

Али Аминиан, Алекс Сюй

System Design

Машинное обучение.
Подготовка к сложному интервью



Machine Learning System Design Interview

Ali Aminian | Alex Xu

Али Аминиан, Алекс Сюй

System Design

Машинное обучение.
Подготовка к сложному интервью



Санкт-Петербург · Москва · Минск

2024

ББК 32.973.2-02
УДК 004.41
С98

Сюй Алекс, Аминиан Али

С98 System Design. Машинное обучение. Подготовка к сложному интервью. — СПб.: Питер, 2024. — 320 с.: ил. — (Серия «Библиотека программиста»).
ISBN 978-5-4461-2130-4

Собеседования по проектированию системы машинного обучения — самые сложные из всех вопросов технического собеседования. Эта книга предоставляет надежную стратегию и базу знаний для решения широкого круга вопросов проектирования систем машинного обучения. Пошаговый подход формирует основу для решения любого вопроса проектирования, используя множество реальных примеров.

Эта книга поможет всем, кто интересуется проектированием систем машинного обучения, будь то новички или опытные инженеры. Если вам нужно подготовиться к собеседованию по данной теме, эта книга создана специально для вас.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.2-02
УДК 004.41

Права на издание получены по соглашению с Ali Aminian. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги. В книге возможны упоминания организаций, деятельность которых запрещена на территории Российской Федерации, таких как Meta Platforms Inc., Facebook, Instagram и др. Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

ISBN 978-1736049129 англ.
ISBN 978-5-4461-2130-4

© 2023 Ali Aminian
© Перевод на русский язык ООО «Прогресс книга», 2023
© Издание на русском языке, оформление ООО «Прогресс книга», 2023
© Серия «Библиотека программиста», 2023

ОГЛАВЛЕНИЕ

Предисловие	14
Что такое собеседование по проектированию систем МО (ML System Design interview).....	14
Почему это важно	14
Для кого эта книга	15
Чего нет в книге	15
Дополнительные ресурсы	15
Благодарности.....	16
От издательства	18
Глава 1. Введение и общие сведения	19
Прояснение требований.....	20
Формулировка проблемы в виде задачи МО.....	21
Определение цели МО	22
Определение входных и выходных данных системы	22
Выбор подходящей категории МО	24
Темы для обсуждения	25
Подготовка данных	25
Инженерия данных	26
Типы данных	27
Конструирование признаков.....	29
Операции конструирования признаков	30
Темы для обсуждения	34
Разработка модели	34
Выбор модели.....	34
Обучение модели	36
Темы для обсуждения	40
Оценка	41
Автономная (offline) оценка	41
Оперативная (online) оценка	42
Темы для обсуждения	43

Развертывание и эксплуатация.....	43
Развертывание в облаке или на устройстве	43
Сжатие модели.....	44
Тестирование при эксплуатации	44
Пайплайн предсказаний	46
Темы для обсуждения	48
Мониторинг	48
Почему в действующей системе происходят сбои	49
Какие показатели нужно отслеживать.....	50
Инфраструктура.....	50
Итоги	51
Ссылки	51
Глава 2. Система визуального поиска	54
Прояснение требований.....	54
Формулировка проблемы в виде задачи МО.....	56
Определение цели МО	56
Определение входных и выходных данных системы.....	56
Выбор подходящей категории МО	56
Подготовка данных	58
Инженерия данных	58
Конструирование признаков.....	60
Разработка модели	60
Выбор модели.....	60
Обучение модели	61
Построение датасета.....	62
Выбор функции потерь	64
Оценка	66
Автономные метрики	66
Оперативные метрики	72
Эксплуатация.....	72
Предсказательный пайплайн	73
Пайплайн индексации.....	74
Эффективность алгоритмов поиска ближайшего соседа (NN).....	75
Какой алгоритм использовать?	77
Другие темы для обсуждения.....	78
Итоги	79
Ссылки	80

Глава 3. Система размытия в Google Street View	81
Прояснение требований.....	81
Формулировка проблемы в виде задачи МО.....	82
Определение цели МО	82
Определение входных и выходных данных системы.....	83
Выбор подходящей категории МО	83
Двухступенчатые сети.....	83
Одноступенчатые сети	84
Сравнение одноступенчатых и двухступенчатых сетей	84
Подготовка данных	85
Инженерия данных	85
Конструирование признаков.....	86
Разработка модели	88
Выбор модели.....	88
Обучение модели	90
Оценка.....	91
Автономные метрики	93
Оперативные метрики	95
Эксплуатация.....	95
Перекрытие ограничительных прямоугольников.....	95
Проектирование системы МО	96
Пайплайн пакетных предсказаний	96
Другие темы для обсуждения.....	98
Итоги	99
Ссылки	100
Глава 4. Поиск видео на YouTube	101
Прояснение требований.....	101
Формулировка проблемы в виде задачи МО.....	102
Определение цели МО	102
Определение входных и выходных данных системы.....	102
Выбор категории МО	103
Подготовка данных	105
Инженерия данных	105
Конструирование признаков.....	106
Разработка модели	110
Выбор модели.....	110
Обучение модели	116

Оценка	117
Автономные метрики	117
Оперативные метрики	118
Эксплуатация.....	119
Предсказательный пайплайн	120
Пайплайн индексации видео	121
Пайплайн индексации текста	121
Другие темы для обсуждения.....	121
Итоги	123
Ссылки	124
Глава 5. Обнаружение вредоносного контента	125
Прояснение требований.....	126
Формулировка проблемы в виде задачи МО.....	127
Определение цели МО	127
Определение входных и выходных данных системы.....	127
Выбор категории МО.....	131
Подготовка данных.....	137
Инженерия данных	137
Конструирование признаков.....	138
Разработка модели	143
Выбор модели.....	143
Обучение модели	144
Оценка	146
Автономные метрики	146
Оперативные метрики	147
Эксплуатация.....	148
Сервис обнаружения вредоносного контента	148
Сервис обработки нарушений.....	148
Сервис понижения приоритета	148
Другие темы для обсуждения.....	149
Итоги	150
Ссылки	151
Глава 6. Система рекомендации видео.....	153
Прояснение требований.....	154
Формулировка проблемы в виде задачи МО.....	155
Определение цели МО	155
Определение входных и выходных данных системы.....	156
Выбор категории МО.....	156

Подготовка данных	161
Инженерия данных	161
Конструирование признаков.....	163
Разработка модели.....	168
Матричная факторизация.....	168
Двухбашенная нейронная сеть	175
Работа двухбашенной нейронной сети	177
Матричная факторизация или двухбашенная нейронная сеть?	179
Оценка.....	179
Автономные метрики	179
Оперативные метрики	180
Эксплуатация.....	181
Генерация кандидатов	182
Скоринг	184
Повторное ранжирование.....	184
Трудности при разработке систем рекомендации видео	185
Другие темы для обсуждения.....	186
Итоги	187
Ссылки	188
Глава 7. Система рекомендации событий.....	189
Прояснение требований.....	190
Формулировка проблемы в виде задачи МО.....	191
Определение цели МО	191
Определение входных и выходных данных системы.....	191
Выбор подходящей категории МО	192
Списочные методы LTR	193
Подготовка данных	194
Инженерия данных	194
Конструирование признаков.....	196
Разработка модели	204
Выбор модели.....	204
Обучение модели	212
Оценка.....	214
Автономные метрики	214
Оперативные метрики	214
Эксплуатация.....	215
Пайплайн оперативного обучения	215
Предсказательный пайплайн	216

10 Оглавление

Другие темы для обсуждения.....	218
Итоги	219
Ссылки	220
Глава 8. Предсказание кликов по рекламе на социальных платформах	221
Введение	221
Прояснение требований.....	222
Формулировка проблемы в виде задачи МО.....	223
Определение цели МО	223
Определение входных и выходных данных системы.....	223
Выбор категории МО.....	224
Подготовка данных.....	225
Инженерия данных	225
Конструирование признаков.....	226
Признаки пользователя.....	228
Разработка модели	230
Выбор модели.....	230
Обучение модели	241
Оценка	242
Автономные метрики	242
Оперативные метрики	244
Эксплуатация.....	245
Пайплайн подготовки данных	246
Пайплайн непрерывного обучения	247
Предсказательный пайплайн	247
Другие темы для обсуждения.....	248
Итоги	249
Ссылки	250
Глава 9. Похожие объекты на платформах краткосрочной аренды жилья.....	251
Прояснение требований.....	252
Формулировка проблемы в виде задачи МО.....	253
Определение цели МО	253
Определение входных и выходных данных системы.....	253
Выбор категории МО.....	254
Подготовка данных.....	255
Инженерия данных	255
Конструирование признаков.....	256

Разработка модели	257
Выбор модели.....	257
Обучение модели	257
Построение датасета.	258
Оценка.....	262
Автономные метрики	262
Оперативные метрики	264
Эксплуатация.....	264
Обучающий пайплайн	264
Индексирующий пайплайн.....	265
Предсказательный пайплайн	266
Другие темы для обсуждения.....	267
Итоги	268
Ссылки	269
Глава 10. Персонализированная лента новостей	270
Введение	270
Прояснение требований.....	271
Формулировка проблемы в виде задачи МО.....	272
Определение цели МО	272
Определение входных и выходных данных системы.....	274
Выбор категории МО	274
Подготовка данных.....	275
Инженерия данных	275
Конструирование признаков.....	277
Признаки пользователей	280
Разработка модели	282
Выбор модели.....	282
Обучение модели	285
Оценка.....	287
Автономные метрики	287
Оперативные метрики	287
Эксплуатация.....	289
Предсказательный пайплайн	289
Другие темы для обсуждения.....	290
Итоги	291
Ссылки	292

Глава 11. Списки возможных знакомых	293
Введение	293
Прояснение требований.....	293
Формулировка проблемы в виде задачи МО.....	294
Определение цели МО	294
Выбор категории МО.....	295
Подготовка данных	299
Инженерия данных	299
Конструирование признаков.....	301
Разработка модели	303
Выбор модели.....	303
Графовые нейронные сети (GNN)	303
Обучение модели	304
Оценка	308
Автономные метрики	308
Оперативные метрики	308
Эксплуатация.....	309
Эффективность.....	309
Проектирование системы МО	311
Другие темы для обсуждения.....	314
Итоги	315
Ссылки	316
Послесловие.....	317

Посвящается Нилуфар, которая была и остается
моим лучшим другом.

Али Аминиан

Посвящается Джулии.

Алекс Сюй

ПРЕДИСЛОВИЕ

Мы рады, что вы читаете эту книгу, чтобы лучше подготовиться к собеседованию по проектированию систем машинного обучения (МО). **Проектирование систем (system design)** — одна из самых сложных тем на любом собеседовании из области МО, так что без хорошей подготовки не обойтись.

Что такое собеседование по проектированию систем МО (ML System Design interview)

Собеседование по проектированию систем МО, как правило, обязательно для претендентов на вакансии, связанные с проектированием и реализацией систем МО: инженер данных, data-сайентист, инженер машинного обучения и т. д.

На таком собеседовании оценивается умение кандидата проектировать комплексные системы МО — визуальный поиск, рекомендации видео, предсказание кликов по рекламе и т. д. Вопросы на собеседовании бывают непростыми, потому что у них обычно нет четкой структуры. Часто однозначных ответов не существует, потому что темы вопросов оказываются широкими и затрагивают разные области, что открывает возможность для различных интерпретаций и решений.

Чтобы успешно пройти собеседование по проектированию систем МО, надо хорошо понимать фундаментальные концепции и методы МО, а также уметь их применять, чтобы решать практические задачи. На собеседовании обычно необходимо продемонстрировать, что вы разбираетесь в пайплайнах данных и конструировании признаков, а также умеете проектировать эффективные системы МО. Возможно, вам еще придется проявить умение выбирать подходящие модели для конкретных задач, настраивать их параметры и оценивать производительность. В принципе, цель собеседования состоит в том, чтобы оценить, насколько хорошо соискатель применяет теоретические знания МО, чтобы проектировать и реализовывать эффективные системы.

Почему это важно

Большинство соискателей, которые проходят собеседования по МО, владеют теоретическими основами, но затрудняются в области проектирования систем

МО, где нет единых руководящих принципов. Тем не менее умение проектировать такие системы — это важнейший навык для инженеров, особенно в контексте карьерного роста. Неправильно выбранная архитектура системы МО может привести к значительным потерям времени и других ресурсов.

Очевидно, собеседование по проектированию систем МО — это важнейшая часть процедуры приема на работу. Чем лучше вы себя покажете, тем вернее можете рассчитывать на более привлекательную позицию и более высокий заработок.

Для кого эта книга

Книга будет ценным источником информации для всех, кто интересуется проектированием систем МО, будь то новички или опытные инженеры. А если вам нужно подготовиться к собеседованию по МО, то эта книга написана специально для вас.

Чего нет в книге

Эта книга — не пособие по основам машинного обучения. Она написана для дата-сайентистов, инженеров данных и инженеров МО, которым нужна помощь, чтобы подготовиться к собеседованию по проектированию систем МО. Книга предназначена в первую очередь для инженеров МО в бизнесе и в меньшей степени для ученых в области МО в образовательных учреждениях или НИИ.

Дополнительные ресурсы

В конце каждой главы приводится длинный список ссылок на дополнительные материалы. Все активные ссылки доступны в репозитории GitHub:

<https://bit.ly/ml-bytebytogo>



БЛАГОДАРНОСТИ

Как бы нам ни хотелось похвастаться, что все примеры проектирования в этой книге полностью оригинальны, все же придется признаться, что большинство идей, которые здесь рассматриваются, можно найти и в других местах: технических блогах, исследовательских статьях, коде, презентациях на YouTube и прочих источниках. Мы собрали эти блестящие идеи, исследовали их и дополнили собственными наблюдениями и опытом, чтобы представить в простой и понятной форме. Авторы хотели бы сказать огромное спасибо многим техническим специалистам и руководителям, которые внесли вклад в работу над этой книгой и рецензировали ее:

- Винит Ахлувалия (Vineet Ahluwalia) (Стэнфордский университет)
- Топоджой Бисвас (Topojoy Biswas) (Walmart)
- Да Чэн (Da Cheng) (Tiktok)
- Рохит Джайн (Rohit Jain) (Twitter)
- Кальян Дипак (Kalyan Deepak) (Flipkart)
- Димитрис Котсакос (Dimitris Kotsakos) (Elastic)
- Субхам Кумар (Subham Kumar) (Amazon)
- Джастин Ли (Justin Li) (Discord)
- Ли Сюй (Li Xu) (TikTok)
- Рави Мандлия (Ravi Mandliya) (Discord)
- Саранг Меткар (Sarang Metkar) (Meta)
- Шабаз Патель (Shabaz Patel) (One Concern)
- Каустубх Пхаднис (Kaustubh Phadnis) (Walmart)
- Рави Рамчандран (Ravi Ramchandran) (Walmart Labs)
- Дэван Султания (Dewang Sultania) (Adobe)
- Сяо Чжу (Xiao Zhu) (Databricks)
- Цзяин Ши (Jiaying Shi) (Amazon)
- Цзяньцян Ван (Jianqiang Wang) (Snapchat)
- Чжэхуэй Ван (Zhehui Wang) (Amazon)

- Шо Сян (Shuo Xiang) (Parafin)
- Бонту Шридеви (Bonthu Sridevi) (Технологический институт Вишну)
- Диала Эззеддин (Diala Ezzeddine) (Tao Media)
- Юаньцзюнь Ян (Yuanjun Yang) (Twitter)

Наконец, мы хотим особо поблагодарить Элвиса Жэня (Elvis Ren), Хя Ли (Hua Li) и Сань Лама (Sahn Lam) за неоценимый вклад в работу над книгой.

ОТ ИЗДАТЕЛЬСТВА

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

1 ВВЕДЕНИЕ И ОБЩИЕ СВЕДЕНИЯ

Мы написали эту книгу, чтобы помочь инженерам по машинному обучению и data-сайентистам успешно пройти собеседование по проектированию систем МО. Книга также может пригодиться всем, кто хочет получить общее представление о том, как МО применяется в реальном мире.

