

[КАК СТАТЬ АВТОРОМ](#)[Все потоки](#)   [Разработка](#)   [Администрирование](#)   [Дизайн](#)   [Менеджмент](#)   [Маркетинг](#)   [Научноп](#)**194.42**

Рейтинг

## ГК ЛАНИТ

Ведущая многопрофильная группа ИТ-компаний в РФ

crazyhatter 21 августа 2018 в 11:13

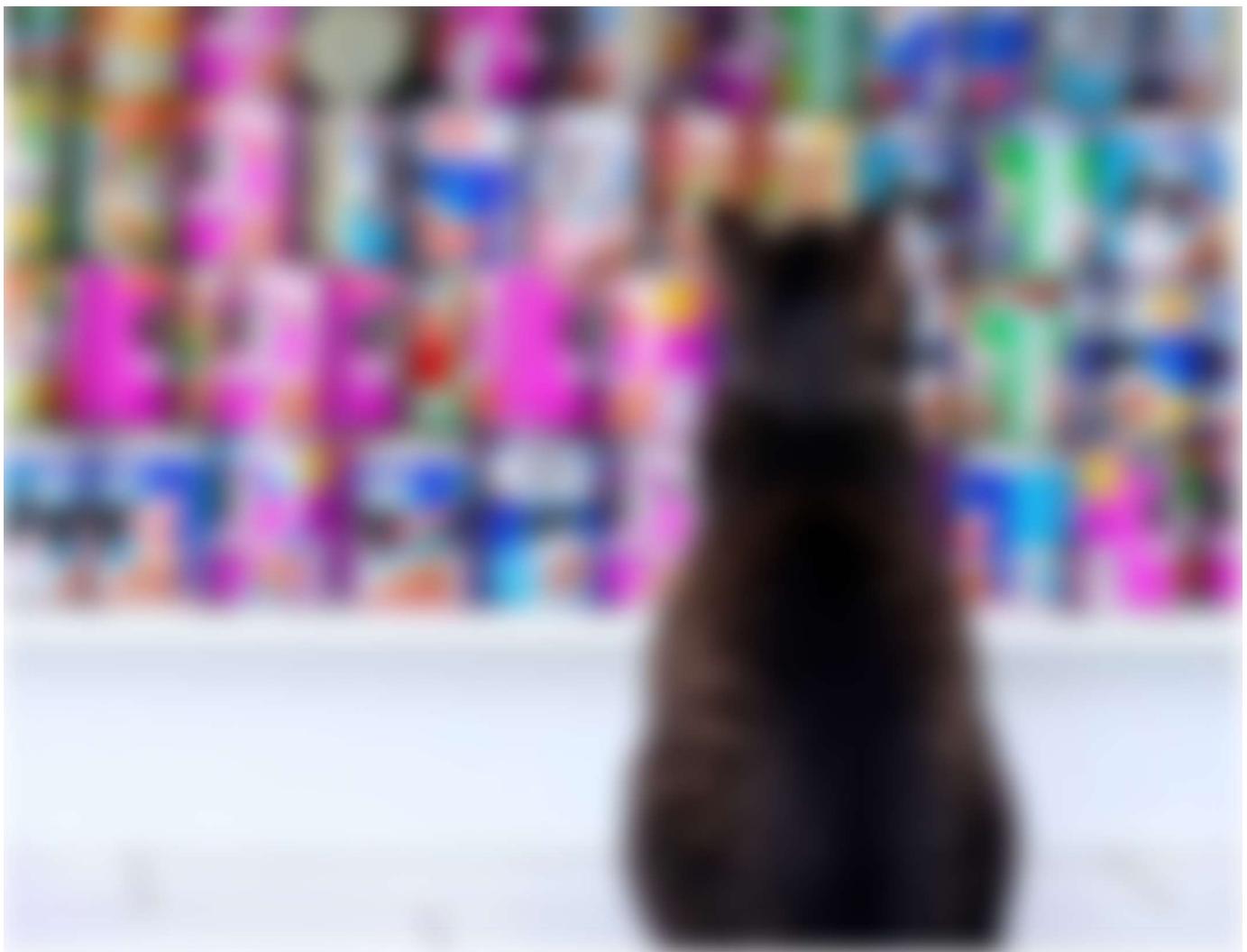
## Анатомия рекомендательных систем. Часть первая

Блог компании ГК ЛАНИТ , Data Mining \*, Алгоритмы \*, Big Data \*, Машинное обучение \*

Я работаю data-саентистом в компании CleverDATA. Мы занимаемся проектами в области машинного обучения, и один из наиболее частых запросов на разработку основанных на машинном обучении маркетинговых решений — это разработка рекомендательных моделей.

В данной статье я расскажу о рекомендательных системах, постараюсь дать максимально полный обзор существующих подходов и на пальцах объясню принципы работы алгоритмов. Часть материала базируется на неплохом курсе по рекомендательным системам лаборатории MovieLens (которая большинству знакома по одноименному датасету для тестирования рекомендаций), остальное — из личного опыта. Статья состоит из двух частей. В первой описана постановка задачи и дан обзор простых (но популярных) алгоритмов рекомендаций. Во второй статье я расскажу о более продвинутых методах и некоторых практических аспектах реализации.

