Введение в рекомендательные системы

Где можно встретить рекомендательные системы

Много где: ютуб, спотифай, яндекс маркет и т.д.

Поиск vs Рекомендации

Задачи похожие, но если в поиске есть формализованный запрос, то в рекомендациях его нет.

Формализация задачи

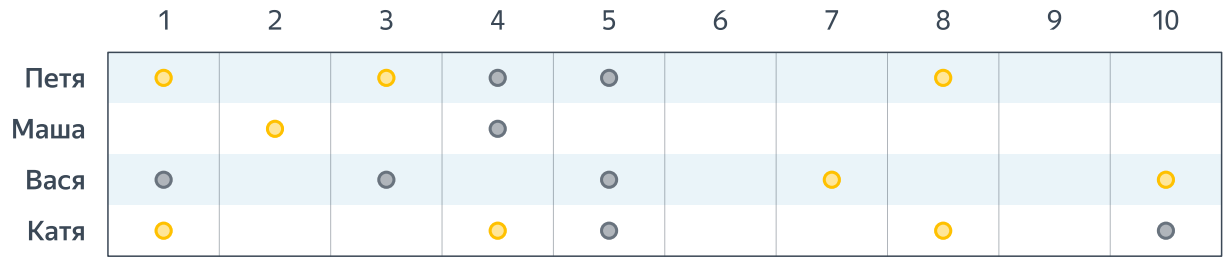
Explicit и Implicit feedback

Формально нам нужно для каждого пользователя `u` из U необходимо оценить значение r_{ui} (его оценку для товара i из I) и выбрать несколько товаров с набольшей оценкой. Причём, оценка, фидбек, может быть как явной (explicit) (ex. оценка фильма, лайк/дизлайк, отзыв, как правило явного фидбека немного), так и неявной (implicit) (ex. продолжительность просмотра, кликабельность и т.д.).

Ранжирующая модель

Заметим, что для нам нестолько важно очень точно посчитать r_{ui} , сколько правильно определить топ n товаров по рейтингу, то есть, фактически решить задачу ранжирования - то есть высчитываем все r_{ui}, а затем сортируем их.

Коллаборативная фильтрация



Суть в том, что мы будем рекомендовать юзеру те объекты, которые нравятся похожим юзерам. Или наоборот, отталкиваться от товаров.

User2User рекомендации

Введём меру схожести 2 юзеров $s(u, v)$, чем она больше, тем больше они похожи. Выберем какого-то юзера, к нему в соотвествие возьмём группу юзеров $N(u)$, с которыми показатель схожести больше какого-то порога(гиперпарметра). Тогда можем посчитать оценку, как средневзвешенное:

$$\hat{r}_{ui} = \frac{\sum_{v \in N(u)} s(u, v) r_{vi}}{\sum_{v \in N(u)} |s(u, v)|}$$

Модуль в знаменателе на случай отрицательной схожести. Далее можно учитывать, что числовые оценки r для пользователей u неодинаковы, то есть "хорошо" для каждого значит что-то своё, тогда можно учитывать ещё и статистику каждого юзера, его среднее и дисперсию. Можно брать и другие формулы схожести, в зависимости от задачи.

Item2Item рекомендации

Тут также можно ввести меру схожести между 2 товарами $s(i, j)$. Выберем какой-то товар, поставим ему в соотвествие группу похожих товаров $N(i)$, тогда

$$\hat{r}_{ui} = \frac{\sum_{j \in N(i)} s(i, j) r_{uj}}{\sum_{j \in N(i)} |s(i, j)|}$$

Тут также можно использовать различные модификации для этой формулы.

Особенности коллаборативной фильтрации

- Для работы нужна только матрица оценок
- Не работает для новых товаров и юзеров, так как для них нет истории

- Так как основывается только на истории взаимодействий, то она не может предложить что-то новое, вгоняет в "инфорационный пузырь"

