Титульный лист материалов по дисциплине

(заполняется по каждому виду учебного материала)

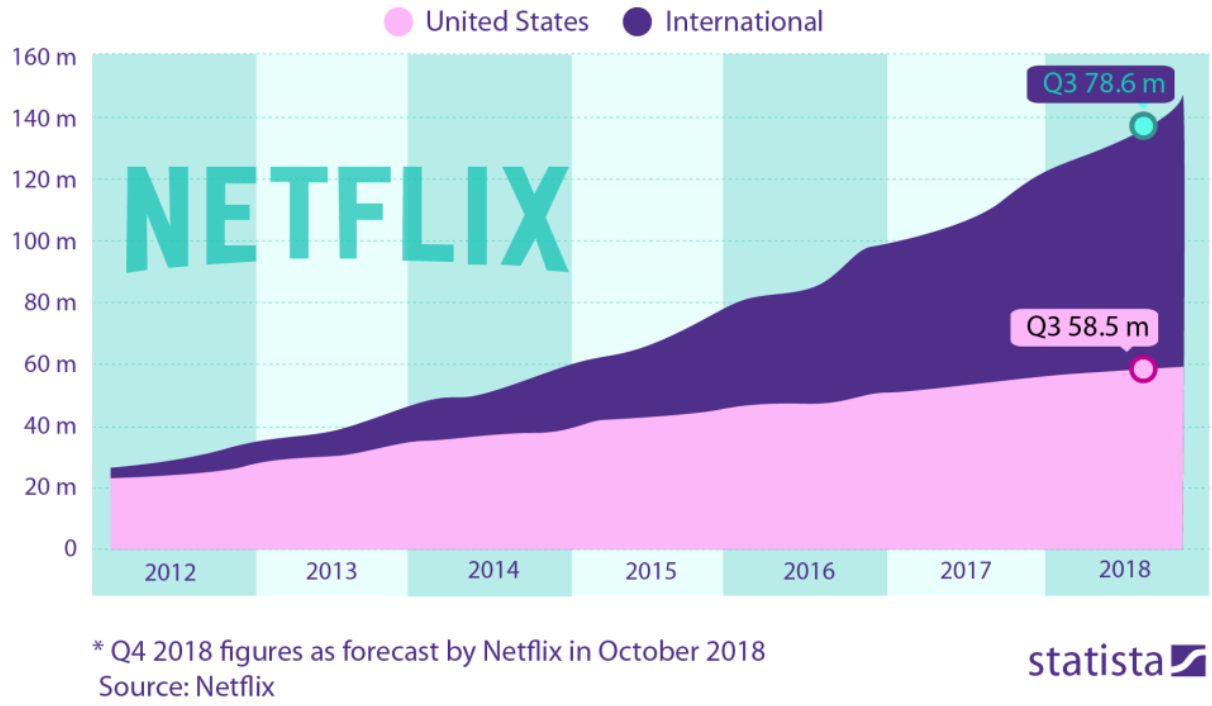
ДИСЦИПЛИНА	**Технологии извлечения знаний из больших данных** (полное наименование дисциплины без сокращений)
ИНСТИТУТ	ИКБ
КАФЕДРА	**Кафедра КБ-14 «Цифровые технологии обработки данных»** полное наименование кафедры)
ВИД УЧЕБНОГО МАТЕРИАЛА	**Лекция** (в соответствии с пп.1-11)
ПРЕПОДАВАТЕЛЬ	**Никонов В.В.** (фамилия, имя, отчество)
СЕМЕСТР	**3 семестр 2023/2024 уч. года** (указать семестр обучения, учебный год)