[+45](#)[49K](#)[217](#)



Источник

## Обзор и постановка задачи

Задача рекомендательной системы – проинформировать пользователя о товаре, который ему может быть наиболее интересен в данный момент времени. Клиент получает информацию, а сервис зарабатывает на предоставлении качественных услуг. Услуги — это не обязательно прямые продажи предлагаемого товара. Сервис также может зарабатывать на комиссионных или просто увеличивать лояльность пользователей, которая потом выливается в рекламные и иные доходы.

В зависимости от модели бизнеса рекомендации могут быть его основой, как, например, у TripAdvisor, а могут быть просто удобным дополнительным сервисом (как, например, в каком-нибудь интернет-магазине одежды), призванным улучшить Customer Experience и сделать навигацию по каталогу более удобной.

Персонализация онлайн-маркетинга – очевидный тренд последнего десятилетия. По оценкам McKinsey, 35% выручки Amazon или 75% Netflix приходится именно на рекомендованные товары и процент этот, вероятно, будет расти. Рекомендательные системы – это про то, что предложить клиенту, чтобы сделать его счастливым.

Чтобы проиллюстрировать всё многообразие рекомендательных сервисов, приведу список основных характеристик, с помощью которых можно описать любую рекомендательную систему.

## 1. Предмет рекомендации – что рекомендуется.

Здесь большое разнообразие – это могут быть товары (Amazon, Ozon), статьи (Arxiv.org), новости (Surfingbird, Яндекс.Дзен), изображения (500px), видео (YouTube, Netflix), люди (Linkedin, LonelyPlanet), музыка (Last.fm, Pandora), плейлисты и прочее. В целом, рекомендовать можно что угодно.

## 2. Цель рекомендации – зачем рекомендуется.

Например: покупка, информирование, обучение, заведение контактов.

## 3. Контекст рекомендации – что пользователь в этот момент делает.

Например: смотрит товары, слушает музыку, общается с людьми.

## 4. Источник рекомендации – кто рекомендует:

- аудитория (средний рейтинг ресторана в TripAdvisor),
- схожие по интересам пользователи,
- экспертное сообщество (бывает, когда речь о сложном товаре, таком, как, например, вино).

## 5. Степень персонализации.

Неперсональные рекомендации – когда вам рекомендуют то же самое, что всем остальным. Они допускают таргетинг по региону или времени, но не учитывают ваши личные предпочтения.

Более продвинутый вариант – когда рекомендации используют данные из вашей текущей сессии. Вы посмотрели несколько товаров, и внизу страницы вам предлагаются похожие.

Персональные же рекомендации используют всю доступную информацию о клиенте, в том числе историю его покупок.

## 6. Прозрачность.

Люди больше доверяют рекомендации, если понимают, как именно она была получена. Так меньше риск нарваться на «недобросовестные» системы,

продвигающие проплаченный товар или ставящие более дорогое товары выше в рейтинге. Кроме того, хорошая рекомендательная система сама должна уметь бороться с купленными отзывами и накрутками продавцов.

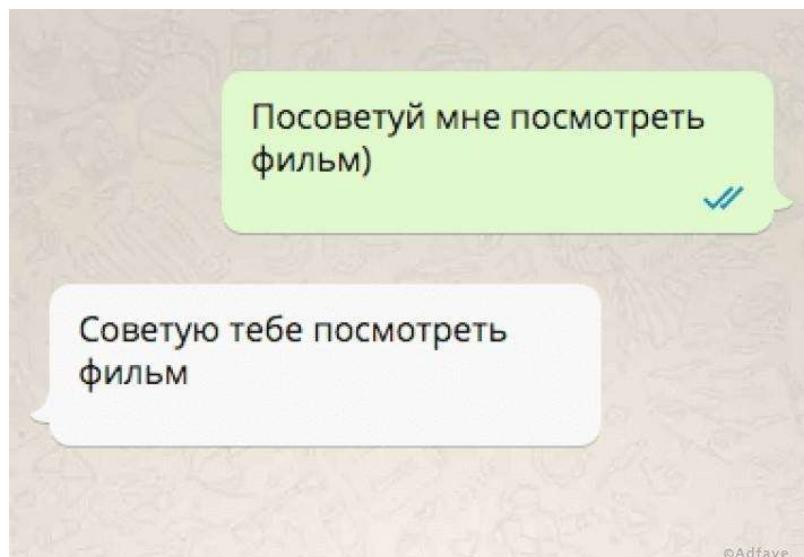
Манипуляции кстати бывают и непреднамеренными. Например, когда выходит новый блокбастер, первым делом на него идут фанаты, соответственно, первую пару месяцев рейтинг может быть сильно завышен.

## 7. Формат рекомендации.

Это может быть всплывающее окошко, появляющийся в определенном разделе сайта отсортированный список, лента внизу экрана или что-то еще.

## 8. Алгоритмы.

Несмотря на множество существующих алгоритмов, все они сводятся к нескольким базовым подходам, которые будут описаны далее. К наиболее классическим относятся алгоритмы Summary-based (неперсональные), Content-based (модели основанные на описании товара), Collaborative Filtering (коллаборативная фильтрация), Matrix Factorization (методы основанные на матричном разложении) и некоторые другие.



