

Введение в RecSys

02.10.2024

Где можно встретить recsys

С рекомендательными системами можно столкнуться там, где есть большое множество **товаров** и **пользователей**, которые хотят найти нужные для себя товары. Рекомендательные системы помогают отобрать **наиболее релевантные** для пользователя **объекты**, тем самым экономя его время.

Что такое «релевантные для пользователя товары» – это нетривиальный вопрос, который решается отдельно для каждой задачи исходя из бизнес-логики.

Приведите несколько примеров рекомендательных систем

Поиск vs рекомендации

Хоть и задачи поиска и рекомендаций кажутся похожими, у них есть одно важное отличие:

В задаче поиска есть **сформулированный запрос** от пользователя, а в задаче рекомендаций **явного запроса нет**, есть только история взаимодействий пользователя с объектами и наша надежда на то, что мы верно распознали его скрытые желания.

Это различие объясняет некоторые особенности дизайна рекомендательных систем.

Задача recsys

Необходимо научиться среди **непоказанных** пользователю товаров **находить** те, которые **заинтересовали** бы его больше всего.

Фидбек от пользователей

Примеры фидбека:

- Для товара – факт добавления в корзину;
- Для музыки – дослушали ли трек до конца;
- Для статьи – лайк/дизлайк;
- Для видео – время его просмотра или факт просмотра, например, наполовину.

Явный фидбек

Явный (Explicit) фидбек – это такие действия пользователя, по которым **точно** можно понять, **понравился** ли ему объект.

Это может быть оценка, поставленная, фильму, лайк/дизлайк к видео или рецензия на купленный товар.

Такого фидбека **очень мало**, но он наиболее **точно характеризует** отношение пользователя к товару.

Неявный фидбек

Неявный (Implicit) фидбек – это любая **другая информация** о действиях пользователя на сайте. Он выступает в качестве прокси к явному фидбеку.

Например, факт того, что пользователь досмотрел видео до конца, не говорит о том, понравилось ли оно ему, однако можно сделать предположение, что большинству посмотревших видео до конца оно понравилось.

Примеры неявного фидбека

Клик на статью, время просмотра видео, покупка товара.

Обычно такого фидбека **в разы больше**, чем явного, однако он более **шумный**, и не стоит доверять ему так же, как явному.

Например, при оптимизации кликов на статью может получиться так, что рекомендательная система научится находить **кликбейт**, а не **интересные** пользователю статьи – это может плохо отразиться на сервисе в долгосрочной перспективе.

Ранжирующая модель

Задачу построения рекомендательной системы можно сформулировать в качестве задачи **классификации** (клик/не клик) или **регрессию** (сколько звёзд пользователь поставит объекту), но это не самые распространённые стратегии.

Обратим внимание, что нам на самом деле **не обязательно** уметь **точно** оценивать рейтинги. Достаточно уметь для пользователя и набора объектов генерировать перестановку этих объектов **в порядке убывания** рейтинга.

Модель, решающую данную задачу, называют **ранжирующей**.

Классический пайплайн ранжирующей модели

На **вход** подаются признаки **пользователя** и **объекта**, и для пары пользователь-объект на основе этих признаков выдается некоторое **число**, ответ модели.

Далее мы **сортируем** объекты в порядке его **убывания**. Из полученной перестановки обычно берут несколько первых объектов для показа пользователю.

Коллаборативная фильтрация

	1	2	3	4	5	6	7	8	9	10
Петя	●		●	●	●			●		
Маша		●		●						
Вася	●		●		●		●			●
Катя	●			●	●			●		●

Рассмотрим матрицу взаимодействий пользователя, приведенную выше. Что можно порекомендовать **Кате**, исходя из исторических данных?

Коллаборативная фильтрация

	1	2	3	4	5	6	7	8	9	10
Петя	●		●	●	●			●		
Маша		●		●						
Вася	●		●		●		●			●
Катя	●			●	●			●		●

Транспонированная задача: для лайкнутого пользователем объекта искать похожие, то есть те, которые пользователи достаточно часто лайкали вместе с ним.