Рекомендательные системы

Постановка задачи и особенности оценки качества

Где применяются рекомендательные системы

Задача построения рекомендаций широко встречается в онлайн-торговле при покупке товаров, в лентах новостей при подборе контента, на видеохостингах, новостных сайтах, при рассылке рекламы. Когда вы заходите на современный маркетплейс, например Amazon, «Яндекс.Маркет» или «СберМегаМаркет», очень часто вас встречает раздел «Вас могут заинтересовать эти товары». Как раз за подборку таких товаров и за их порядок отвечают алгоритмы построения рекомендаций. В этих областях хорошие рекомендации помогают продавать больше. Например, Amazon после внедрения рекомендательной системы получил дополнительный доход $2,93 млрд. В компании Netflix 80 % просматриваемых фильмов и сериалов — это фильмы и сериалы по рекомендациям, что позволило значительно увеличить количество пользователей платформы.

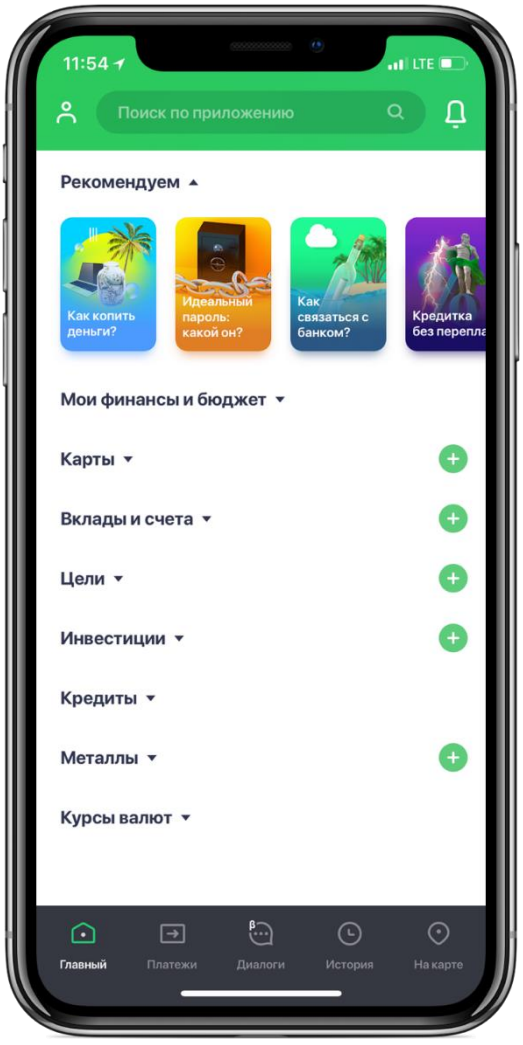


Рост аудитории компании Netflix. Источник: statista.com

Похожие системы есть в онлайн-кинотеатрах, на платформах прослушивания музыки, в агрегаторах текстового контента, таких как «Хабр», «Яндекс.Дзен» или Medium. Люди хотят получать полезный им контент, и

задача платформ — рекомендовать актуальные, интересные и разнообразные видео, аудио или тексты, иначе пользователи предпочтут купить подписку другого сервиса.

Также существует много других областей, таких как подбор возможных друзей в социальных сетях, выдача интересных объявлений, рекомендации других продуктов экосистемы. Например, в мобильном приложении «Сбербанка» продвигаются другие продукты банка.



Рекомендации продуктов экосистемы «Сбербанка» в историях банковского приложения

Задача построения рекомендаций

Задачу построения рекомендаций легко представить следующим образом: мы хотим предоставить **пользователю** из большого списка небольшое подмножество **предметов** (5–50 наиболее подходящих объектов),

которые будут интересны ему, причем чем выше **предмет** в списке, тем он должен лучше подходить **пользователю**.

Так у нас появляются две основные сущности:

- Пользователи (users)

Активные агенты, совершающие взаимодействие с предметами. Обычно это реальные люди, которые покупают товары, смотрят видео, читают новостные статьи и т. д.

- Предметы (товары, **items**)

Пассивные объекты, с которыми пользователи взаимодействуют. Например, товары на маркетплейсах, видео на YouTube, музыка в стриминговых сервисах, услуги и т. д. Все это мы будем в дальнейшем называть «предметами».