Источник

В центре любой рекомендательной системы находится так называемая матрица предпочтений. Это матрица, по одной из осей которой отложены все клиенты сервиса (Users), а по другой – объекты рекомендации (Items). На пересечении некоторых пар (user, item) данная матрица заполнена оценками (Ratings) – это известный нам показатель заинтересованности пользователя в данном товаре, выраженный по заданной шкале



Пользователи обычно оценивают лишь небольшую часть товаров, что есть в каталоге, и задача рекомендательной системы – обобщить эту информацию и предсказать отношение клиента к другим товарам, про которые ничего не известно. Другими словами нужно заполнить все незаполненные ячейки на картинке выше.

Шаблоны потребления у людей разные, и не обязательно должны рекомендоваться новые товары. Можно показывать повторные позиции, например, для пополнения запаса. По этому принципу выделяют две группы товаров.

- **Повторяемые.** Например, шампуни или бритвенные станки, которые нужны всегда.
- **Неповторяемые.** Например, книги или фильмы, которые редко приобретают повторно.

Если продукт нельзя явно отнести к одному из классов, имеет смысл определять допустимость повторных покупок индивидуально (кто-то ходит в магазин только ради арахисового масла определенной марки, а кому-то важно попробовать все, что есть в каталоге).

Понятие «интересности» тоже субъективное. Некоторым пользователям нужны вещи только из их любимой категории (conservative recommendations), а кто-то, наоборот, больше откликается на нестандартные товары или группы товаров (risky

новые серии любимого сериала, а может периодически засыпывать ему новые шоу или вообще новые жанры. В идеале стоит выбирать стратегию показа рекомендаций под каждого клиента отдельно, с помощью моделирования категории клиента.

Пользовательские оценки можно получить двумя способами:

- явно (explicit ratings) – пользователь сам ставит рейтинг товару, оставляет отзыв, лайкает страницу,
- неявно (implicit ratings) – пользователь явно свое отношение не выражает, но можно сделать косвенный вывод из его действий: купил товар – значит он ему нравится, долго читал описание – значит есть интерес и т.п.

Конечно, явные предпочтения лучше – пользователь сам говорит о том, что ему понравилось. Однако на практике далеко не все сайты предоставляют возможность явно выражать свой интерес, да и не все пользователи имеют желание это делать. Чаще всего используются сразу оба типа оценок и хорошо дополняют друг друга.

Также важно отличать термины Prediction (предсказание степени интереса) и собственно Recommendation (показ рекомендации). Что и как показывать – это отдельная задача, которая использует полученные на шаге Prediction оценки, но может быть реализована по-разному.

Иногда термин “рекомендация” употребляют в более широком смысле и имеют в виду любую оптимизацию, будь то выборка клиентов для рекламной рассылки, определение оптимальной цены предложения или просто выбор наилучшей стратегии коммуникаций с клиентом. В статье я ограничусь классическим определением данного термина, обозначающего выбор наиболее интересного товара для клиента.

## Неперсонализированные рекомендации

Начнем с неперсонализированных рекомендаций, поскольку они самые простые в реализации. В них потенциальный интерес пользователя определяется просто средним рейтингом товара: «Всем нравится – значит понравится и вам». По этому принципу работает большинство сервисов, когда пользователь не авторизуется в системе, например, тот же TripAdvisor.

Показываться рекомендации могут по-разному – как баннер сбоку от описания товара (Amazon), как результат запроса, отсортированный по определенному параметру (TripAdvisor), или как-то еще.

Рейтинг товара также может изображаться разными способами. Это могут быть звездочки рядом с товаром, количество лайков, разница положительных и отрицательных голосов (как обычно делают на форумах), доля высоких оценок или вообще гистограмма оценок. Гистограммы – наиболее информативный способ, но у них есть один минус – их сложно сравнивать между собой или сортировать, когда нужно вывести товары списком.