Многие технические специалисты полагают, что системы МО исчерпываются такими алгоритмами МО, как логистическая регрессия или нейронные сети. Тем не менее реальные системы МО далеко не ограничиваются разработкой моделей. Эти системы обычно весьма сложны; они состоят из множества компонентов, включая стеки данных, служебную инфраструктуру (благодаря которой система становится доступной миллионам пользователей), пайплайн для оценки ее эффективности, а также средства мониторинга, которые следят за тем, чтобы качество модели не ухудшалось со временем.

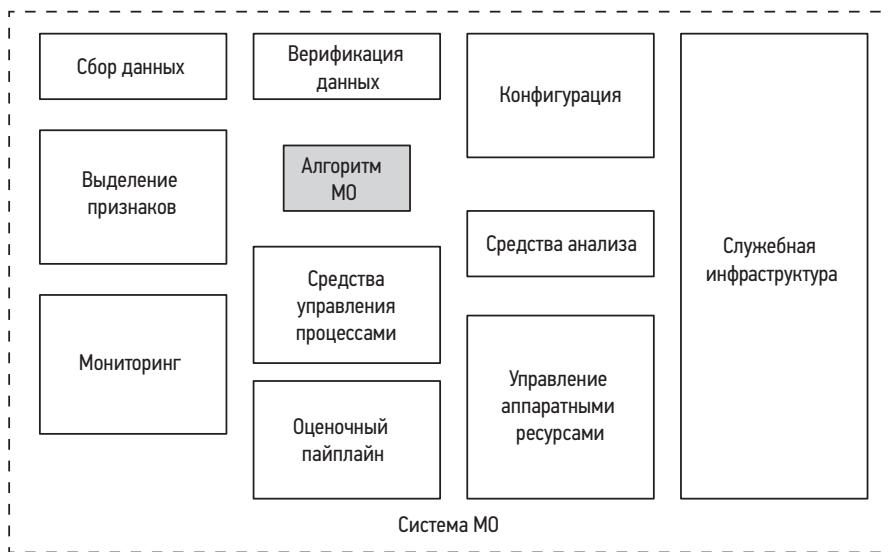


Рис. 1.1. Компоненты системы МО, готовой к эксплуатации

Скорее всего, на собеседовании по проектированию систем МО вам предстоит отвечать на вопросы открытого типа. Например, вам могут предложить спроектировать систему для рекомендаций фильмов или службу поиска видео. У таких задач нет единственно правильного решения. Эксперт, проводящий собеседование,

хочет посмотреть, как вы размышляете, глубоко ли понимаете различные темы из области МО, умеете ли проектировать комплексные системы и находить компромиссы между конфликтующими факторами, которые влияют на проектирование.

Чтобы успешно проектировать сложные системы МО, очень важно придерживаться определенной логики. Без структуры сложно разобраться в проектировочных решениях. Здесь мы предлагаем схему, на которую в этой книге будут опираться примеры проектирования систем МО. Схема состоит из семи основных шагов.

1. Прояснение требований
2. Формулировка проблемы как задачи МО
3. Подготовка данных
4. Разработка модели
5. Оценка
6. Развёртывание и эксплуатация
7. Мониторинг и инфраструктура

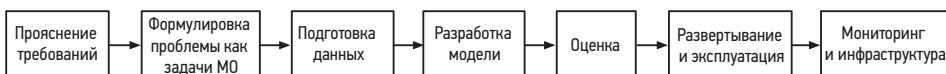


Рис. 1.2. Основные шаги проектирования систем МО

Каждое собеседование по проектированию систем МО отличается от других, потому что вопросы носят открытый характер и не существует универсальных удачных решений. Эта схема помогает упорядочить мысли, но строго следовать ей необязательно. Сохраняйте гибкость. Если эксперта, проводящего собеседование, в первую очередь интересует разработка модели, почти всегда стоит подстраиваться под его запросы.

Давайте подробнее рассмотрим каждый шаг этой схемы.

Прояснение требований

Вопросы на собеседовании по проектированию систем МО обычно намеренно ставятся нечетко, с минимумом информации. Например, вопрос может звучать так: «Спроектируйте систему для рекомендации событий». Прежде всего стоит задать уточняющие вопросы. Но какие именно? Нужны такие вопросы, которые помогут понять конкретные требования. Этот систематизированный список вопросов можно взять за основу.

- **Бизнес-цель.** Если система должна рекомендовать отпускное жилье для бронирования, то цели могут заключаться в том, чтобы увеличить количество бронирований и выручку.

- **Функции, которые должна поддерживать система.** Какие из требуемых функциональных возможностей могут повлиять на проектирование системы МО? Допустим, вам предложено спроектировать систему для рекомендации видео. Вероятно, стоит уточнить, могут ли пользователи ставить рекомендуемому контенту лайки или дизлайки, потому что этими оценками можно размечать обучающие данные.
- **Данные.** Откуда поступают данные? Каков размер датасета? Размечены ли данные?
- **Ограничения.** Какая вычислительная мощность доступна? Будет ли система работать в облаке или на локальном устройстве? Планируется ли, что со временем модель будет автоматически совершенствоваться?
- **Масштаб системы.** Сколько пользователей будет у системы? С каким количеством объектов (например, видеороликов) придется иметь дело? С какой скоростью растут эти показатели?
- **Производительность.** Насколько быстрыми должны быть предсказания? Должна ли система работать в реальном времени? Что важнее — точность или низкая задержка?

Это не исчерпывающий список, но его можно принять за отправную точку. Не забывайте, что могут быть и другие важные аспекты — например, конфиденциальность и этика.

Предполагается, что к концу этого этапа вы согласуете с экспертом рамки системы и требования к ней. Обычно имеет смысл составить список требований и ограничений: это поможет убедиться, что все одинаково представляют себе задачу.

Формулировка проблемы в виде задачи МО

При решении задач МО крайне важно правильно сформулировать проблему. Допустим, эксперт предлагает вам увеличить степень вовлеченности пользователей на платформе видеостриминга. Безусловно, недостаточная вовлеченность — это проблема, но это не задача МО. Таким образом, чтобы решить проблему, ее следует переформулировать в виде задачи МО.

В реальности сначала надо выяснить, действительно ли для решения проблемы нужно МО. Однако на собеседовании по проектированию систем МО логично допустить, что от МО все-таки будет польза. Таким образом, чтобы сформулировать проблему как задачу МО, можно поступить так:

- Определить цель МО.
- Определить входные и выходные данные системы.
- Выбрать подходящую категорию МО.

Определение цели МО

Бизнес-цель может состоять в том, чтобы повысить продажи на 20 % или увеличить степень удержания пользователей. Однако цели не всегда определяются четко, а модель невозможно обучить, просто приказав ей увеличить продажи на 20 %. Чтобы система МО решила задачу, нужно преобразовать бизнес-цель в точно определенную цель МО. Хорошой целью МО будет такая, которую можно решить с помощью моделей МО. Некоторые примеры перечислены в табл. 1.1, а в дальнейших главах встречаются и другие примеры.

Таблица 1.1. Преобразование бизнес-целей в цели МО

Приложение	Бизнес-цель	Цель МО
Приложение для продажи билетов на мероприятия	Повысить продажи билетов	Максимизировать количество регистраций на мероприятия
Видеостриминговое приложение	Повысить степень вовлеченности пользователей	Максимизировать время, которое пользователи проводят за просмотром видео
Система прогнозирования кликов по рекламе	Увеличить количество кликов	Максимизировать CTR (кликаемость)
Выявление вредоносного контента в социальной сети	Сделать платформу безопаснее	Точно предсказывать, является ли контент вредоносным
Система рекомендации друзей	Увеличить темп расширения сети пользователя	Максимизировать количество установленных связей

Определение входных и выходных данных системы

Когда цель МО ясна, нужно определить входные и выходные данные системы. Например, для системы выявления вредоносного контента в социальной сети входными данными является пост, а выходными — решение о том, считать ли его вредоносным.

Иногда система может состоять более чем из одной модели МО. В таком случае требуется определить входные и выходные данные для каждой модели. Например, в примере с выявлением вредоносного контента одна модель может выявлять призыв к насилию, а другая — непристойные изображения. Система опирается на обе эти модели, чтобы решить, считать ли пост вредоносным.

Еще одно важное соображение состоит в том, что может существовать несколько способов определить входные и выходные данные модели — см. пример на рис. 1.4.



Рис. 1.3. Входные и выходные данные системы выявления вредоносного контента

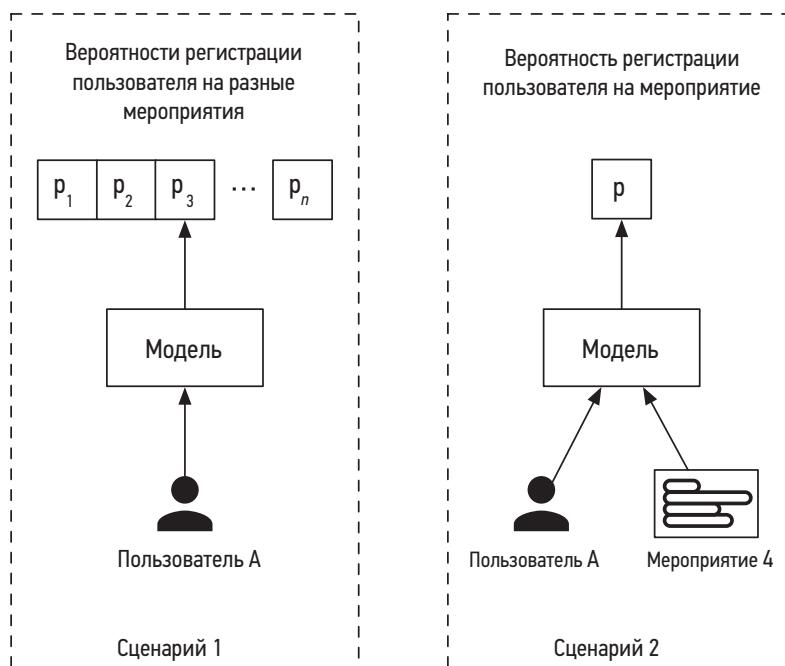


Рис. 1.4. Разные способы определения входных и выходных данных модели

Выбор подходящей категории МО

Существует много способов переформулировать проблему в виде задачи МО. Большинство проблем можно представить так, чтобы они относились к одной из категорий МО, изображенных на рис. 1.5. Поскольку эти категории, вероятно, уже знакомы большинству читателей, мы ограничимся краткой сводкой.

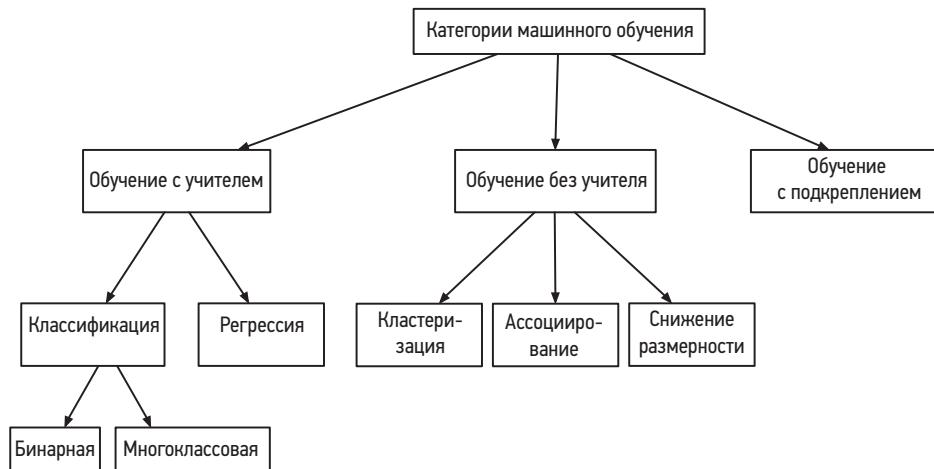


Рис. 1.5. Распространенные категории МО

Обучение с учителем. В этих моделях используется обучающий набор данных. На практике многие проблемы относятся к этой категории, потому что обучение на размеченном наборе данных обычно приводит к лучшим результатам.

Обучение без учителя. Чтобы делать предсказания, такие модели обрабатывают данные, которые не содержат правильных ответов. Цель обучения — выявить осмыслиенные закономерности в данных. Популярные алгоритмы обучения без учителя — кластеризация, ассоциирование и снижение размерности.

Обучение с подкреплением. Система учится решать задачу, многократно взаимодействуя со средой методом проб и ошибок. Например, таким способом можно научить робота ходить по комнате или натренировать такую программу, как AlphaGo, чтобы она успешно соревновалась с человеком в игре го.

По сравнению с обучением с учителем, обучение без учителя и обучение с подкреплением менее популярны в реальных системах, потому что модели МО обычно лучше обучаются, если есть обучающие данные. Поэтому для большинства проблем, которые рассматриваются в этой книге, применяется обучение с учителем. Давайте поближе познакомимся с разными его видами.

Регрессионная модель. Регрессия предсказывает непрерывное числовое значение — например, ожидаемую стоимость дома.

Классификационная модель. Классификация предсказывает дискретную метку класса — например, следует ли отнести входное изображение к классу «собака», «кошка» или «кролик». Классификационные модели можно разделить на две группы.

- **Бинарная классификация** предсказывает бинарный результат — например, есть на изображении собака или нет.
- **Многоклассовая классификация** разбивает входные данные на несколько классов: например, можно классифицировать объект на изображении как собаку, кошку или кролика.

Предполагается, что на этом шаге вы выберете правильную категорию МО. В следующих главах приводятся примеры того, как выбрать подходящую категорию во время собеседования.

Темы для обсуждения

Вот некоторые из тем, которые могут обсуждаться во время собеседования.

- Что такая хорошая цель МО? Как сравнить между собой разные цели МО? Какие у них плюсы и минусы?
- Какие входные и выходные данные будут у системы для конкретной цели МО?
- Если в системе МО задействованы несколько моделей, каковы входные и выходные данные у каждой из них?
- Как должно проводиться обучение — с учителем или без?
- Какая модель лучше поможет решить проблему — регрессия или классификация? Если используется классификация, то должна ли она быть бинарной или многоклассовой? А если регрессия, то каким должен быть диапазон выходных значений?

Подготовка данных

Модели МО обучаются непосредственно на данных, а это значит, что для обучения чрезвычайно важны данные с высокой предсказательной способностью. Этот раздел посвящен тому, как готовить качественные входные данные для моделей МО с помощью двух основных процессов — инженерии данных (data engineering) и конструирования признаков (feature engineering). Мы рассмотрим важные аспекты того и другого.

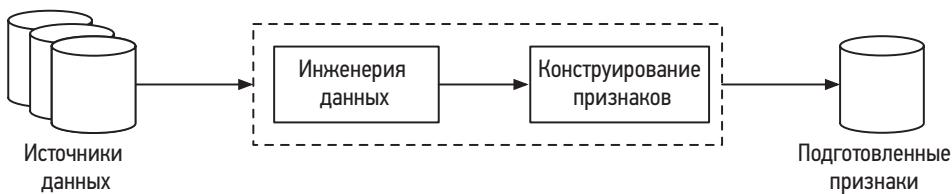


Рис. 1.6. Процесс подготовки данных

Инженерия данных

Инженерия данных заключается в том, чтобы проектировать и строить пайплайны для сбора, хранения, извлечения и обработки данных. Кратко рассмотрим основные принципы инженерии данных, чтобы понять, какие основные компоненты для нее могут понадобиться.

Источники данных

Система МО может работать с данными из многих источников. Полезно разбираться в источниках данных, чтобы отвечать на различные контекстные вопросы, например: кто собирал данные? Насколько они чисты? Можно ли доверять источнику? Данные созданы пользователями или сгенерированы машиной?

Хранилище данных

Хранилище данных (или база данных, БД) — это репозиторий, который позволяет долгосрочно хранить коллекции данных и управлять ими. Для разных сценариев использования применяются разные БД, поэтому важно понимать на высоком уровне, как работают те или иные базы данных. Для собеседования по проектированию систем МО обычно не требуется разбираться во внутреннем устройстве баз данных.