Обычно количество пользователей измеряется тысячами или десятками тысяч, и для каждого пользователя мы хотим строить рекомендации, причем в некоторых ситуациях мы хотим, чтобы рекомендации для одного и того же пользователя менялись со временем.

Количество предметов обычно значительно больше, чем количество пользователей, и может измеряться миллионами или даже миллиардами. Основное отличие предметов от пользователей в том, что не обязательно каждый предмет должен быть рекомендован какому-либо пользователю.

Таким образом, в общем смысле под **рекомендацией** мы подразумеваем список из K объектов, которые мы показываем или рекомендуем пользователю. Причем для нас важен порядок предметов в этом списке, так как пользователь, скорее всего, обратит больше внимания на первые предметы в списке и с большей вероятностью совершит действие, которое мы от него ждем: послушает песню, совершит покупку и т. д.

Для построения рекомендаций часто используется история **взаимодействия** пользователей с предметами. Обратную связь от пользователя по взаимодействию можно получить в двух форматах:

1. Когда у нас есть **явный** (explicit) фидбек от пользователя: рейтинг, оценка, лайк/дизлайк

2. Когда наш фидбек **неявный** (implicit): тот, который нам пользователь напрямую не сообщал, но мы сами по его действиям сделали выводы. Неявным фидбеком может быть количество прослушиваний песни, сколько раз пользователь покупал товар, на какой секунде пользователь переключил видео на другое и др.

Базовые подходы к построению рекомендаций

Рекомендации могут быть **персонализированными** и **неперсонализированными**. Если мы делаем неперсонализированне рекомендации, то мы будем рекомендовать одни и те же товары для всех пользователей. Так, если мы рассмотрим сервис рекомендаций музыки, мы можем рекомендовать всем пользователям треки, которые популярны на платформе. Этот подход алгоритмически проще в реализации, и его часто используют как базовое решение. К сожалению, в реальном бизнесе не всегда бывает так, что персонализированные рекомендации работают лучше неперсонализированных. Поэтому, прежде чем строить сложную модель, лучше построить базовую, чтобы потом можно было сравнивать с ней более сложную модель.

В персонализированных рекомендациях мы будем менять рекомендации в зависимости от характеристик и поведения пользователя, учитывая его индивидуальность и интересы. Причем рекомендации могут меняться для определенных групп пользователей или быть индивидуальными для каждого пользователя.

Существуют два базовых подхода к персонализации рекомендаций: **content-based** и **коллаборативная фильтрация**. Идея подхода **content-based** в том, что мы принимаем решение о рекомендации в зависимости от характеристик товаров и пользователей без учета истории взаимодействий товаров и пользователей. Так, например, мы можем спрашивать у пользователя, какие жанры ему нравятся, смотреть на жанры музыкальных

треков и осуществлять рекомендации. Этот подход очень хорош для новых пользователей и товаров, когда у нас нет истории взаимодействий.

При использовании **коллаборативной фильтрации** мы не смотрим на характеристики пользователей и предметов, с которыми они взаимодействуют, а опираемся на историю взаимодействий пользователей и предметов.

Конечно, существует большое количество смешанных моделей, которые черпают идеи из обоих подходов. В реальных промышленных рекомендациях используются сложные модели, которые опираются как на взаимодействие пользователей с предметами, так и на характеристики предметов и пользователей.

Метрики качества рекомендаций

Как и в любой области машинного обучения, в задаче рекомендаций используются специфические метрики качества: **hitrate, MRR, precision@k, recall@k** (последние две метрики — аналоги precision и recall). В основе этих метрик лежит идея сравнения списка рекомендуемых нами предметов с предметами из **отложенной выборки**, с которыми пользователь взаимодействовал, но система не знала об этих взаимодействиях при обучении. Предметы из отложенной выборки, которые мы рекомендовали и с которыми пользователь реально взаимодействовал, мы будем называть **релевантными рекомендациями**. Количество рекомендованных нами объектов мы будем обозначать *K*.