Например, объекты 1 и 8 похожи друг на друга, так как их лайкали одни и те же пользователи, и точно так же похожи 1 и 3.

Предыдущая задача называется **user2user**, транспонированная **item2item**.

Особенности коллаборативной фильтрации

Методы **не опираются** ни на какую **дополнительную** информацию кроме **матрицы оценок**, предполагая, что этого должно быть достаточно для улавливания качественного сигнала о схожести пользователей и товаров;

Предложенные методы **не применимы** для новых объектов и пользователей – для них просто нет истории или она недостаточно информативна для того, чтобы методы могли давать более-менее точные оценки;

Так как методы основаны только на **истории прошлых взаимодействий**, рекомендательная система, построенная исключительно на их основе будет постепенно **вгонять** пользователя в **информационный пузырь**: эти методы не предполагают открытия новых интересов у пользователя, они способны только эксплуатировать уже имеющиеся.

Content-based рекомендации

Методы основаны на измерении схожести между объектами на основе их содержания.

Входом для content-based модели являются **разные контентные признаки и характеристики** товара (например, текст статьи, время публикации, картинки), а выходом является некоторое **числовое представление** объекта (эмбеддинг).

Важно, что никакую коллаборативную информацию такие модели не используют, они **ничего не знают про других пользователей** и про их взаимодействие с объектами.

Например, Bert является чисто контентной моделью – он переводит текст в эмбеддинг.

Особенности content-based рекомендаций

Плюс контентного подхода в том, что, в отличие от чисто коллаборативного подхода, он одинаково **хорошо работает** на **новых** и **старых** объектах, так как контентные модели основаны только на статичной контентной информации, которая всегда доступна.

Из минусов можно отметить, что **схожесть по контенту** может ещё больше загонять пользователя в информационный пузырь.

Например, контентная модель вряд ли сможет к кофемашине порекомендовать кофейные зерна, в то время как коллаборативный подход получит сигнал о том, что товары являются дополняющими напрямую из действий других пользователей.

Классический пайплайн рекомендательной системы



Отбор кандидатов

Пользователю наверняка **интересна** лишь **небольшая часть** имеющихся у нас товаров. Можно попытаться **сузить** множество до потенциально интересных пользователю объектов и уже для них **применить** «тяжёлую» **ранжирующую** модель, которая определит **финальную выдачу**.

К отбору кандидатов предъявляют два требования:

- он должен быть **быстрым**;
- в полученной после отбора кандидатов подмножестве должны в **избытке** находиться **интересные** пользователю статьи/книги/продукты;

Подходы к отбору кандидатов

Эвристики: самые популярные товары, популярные за X последних дней, популярные среди жителей этого города, недавно опубликованные;

Коллаборативные: item2item или user2user рекомендации. Также есть более сложные подходы на основе матричных разложений (SVD).

Контентные методы: берём content-based эмбединги объектов и строим быстрый индекс для поиска ближайших объектов (ex. **HNSW**). Далее, можем взять понравившиеся пользователю товары и найти похожие на них.

Реранкинг

Так же необходимо учесть механизм, который позволит **учитывать бизнес-логику**, т. е. некоторое **качество** рекомендательной системы, которое хотелось бы иметь, но которое достаточно **нетривиально**, чтобы мы не стали **зашивать** его в саму **ранжирующую** модель.

Примеры возможных пожеланий:

- Реже показывать старые видео в ленте;
- Реже показывать слишком длинные видео или видео, снятые в плохом качестве;
- Обеспечить разнообразную для пользователя выдачу.

Все эти свойства **подразумевают** под собой небольшое **переупорядочивание** объектов после применения **ранжирующей** формулы.

Как применить в виртуальных
ассистентах?

Статус проектных задач?