SQL	NoSQL			
Реляционные 	Ключ — значение 	Семейства столбцов 	Графовые 	Документо-ориентированные
• MySQL • PostgreSQL	• Redis • DynamoDB	• Cassandra • HBase	• Neo4J	• MongoDB • CouchDB

Рис. 1.7. Разные виды баз данных

Извлечение, преобразование и загрузка (ETL)

Процедура ETL (Extract, Transform, Load — «извлечение, преобразование, загрузка») состоит из трех фаз:

- **извлечение:** данные извлекаются из разных источников;
- **преобразование:** в этой фазе данные обычно очищаются, приводятся в порядок и преобразуются в тот формат, который нужен для выполняемых задач;
- **загрузка:** преобразованные данные загружаются в приемник — файл, базу данных или хранилище данных [1].

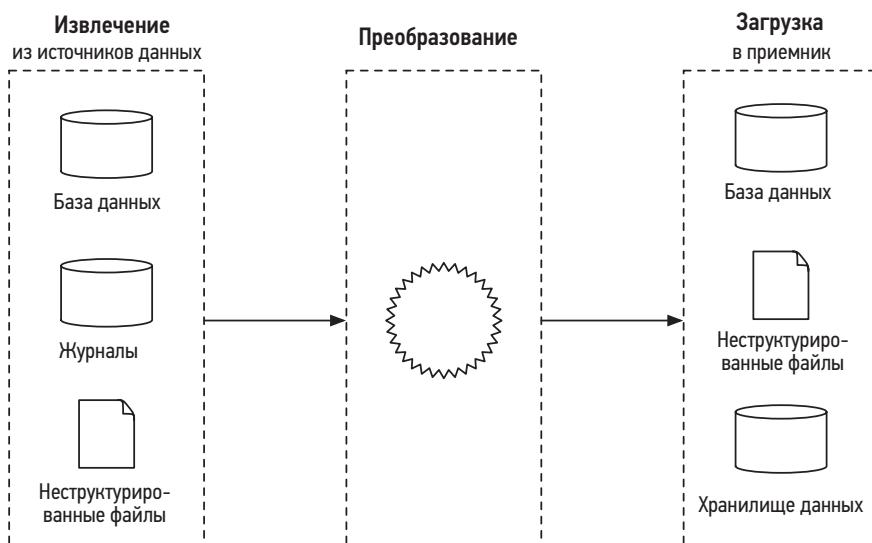


Рис. 1.8. Обзор процесса ETL

Типы данных

Типы данных в машинном обучении отличаются от типов в языках программирования (`int`, `float`, `string` и т. д.). На высоком уровне типы данных можно разделить на две категории: структурированные и неструктурированные (рис. 1.9).

Структурированные данные подчиняются заранее определенной схеме. Например, структурированными данными можно считать даты, имена, адреса, номера кредитных карт и вообще все, что можно представить в табличном формате со строками и столбцами. Неструктурированные данные не соответствуют какой-то конкретной схеме; к этой категории относятся текст, изображения, аудио- и видеозаписи. В табл. 1.2 перечислены основные различия между структурированными и неструктурированными данными.

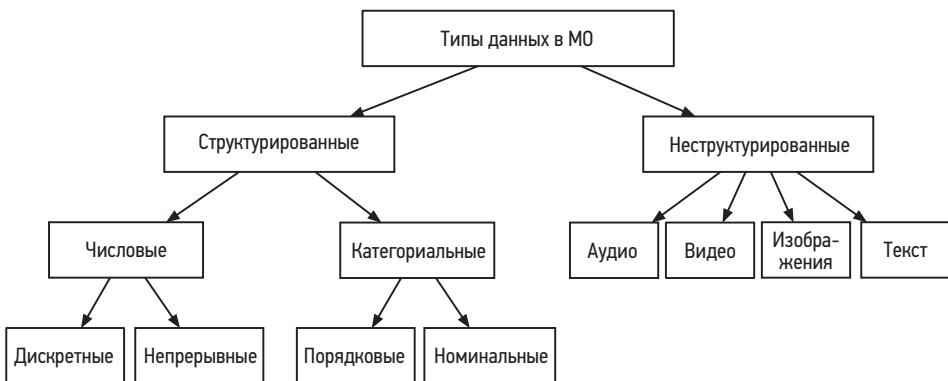


Рис. 1.9. Типы данных в МО

Таблица 1.2. Структурированные и неструктурированные данные

	Структурированные данные	Неструктурированные данные
Характеристики	Заранее определенная схема. Просто выполнять поиск	Нет определенной схемы. Трудно выполнять поиск
Место хранения	Реляционные базы данных. Во многих базах данных NoSQL могут храниться структурированные данные. Хранилища данных	Базы данных NoSQL. Озера данных
Примеры	Даты. Телефонные номера. Номера кредитных карт. Адреса. Имена	Текстовые файлы. Аудиофайлы. Изображения. Видео

Как показано на рис. 1.10, для разных типов данных подходят разные модели МО. Важно понимать, структурированы данные или нет, чтобы выбрать подходящую модель МО на шаге разработки модели.

Числовые данные

К числовым данным относятся любые значения, представленные числами. Как показано на рис. 1.9, числовые данные делятся на непрерывные и дискретные. Например, цены на недвижимость можно считать непрерывными, потому что цена может принимать любое значение из соответствующего диапазона. С другой стороны, количество домов, проданных за последний год, можно считать примером дискретных числовых данных, так как оно принимает только целые значения.

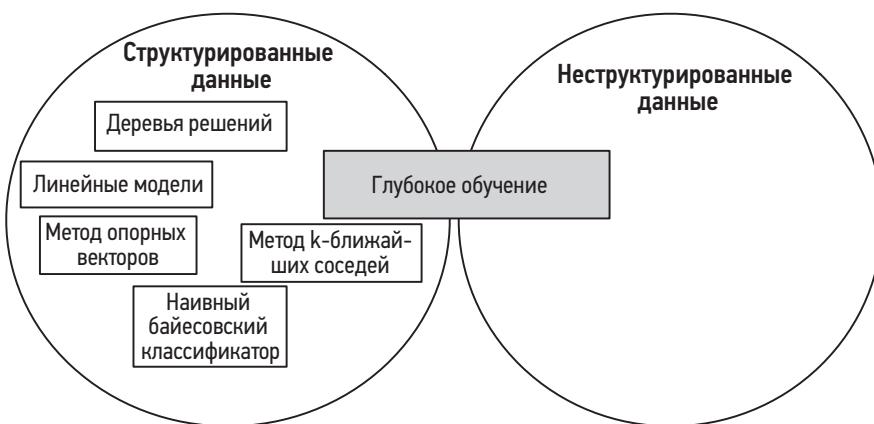


Рис. 1.10. Модели для структурированных и неструктурированных данных (см. [2])

Категориальные данные

Категориальные данные – это данные, которые можно хранить и идентифицировать с помощью присвоенных им имен или меток. Например, пол относится к категориальным данным, потому что его значение берется из ограниченного набора нечисловых значений. Категориальные данные можно разделить на две группы: номинальные и порядковые.

Под номинальными данными понимаются данные, между категориями которых нет числовой связи. Например, сюда относится пол, потому что между значениями «мужской» и «женский» нет числовых отношений. Порядковые данные состоят из значений, между которыми есть заранее определенный или последовательный порядок, — например, сюда относятся оценки с тремя уникальными значениями: «недоволен», «нейтрально» и «доволен».

Конструирование признаков

К конструированию признаков относятся два процесса:

- использование знаний о предметной области, чтобы выбирать и извлекать предсказательные признаки из необработанных данных;
- преобразование предсказательных признаков в формат, пригодный для модели.

Выбрать подходящие признаки — одна из главных задач при разработке и обучении моделей МО. Важно выбрать признаки, которые приносят наибольшую практическую пользу. На этом этапе нужно хорошо знать предметную область, причем процесс также сильно зависит от конкретной задачи. Чтобы помочь вам его освоить, в книге мы приводим многочисленные примеры.

После того как предсказательные признаки выбраны, их нужно преобразовать в подходящие форматы с помощью операций конструирования признаков, которые рассматриваются ниже.

Операции конструирования признаков

Нередко выясняется, что некоторые из выбранных признаков представлены не в том формате, который может использовать модель. Операции конструирования признаков преобразуют их в подходящий формат. Среди таких операций — обработка отсутствующих значений, масштабирование значений со смещенным распределением, а также кодирование категориальных признаков. Хотя приведенный ниже список не исчерпывающий, в него включены самые распространенные операции со структурированными данными.

Обработка отсутствующих значений

В реальных данных часто встречаются отсутствующие значения. Эту проблему обычно решают одним из двух способов: удалением или импутацией.

Удаление. В этом случае удаляются все записи, где отсутствуют значения каких-либо признаков. Удалять можно столбцы или строки. В первом случае удаляется весь столбец, соответствующий признаку, если в нем слишком много пропущенных значений. Во втором случае удаляется строка, представляющая один объект предметной области, у которого не хватает слишком большого количества значений.

Данные				
ID	Признак 1	Признак 2	Признак 3	Признак 4
1	2	N/A	6	6
2	9	N/A	8	7
3	18	N/A	11	21
4	2	11	5	6

Данные			
ID	Признак 1	Признак 3	Признак 4
1	2	6	6
2	9	8	7
3	18	11	21
4	2	5	6

Рис. 1.11. Удаление столбцов

Недостаток удаления заключается в том, что сокращается объем данных, которые модель потенциально может использовать для обучения, — притом, что модели МО обычно работают лучше, когда располагают большим объемом данных.

Импутация (восполнение). Также можно попытаться восполнить отсутствующие данные, подставив на пустые места те или иные значения. Некоторые распространенные приемы:

- отсутствующие значения заменяются значениями по умолчанию;
- отсутствующие значения заменяются средним, медианой или модой (наиболее часто встречающимся значением).

Недостаток импутации в том, что она может повышать уровень шума в данных. Имейте в виду, что не существует идеального способа обработки отсутствующих значений: у каждого варианта есть свои достоинства и недостатки.

Масштабирование признаков

Масштабирование признаков заключается в том, что они приводятся к стандартному диапазону и распределению. Для начала разберемся, почему это вообще может понадобиться.

Многие модели МО плохо обучаются, когда признаки датасета лежат в разных диапазонах. Например, таким признакам, как возраст и доход, соответствуют разные числовые диапазоны. Кроме того, некоторые модели плохо обучаются, если признаки имеют смещенное распределение. Рассмотрим несколько методов масштабирования признаков, которые применяются на практике.

Нормализация (min-max масштабирование). Признаки масштабируются так, чтобы все значения принадлежали отрезку $[0, 1]$, по следующей формуле:

$$z = \frac{x - x_{\min}}{x_{\max} - x_{\min}}.$$

Обратите внимание, что распределение признака при нормализации не меняется. Чтобы изменить распределение и привести его к стандартному виду, применяется стандартизация.

Стандартизация (нормализация z-оценки). Распределение признака изменяется так, чтобы его среднее значение было равно 0, а стандартное отклонение было равно 1. Для стандартизации используется следующая формула:

$$z = \frac{x - \mu}{\sigma},$$

где μ — среднее значение признака, а σ — стандартное отклонение.

Логарифмическое масштабирование. Этот распространенный прием позволяет компенсировать асимметричное распределение признака, преобразовав его по следующей формуле:

$$z = \log(x).$$

В результате логарифмического масштабирования распределение данных может стать более симметричным, и тогда алгоритм оптимизации будет сходиться быстрее.

Дискретизация (*bucketing*)

Дискретизация — это преобразование непрерывного признака в категориальный. Например, вместо того чтобы представлять рост как непрерывный признак, можно разделить шкалу роста на интервалы, или бакеты (*buckets*), и сопоставить с каждым ростом интервал, к которому он относится. Это позволяет обучать модель на нескольких категориях вместо того, чтобы заставлять ее учитывать бесконечное количество возможных вариантов.

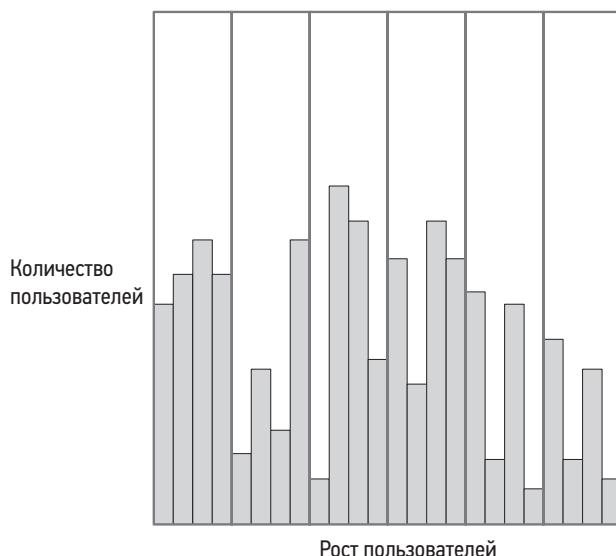


Рис. 1.12. Дискретизация роста пользователей по 6 интервалам (бакетам)

Дискретизацию можно применять и к дискретным признакам. Например, возраст пользователя — дискретный признак, но дискретизация позволяет сократить количество категорий, как показано в табл. 1.3.

Таблица 1.3. Дискретизация возраста

Номер интервала	Диапазон возрастов
1	0–9
2	10–19
3	20–39
4	40–59
5	60+

Кодирование категориальных признаков

В большинстве моделей МО все входные и выходные данные должны быть числовыми. Это значит, что категориальные признаки нужно закодировать в числовой форме, прежде чем передавать их модели. Существует три распространенных метода преобразования категориальных признаков в числовые представления: целочисленное кодирование, унитарное кодирование (one-hot coding) и обучение с эмбеддингами.

Целочисленное кодирование. Каждому уникальному категориальному значению ставится в соответствие целое число. Например, «Отлично» — 1, «Хорошо» — 2, «Плохо» — 3. Этот способ особенно полезен, если порядок целых значений соответствует естественному порядку категорий.



Рис. 1.13. Целочисленное кодирование

Однако если между значениями категориального признака нет порядка, то целочисленное кодирование — не лучший вариант. Проблему решает унитарное кодирование.

Унитарное (one-hot) кодирование. Для каждого уникального значения создается новый бинарный признак. Как показано на рис. 1.14, исходный признак (цвет) заменяется тремя новыми бинарными признаками (красный, зеленый и синий). Например, если точке данных соответствует красный цвет, он заменяется тройкой «1, 0, 0».



Рис. 1.14. Унитарное кодирование

Эмбеддинги (embeddings). Другой способ кодирования категориальных признаков основан на обучении с эмбеддингами. Эмбеддинг (вложение) — это отображение категориального признака на N -мерный вектор. При обучении с эмбеддингами подбирается N -мерный вектор для каждого уникального

значения, которое может принимать категориальный признак. Такой подход полезен, когда уникальных значений признака очень много. В этом случае унитарное кодирование недостаточно эффективно, потому что векторы получаются чересчур длинными. Другие примеры будут приведены в следующих главах.

Темы для обсуждения

Вот возможные темы, которые могут обсуждаться во время собеседования:

- **Доступность данных и их сбор.** Из каких источников происходят данные? Какие данные доступны и как вы будете их собирать? Каков объем данных? Насколько часто появляются новые данные?
- **Хранение данных.** Где хранятся данные? Они находятся в облаке или на пользовательских устройствах? В каком формате лучше хранить данные? Как хранить комбинированные данные (например, точку данных, в которую могут входить и изображения, и текст)?
- **Конструирование признаков.** Как преобразовать сырье данные в форму, удобную для моделей? Что делать с отсутствующими данными? Нужно ли конструировать признаки для этой задачи? Какими операциями преобразовывать необработанные данные в формат, пригодный для модели МО? Нужно ли нормализовывать признаки? Какие признаки следует сформировать из необработанных данных? Как объединять данные разных типов (например, текст, числа и графику)?
- **Конфиденциальность.** Насколько конфиденциальны доступные данные? Заинтересованы ли пользователи в конфиденциальности своих данных? Нужно ли анонимизировать пользовательские данные? Можно ли хранить данные пользователей на серверах или к данным можно обращаться только на устройствах пользователей?
- **Смещения.** Наблюдаются ли смещения в данных? Если да, то какие именно и как их исправить?