## Проблема холодного старта

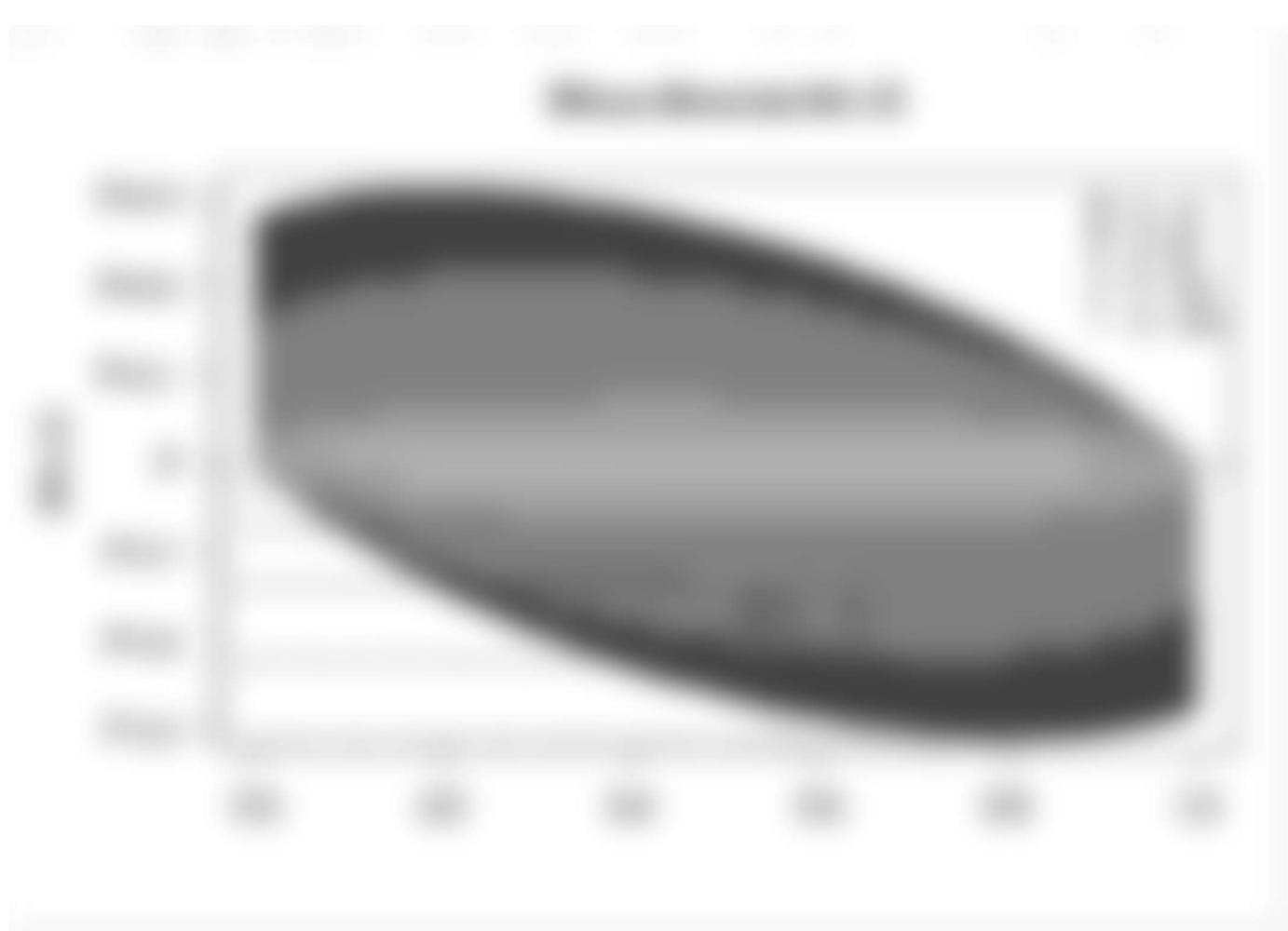
Холодный старт – это типичная ситуация, когда ещё не накоплено достаточно большое количество данных для корректной работы рекомендательной системы (например, когда товар новый или просто его очень редко покупают). Если средний рейтинг посчитан по оценкам всего трёх пользователей (`igor92`, `xuz_111` и `oleg_s`), такая оценка явно не будет достоверной, и пользователи это понимают. Часто в таких ситуациях рейтинги искусственно корректируют.

Первый способ – показывать не среднее значение, а сглаженное среднее (Damped Mean). Смысл таков: при малом количестве оценок отображаемый рейтинг больше тяготеет к некому безопасному «среднему» показателю, а как только набирается достаточное количество новых оценок, «усредняющая» корректировка перестает действовать.

Другой подход – рассчитывать по каждому рейтингу интервалы достоверности (confidence Intervals). Математически, чем больше оценок, тем меньше вариация среднего и, значит, большее уверенность в его корректности. А в качестве рейтинга можно выводить, например, нижнюю границу интервала (Low CI Bound). При этом понятно, что такая система будет достаточно консервативной, с тенденцией к занижению оценок по новым

товарам (если конечно это не хит)

Поскольку оценки ограничены определенной шкалой (например от 0 до 1), обычный способ расчета интервала достоверности здесь плохо применим: из-за хвостов распределения, уходящих за бесконечность и симметричности самого интервала. Есть альтернативный и более точный способ его посчитать — *Wilson Confidence Interval*. При этом получаются несимметричные интервалы примерно такого вида.



На картинке выше по горизонтали отложена оценка среднего значения рейтинга, по вертикали — разброс вокруг среднего значения. Цветом выделены различные размеры выборки (очевидно, чем выборка больше, тем меньше интервал достоверности).

Проблема холодного старта так же актуальна и для неперсонализированных рекомендаций. Общий подход здесь — заменять то, что в данный момент не может быть посчитано, различными эвристиками (например, заменять средним рейтингом, использовать алгоритм попроще, или вообще не использовать товар, пока не соберутся данные).

## Актуальность рекомендаций

В некоторых случаях также важно учитывать «свежесть» рекомендации. Это особенно

актуально для статей или постов на форумах. Свежие записи должны чаще попадать в

топ. Для этого используются корректирующие коэффициенты (damping factors). Ниже пара формул для расчета рейтинга статей на медиа сайтах.

Пример расчета рейтинга в журнале Hacker news:



где  $U$  = upvotes,  $D$  = downvotes, а  $P$  (Penalty) — дополнительная корректировка для имплементации иных бизнес-правил

Расчет рейтинга в Reddit:



где  $U$  = число голосов «за»,  $D$  = число голосов «против»,  $T$  = время записи. Первое слагаемое оценивает «качество записи», а второе делает поправку на время.

Очевидно, что универсальной формулы не существует, и каждый сервис изобретает ту формулу, которая лучше всего решает его задачу — проверяется это эмпирически.

## Content-based рекомендации

Персональные рекомендации предполагают максимальное использование информации о самом пользователе, в первую очередь о его предыдущих покупках. Одним из первых появился подход content-based filtering. В рамках данного подхода описание товара (content) сопоставляется с интересами пользователя, полученными из его предыдущих оценок. Чем больше товар этим интересам соответствует, тем выше оценивается потенциальная заинтересованность пользователя. Очевидное требование здесь — у всех товаров в каталоге должно быть описание.