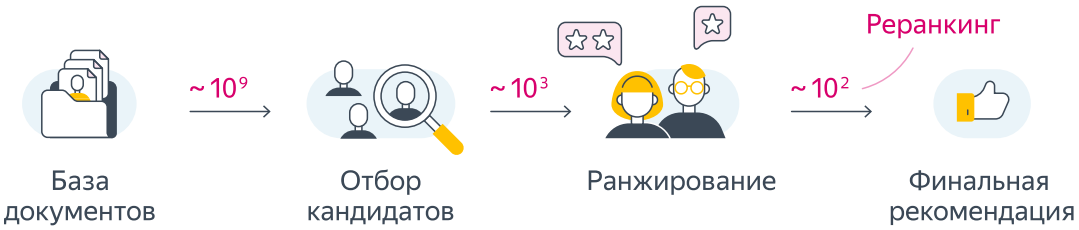
Content-based рекомендации

Суть в том, что мы измеряем схожесть объектов по их содержанию (ex. статьи о том, как поменять колесо на велосипеде). То есть на вход модели подаются фичи товара (информация о нём), на выходе получаем эмбеддинг (числовое представление объекта). Такие модели не используют коллаборативную информацию, так как не знают ничего о других объектах и их взаимодействии (ex. Bert - чисто контентная модель, которая переводит текст в эмбеддинг). Итак, пусть у нас есть контекстные эмбединги для каждого товара $e_i \in \mathbb{R}^n$ (ex. мы применили Bert для получения эмбединга статей), тогда можем определить схожесть через косинусное расстояние

$$\hat{r}_{ui} = \max_{j \in I_u, r_{uj} > \alpha} \rho(e_i, e_j) r_{uj}$$

где ρ – скалярное произведение или косинусное расстояние между двумя векторами, Iu – множество оценённых пользователем объектов, а α – гиперпараметр. Таким образом, высокие рейтинги получат объекты, похожие на те, что понравились пользователю – мы получили простую ранжирующую модель.
Контентный метод хорошо работает как со старыми, так и с новыми объектами, так как не использует историю взаимодействия, но ещё больше вгоняет юзера в "информационный пузырь".
Коллаборативный и контентный методы можно комбинировать (ex. DSSM).

Классический пайплайн рекомендательной системы



Хорошая рекомендательная система должна обладать свойствами:

- при ранжировании товаров в порядке убывания оценки нам бы хотелось учитывать как можно больше фичей как юзера, так и товара
- рекомендательная система должна быть быстрой
- должен быть механизм, позволяющий постоянно учитывать бизнес логику

Для первого свойства, при ранжировании, обычно применяют бустинг из-за скорости работы.

Не стоит попадать в feedback loop - ситуацию, когда при тестировании новой модели мы высчитываем метрики, которые зависят от результата работы старой модели, то есть новая модель будет учиться предсказывать старую. Для того, чтобы это не случалось, можно подмешивать случайные объекты и давать им большой вес в функции потерь, то есть давать на вход некоторое "свежее" подмножество объектов, и выводить юзера из "информационного пузыря".

Теперь про второе свойство, обычно в рекомендательной системе крутятся сотни тысяч, а зачастую и миллионы товаров и юзеров, фичи пользователей и объектов постоянно меняются, поэтому при каждом запросе юзера мы должны заново скорить все объекты, но это очень долго и дорого, поэтому, в действительности, нас интересует только небольшая подвыборка товаров, которые могут заинтересовать юзера, то есть сперва мы определим потенциально интересные товары, а потом из них тяжёлой моделью определим финальную выдачу. Но процесс выбора потенциально интересных товаров (отбор кандидатов) также должен быть быстрым, давать достаточное количество наиболее подходящих товаров.

Подходы к отбору кандидатов:

- Эвристики: самые популярные товары, недавно опубликованные и т.д.
- Коллаборативные методы: item2item и user2user рекомендации. Пока юзер в оффлайне мы можем строить все необходимые таблички.
- Контентные методы: берём content-based эмбединги объектов и строим быстрый индекс для поиска ближайших объектов (ex. HNSW) (см. главу про метрические методы). \

Обычно кандидаты набираются из разных источников, где каждый источник закрывает какой-то пользовательский аспект.

Теперь третье свойство, оно хорошо достигается при использовании двухступенчатой рекомендательной системы. Под бизнес-логикой понимается какое-то качество рекомендательной системы, которые слишком нетривиально, чтобы зашивать его в ранжирующую модель. (ex. реже показывать старые видео, реже показывать плохие по качеству видео, обеспечение разнообразия рекомендаций). То есть, подразумевается некоторый реранкинг после применения ранжирующей формулы.