Разработка модели

Разработка модели заключается в том, чтобы выбрать подходящую модель МО и обучить ее для решения текущей задачи.

Выбор модели

На этом этапе мы выбираем алгоритм и архитектуру МО, которые лучше всего подойдут для текущей задачи моделирования. На практике типичный процесс выбора модели выглядит так:

- **Создать простое базовое решение (baseline).** Например, в системе рекомендации видео базовое решение может рекомендовать самые популярные видеоролики.
- **Поэкспериментировать с простыми моделями.** После того как создано базовое решение, имеет смысл попробовать алгоритмы МО, которые быстро обучаются, например логистическую регрессию.
- **Перейти на более сложные модели.** Если простые модели не обеспечивают удовлетворительных результатов, можно рассмотреть более сложные модели — например, глубокие нейронные сети.
- **Использовать ансамбли моделей, если нужны более точные предсказания.** Качество предсказаний можно улучшить, если использовать ансамбль из нескольких моделей вместо одной. Ансамбли создаются тремя основными способами: бэггинг [3], бустинг [4] и стекинг [5], которые будут рассмотрены в следующих главах.

На собеседовании важно исследовать разные варианты моделей, обсудить их достоинства и недостатки. Вот некоторые типичные примеры моделей:

- логистическая регрессия;
- линейная регрессия;
- деревья решений;
- градиентный бустинг деревьев решений и случайных лесов;
- метод опорных векторов;
- наивный байесовский классификатор;
- факторизационные машины (FM);
- нейронные сети.

Анализируя разные варианты, стоит кратко объяснить алгоритм и обсудить его ограничения. Например, логистическая регрессия часто хорошо подходит для обучения линейным задачам, но для сложных задач, возможно, стоит предпочтеть другую модель. Выбирая алгоритм МО, важно рассмотреть разные аспекты модели, например:

- сколько данных нужно для обучения модели;
- скорость обучения;
- какие выбрать гиперпараметры и методы их настройки;
- возможность непрерывного обучения;
- вычислительные требования. Более сложная модель может обеспечить более высокую точность, но также требует больше вычислительной мощности — например, графический процессор (GPU) вместо центрального (CPU);

- интерпретируемость модели [6]. Производительность более сложной модели может быть выше, но, возможно, ее результаты будут хуже интерпретироваться.

Не существует универсального алгоритма, который решал бы все проблемы. Эксперт хочет видеть, хорошо ли вы понимаете разные алгоритмы МО, их достоинства и недостатки, а также как вы выбираете модель в зависимости от требований и ограничений. Чтобы улучшить ваши навыки выбора модели, в эту книгу включены разнообразные примеры. Книга предполагает, что вы знакомы с распространенными алгоритмами МО. Если вам потребуется освежить их в памяти, обращайтесь к [7].

Обучение модели

После того как модель выбрана, наступает время ее обучать. На этом шаге возникает ряд вопросов, которые могут обсуждаться на собеседовании, например:

- построение датасета;
- выбор функции потерь;
- обучение с нуля или тонкая настройка;
- распределенное обучение.

Рассмотрим каждый из этих вопросов.

Построение датасета

На собеседовании обычно стоит обсудить, как строится датасет для обучения и оценки модели. Как показано на рис. 1.15, построение датасета состоит из 5 шагов.

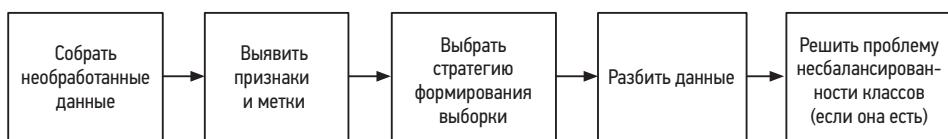


Рис. 1.15. Шаги построения датасета

Все шаги, кроме выявления признаков и меток, — это обобщенные операции, которые можно применять к любой задаче проектирования систем МО. В этой главе подробно рассматривается каждый из этих шагов, но дальше мы в основном будем сосредотачиваться на выявлении признаков и меток в каждой конкретной задаче.

Сбор необработанных данных

Этот шаг подробно рассматривается в разделе о подготовке данных, поэтому мы не будем повторяться.

Выявление признаков и меток

Вопрос о том, какие признаки стоит использовать, уже обсуждался ранее на шаге конструирования признаков, так что здесь мы сосредоточимся на создании меток для данных. Для этого можно применять ручную или автоматическую разметку.

Ручная разметка. Живые люди размечают данные вручную: например, специалист помечает, содержит ли пост неверную информацию. Ручная разметка обеспечивает высокую точность, потому что ее контролирует человек. Однако у нее есть ряд недостатков: это затратный и медленный процесс, который создает смещение, требует знания предметной области и создает угрозу для конфиденциальности данных.

Автоматическая разметка. Эталонные метки определяются автоматически, без участия человека. Чтобы лучше понять, что такое естественные метки, рассмотрим пример.

Допустим, мы проектируем систему МО, которая ранжирует новостную ленту по релевантности. Одно из возможных решений — обучить модель, которая принимает на вход пользователя и пост и возвращает вероятность того, что пользователь, просмотрев пост, поставит лайк. В этом случае обучающими данными будут пары <пользователь, пост>, каждой из которых соответствует метка 1, если пользователь поставил лайк, или 0, если не поставил. При таком подходе можно разметить обучающие данные, не полагаясь на человека.

На этом шаге важно четко сформулировать, откуда берутся метки для обучения и как выглядят обучающие данные.

Выбор стратегии формирования выборки

Не всегда целесообразно собирать абсолютно все данные: иногда лучше использовать выборку, чтобы сократить объем данных в системе. Выборки часто формируются с помощью таких стратегий, как выборка по принципу удобства, «снежный ком», районирование (стратифицирование), резервуарная выборка и выборка по значимости. За дополнительной информацией о разных методах формирования выборки можно обратиться к [8].

Разбиение данных

Разбиение данных — это деление набора данных на обучающий, оценочный (валидационный) и тестовый наборы. За дополнительной информацией о методах разбиения данных обращайтесь к [9].

Решение проблем несбалансированности классов

Набор данных со смещенным распределением меток классов называется несбалансированным. Класс, составляющий большую долю набора, называется мажоритарным, а класс, составляющий меньшую долю, — миноритарным.

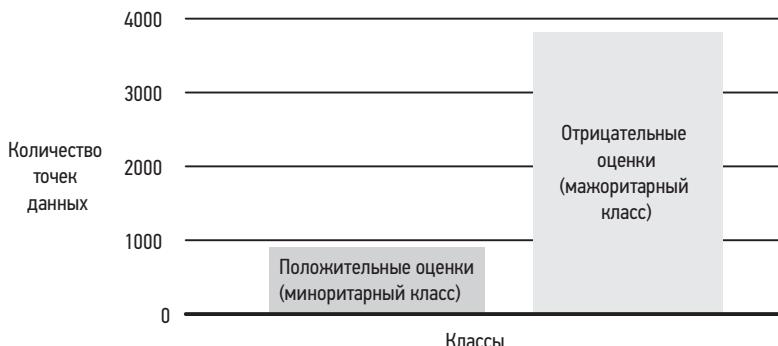


Рис. 1.16. Несбалансированный набор данных с двумя классами

Несбалансированный набор данных — серьезная проблема при обучении модели, потому что при этом ей может не хватить данных для обучения по миноритарному классу. Рассмотрим два популярных метода борьбы с этой проблемой: повторная выборка обучающих данных и изменение функции потерь.

Повторная выборка обучающих данных

С помощью повторной выборки можно регулировать соотношение объема классов, что помогает сбалансировать данные. Например, можно расширить выборку для миноритарного класса (рис. 1.17) или сократить выборку для мажоритарного класса (рис. 1.18).

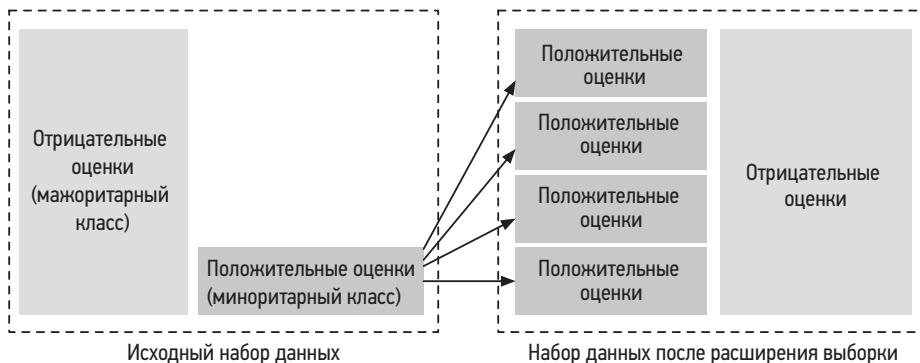


Рис. 1.17. Расширение выборки для миноритарного класса

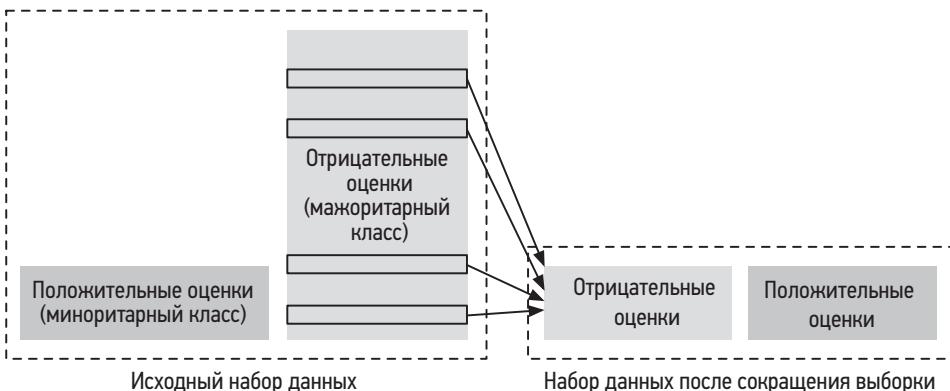


Рис. 1.18. Сокращение выборки для мажоритарного класса

Изменение функции потерь

Этот метод изменяет функцию потерь так, чтобы она стала устойчивее против несбалансированности классов. Общая идея заключается в том, чтобы придать больше веса точкам данных из миноритарного класса. Увеличенный вес в функции потерь сильнее наказывает модель за неверные предсказания относительно миноритарного класса, а это заставляет ее эффективнее анализировать миноритарные классы. Две часто используемые функции потерь, которые применяются для несбалансированных классов, — функция потерь, сбалансированная по классам [10], и фокальная функция потерь [11], [12].

Выбор функции потерь

После того как датасет построен, нужно выбрать подходящую функцию потерь для обучения модели. Функция потерь оценивает, насколько точно модель предсказывает ожидаемый результат, и позволяет алгоритму оптимизации обновлять параметры модели в процессе обучения, чтобы минимизировать потери.

Изобрести новую функцию потерь непросто. На собеседовании по МО обычно ожидается, что вы выберете функцию потерь из списка существующих функций в соответствии с тем, как вы сформулировали проблему. Иногда приходится вносить в функцию потерь небольшие изменения, чтобы адаптировать ее для конкретной задачи — примеры будут приведены в следующих главах.

Обучение с нуля или тонкая настройка (fine-tuning)

Одна из тем, которые стоит обсудить хотя бы вкратце, — выбор между обучением с нуля и тонкой настройкой, или адаптацией параметров (fine-tuning). Тонкая настройка означает, что мы продолжаем обучать модель на новых данных, внося

небольшие изменения в параметры, которые были получены на предыдущих итерациях обучения. Это одно из проектировочных решений, которые вам, возможно, стоит обсудить с экспертом.

Распределенное обучение

Важно, чтобы обучение можно было масштабировать, потому что со временем модели увеличиваются, и размер датасета тоже значительно возрастает. При распределенном обучении работа по обучению модели обычно распределяется между несколькими рабочими узлами, которые функционируют параллельно, чтобы ускорить обучение. Существует два основных типа распределенного обучения: параллелизм данных [13] и параллелизм модели [14].

Для некоторых задач распределенное обучение может оказаться необходимостью. В таких случаях очень важно обсудить эту тему с экспертом. Следует учитывать, что распределенное обучение — общая тема, которую можно рассматривать независимо от конкретной задачи.

Темы для обсуждения

Вот возможные темы, которые могут обсуждаться во время собеседования.

- **Выбор модели.** Какие модели МО подходят для задачи, каковы их достоинства и недостатки? При выборе модели учитывайте такие вопросы:
 - Сколько времени необходимо для обучения?
 - Сколько обучающих данных нужно для модели?
 - Сколько вычислительных ресурсов может понадобиться для модели?
 - Какова задержка обученной модели во время предсказаний?
 - Можно ли развернуть модель на пользовательском устройстве?
 - Интерпретируемость модели. Усложняя модель, можно улучшить ее производительность, но результаты могут хуже интерпретироваться.
 - Можно ли дообучать модель или обучение должно проводиться с нуля?
 - Сколько параметров у модели? Сколько памяти ей требуется?
 - Для нейронных сетей, возможно, придется обсудить типичные архитектуры и блоки — например, ResNet или архитектуры на базе Transformer. Также можно обсудить выбор гиперпараметров: количество скрытых слоев, количество нейронов, функции активации и т. д.
- **Метки набора данных.** Откуда возьмутся метки? Есть ли уже у данных аннотации, и если да, то насколько они хороши? Если доступны естественные метки, то как их получить и сколько времени для этого потребуется? Как получить обратную связь от пользователей?

- **Обучение модели:**

- Какую функцию потерь следует выбрать? (Например, модель перекрестной энтропии [15], среднеквадратичная ошибка (MSE) [16], средняя абсолютная ошибка (MAE) [17], функция потерь Хьюбера [18] и т. д.)
- Какую регуляризацию следует применить? (Например, $L1$ [19], $L2$ [19], энтропийная регуляризация [20], k -кратная перекрестная валидация [21] или исключение (dropout) [22].)
- Как устроен метод обратного распространения ошибки?
- Возможно, вам придется описать распространенные методы оптимизации [23] — такие, как стохастический градиентный спуск (SGD) [24], адаптивный градиентный метод (AdaGrad) [25], Momentum [26] и RMSProp [27].
- Какую функцию активации следует использовать и почему? (Например, экспоненциальную линейную (ELU) [28], линейный выпрямитель (ReLU) [29], гиперболический тангенс [30], логистическую (сигмоиду) [31].)
- Что делать с несбалансированным набором данных?
- Как достичь компромисса между смещением и дисперсией?
- Каковы возможные причины переобучения и недообучения? Что с ними делать?
- **Непрерывное обучение.** Нужно ли дообучать модель с каждой новой точкой данных? Следует ли персонализировать модель для каждого пользователя? Как часто будет дообучаться модель? Одни модели нужно дообучать ежедневно или еженедельно, а другие — раз в месяц или в год.

Оценка

Следующий шаг после разработки модели — оценка: на этом этапе используются разные метрики, чтобы охарактеризовать производительность модели МО. В этом разделе рассматриваются два способа оценки: автономный и оперативный.

Автономная (offline) оценка

Это оценка производительности моделей МО на этапе их разработки. Чтобы оценить модель, мы обычно сначала делаем предсказания на валидационном наборе данных, а затем с помощью различных формальных метрик выясняем, насколько эти предсказания близки к эталонным. В табл. 1.4 приведены некоторые популярные метрики для разных задач.