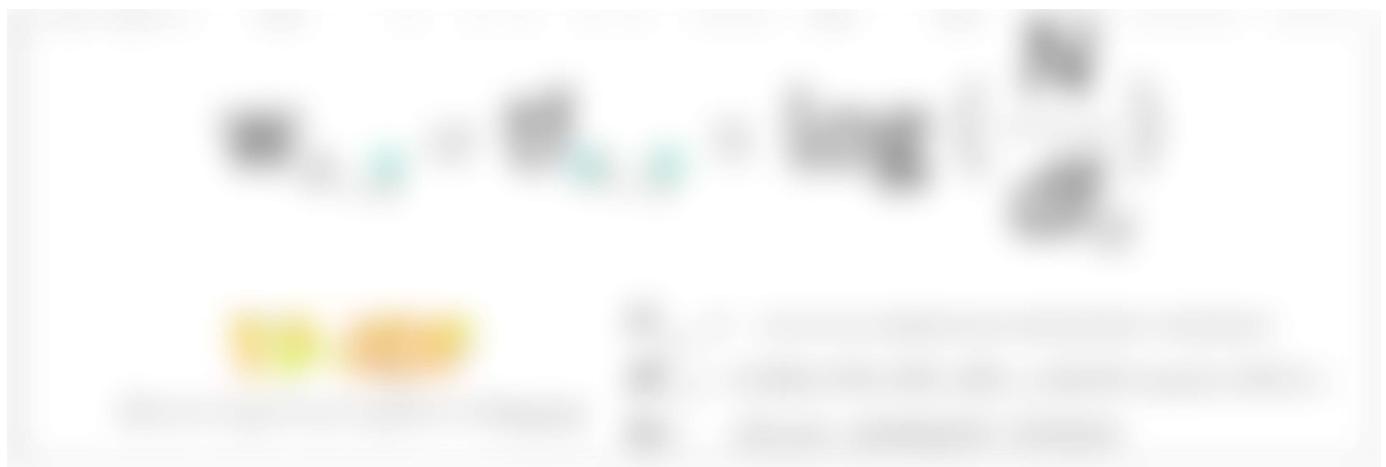
Исторически предметом Content-based рекомендаций чаще были товары с неструктурированным описанием: фильмы, книги, статьи. Такими признаками могут быть, например, текстовые описания, рецензии, состав актеров и прочее. Однако ничто не мешает использовать и обычные числовые или категориальные признаки.

Неструктурированные признаки описываются типичным для текста способом — векторами в пространстве слов (Vector-Space model). Каждый элемент такого вектора — признак, потенциально характеризующий интерес пользователя. Аналогично, продукт — вектор в том же пространстве.

По мере взаимодействия пользователя с системой (скажем, он покупает фильмы),

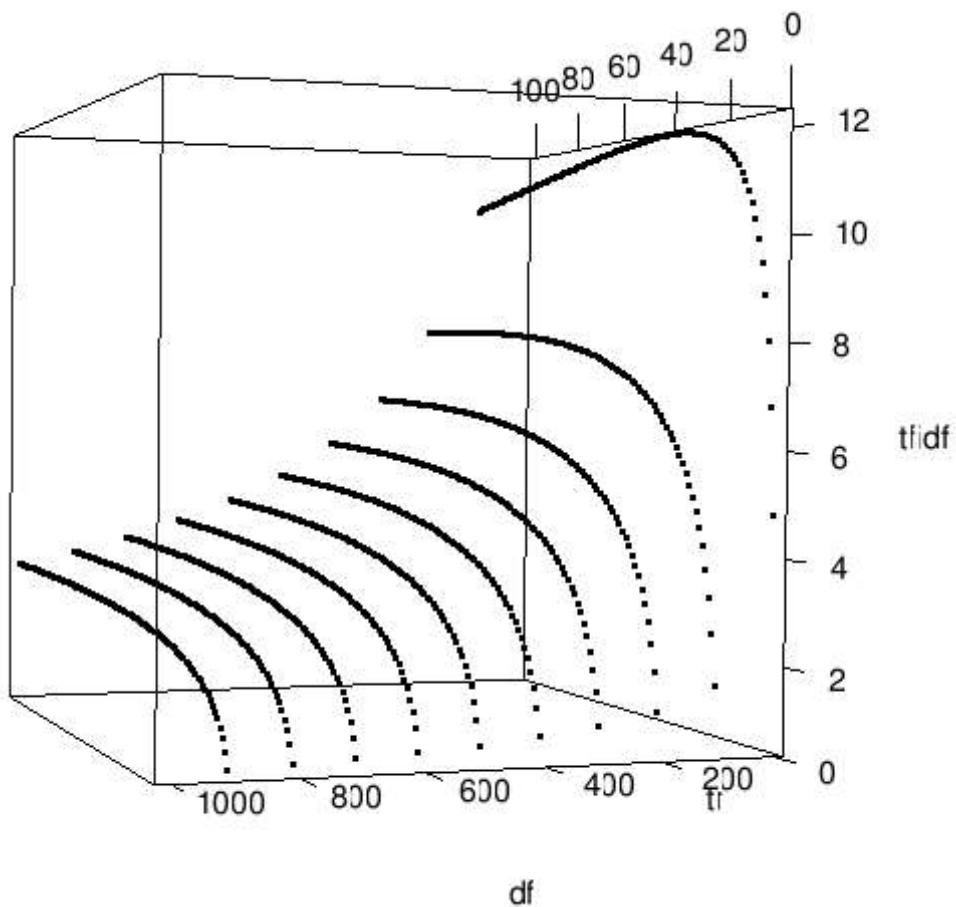
векторные описания приобретенных или товаров соединяются (суммируются и нормализуются) в единый вектор и, таким образом, формируется вектор его интересов. Далее достаточно найти товар, описание которого наиболее близко к вектору интересов, т.е. решить задачу поиска n ближайших соседей.