Самая базовая идея оценки качества: давайте считать, для скольких пользователей в среднем мы рекомендовали релевантные товары. Эта метрика называется hit rate (HR):

$$HR = \frac{\sum_{i=1}^{N} relevance_i}{N}, relevance_{ij} = \{1, \text{в рекомендации для пользователя } i \text{ был релевантный объект } 0, \text{иначе}\}$$

где N — общее количество пользователей; K — количество предметов, рекомендуемых пользователю; j — рекомендуемый предмет.

Однако HR не учитывает позицию нашего объекта в рекомендованном листе, а пользователь, скорее, обратит внимание на первые рекомендованные объекты, чем на последние. Поэтому можно доработать эту метрику и учитывать не просто наличие релевантной рекомендации, а обратный ранг *(rank_i)* первого релевантного рекомендованного объекта:

$$MRR = \frac{\sum_{i=1}^{N} RR_i}{N} \quad RR_i = \frac{relevance_i}{rank_i},$$

где $rank_i$ — номер первого релевантного товара в рекомендации для пользователя $_i$.

Метрики **precision@k** и **recall@k** похожи на своих старших сестер — метрики **precision** и **recall**, которые мы рассматривали в разделе про классификацию. **Precision@k** и **recall@k** рассчитываются следующим образом: $Precision@K = \frac{\sum_{j<K} relevance_{ij}}{K}$

$Recall@K = \frac{\sum_{j<K} relevance_{ij}}{L}$,

где L — число объектов в отложенной выборке, с которыми взаимодействовал пользователь; K — количество предметов, рекомендуемых пользователю.

Precision показывает долю предметов, которые были релевантны для пользователя, из всех рекомендованных нами, а recall — долю рекомендованных нами предметов из всех, с которыми пользователь провзаимодействовал в отложенной выборке. В последствии эти метрики усредняются по всем пользователям.

Буква K в названии формулы обозначает размер списка рекомендуемых предметов, который мы рассматриваем. Обычно K берут размером 5, 10, 20 или 50, хотя нет четких правил выбора K — можете брать любое число, которое релевантно нашей задаче. Например, количество товаров, которые показываются на первой странице. Однако заметим, что с ростом K растет метрика recall@k, потому что чем внушительнее список наших рекомендаций,

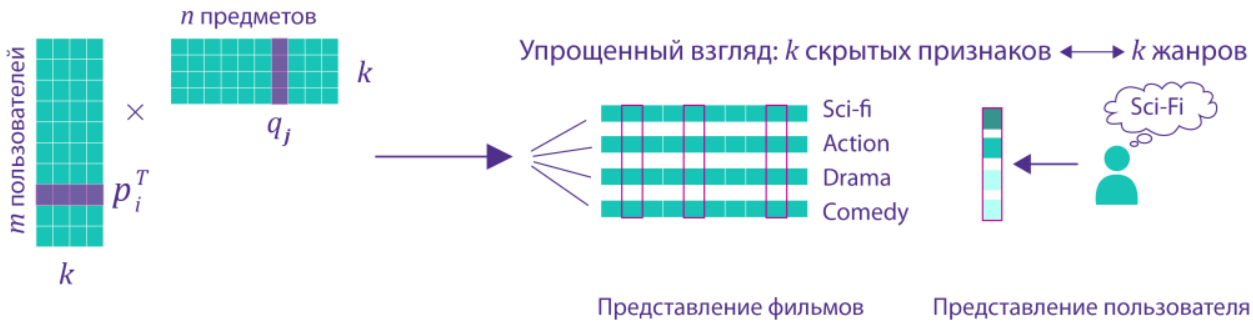
тем выше вероятность охватить большее число предметов, с которыми пользователь взаимодействовал. Соответственно, метрика precision@k снижается, потому что чем больше предметов мы рекомендуем, тем ниже шанс, что в последних, менее релевантных предметах будут актуальные.