Таблица 1.4. Популярные метрики для автономной оценки качества моделей

Задача	Автономные метрики
Классификация	Точность (precision), полнота (recall), F -мера, доля верных результатов (accuracy), ROC-AUC, PR-AUC, матрица ошибок
Регрессия	MSE, MAE, RMSE
Ранжирование	Точность на k элементах (precision@ k), полнота на k элементах (recall@ k), среднеобратный ранг (MRR), mAP, nDCG
Генерирование изображений	Расстояние Фреше (FID) [32], «мера внедрения» (Inception Score) [33]
Обработка естественного языка	BLEU [34], METEOR [35], ROUGE [36], CIDEr [37], SPICE [38]

На собеседовании важно идентифицировать метрики, которые подходят для автономной оценки. Они зависят от текущей задачи и от того, как мы ее сформулировали. Например, если вы решаете задачу ранжирования, то, возможно, стоит обсудить метрики ранжирования и их ограничения.

Оперативная (online) оценка

Это оценка качества моделей на этапе эксплуатации после развертывания. Метрики, которые применяются для оперативной оценки, обычно связаны с бизнес-целями. В табл. 1.5 представлены метрики для разных задач.

Таблица 1.5. Метрики для оперативной оценки качества моделей

Задача	Оперативные метрики
Предсказание кликов на рекламе	Кликабельность (CTR), повышение выручки и т. д.
Выявление вредоносного контента	Степень распространения, количество обжалований и т. д.
Рекомендации видео	Кликабельность (CTR), общее время просмотра, количество завершенных просмотров и т. д.
Рекомендации друзей	Количество запросов, отправляемых и принимаемых за день, и т. д.

На практике компании обычно отслеживают много оперативных метрик. На собеседовании нужно выбрать самые важные из них, чтобы измерить эффект от системы МО. В отличие от автономных метрик, оперативные метрики часто выбираются субъективно в зависимости от видения владельцев продукта и ключевых участников со стороны бизнеса.

На этом шаге эксперт оценивает ваше деловое чутье и хочет получить представление о том, как вы размышляете и почему выбираете те или иные метрики.

Темы для обсуждения

Вот возможные темы, которые могут обсуждаться в контексте оценки моделей.

- **Оперативные метрики.** Какие метрики позволяют оценивать эффективность работающей системы МО? Как эти метрики соотносятся с бизнес-целями?
- **Автономные метрики.** Какие автономные метрики хорошо подходят, чтобы оценивать предсказания модели на этапе разработки?
- **Объективность и смещение.** Угрожает ли модели смещение по таким признакам, как возраст, пол, раса и т. д.? Как это исправить? Что произойдет, если доступ к вашей системе получит кто-то с недобрыми намерениями?

Развертывание и эксплуатация

Естественный следующий шаг после выбора подходящих метрик для оперативной и автономной оценки качества — ввод модели в эксплуатацию для обслуживания миллионов пользователей. Здесь стоит уделить внимание, в частности, таким вопросам:

- развертывание в облаке или на устройстве;
- сжатие модели;
- тестирование в режиме эксплуатации;
- пайплайн предсказаний.

Рассмотрим каждый из этих вопросов.

Развертывание в облаке или на устройстве

Развертывание модели в облаке отличается от развертывания на пользовательском устройстве. В табл. 1.6 приведена сводка основных различий.

Таблица 1.6. Достоинства и недостатки развертывания на устройстве и в облаке

	Облако	Устройство
Простота	✓ Простота развертывания и управления с помощью облачных служб	✗ Развертывание модели на устройстве может быть сложным
Затраты	✗ Затраты на облачные ресурсы могут быть высокими	✓ Вычисления выполняются на устройстве, нет затрат на аренду облачных ресурсов
Сетевая задержка	✗ Есть сетевая задержка	✓ Нет сетевой задержки

Таблица 1.6 (окончание)

	Облако	Устройство
Задержка ответа	✓ Обычно ответ выдается быстрее благодаря использованию более мощного оборудования	✗ Модели МО работают медленнее
Аппаратные ограничения	✓ Меньше ограничений	✗ Больше ограничений по памяти, потреблению аккумулятора и т. д.
Конфиденциальность	✗ Конфиденциальность ослабляется, потому что пользовательские данные переносятся в облако	✓ Более высокий уровень конфиденциальности, потому что данные не покидают устройство
Зависимость от подключения к интернету	✗ Чтобы отправлять и получать данные из облачных служб, необходимо подключение к интернету	✓ Подключение к интернету не требуется

Сжатие модели

Сжатие модели состоит в том, чтобы сделать ее меньше: это необходимо, чтобы сократить задержку ответа и сэкономить пространство. Для сжатия моделей обычно применяются три метода:

- **дистилляция знаний** (distillation) — обучение меньшей модели (ученика), которая имитирует большую модель (учителя);
- **прореживание** (pruning) — процесс, который находит наименее полезные параметры и обнуляет их. От этого модели становятся более разреженными, что позволяет хранить их эффективнее;
- **квантование** (quantization) — параметры модели часто представляются 32-разрядными числами с плавающей точкой. Квантование сокращает количество разрядов в представлении параметров, что уменьшает размер модели. Квантование может происходить во время обучения или после него [39].

За дополнительной информацией о сжатии моделей обращайтесь к [40].

Тестирование при эксплуатации

Единственный способ убедиться, что модель показывает хорошие результаты при эксплуатации, — протестировать ее на реальном трафике. Среди популярных способов тестирования моделей — теневое развертывание (shadow deployment) [41], А/В-тестирование [42], канареечные релизы (тестирование модели на

ограниченной аудитории) [43], эксперименты с перемежением [44], многорукие бандиты [45] и т. д.

Чтобы продемонстрировать, что вы понимаете, как проводить тестирование в условиях эксплуатации, желательно упомянуть на собеседовании хотя бы один из этих методов. Кратко рассмотрим теневое развертывание и А/В-тестирование.

Теневое развертывание

В этом варианте новая модель развертывается параллельно с существующей. Каждый входной запрос передается обеим моделям, но пользователь получает предсказания только от существующей модели.

При теневом развертывании мы сводим к минимуму риск ненадежных предсказаний, пока новая модель не будет тщательно протестирована. Однако это довольно затратный подход, потому что количество предсказаний удваивается.

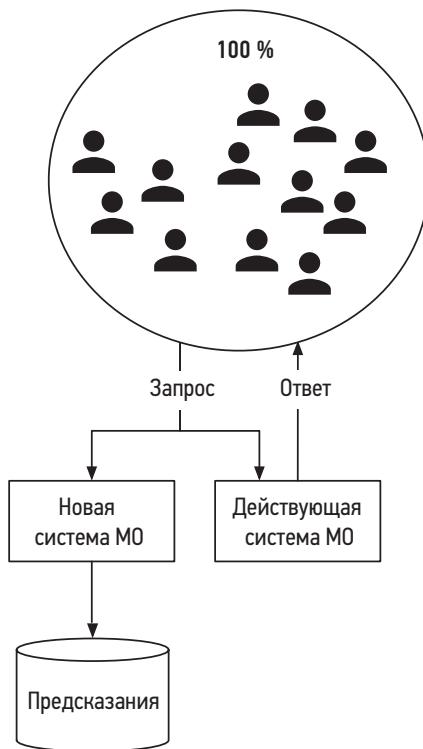


Рис. 1.19. Теневое развертывание

A/B-тестирование

В таком формате тестирования новая модель развертывается параллельно с существующей. Часть трафика передается новой модели, а остальные запросы — существующей (рис. 1.20).

Чтобы правильно провести А/В-тестирование, важно учесть два фактора. Во-первых, трафик, который передается каждой модели, должен быть случайным. Во-вторых, чтобы считать результаты А/В-тестов состоятельными, тесты нужно проводить на достаточном объеме данных.

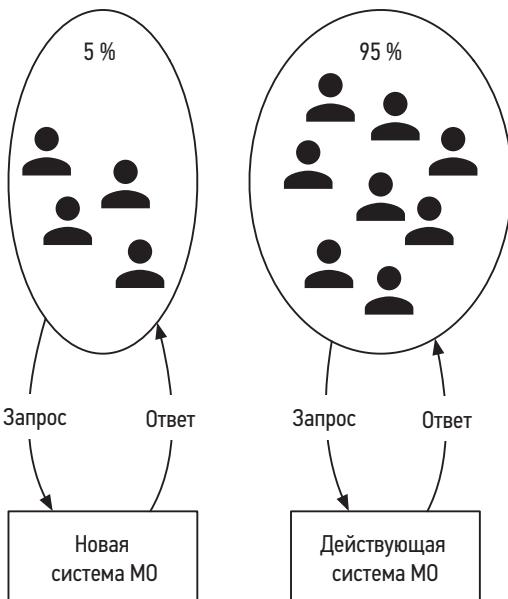


Рис. 1.20. А/В-тестирование

Пайплайн предсказаний

Чтобы обрабатывать запросы на стадии реальной эксплуатации, необходим пайплайн предсказаний. Одно из важных проектировочных решений — будут ли предсказания вырабатываться в пакетном или оперативном режиме.

Пакетные предсказания (batch prediction). Модель выдает предсказания с некоторой периодичностью. Поскольку предсказания вычисляются заранее, вам не нужно беспокоиться о том, сколько времени потребуется модели, чтобы их сгенерировать после того, как они вычислены.

Однако у пакетного предсказания есть два серьезных недостатка. Во-первых, модель не так быстро реагирует на изменяющиеся предпочтения пользователей.

Во-вторых, пакетные предсказания возможны, только если заранее известно, какие результаты нужно предварительно вычислять. Например, в системе машинного перевода невозможно генерировать переводы заранее, потому что они полностью зависят от пользовательского ввода.

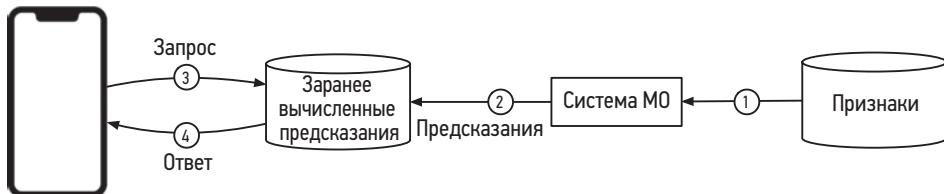


Рис. 1.21. Процесс пакетного предсказания

Оперативные предсказания (online prediction). В этом режиме новые предсказания генерируются и возвращаются сразу же после поступления запросов. Главный недостаток оперативных предсказаний в том, что модель может вырабатывать их слишком долго.

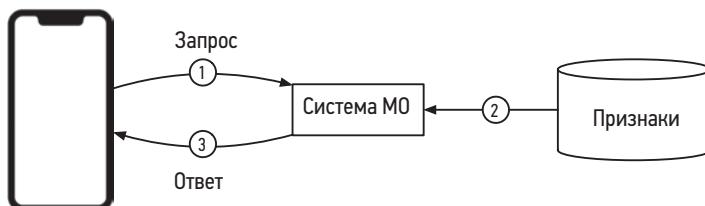


Рис. 1.22. Процесс оперативного предсказания

Выбор между пакетными и оперативными предсказаниями в основном определяется требованиями к продукту. Оперативные предсказания обычно предпочтительны в ситуациях, когда заранее неизвестно, какие результаты понадобится вычислять. Пакетные предсказания идеально подходят, когда система обрабатывает большие объемы данных, а результаты не нужны в реальном времени.

Как упоминалось ранее, разработка систем МО не сводится лишь к построению модели МО. В ходе собеседования часто стоит предложить общую структуру системы МО, чтобы продемонстрировать глубокое понимание того, как различные компоненты работают как единое целое. Для экспертов это нередко становится важнейшим сигналом.

На рис. 1.23 изображена возможная архитектура системы МО для персонализированной ленты новостей. Мы рассмотрим ее подробнее в главе 10.

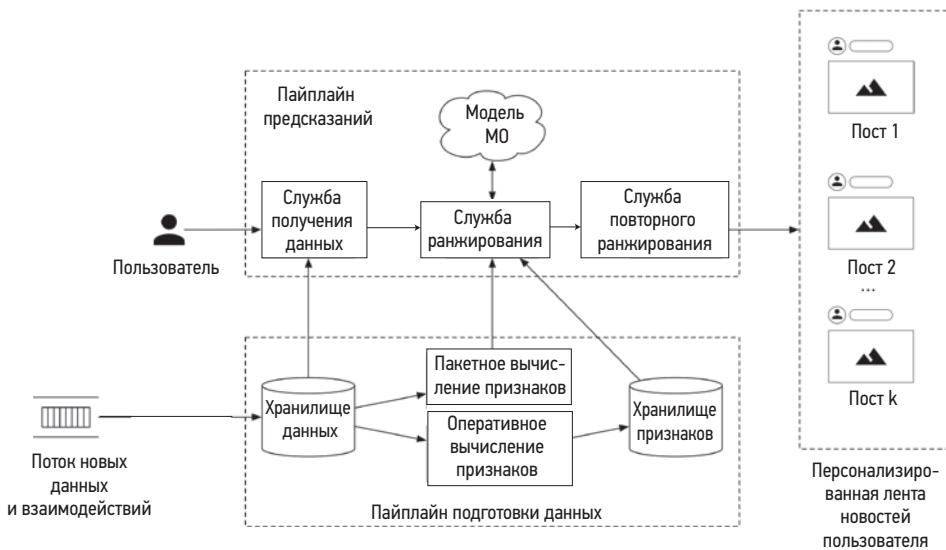


Рис. 1.23. Архитектура системы МО для персонализированной ленты новостей

Темы для обсуждения

- Где должны вычисляться результаты модели — в облаке или на пользовательском устройстве?
- Нужно ли сжимать модель? Какие методы сжатия часто применяются на практике?
- Какой тип предсказаний лучше подойдет — оперативные или пакетные? Каковы достоинства и недостатки каждого варианта?
- Можно ли обращаться к признакам в реальном времени? Какие проблемы при этом возникают?
- Как протестировать развернутую модель в условиях реальной эксплуатации?
- Система МО состоит из различных компонентов, которые совместно обслуживают запросы. Какие функции выполняет каждый компонент в предложенном решении?
- Какие технологии следует применить, чтобы обслуживание было быстрым и масштабируемым?

Мониторинг

Задача мониторинга — отслеживать, измерять и регистрировать различные метрики. В действующих системах МО могут происходить сбои по разным при-

чинам. Мониторинг помогает выявлять сбои по мере их возникновения, чтобы проблемы можно было исправить как можно быстрее.

Мониторинг — важная тема для обсуждения на собеседовании по проектированию систем МО. Вот два основных вопроса, которые имеет смысл обсудить:

- почему в действующей системе происходят сбои;
- какие показатели следует отслеживать.

Почему в действующей системе происходят сбои

Существует немало причин, по которым в системе МО может произойти сбой после того, как она введена в эксплуатацию. Одна из самых распространенных причин — смещение распределения данных.

Распределение данных смещается, когда данные, которые передаются модели при эксплуатации, отличаются от данных, которые использовались для обучения. В примере на рис. 1.24 обучающие данные содержали фронтальные изображения чашек, а действующая модель МО получает изображения чашек под другим углом.

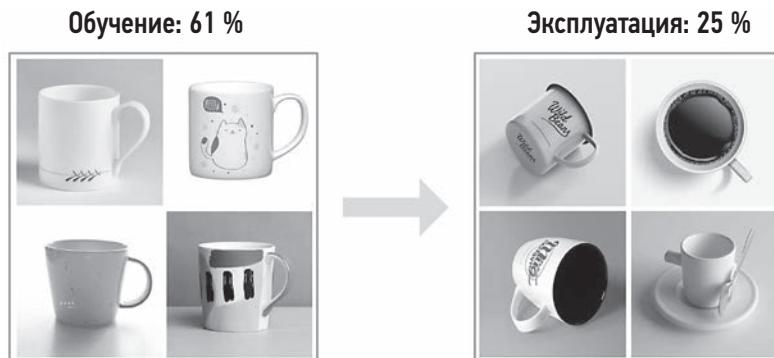


Рис. 1.24. Смещение распределения данных

В реальном мире распределения данных постоянно меняются. Иначе говоря, данные, которые использовались для обучения, обычно становятся менее актуальными с течением времени, отчего модель устаревает и ее производительность снижается. А значит, нужно постоянно следить за системой, чтобы выявлять такие проблемы. Чтобы бороться со смещением распределений данных, используются два основных метода.