Не все элементы одинаково значимы: например, союзные слова, очевидно, не несут никакой полезной нагрузки. Поэтому при определении числа совпадающих элементов в двух векторах все измерения нужно предварительно взвешивать по их значимости. Данную задачу решает хорошо известное в Text Mining преобразование TF-IDF, которое назначает больший вес более редким интересам. Совпадение таких интересов имеет большее значение при определении близости двух векторов, чем совпадение популярных.



Принцип TF-IDF здесь в той же мере применим и к обычным номинальным атрибутам, таким, как например, жанр, режиссер, язык. TF — мера значимости атрибута для пользователя, IDF — мера «редкости» атрибута.

Существует целое семейство похожих преобразований (например, BM25 и аналогичные), но содержательно все они повторяют ту же логику, что TF-IDF: редкие атрибуты должны иметь больший вес при сравнении товаров. Картинка ниже иллюстрирует, как именно зависит вес TF-IDF от показателей TF и IDF. Ближняя горизонтальная ось — это DF: частота атрибута среди всех товаров, дальняя горизонтальная ось — TF: логарифм частоты атрибута у пользователя.



Некоторые моменты которые можно учесть при реализации.

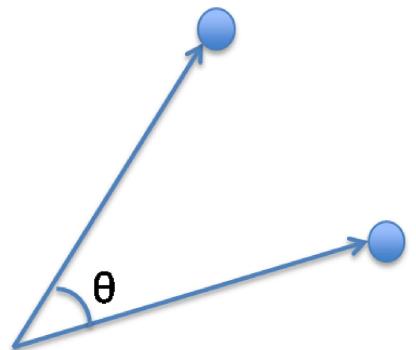
- При формировании vector-space представления товара вместо отдельных слов можно использовать шинглы или n-граммы (последовательные пары слов, тройки и т.д.). Это сделает модель более детализированной, однако потребуется больше данных для обучения.
- В разных местах описания товара вес ключевых слов может отличаться (например описание фильма может состоять из заголовка, краткого описания и детального описания).
- Описания товара от разных пользователей можно взвешивать по-разному. Например, можем давать больший вес активным пользователям, у которых много оценок.
- Аналогично можно взвешивать и по товару. Чем больше средний рейтинг объекта, тем больше его вес (аналог PageRank).
- Если описание товара допускает ссылки на внешние источники, то можно заморочиться и анализировать также всю связанную с товаром стороннюю информацию.

Видно, что content-based фильтрация почти полностью повторяет механизм query-document matching, используемый в поисковых системах типа Яндекс и Google. Отличие

пользователя, а там — ключевые слова запрашиваемого документа. Когда поисковики стали добавлять персонализацию, различие стерлось еще больше.

В качестве меры близости двух векторов чаще всего используется косинусное расстояние.