Коллаборативная фильтрация и матричная факторизация

Ключевым подходом для построения рекомендаций является коллаборативная фильтрация. В отличии от content-based подхода в основе коллаборативной фильтрации лежит идея о том, что похожим пользователям нравятся одни и те же предметы. Представим, что мы имеем двух похожих пользователей онлайн-кинотеатра, которые любят фильмы про приключения. Они смотрели похожие фильмы, но первый пользователь уже смотрел «Индиану Джонса», а второй еще нет. В таком случае было бы логично порекомендовать второму пользователю этот фильм.

В основе коллаборативной фильтрации лежит **матричная факторизация**. При помощи нее мы можем получить **латентные векторы**, характеризующие пользователей и предметы.

Давайте попытаемся разобраться, что означают все эти сложные термины. Начнем с **латентных векторов**. Базовая идея такая: мы хотим каждого пользователя и каждый предмет представлять вектором пространства размером *k*. Для этого у нас есть модель, которая создает эти **латентные векторы**. Иными словами, для каждого предмета и пользователя есть вектор из *k* чисел, которые как-то его характеризуют, причем мы не понимаем как (а модель, которая строит эти векторы, понимает). Пример: мы делаем рекомендации фильмов при помощи модели, которой мы сообщаем число k «жанров фильмов». Модель будет представлять каждый фильм и каждого пользователя как сочетание всех *k* жанров сопределеннымикоэффициентами. Так, если Маша больше любит романтические комедии и мелодрамы, а ужасы не любит, у нее будут большие коэффициенты для комедий и мелодрам, а для ужасов маленькие или отрицательные.