- Обучение на больших датасетах. Большой обучающий набор данных позволяет модели охватить достаточно полное распределение, так что данные,

которые поступают при эксплуатации, с большой вероятностью будут относиться к изученному распределению.

- Регулярное дообучение модели с использованием размеченных данных из нового распределения.

Какие показатели нужно отслеживать

Это весьма обширная тема, и здесь мы сосредоточимся на методах мониторинга после ввода в эксплуатацию. Наша цель — обнаруживать сбои и выявлять смещения в системе МО. В широком смысле методы мониторинга в системах МО можно разделить на две категории: операционные и относящиеся к специфике МО.

Операционные метрики проверяют, что система штатно функционирует. К этой категории относится среднее время обслуживания, пропускная способность, количество запросов на предсказания, степень загрузки центрального/графического процессора и т. д.

Специфические метрики МО

- Мониторинг ввода/вывода. Качество модели ограничено качеством данных, с которыми она работает, поэтому крайне важно следить за входными и выходными данными.
- Смещение. Мониторинг входных и выходных данных позволяет обнаружить изменения в их распределении.
- Точность модели. Например, может потребоваться, чтобы точность лежала в определенном диапазоне.
- Версии модели. Отслеживается, какая версия модели развернута.

Инфраструктура

Инфраструктура — это основа для обучения, развертывания и сопровождения систем МО.

На многих собеседованиях по МО вам не будут задавать вопросы, относящиеся к инфраструктуре. Однако некоторые должности в сфере МО — например, DevOps и MLOps — могут требовать знания инфраструктуры. Таким образом, важно прояснить ожидания эксперта по этой теме.

Инфраструктура — очень обширная тема, которую невозможно кратко охарактеризовать в нескольких строках. Чтобы больше узнать об инфраструктуре МО, обращайтесь к [46], [47] и [48].

Итоги

В этой главе мы предложили схему для собеседования по проектированию систем МО. Хотя многие темы, которые здесь рассматривались, связаны с конкретными задачами, другие носят общий характер и могут применяться в широком диапазоне задач. Чтобы избежать повторений, в этой книге мы сосредоточимся только на уникальных темах, которые относятся к конкретным задачам. Наоборот, вопросы развертывания, мониторинга и инфраструктуры часто одинаковы в разных задачах. Поэтому в последующих главах мы не будем повторять общие темы, хотя их обычно стоит затронуть во время собеседования.

Наконец, ни один специалист не может быть экспертом по всем аспектам жизненного цикла МО. Одни инженеры специализируются на развертывании и эксплуатации, а другие — на разработке модели. Одни компании не придают значения инфраструктуре, а другие могут уделять инфраструктуре и мониторингу особое внимание. Должности, связанные с data science, обычно требуют квалификации в работе с данными, а должности в сфере прикладного МО в большей степени ориентированы на разработку и практическое внедрение моделей. В зависимости от вакансии и от предпочтений эксперта некоторые шаги могут обсуждаться подробнее, а некоторые рассматриваются кратко или вовсе пропускаются. В общем случае кандидату следует стремиться направлять ход обсуждения, но быть готовым адаптироваться к интересам эксперта, если тот задаст вопрос.

Разобравшись с основами, можно перейти к вопросам, которые наиболее часто встречаются на собеседованиях по проектированию систем МО.

Ссылки

- [1] Хранилище данных. <https://cloud.google.com/learn/what-is-a-data-warehouse>
- [2] Структурированные и неструктурированные данные. <https://signal.onepointltd.com/post/102gjab/machine-learning-libraries-for-tabular-data-problems>
- [3] Бэггинг в ансамблевом обучении. https://en.wikipedia.org/wiki/Bootstrap_aggregating
- [4] Бустинг в ансамблевом обучении. <https://aws.amazon.com/what-is/boosting/>
- [5] Стекинг в ансамблевом обучении. <https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/>
- [6] Интерпретируемость в МО. <https://blog.ml.cmu.edu/2020/08/31/6-interpretability/>
- [7] Традиционные алгоритмы МО. <https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/>
- [8] Стратегии выборки данных. <https://www.scribbr.com/methodology/sampling-methods/>
- [9] Методы разбиения данных. <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/>

- [10] Функция потерь, сбалансированная по классам. <https://arxiv.org/pdf/1901.05555.pdf>
- [11] Фокальная функция потерь для плотных выборок. <https://arxiv.org/pdf/1708.02002.pdf>
- [12] Фокальная функция потерь. <https://medium.com/swlh/focal-loss-an-efficient-way-of-handling-class-imbalance-4855ae1db4cb>.
- [13] Параллелизм данных. <https://www.telesens.co/2017/12/25/understanding-data-parallelism-in-machine-learning/>
- [14] Параллелизм модели. <https://docs.aws.amazon.com/sagemaker/latest/dg/model-parallel-intro.html>
- [15] Перекрестная энтропия. https://en.wikipedia.org/wiki/Cross_entropy
- [16] Среднеквадратичная ошибка. https://en.wikipedia.org/wiki/Mean_squared_error
- [17] Средняя абсолютная ошибка. https://en.wikipedia.org/wiki/Mean_absolute_error
- [18] Функция потерь Хьюбера. https://en.wikipedia.org/wiki/Huber_loss
- [19] Регуляризация L1 и L2. <https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning>
- [20] Энтропийная регуляризация. <https://paperswithcode.com/method/entropy-regularization>
- [21] К-кратная перекрестная валидация. [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics))
- [22] Исключение (dropout). <https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>
- [23] Обзор алгоритмов градиентного спуска. <https://ruder.io/optimizing-gradient-descent/>
- [24] Стохастический градиентный спуск. https://en.wikipedia.org/wiki/Stochastic_gradient_descent
- [25] Алгоритм оптимизации AdaGrad. <https://optimization.cbe.cornell.edu/index.php?title=AdaGrad>
- [26] Алгоритм оптимизации Momentum. <https://optimization.cbe.cornell.edu/index.php?title=Momentum>
- [27] Алгоритм оптимизации RMSProp. <https://optimization.cbe.cornell.edu/index.php?title=RMSProp>
- [28] Функция активации ELU. https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#elu
- [29] Функция активации ReLU. https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#relu
- [30] Функция активации «гиперболический тангенс». https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#tanh
- [31] Логистическая функция активации (сигмоида). https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#softmax
- [32] Метрика FID. https://en.wikipedia.org/wiki/Fr%C3%A9chet_inception_distance
- [33] Метрика внедрения (Inception). https://en.wikipedia.org/wiki/Inception_score
- [34] Метрика BLEU. <https://en.wikipedia.org/wiki/BLEU>
- [35] Метрика METEOR. <https://en.wikipedia.org/wiki/METEOR>
- [36] Метрика ROUGE. [https://en.wikipedia.org/wiki/ROUGE_\(metric\)](https://en.wikipedia.org/wiki/ROUGE_(metric))
- [37] Метрика CIDEr. <https://arxiv.org/pdf/1411.5726.pdf>
- [38] Метрика SPICE. <https://arxiv.org/pdf/1607.08822.pdf>

- [39] Обучение с поддержкой квантования. <https://pytorch.org/docs/stable/quantization.html>
- [40] Исследование сжатия моделей. <https://arxiv.org/pdf/1710.09282.pdf>
- [41] Теневое развертывание. <https://christophergs.com/machine%20learning/2019/03/30/deploying-machine-learning-applications-in-shadow-mode/>
- [42] А/В-тестирование. https://en.wikipedia.org/wiki/A/B_testing
- [43] Канареечный релиз. <https://blog.getambassador.io/cloud-native-patterns-canary-release-1cb8f82d371a>
- [44] Эксперименты с перемежением. <https://netflixtechblog.com/interleaving-in-online-experiments-at-netflix-a04ee392ec55>
- [45] Многорукие бандиты. <https://vwo.com/blog/multi-armed-bandit-algorithm/>
- [46] Инфраструктура МО. <https://www.run.ai/guides/machine-learning-engineering/machine-learning-infrastructure>
- [47] Интерпретируемость в МО. <https://fullstackdeeplearning.com/spring2021/lecture-6/>
- [48] Чип Хуэн (Chip Huyen). «Designing Machine Learning Systems: An Iterative Process for Production-Ready Application» («Проектирование систем МО: Итеративный процесс для приложений, готовых к эксплуатации»). O'Reilly Media, Inc., 2022.

2

СИСТЕМА ВИЗУАЛЬНОГО ПОИСКА

Система визуального поиска помогает пользователям находить изображения, которые визуально похожи на выбранное изображение. В этой главе мы спроектируем систему визуального поиска, похожую на ту, которую использует Pinterest [1], [2].

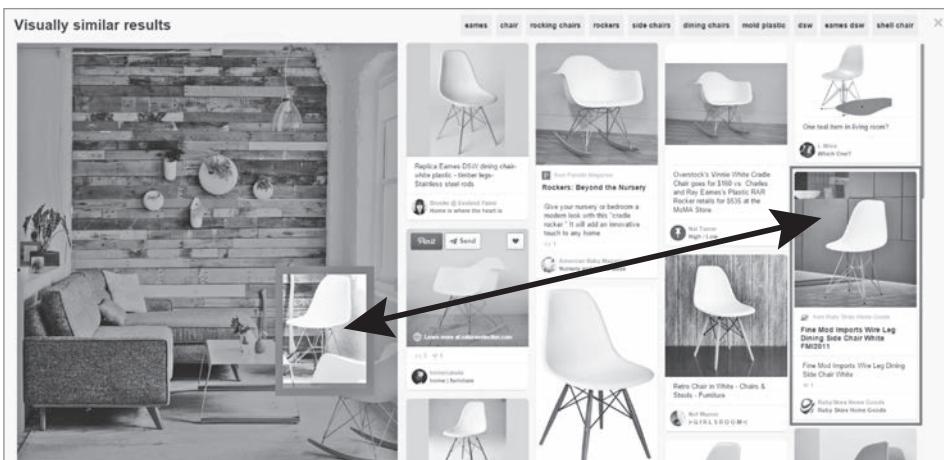


Рис. 2.1. Найденные изображения, визуально сходные с выбранным фрагментом

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: Нужно ли ранжировать результаты от более похожих к менее похожим?

Эксперт: Изображения в начале подборки должны быть более похожими на изображение из запроса.

Соискатель: Должна ли система также поддерживать видео?

Эксперт: Ограничимся только изображениями.

Соискатель: Такая платформа, как Pinterest, позволяет выделить фрагмент картинки и найти похожие изображения. Нам нужно поддерживать эту функциональность?

Эксперт: Да.

Соискатель: Надо ли персонализировать результаты поиска под конкретного пользователя?

Эксперт: Для простоты не будем отвлекаться на персонализацию. Для одного и того же запроса должны возвращаться одни и те же результаты независимо от того, кто проводит поиск.

Соискатель: Может ли модель использовать метаданные исходного изображения — например, теги?

Эксперт: На практике модель будет использовать метаданные. Но для простоты допустим, что в запросе они не используются, а при поиске учитываются только пиксели изображения.

Соискатель: Может ли пользователь выполнять другие действия — сохранить, поделиться или поставить лайк? Эти действия могут помочь разметить обучающие данные.

Эксперт: Верно замечено. Для простоты будем считать, что поддерживаются только клики по изображению.

Соискатель: Нужно ли модерировать изображения?

Эксперт: Обеспечивать безопасность платформы важно, но модерация контента выходит за рамки задачи.

Соискатель: Обучающие данные можно конструировать во время эксплуатации и размечать в зависимости от действий пользователей. Годится ли такой механизм?

Эксперт: Да, звучит разумно.

Соискатель: Насколько быстрым должен быть поиск? Если предположить, что на платформе 100–200 миллиардов изображений, система должна находить похожие изображения достаточно быстро. Можно ли считать это разумным предположением?

Эксперт: Да, можно.

Резюмируем описание проблемы. Нужно спроектировать систему визуального поиска. Система находит изображения, похожие на то, которое пользователь предоставил в запросе, ранжирует их по сходству с исходным изображением, а затем выводит пользователю. Платформа работает только с графикой; запросы в виде видео или текста не поддерживаются. Для простоты персонализация не требуется.

Формулировка проблемы в виде задачи МО

В этом разделе мы четко определим цель МО и сформулируем проблему визуального поиска в виде задачи МО.

Определение цели МО

Чтобы решить проблему с помощью модели МО, необходимо четко определить цель МО. Потенциальная цель — с высокой степенью точности подбирать изображения, визуально сходные с тем, которое предоставил пользователь в запросе.

Определение входных и выходных данных системы

На вход системы визуального поиска подается исходное изображение, которое предоставил пользователь. Система выводит изображения, визуально похожие на исходное, а результаты ранжируются по степени сходства. На рис. 2.2 изображен ввод и вывод системы визуального поиска.

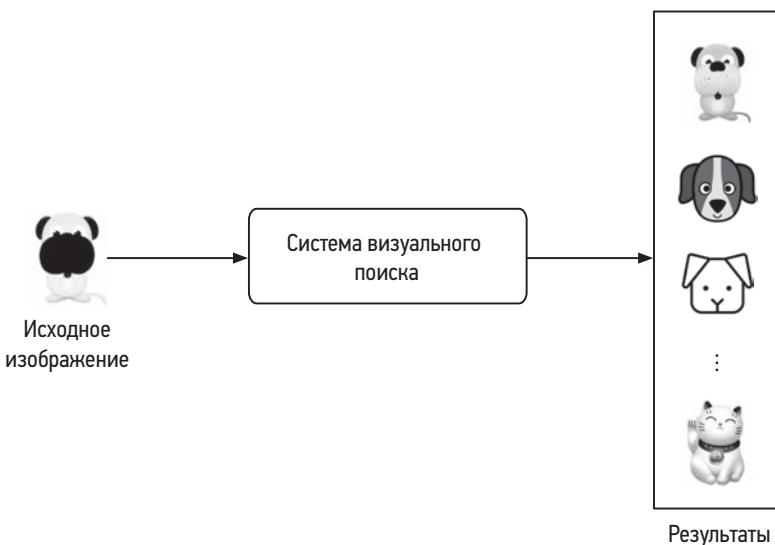


Рис. 2.2. Ввод и вывод системы визуального поиска

Выбор подходящей категории МО

Выходные данные модели — ранжированный набор изображений, похожих на исходное. Значит, систему визуального поиска можно описывать через задачи ранжирования. В общем случае цель ранжирования — отсортировать

совокупность элементов (изображений, сайтов, товаров и т. д.) по их релевантности запросу так, чтобы более релевантные элементы располагались выше в результатах поиска. Многие приложения МО — такие, как рекомендательные системы, поисковые системы, службы поиска документов и системы сетевой рекламы, — можно описывать на основе задач ранжирования. В этой главе мы воспользуемся популярным методом, который называется обучением представлений. Рассмотрим его подробнее.

Обучение представлений (representation learning). В обучении представлений [3] модель обучается преобразовывать входные данные (например, изображения) в специальные представления, называемые эмбеддингами (embeddings). Иначе говоря, модель отображает входные изображения на точки N -мерного пространства, которое называется пространством эмбеддингов. Модель обучается так, чтобы похожим изображениям соответствовали эмбеддинги, которые находятся в этом пространстве поблизости друг от друга. На рис. 2.3 показано, как два похожих изображения отображаются на две точки, находящиеся в непосредственной близости друг от друга в пространстве эмбеддингов. Для демонстрационных целей эмбеддинги (обозначенные X) изображены в двухмерном пространстве. На самом деле это пространство является N -мерным, где N — размерность вектора эмбеддинга.