$$\text{sim}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

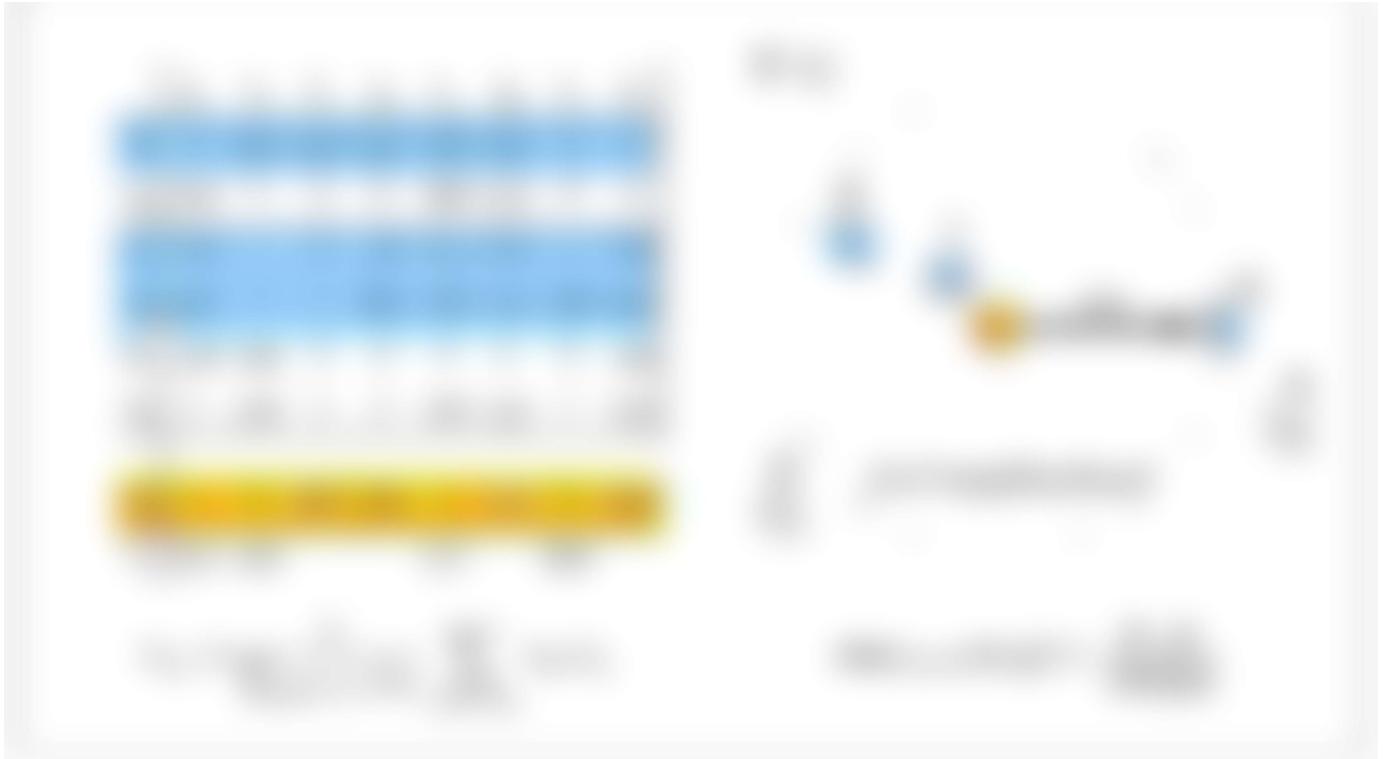


При добавлении новой оценки вектор интересов обновляется инкрементально (только по тем элементам, которые изменились). При пересчете имеет смысл давать новым оценкам чуть больше веса, поскольку предпочтения могут меняться.

## Коллаборативная фильтрация (User-based вариант)

Данный класс систем начал активно развиваться в 90-е годы. В рамках подхода рекомендации генерируются на основании интересов других похожих пользователей. Такие рекомендации являются результатом «коллaborации» множества пользователей. Отсюда и название метода.

Классическая реализация алгоритма основана на принципе k ближайших соседей. На пальцах — для каждого пользователя ищем k наиболее похожих на него (в терминах предпочтений) и дополняем информацию о пользователе известными данными по его соседям. Так, например, если известно, что ваши соседи по интересам в восторге от фильма «Кровь и бетон», а вы его по какой-то причине еще не смотрели, это отличный повод предложить вам данный фильм для субботнего просмотра.



На картинке выше проиллюстрирован принцип работы метода. В матрице предпочтений желтым цветом выделен пользователь, для которого мы хотим определить оценки по новым товарам (знаки вопроса). Синим цветом выделены три его ближайших соседа.

«Похожесть» – в данном случае синоним «корреляции» интересов и может считаться множеством способов (помимо корреляции Пирсона, есть еще косинусное расстояние, есть расстояние Жаккара, расстояние Хэмминга и пр.).

У классической реализации алгоритма есть один явный минус – он плохо применим на практике из-за квадратичной сложности. Действительно, как любой метод ближайшего соседа, он требует расчета всех попарных расстояний между пользователями (а пользователей могут быть миллионы). Нетрудно посчитать, что сложность расчета матрицы расстояний будет  $O(n^2m)$ , где  $n$  — число пользователей, а  $m$  — число товаров. При миллионе пользователей для хранения матрицы расстояний в сыром виде, потребуется минимум 4ТБ.

Данная проблема отчасти может быть решена покупкой высокопроизводительного железа. Но если подходить с умом, то лучше ввести корректировки в алгоритм:

- обновлять расстояния не при каждой покупке, а батчами (например, раз в день),
- не пересчитывать матрицу расстояний полностью, а обновлять ее инкрементально,
- сделать выбор в пользу итеративных и приближенных алгоритмов (например ALS).

Для того чтобы алгоритм был эффективен, важно чтобы выполнялось несколько допущений.

- Вкусы людей не меняются временем (или меняются, но для всех одинаково).
- Если вкусы людей совпадают, то они совпадают во всем.

Например, если два клиента предпочитают одни фильмы, то книги им тоже нравятся одинаковые. Так часто бывает, когда рекомендуемые товары однородны (например, только фильмы). Если это же не так, то у пары клиентов вполне могут совпадать предпочтения в еде, а политические взгляды быть прямо противоположными — здесь алгоритм будет менее эффективным.

Окрестность пользователя в пространстве предпочтений (его соседи), которую мы будем анализировать для генерации новых рекомендаций, можно выбирать по-разному. Мы можем работать вообще со всеми пользователями системы, можем задать некий порог близости, можем выбрать несколько соседей случайным образом или брать n наиболее похожих соседей (это наиболее популярный подход).

Авторы из MovieLens в качестве оптимального количества соседей приводят цифры в 30-50 соседей для фильмов и 25-100 для произвольных рекомендаций. Здесь понятно, что если возьмем слишком много соседей, то получим больше вероятность случайного шума. И наоборот, если возьмем слишком мало, то получим более точные рекомендации, но меньшее количество товаров можно рекомендовать.

Важный этап подготовки данных — нормализация оценок.