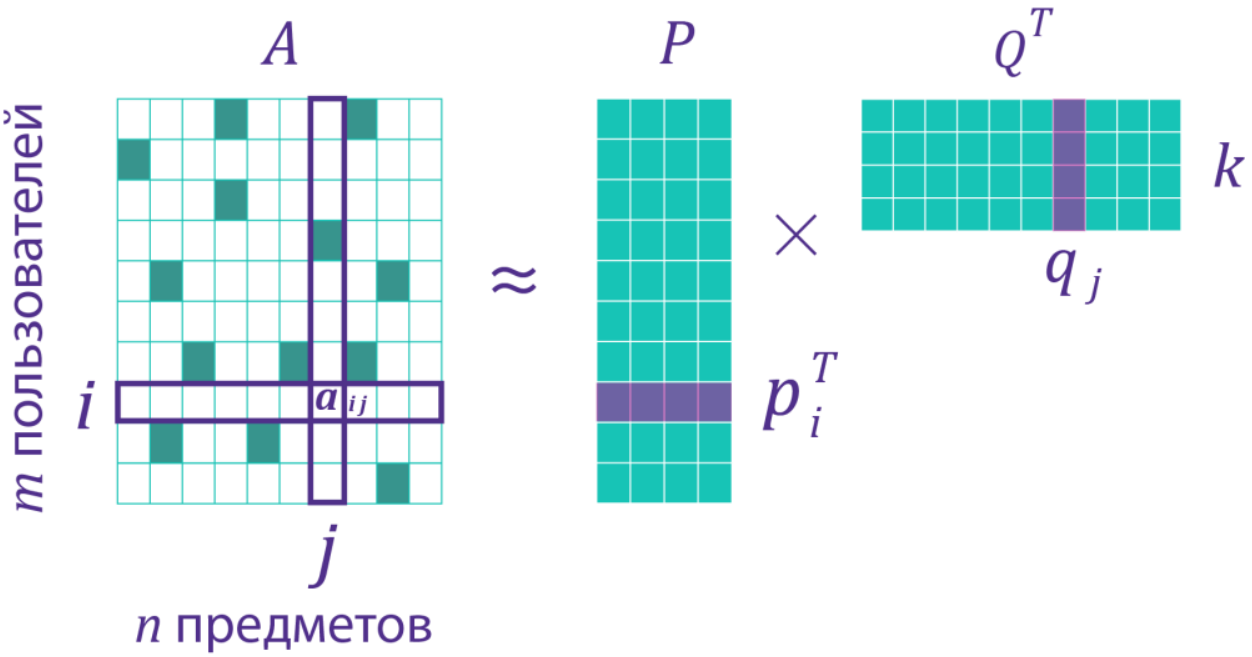


Введем обозначения: латентный вектор пользователя i мы будем обозначать как p_i, латентный вектор предмета (фильма) j будем обозначать как q_j. Всего у нас n пользователей и m фильмов.

Окей, а как наша модель должна строить данные векторы? Давайте вспомним, что коллаборативная фильтрация отражает взаимодействие пользователей и товаров. Поэтому наша модель должна отражать взаимодействия пользователей и фильмов. Таким взаимодействием в онлайн-кинотеатрах могут быть рейтинги фильмов. Рейтинг фильма j, оставленного пользователем i, обозначим как r_{ij}. Тогда мы хотим, чтобы скалярное произведение векторов p_i и q_j было как можно ближе к рейтингу r_{ij}:

$$r_{ij} \approx p_i^T q_j = \sum_{t=1}^{k} p_{it} q_{jt}.$$

Теперь давайте искать, где в этом всем есть **матрица**. Соберем таблицу, в которой строки — наши пользователи, столбцы — фильмы, а в ячейках — рейтинги от пользователей для фильмов. Таким образом, наша таблица, или **матрица**, имеет размерность n x m. Надо помнить, что далеко не все пользователи смотрели все фильмы. Поэтому эта матрица (назовем ее A) является крайне разреженной: большинство ячеек в ней пустые. В реальности доля заполненных ячеек может быть менее 1 %.



Матрица есть, латентные векторы есть, теперь поговорим о **факторизации**, т. е. о том, как нам связать матрицу и векторы. Давайте все векторы пользователей p_i соберем в единую матрицу P размерностью (n, k). То же самое сделаем для векторов q_j. В результате получим матрицу Q размерностью (m, k) и умножим матрицу P на матрицу $Q.T$. По правилам линейной алгебры для матричного перемножения количество столбцов первой матрицы должно быть равно количеству строк второй матрицы, поэтому нам нужно транспонировать (поменять местами строки и столбцы) матрицу Q. В результате мы получим матрицу размерами (n, m). Кажется, чем-то напоминает нашу старую добрую матрицу A.

Назовем новую матрицу $\hat{A}$. Причем нам хочется, чтобы заполненные ячейки старой матрицы были максимально близки к соответствующим ячейкам новой матрицы. Иными словами, чтобы разница в таких ячейках была минимальна. Давайте запишем это в формулах:

$$L(A,\ P,\ Q) \rightarrow \min_{P,\,Q}$$

$$L(A,\ P,\ Q) = \left|A - \hat{A}\right|_2$$

$$\hat{A} = PQ^T$$

L в нашем случае — правило, по которому мы получаем матрицу P и Q, или наша модель. Причем в математике подбор матриц P и Q для исходной

матрицы A называется матричным разложением, или матричной факторизацией. Вот и нашелся последний «герой» нашего квеста по матричной факторизации.

Стоит упомянуть, какие модели могут быть использованы для матричных факторизаций. В первую очередь это SVD-разложение, ALS-алгоритм, NNFM-разложение, модель LightFM, а также модели автоэнкодеров и нейросетевые модели.

Хорошо, а как нам это может помочь в построении

рекомендаций? Если в матрице $\hat{A}$ у нас на месте ячеек, где в матрице A не было пропусков, стоят приближения рейтингов пользователей, то на месте других ячеек стоят (вы не поверите) предсказанные рейтинги для фильмов, которые пользователь не видел. Их мы можем использовать как оценки релевантности для подборок фильмов для пользователей.