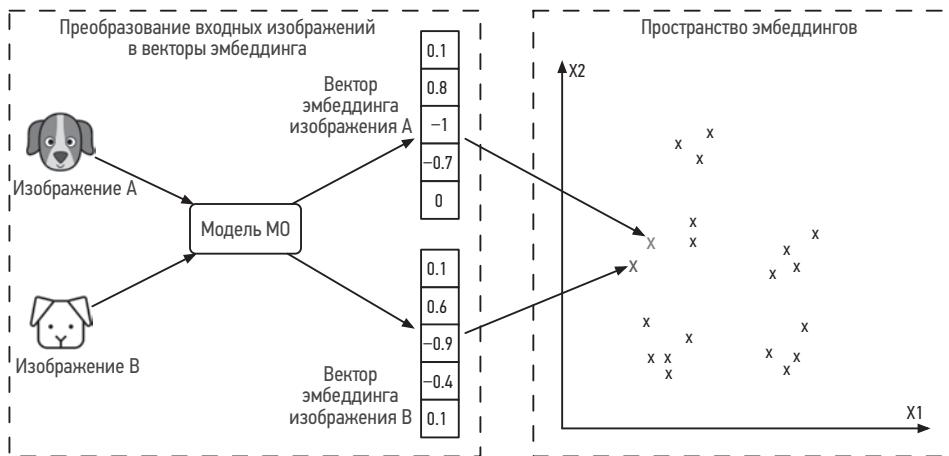


Рис. 2.3. Похожие изображения в пространстве эмбеддингов

Как ранжировать изображения с помощью обучения представлений?

Сначала входные изображения преобразуются в векторы эмбеддингов. Затем вычисляется показатель сходства между исходным изображением и другими изображениями на платформе, для чего измеряются расстояния между ними в пространстве эмбеддингов. Изображения ранжируются по показателю сходства, как показано на рис. 2.4.

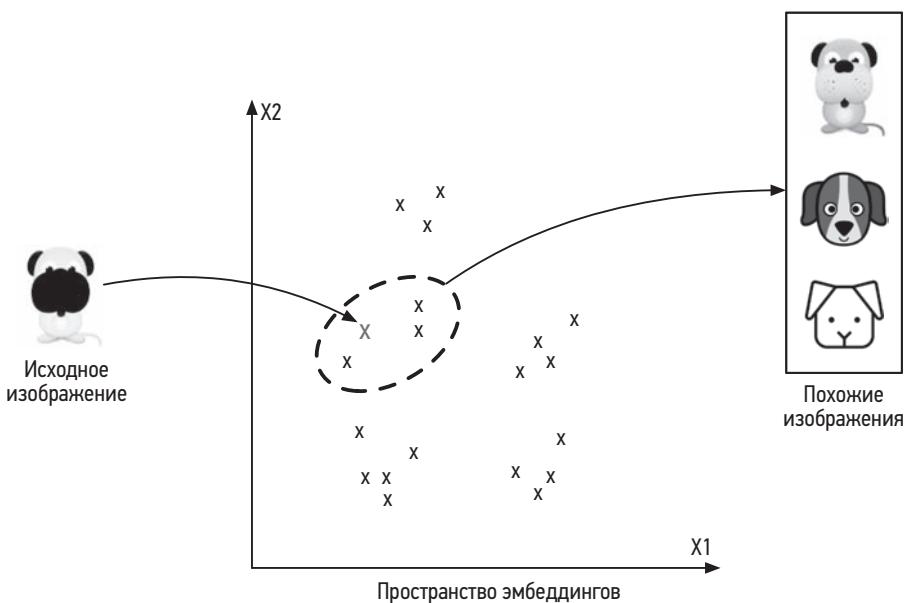


Рис. 2.4. Первые три изображения, наиболее похожие на исходное

На этом месте у вас может возникнуть много вопросов: как сделать, чтобы похожие изображения располагались поблизости друг от друга в пространстве эмбеддингов? Как определить метрику сходства? Как обучать такие модели? Мы рассмотрим эти вопросы подробнее в разделе, посвященном разработке модели.

Подготовка данных

Инженерия данных

Важно не только разбираться в инженерии данных как таковой, но и понимать, какие данные вам доступны. Поскольку система визуального поиска имеет дело прежде всего с пользователями и изображениями, в этом случае доступны такие данные:

- изображения;
- пользователи;
- взаимодействия пользователей с изображениями.

Изображения

Создатели загружают изображения, а система сохраняет их вместе с метаданными — такими, как идентификатор (ID) владельца, контекстная информация

(например, время загрузки), теги и т. д. В табл. 2.1 приведен упрощенный пример метаданных изображения.

Таблица 2.1. Метаданные изображения

ID изображения	ID владельца	Время загрузки	Теги, назначенные вручную
1	8	1658451341	Зебра
2	5	1658451841	Паста, Еда, Кухня
3	19	1658821820	Дети, Семья, Праздник

Пользователи

В данных пользователей хранятся демографические атрибуты: возраст, пол и т. д. Пример данных пользователей приведен в табл. 2.2.

Таблица 2.2. Данные пользователей

ID пользователя	Имя пользователя	Возраст	Пол	Город	Страна	Электронная почта
1	johnduo	26	М	Сан-Хосе	США	john@gmail.com
2	hsieh2008	49	М	Париж	Франция	hsieh@gmail.com
3	alexish	16	Ж	Рио-де-Жанейро	Бразилия	alexh@yahoo.com

Взаимодействия пользователей с изображениями

Данные взаимодействий отражают разные типы пользовательских взаимодействий. Согласно собранным требованиям, основные типы взаимодействий — показы и клики. В табл. 2.3 приведена соответствующая сводка.

Таблица 2.3. Данные взаимодействий пользователей с изображениями

ID пользователя	ID исходного изображения	ID выводимого изображения	Позиция в отображаемом списке	Тип взаимодействия	Геопозиция (широта, долгота)	Временная метка
8	2	6	1	Клик	38.8951 -77.0364	1658450539
6	3	9	2	Клик	38.8951 -77.0364	1658451341
91	5	1	2	Показ	41.9241 -89.0389	1658451365

Конструирование признаков

Здесь предполагается, что вы предложите, как сконструировать эффективные признаки и подготовить их в качестве входных данных модели. Обычно это зависит от того, как сформулирована задача и какие входные данные принимает модель. В предыдущем разделе «Формулировка проблемы как задачи МО» система визуального поиска была описана в терминах задачи ранжирования, которая решается с помощью обучения представлениям. В частности, мы решили, что модель принимает на вход изображение. Прежде чем его можно будет передать модели, изображение следует предварительно обработать. Вот некоторые распространенные операции предварительной обработки изображений.

- **Изменение размеров.** Для модели обычно требуются изображения фиксированного размера (например, 224×224).
- **Масштабирование.** Значения пикселей масштабируются в диапазоне от 0 до 1.
- **Нормализация Z-оценки.** Значения пикселей масштабируются так, чтобы их среднее было равно 0, а стандартное отклонение — 1.
- **Единая цветовая модель.** Изображения приводятся к единой цветовой модели (например, RGB или CMYK).

Разработка модели

Выбор модели

Мы выбрали нейронные сети по следующим причинам:

- нейронные сети хорошо подходят для работы с неструктуризованными данными (такими, как изображения и текст);
- в отличие от многих традиционных методов МО, нейронные сети могут создавать эмбеддинги, которые нужны для обучения представлений.

Какую архитектуру нейронной сети использовать? Очень важно, чтобы архитектура могла работать с изображениями. Архитектуры на основе сверточных нейронных сетей (CNN) — такие, как ResNet [4], или более поздние архитектуры на базе Transformer [5] — такие, как ViT [6], хорошо справляются с графическими входными данными. На рис. 2.5 представлена упрощенная архитектура модели, которая преобразует входное изображение в векторное представление. Количество сверточных слоев, количество нейронов в полно связанных слоях и размер векторного представления являются гиперпараметрами, которые обычно подбираются опытным путем.

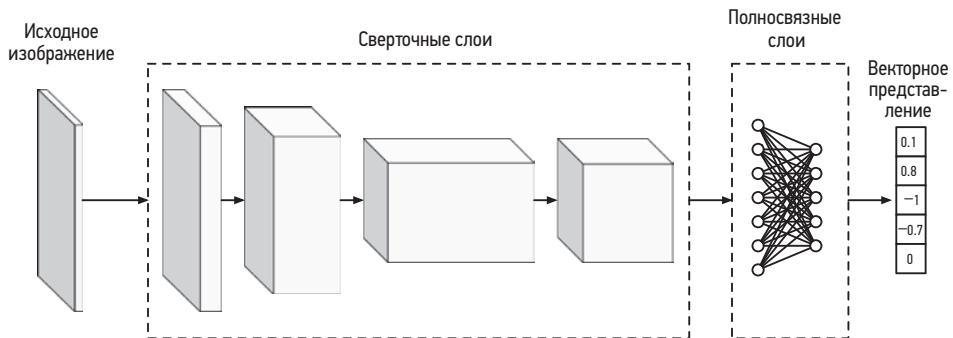


Рис. 2.5. Упрощенная архитектура модели на основе нейронной сети

Обучение модели

Чтобы находить визуально сходные изображения, модель должна изучить представления (эмбеддинги) в ходе обучения. В этом разделе мы обсудим, как этого добиться.

Стандартный прием обучения представлениям изображений — контрастное обучение [7]. По этому методу модель учится различать похожие и непохожие изображения. Как показано на рис. 2.6, мы предоставляем модели исходное изображение (слева), одно похожее изображение (выделенное изображение собаки справа) и несколько непохожих изображений (тоже справа). В ходе обучения модель учится вырабатывать представления, в которых похожее изображение оказывается ближе к исходному, чем другие изображения в правой части рис. 2.6.

Чтобы обучить модель методом контрастного обучения, сначала нужно сконструировать обучающие данные.



Рис. 2.6. Контрастное обучение

Построение датасета

Как упоминалось ранее, каждая точка данных, используемая для обучения, содержит:

- исходное изображение;
- одно изображение, похожее на исходное (положительное изображение);
- $n - 1$ изображений, не похожих на исходное (отрицательные изображения).

Эталонной меткой точки данных является индекс положительного изображения. Как показано на рис. 2.7, наряду с исходным изображением (q) также есть n других изображений, из которых одно похоже на q (изображение собаки), а остальные $n - 1$ изображений не похожи. Эталонная метка этой точки — индекс положительного изображения, который равен 2 (второе из n изображений на рис. 2.7).



Рис. 2.7. Точка данных обучающего набора

Чтобы построить обучающую точку данных, мы случайно выбираем исходное изображение и $n - 1$ изображений в качестве отрицательных. Выбрать положительное изображение можно тремя способами.

1. Предоставить выбор человеку.
2. Условно оценивать схожесть по взаимодействиям — например, пользовательским кликам.
3. Искусственно сконструировать похожее изображение из исходного (это называется самообучением).

Рассмотрим каждый из этих вариантов.

Предоставить выбор человеку

В этом случае обычно нанимают специальных людей, которые вручную ищут похожие изображения. В результате обучающие данные получаются точнее, но услуги живых разметчиков дороже и затратнее по времени.

Условно оценивать схожесть по взаимодействиям — например, пользовательским кликам

При таком подходе схожесть оценивается на основе данных о пользовательских взаимодействиях. Например, если пользователь щелкнул по изображению, это изображение считается похожим на исходное изображение q .

Этот метод не требует ручной работы и может автоматически создавать обучающие данные. Однако сведения о кликах обычно сильно зашумлены. Пользователи часто щелкают по изображениям, даже если они не похожи на исходное. Кроме того, эти данные очень разрежены, и для большинства изображений может не быть сведений о кликах. Зашумленные и разреженные обучающие данные снижают качество модели.

Искусственно сконструировать похожее изображение из исходного

Здесь мы искусственно создаем похожее изображение из исходного. Например, можно повернуть изображение и использовать его в качестве похожего. Этот подход используют такие недавно появившиеся фреймворки, как SimCLR [7] и MoCo [8].

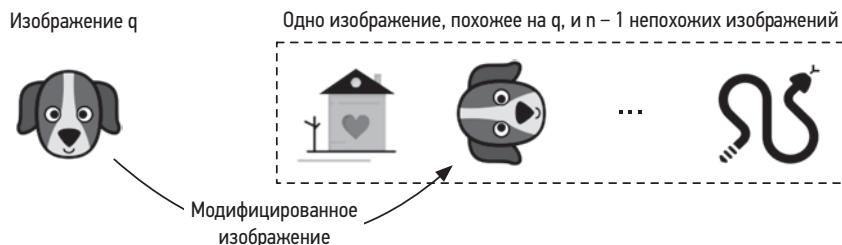


Рис. 2.8. Запись обучающих данных

Преимущество этого метода заключается в том, что он не требует никакой ручной работы. Чтобы создавать похожие изображения, можно реализовать простую логику модификации (аугментации) данных. Кроме того, созданные обучающие данные не зашумлены, потому что изображения, полученные путем аугментации, всегда похожи на исходное. Главный недостаток такого подхода в том, что сконструированные обучающие данные отличаются от реальных. На практике похожие изображения — это не модифицированные версии исходного; они визуально и семантически близки, но отличаются от него.

Какой метод лучше подойдет для нашего случая?

На собеседовании очень важно предложить разные варианты и обсудить их достоинства и недостатки. Обычно не существует единственного лучшего решения, которое всегда работает. В нашем примере самообучение используется по двум

причинам. Во-первых, оно не требует никаких первоначальных затрат, потому что процесс можно автоматизировать. Во-вторых, различные фреймворки (такие, как SimCLR [7]) показывают перспективные результаты, обучаясь на больших датасетах. Поскольку на платформе доступны миллиарды изображений, такой метод может для нее хорошо подойти.

Если результаты экспериментов нас не устроят, всегда можно будет переключиться на другие методы разметки. Например, можно начать с самообучения, а потом задействовать для разметки данные кликов. Также можно комбинировать варианты: например, построить исходный обучающий набор данных на основе кликов, а затем поручить живым людям выявлять и удалять зашумленные точки данных.

После того как датасет построен, наступает время обучать модель с подходящей функцией потерь.

Выбор функции потерь

Как показано на рис. 2.9, модель принимает на вход изображения и создает эмбеддинг для каждого входного изображения. E_x обозначает эмбеддинг изображения x .

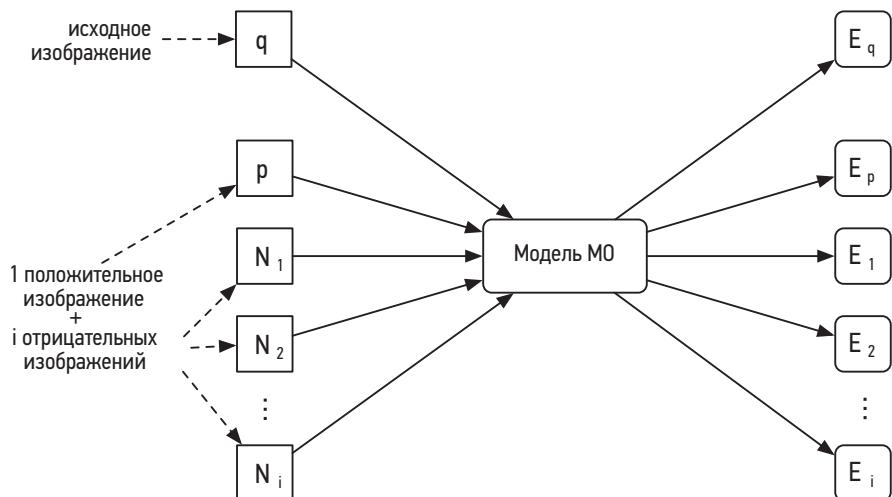


Рис. 2.9. Входные и выходные данные модели

Цель обучения — оптимизировать параметры модели так, чтобы эмбеддинги похожих изображений располагались поблизости друг от друга в пространстве эмбеддингов. Как показано на рис. 2.10, положительное и исходное изображения сблизились во время обучения.