## Стандартизация данных (scaling)

Поскольку все пользователи оценивают по-разному — кто-то всем подряд пятерки ставит, а от кого-то четверки редко дождешься — перед расчетом данные лучше нормализовать, т.е. привести к единой шкале, чтобы алгоритм мог корректно сравнивать их между собой.

Естественно, предсказанную оценку затем нужно будет перевести в исходную шкалу обратным преобразованием (и, если нужно, округлить до ближайшего целого числа).

Нормализовать можно несколькими способами:

- центрированием (mean-centering) — из оценок пользователя просто вычитаем его среднюю оценку,

\* актуально только для небинарных матриц

- стандартизацией (z-score) — в добавок к центрированию делим оценку ее на стандартное отклонение у пользователя,

\* здесь после обратного преобразования рейтинг может выйти за пределы шкалы (т.е. например, 6 по пятибалльной шкале), но такие ситуации довольно редки и решаются просто округлением в сторону ближайшей допустимой оценки.

- двойной стандартизацией — первый раз нормируем оценками пользователя, второй раз — оценками товара.

Если у фильма «Самый лучший фильм» средняя оценка 2.5, а пользователь ей ставит 5, то это сильный фактор, говорящий о том, что такие фильмы ему явно по вкусу.

«Похожесть» или корреляцию предпочтений двух пользователей можно считать разными способами. По сути нам надо просто сравнить два вектора. Перечислим несколько наиболее популярных.

1. Корреляция Пирсона — классический коэффициент, который вполне применим и при сравнении векторов.

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Основной его минус — когда пересечение по оценкам низкое, корреляция может быть высокой просто случайно.

Для борьбы со случайно завышенной корреляцией можно домножить на коэффициент  $50 / \min(50, \text{Rating intersection})$  или любой другой damping factor, влияние которого уменьшается с ростом числа оценок.

2. Корреляция Спирмана

Основное отличие — коэффициент ранговый, т.е. работает не с абсолютными значениями рейтингов, а с их порядковыми номерами. В целом дает результат очень близкий к корреляции Пирсона.

$$\rho = 1 - \frac{\sum_{i=1}^n u_i}{n(n^2 - 1)}.$$

### 3. Косинусное расстояние

Еще один классический коэффициент. Если приглядеться, косинус угла между стандартизованными векторами — это и есть корреляция Пирсона, одна и та же формула.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

Почему косинусное — потому что, если два вектора сонаправлены (т.е. угол между ними нулевой), то косинус угла между ними равен единице. И наоборот, косинус угла между перпендикулярными векторами равен нулю.

Интересное развитие коллаборативного подхода — так называемые Trust-based recommendations, в которых учитывается не только близость людей по интересам, но также их «социальная» близость и степень доверия между ними. Если например видим, что на фейсбуке девушка периодически заходит на страницу с аудиозаписями подруги, значит доверяет её музыкальному вкусу. Следовательно в рекомендации девушке можно вполне подмешивать новые песни из плейлиста подруги.



## Обоснования рекомендаций

Важно, чтобы пользователь доверял рекомендательной системе, а для этого она должна быть проста и понятна. При необходимости всегда должно быть доступно понятное объяснение рекомендации (в англ. терминологии explanation).

В рамках объяснения неплохо показывать оценку товара соседями, по какому именно атрибуту (например, актер или режиссер) было совпадение, а также выводить уверенность системы в оценке (confidence). Чтобы не перегружать интерфейс, можно всю эту информацию вынести в кнопку «Tell me more».

Например:

- «Вам может понравиться фильм... поскольку там играет... и ...».
- «Пользователи с похожими на ваш музыкальными вкусами оценили альбом... на 4.5 из 5».

## Резюме

На этом закончу первую часть статьи. Мы рассмотрели общую постановку задачи, поговорили про неперсональные рекомендации, описали два классических подхода (content-based и коллаборативную фильтрацию), а также затронули тему обоснования рекомендаций. В целом двух этих подходов вполне достаточно для построения production-ready рекомендательной системы. В следующей части я продолжу обзор и расскажу о более современных методах, в том числе действующих нейросети и глубокое обучение, а также про гибридные модели.

► **А пока посмотрите наши вакансии.**

**Теги:** ланит, cleverdata, алгоритмы, машинное обучение, data scientist, рекомендательные системы, алгоритмы рекомендаций

**Хабы:** Блог компании ГК ЛАНИТ, Data Mining, Алгоритмы, Big Data, Машинное обучение



ГК ЛАНИТ

Ведущая многопрофильная группа ИТ-компаний в РФ

Сайт

43 0

Карма Рейтинг

Константин Коточигов @crazyhatter

Data Scientist

Комментарии 15

---

## ИНФОРМАЦИЯ

---

Дата основания 16 октября 1989

Местоположение Россия

Сайт lanit.ru

Численность свыше 10 000 человек

Представитель

Ася

**Ваш аккаунт**[Войти](#)[Регистрация](#)**Разделы**[Публикации](#)[Новости](#)[Хабы](#)[Компании](#)[Авторы](#)[Песочница](#)**Информация**[Устройство сайта](#)[Для авторов](#)[Для компаний](#)[Документы](#)[Соглашение](#)[Конфиденциальность](#)**Услуги**[Реклама](#)[Тарифы](#)[Контент](#)[Семинары](#)[Мегапроекты](#)[Настройка языка](#)[О сайте](#)[Техническая поддержка](#)[Вернуться на старую версию](#)