Смешанные модели для рекомендаций

Стоит отдельно рассмотреть модель, которая объединяет в себе идеи подхода content-based и коллаборативной фильтрации. Это модель LightFM. Подробнее про нее можно прочитать:

- Статья на архив: https://arxiv.org/pdf/1507.08439.pdf
- Github-репозиторий: https://github.com/lyst/lightfm
- Официальная документация: https://making.lyst.com/lightfm/docs/quickstart.html

Тем, кто хочет развиваться в рекомендательных системах, будет полезно ознакомиться с пакетом LightFM на Python. Данный пакет включает в себя как модель LightFM, так и датасеты и метрики качества рекомендаций. Интерфейс пакета довольно близок к интерфейсу sklearn.

from lightfm import LightFM

from lightfm.evaluation import precision_at_k from lightfm.evaluation import auc_score

model = LightFM(learning_rate=0.05, loss='bpr') model.fit(train, epochs=10)

train_precision = precision_at_k(model, train, k=10).mean() test_precision = precision_at_k(model, test, k=10).mean()

train_auc = auc_score(model, train).mean() test_auc = auc_score(model, test).mean()

Другие примеры с кодом вы можете найти в официальной документации библиотеки. Таким образом, освоение данного пакета поможет вам быстрее погрузиться в задачу рекомендательных систем.

Особенности задачи и особенности промышленного внедрения

Область рекомендательных систем по сравнению с классическим машинным обучением обладает рядом специфических проблем, особенно когда речь доходит до практики. В этом модуле мы рассмотрим некоторые распространенные трудности, с которыми сталкиваются специалисты по анализу данных при решении задач рекомендации в индустрии.

Проблема холодного старта

Многие модели опираются на историю взаимодействия пользователей и предметов на платформе. Но что, если к нам приходят новые пользователи или появляются новые предметы, по которым у нас нет никакой истории взаимодействия? В этом случае коллаборативная фильтрация не подойдет для моделирования. Однако мы можем делать рекомендации на основании характеристик пользователей и товаров. Например, у нас есть платформа для прослушивания музыки. Если к нам приходит новый пользователь, то мы можем при помощи небольшой анкеты узнать его город, пол, возраст или напрямую указать, музыку каких жанров он предпочитает. На основании этих данных мы будем делать первые рекомендации и после получим первые взаимодействия.

Нужно отметить, что проблема холодного старта для пользователей стоит обычно острее, чем для предметов, так как именно пользователь –

источник дохода для бизнеса. Если не заинтересовать пользователя с первых минут, он может уйти на платформу конкурентов.

Другая проблема, связанная с холодным стартом: новым предметам сложно конкурировать с уже популярными предметами на платформе. Для этого обычно новым предметам дается «лимит доверия», и они активнее показываются пользователям. Подобные эвристики помогли раскрутиться TikTok: на этой платформе каждый новый продукт или автор могут быстро попасть в тренды за счет активного показа нового контента пользователям.

Проблема большого числа пользователей и предметов

Другая, более техническая проблема связана с тем, что на нашей платформе может быть много пользователей и предметов. Причем большая часть взаимодействия обеспечивается очень популярными предметами. Однако есть нишевые товары, которые «широко популярны в узких кругах». Такие товары могут дать рост продажам, если рекомендательная система хорошо находит пользователей, которым они могут быть интересны.

Большое количество пользователей и предметов порождает проблемы отсутствия данных по большему количеству пар «предмет — пользователь». Иногда доля взаимодействий, по которым у нас есть информация, может быть меньше 0,01 % от общего числа пар «предмет — пользователь». Данная проблема может решаться матричными факторизациями, которые помогают смоделировать потенциальные рейтинги для полной матрицы «пользователи — предметы».

