных систем, системы с монолитной организацией представляют собой один модуль, включающий в себя различные методы обработки имеющихся данных (см. рис. 2).

Большинство гибридных систем такого рода реализуют коллаборативную фильтрацию, усиленную особенностями содержимого. Например, если алгоритм коллаборативной фильтрации рекомендует некоторому пользователю несколько книг, то среди них можно выбрать наиболее подходящую с помощью сравнения жанров этих книг с популярными жанрами данного пользователя. Т.е. если пользователь предпочитает фантастику (что легко узнать, просмотрев жанры всех оцененных/прочитанных пользователем книг или его профиль), то среди книг двух жанров: история и фантастика, система предложит второй вариант.

Многие системы используют псевдооценки объектов пользователями:

$$v_{u,i} = \begin{cases} r_{u,i}, \text{если пользователь } u \text{ оценил объект } i \\ c_{u,i}, \text{иначе} \end{cases} \quad (13)$$

Здесь $c_{u,i}$ – предсказание оценки на основе анализа содержимого.

Иногда пользователь запрашивает рекомендации объектов, удовлетворяющих некоторым критериям. Здесь уже могут использоваться методы, основанные на знаниях. Но, сочетание систем, использующих базы знаний с коллаборативными методами, пока не изучено и практически не используется.

6.2. Параллельная организация

Схема гибридной системы с параллельной организацией представлена на рис. 3. Исходные данные поступают на каждый модуль системы. После того, как все подсистемы выдадут ответ, начинает работать агрегирующий модуль. Есть три основных подхода выбрать множество наиболее релевантных рекомендаций.

Первый подход – смешать все рекомендации выдать их пользователю:

$$rec(u) = \bigcup_{k=1}^{n} rec_k(u) \quad (14)$$

rec(u) – результат работы «агрегатора» для пользователя *u*, $rec_k(u)$ – результат работы k-ого модуля системы для пользователя *u*.

Второй подход – для каждого объекта подсчитать сумму рейтингов, полученных параллельными модулями и вывести объекты с наибольшим рейтингом:

$$r(u,i) = \sum_{k=1}^{n} \beta_k \cdot r_k(u,i) \quad (15)$$

Здесь *r(u,i)* – нужная предсказанная оценка объекта *i* пользователем *u*, $r_k(u,i)$ – оценка, полученная *k*-м модулем, β_k – весовые коэффициенты.

И, наконец, третий подход – выбрать одну подходящую систему рекомендации, реализованную в одном из модулей для каждой конкретной ситуации. Например, если модуль коллаборативной фильтрации не дает ответ из-за нехватки данных (проблема «холодного старта»), система выбирает метод анализа содержимого.

6.3. Конвейерная организация

Схема гибридной системы с конвейерной организацией представлена на рис. 4. Задача рекомендации делится на этапы, которые выполняются последовательно. Результат работы очередного этапа используется в последующих.

7. Заключение

В данной работе были кратко изложены основные алгоритмы, используемые в системах рекомендаций. Различные методы используют различные данные о пользователях и об объектах. Каждый подход имеет свои достоинства и недостатки. Например, метод коллаборативной фильтрации рекомендует объекты, не имея никакого представления о том, что они собой представляют. Однако алгоритм не позволяет рекомендовать новые объекты. Проблему рекомендации новых объектов решают методы анализа содержимого. Но для их хорошей работы требуются текстовые данные об объектах. Если информации о пользователях, объектах и оценках недостаточно для этих алгоритмов, применяются методы, использующие базы знаний. При этом интерактивно выявляются требования пользователя.

Чем больше доступно данных, тем более точную систему рекомендации можно разработать, используя различные гибридные методы рекомендации.

Список литературы

[1] P. Melville, V. Sindhwani; *Recommender systems*; Encyclopedia of Machine Learning, 2010

[2] X. Su, T.M. Khoshgoftaar; *A Survey of Collaborative Filtering Techniques*; Advances in Artificial Intelligence, 2009

[3] К. В. Воронцов; *Методы коллаборативной фильтрации и тематического моделирования*; 2011; [PDF] http://www.machinelearning.ru/wiki/images/9/95/Voron-ML-CF.pdf

[4] А.Г. Гомзин; *Collaborative filtering (Коллаборативная фильтрация)*; 2011; [PDF] http://modis.ispras.ru/seminar/wp-content/uploads/2011/11/CFFinal.pdf

[5] D. Turdakov; *Recommender system based on user-generated content*; SYRCODIS, 2007

[6] W.P. Jones, G.W. Furnas; *Pictures of Relevance: A Geometric Analysis of Similarity Measures*; Journal of the American society for information science. 38(6): 420-442, 1987; стр. 420-442.

[7] S. Choi, S. Cha, C.C. Tappert; *A Survey of Binary Similarity and Distance Measures*; Journal of Systemics, Cybernetics and Informatics, Vol 8 No 1 2010, стр. 43-48

[8] G. Shani, D. Heckerman, R. I. Brafman; *An MDP-Based Recommender System*; Journal of Machine Learning Research 6, 2005; стр. 1265–1295

[9] А.В. Коршунов; *Извлечение ключевых терминов из сообщений микроблогов с помощью Википедии*; Труды Института системного программирования РАН, том 20, 2011. стр. 283-296

[10] A. Huang; *Similarity Measures for Text Document Clustering*; NZCSRSC, 2008

[11] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich; *Recommender Systems. An Introduction*; Cambridge University Press 32 Avenue of the Americas, New York, NY 10013-2473, USA, 2011; 352 страниц

[12] R. Burke; *Hybrid Recommender Systems: Survey and Experiments*; User Modeling and User-Adapted Interaction 12, 2002; стр. 331-370