Проблема скорости работы

Как мы уже отметили, на платформах, где происходят рекомендации, могут быть миллионы предметов. Хорошие алгоритмы рекомендации постоянно адаптируются под действия пользователей, что означает потребность в регулярном обновлении рекомендаций. Однако постоянно сортировать миллионы предметов для десятков тысяч пользователей очень вычислительно затратно.

Эту проблему можно решить предварительным расчетом рекомендаций, показом уже заранее подобранных объявлений и обновлением рекомендаций раз в сутки. Такая схема может подойти для маркетплейсов куда пользователи заходят не так часто. Но есть области, где отложенные рекомендации не решают проблему, так как пользователи ожидают постоянного обновления рекомендаций. Такими областями могут быть YouTube, музыкальные платформы, где пользователи пропускают не понравившиеся им материалы и ожидают, что от их действий рекомендации быстро перестроятся. Решением в случаях, когда нужно постоянно обновлять рекомендации, может быть использование **двухэтапных рекомендаций**:

1 этап. Чтобы большая тяжелая модель не сортировала миллионы предметов, мы хотим сначала быстрым алгоритмом отобрать условно 100–200 самых релевантных пользователю предметов. В качестве алгоритма подойдет, например, матричная факторизация. Эти рекомендации могут быть не такими точными по сравнению с рекомендациями, полученными градиентным бустингом или нейронными сетями, но они помогают «расчистить поле» для более мощных и вычислительно сложных алгоритмов.

2 этап. Чтобы наша рекомендация была точной, для 100–200 предметов из этапа 1 мы применяем ранжирование более мощными алгоритмами. Другими словами, мы переставляем, ранжируем заново предметы точной моделью и выводим пользователю только самые релевантные. Здесь нам важна уже точность, а не скорость. Для этого этапа подойдут нейросетевые модели или ансамблевые модели «классического» машинного обучения.

Сочетание двух моделей позволяет обновлять рекомендации в течение нескольких секунд, при этом сохраняя приемлемое качество.

Проблема оценки качества рекомендаций

Как и для любой модели машинного обучения, выводы о пользе нашей рекомендательной системы мы в первую очередь можем делать по показателям, приносящим пользу бизнесу напрямую: объем продаж, прибыль,

количество показов рекламы и т. д. Однако такие метрики крайне сложно быстро оценить. Поэтому в бизнесе могут считать так называемые прокси-метрики: метрики, напрямую не показывающие пользу модели для бизнеса, но сильно скоррелированные с ними. Ими могут быть количество кликов, время пребывания пользователя на платформе, количество товаров в корзине и т. д. Но данные метрики невозможно рассчитать, когда мы готовим и проверяем новую модель.

Также довольно сложно ввести метрики, отвечающие за разнообразие рекомендаций. Пользователи в один момент могут устать от однообразного контента и захотеть чего-то нового, и если модель выдает только однообразные предметы, например только песни одной и той же группы, то пользователь может уйти на другую платформу, где рекомендации разнообразнее и где ему покажут что-то новенькое. Хорошие рекомендации должны обладать следующими свойствами:

- Быть разнообразными
- Регулярно показывать новые предметы, не рекомендованные до этого
- Быть неожиданными, уметь удивить пользователя
- Покрывать потребности пользователя

Также есть другая проблема. При осуществлении рекомендаций пользователи чаще взаимодействуют с рекомендованными предметами, что в свою очередь делает их более популярными и повышает в рейтинге рекомендаций. Так, многие рекомендательные модели могут просто начать выдавать самые популярные предметы, а нишевые предметы не найдут своего пользователя. Эта проблема в англоязычной литературе получила название Inner Loop.

Таким образом, ключевыми проблемами индустрии является сложность валидации результатов относительно метрик, вычислительная сложность алгоритмов и однообразие рекомендаций. Для решения каждой проблемы проводятся исследования, изобретаются новые подходы и модели.