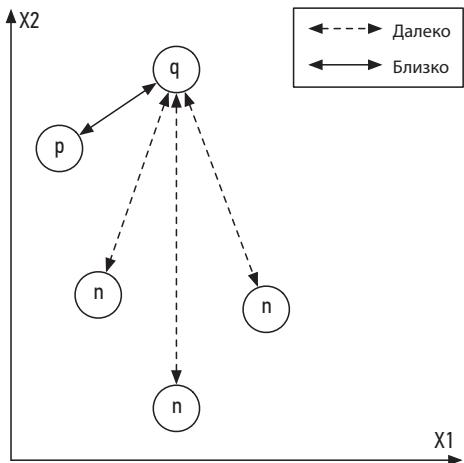


Рис. 2.10. Как входные изображения отображаются в пространство эмбеддингов

Чтобы достичь этой цели, нужно измерять качество полученных эмбеддингов с помощью функции потерь. Для контрастного обучения используется несколько разных функций потерь, и эксперты обычно не требуют, чтобы вы рассказывали о них во всех подробностях. Тем не менее важно иметь хотя бы общее понимание того, как устроены эти функции.

Рассмотрим в упрощенном виде, как работают контрастные функции потерь. Если вам захочется узнать о них больше, обращайтесь к [9].

Как показано на рис. 2.11, контрастная функция потерь вычисляется в три этапа.

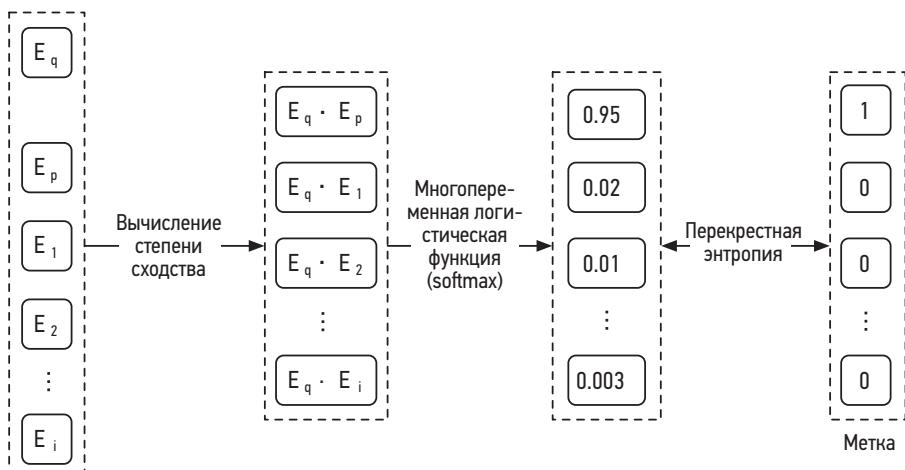


Рис. 2.11. Упрощенная функция потерь для контрастного обучения

Вычисление степени сходства. Сначала вычисляется сходство между эмбеддингами исходного изображения и других изображений. Для оценки сходства между точками пространства эмбеддингов широко применяется скалярное произведение [10] и косинусный коэффициент [11]. Также можно применять евклидово расстояние [12], однако в многомерных пространствах оно обычно оказывается неэффективным из-за «проклятия размерности» [13]. За дополнительной информацией о проблеме «проклятия размерности» обращайтесь к [14].

Многопеременная логистическая функция (softmax). Эта функция применяется к вычисленным расстояниям, в результате чего их значения в сумме дают 1, что позволяет интерпретировать их как вероятности.

Перекрестная энтропия. Перекрестная энтропия [15] измеряет, насколько предсказанные вероятности близки к эталонным меткам. Высокая степень близости показывает, что эмбеддинг достаточно хорошо позволяет отличить положительное изображение от отрицательных.

На собеседовании также можно обсудить, имеет ли смысл использовать предварительно обученные модели. Например, можно взять предварительно обученную контрастную модель и отрегулировать ее на основе обучающих данных. Такие модели уже прошли обучение на больших датасетах, а следовательно, изучили хорошие представления изображений. Так можно значительно сэкономить время по сравнению с обучением модели с нуля.

Оценка

После того как модель разработана, можно поговорить о том, как ее оценивать. В этом разделе рассматриваются важные метрики для автономной и оперативной оценки.

Автономные метрики

Согласно заданным требованиям, для автономной оценки доступен оценочный датасет. Допустим, каждая точка данных содержит исходное изображение, несколько изображений-кандидатов и метрику сходства каждого кандидата с исходным изображением. Эта метрика представляет собой целое число от 0 до 5,



Рис. 2.12. Точка данных из оценочного датасета

где 0 означает полное отсутствие сходства, а 5 – что два изображения визуально и семантически очень близки. Для каждой точки в оценочном датасете ранжирование, вычисленное моделью, сравнивается с идеальным ранжированием, которое основано на эталонных показателях.

Давайте рассмотрим автономные метрики, которые часто используются в поисковых системах. Следует заметить, что поиск, извлечение информации и рекомендательные системы обычно используют одни и те же автономные метрики:

- среднеобратный ранг (MRR);
- полнота на k (recall@ k);
- точность на k (precision@ k);
- усредненная средняя точность (mAP);
- нормализованная дисконтированная накопленная ценность (nDCG).

MRR. Эта метрика берет ранг первого релевантного элемента из каждого выходного списка, возвращаемого моделью, а затем усредняет эти ранги. Формула выглядит так:

$$MRR = \frac{1}{m} \sum_{i=1}^m \frac{1}{rank_i},$$

где m – общее количество выходных списков, а $rank_i$ – ранг первого релевантного элемента в i -м выходном списке.

На рис. 2.13 показано, как работает эта метрика. Для каждого из четырех ранжированных списков вычисляется обратный ранг (RR), а затем вычисляется среднее значение всех этих RR, в результате чего получается MRR.

У MRR есть недостаток: поскольку эта метрика учитывает только первый релевантный элемент и игнорирует все остальные элементы списка, она не характеризует точность и качество ранжирования в списке. Например, на рис. 2.14 показаны выводы двух разных моделей. Вывод модели 1 содержит 3 релевантных элемента, а вывод модели 2 – только 1 релевантный элемент. Тем не менее обратный ранг обеих моделей равен 0.5. Из-за этого недостатка мы не будем использовать эту метрику.

Полнота на k (recall@ k). Эта метрика измеряет отношение количества релевантных элементов в выходном списке к общему количеству релевантных элементов во всем наборе данных. Формула выглядит так:

$$\text{recall}@k = \frac{\text{количество релевантных элементов среди первых } k \text{ элементов}}{\text{общее количество релевантных элементов}}.$$

<input checked="" type="checkbox"/>	Релевантный элемент						
<input type="checkbox"/>	Нерелевантный элемент						
	Обратный ранг						
Ранжированный список 1	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td></td><td></td><td style="text-align: center;">✓</td><td></td><td></td></tr></table>			✓			1/3
		✓					
Ранжированный список 2	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="text-align: center;">✓</td><td></td><td></td><td style="text-align: center;">✓</td><td></td></tr></table>	✓			✓		1
✓			✓				
Ранжированный список 3	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td></td><td></td><td style="text-align: center;">✓</td><td style="text-align: center;">✓</td><td></td></tr></table>			✓	✓		1/3
		✓	✓				
Ранжированный список 4	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td></td><td></td><td></td><td></td><td></td></tr></table>						0
Среднеобратный ранг	$(1/3 + 1 + 1/3 + 0)/4 = 0.417$						

Рис. 2.13. Пример вычисления MRR

		MRR					
Вывод модели 1	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td></td><td style="text-align: center;">✓</td><td style="text-align: center;">✓</td><td style="text-align: center;">✓</td><td></td></tr></table>		✓	✓	✓		1/2
	✓	✓	✓				
Вывод модели 2	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td style="text-align: center;">✓</td><td></td><td></td><td></td><td></td></tr></table>	✓					1/2
✓							

Рис. 2.14. MRR двух разных моделей

Хотя полнота на k измеряет, сколько релевантных элементов модели не удалось включить в выходной список, эта метрика не всегда хороша. Разберемся, с чем это связано. В некоторых системах (например, поисковых) может быть очень много релевантных элементов. Это отрицательно отражается на полноте, потому что знаменатель будет очень большим. Например, в нашем примере, где на исходном изображении представлена собака, в базе данных могут быть миллионы разных картинок с собаками. Цель заключается не в том, чтобы найти все изображения собак, а в том, чтобы получить небольшую подборку самых похожих изображений собак.

Так как полнота на k не измеряет качество ранжирования модели, мы не будем использовать и эту метрику.

Точность на k ($\text{precision}@k$). Эта метрика измеряет долю релевантных элементов среди первых k элементов выходного списка. Формула выглядит так:

$$\text{precision}@k = \frac{\text{количество релевантных элементов среди первых } k \text{ элементов выходного списка}}{k}.$$

Эта метрика измеряет, насколько точны выходные списки, но не учитывает качество ранжирования. Например, на рис. 2.15 показано, что, если более релевантные элементы будут ранжироваться выше в списке, точность не изменится. Метрика не идеальна для нашего примера, потому что нам нужно измерять как точность, так и качество ранжирования результатов.

Точность на 5					
Вывод модели 1	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>				
Вывод модели 2	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>				
	<input type="checkbox"/> 2/5		<input type="checkbox"/> 2/5		

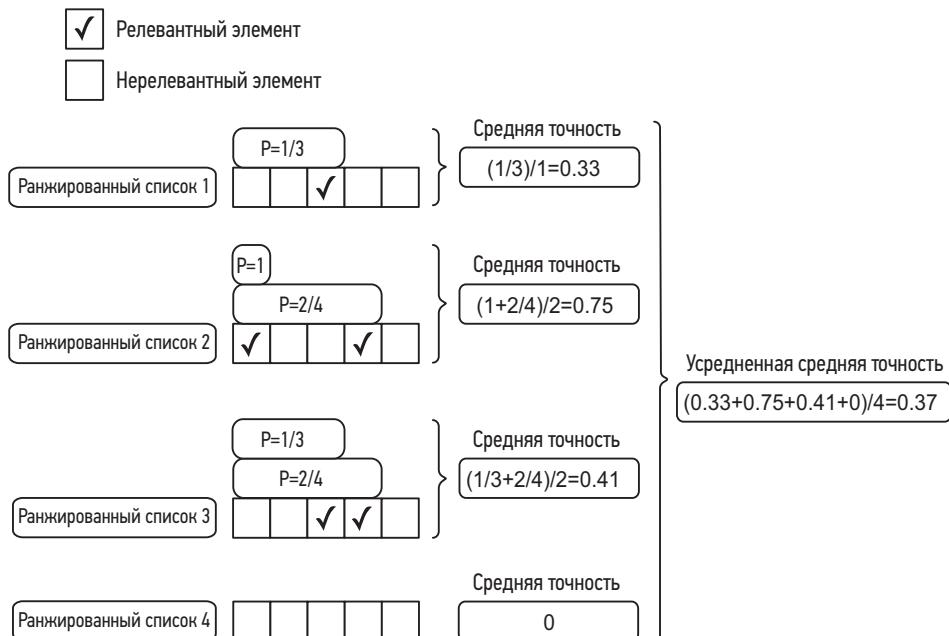
Рис. 2.15. Точность на 5 для двух разных моделей

mAP. Эта метрика сначала вычисляет среднюю точность (AP) для каждого выходного списка, а затем усредняет вычисленные значения.

Сначала разберемся, что такое AP. Метрика берет список из k элементов (например, изображений) и вычисляет точность из k , усредненную по разным значениям k . Высокие значения AP достигаются, когда более релевантные элементы находятся в верхней части списка. Для списка размером k формула AP выглядит так:

$$AP = \frac{\sum_{i=1}^k \text{точность на } i, \text{ если } i\text{-й элемент релевантен для пользователя}}{\text{общее количество релевантных элементов}}.$$

Следующий пример поможет лучше понять смысл этой метрики. На рис. 2.16 показано вычисление AP для каждого из четырех выходных списков, которые возвращает модель.



Поскольку показатели точности усредняются, метрика mAP учитывает общее качество ранжирования списка. Тем не менее она хороша для бинарной релевантности; другими словами, она эффективно работает только в том случае, если каждый элемент либо релевантен, либо нет. Для непрерывной оценки релевантности лучше подойдет nDCG.

nDCG. Метрика оценивает качество ранжирования выходного списка: она показывает, насколько хорошо выполнено ранжирование по сравнению с идеальным. Сначала разберемся, как вычисляется DCG, а потом обсудим nDCG.

Что такое DCG

DCG (Discounted Cumulative Gain, «дисконтированная накопленная ценность») оценивает ценность элементов в списке, взяв за основу показатели релевантности каждого элемента. Показатели для всех элементов суммируются, причем к каждому показателю применяется понижающий коэффициент, который увеличивается от верхних рангов к нижним. Формула выглядит так:

$$\text{DCG}_p = \sum_{i=1}^p \frac{\text{rel}_i}{\log_2(i+1)},$$

где rel_i — эталонная оценка релевантности изображения, которое ранжируется в позиции i .

Что такое nDCG

Поскольку DCG суммирует показатели релевантности элементов и применяет к ним понижающие коэффициенты в зависимости от позиции, значением DCG может быть произвольное число. Чтобы получить более содержательное значение, DCG нужно нормализовать. Для этого применяется nDCG, которое делит DCG на DCG идеального ранжирования. Формула выглядит так:

$$\text{nDCG}_p = \sum_{i=1}^p \frac{\text{DCG}_p}{\text{IDCG}_p},$$

где IDCG_p — это DCG идеального ранжирования (в порядке показателей релевантности элементов). Заметим, что в идеальной системе ранжирования DCG равно IDCG.

Чтобы лучше понять смысл nDCG, рассмотрим пример. На рис. 2.17 представлен список выходных изображений, который вернула поисковая система, и связанные с ними эталонные показатели релевантности.



Рис. 2.17. Ранжированный список, возвращенный поисковой системой

Значение nDCG вычисляется в три шага:

- 1) вычислить DCG;
- 2) вычислить IDCG;
- 3) разделить DCG на IDCG.

Вычисление DCG. Для текущего ранжирования, возвращенного моделью, DCG вычисляется так:

$$DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)} = \frac{0}{\log_2(2)} + \frac{5}{\log_2(3)} + \frac{1}{\log_2(4)} + \frac{4}{\log_2(5)} + \frac{2}{\log_2(6)} = 6.151.$$

Вычисление IDCG. DCG для идеального ранжирования (оно же IDCG) вычисляется так же, как DCG, за исключением того, что элементы ранжируются по показателям релевантности (рис. 2.18).



Рис. 2.18. Идеально ранжированный список

IDCG для идеального ранжирования вычисляется так:

$$IDCG_p = \sum_{i=1}^v \frac{rel_i}{\log_2(i+1)} = \frac{5}{\log_2(2)} + \frac{4}{\log_2(3)} + \frac{2}{\log_2(4)} + \frac{1}{\log_2(5)} + \frac{0}{\log_2(6)} = 8.9543.$$

Деление DCG на IDCG:

$$\text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p} = \frac{6.151}{8.9543} = 0.6869.$$

nDCG хорошо работает в большинстве случаев. Главный недостаток этой метрики в том, что не всегда можно вычислить эталонные показатели релевантности. В нашем случае, поскольку в оценочном наборе данных есть показатели сходства, для автономной оценки можно измерить качество модели с помощью nDCG.

Оперативные метрики

В этом разделе мы рассмотрим несколько распространенных оперативных метрик, которые оценивают, насколько быстро пользователи могут найти подходящие изображения.

Кликабельность (CTR, Click-Through Rate) показывает, как часто пользователи нажимают на отображаемые элементы. CTR вычисляется по следующей формуле:

$$\text{CTR} = \frac{\text{количество изображений, на которые нажали пользователи}}{\text{общее количество предложенных изображений}}.$$

Высокое значение CTR говорит о том, что пользователи часто кликают по отображаемым элементам. CTR обычно используется как оперативная метрика в поисковых и рекомендательных системах, как будет показано в следующих главах.

Среднее время просмотра предлагаемых изображений (в день, неделю, месяц). Эта метрика показывает, насколько пользователи вовлечены в просмотр изображений. Если поисковая система работает с высокой точностью, среднее время просмотра должно возрастать.

Эксплуатация

В режиме эксплуатации система возвращает ранжированный список похожих изображений на основании исходного изображения. На рис. 2.19 показаны предсказательный пайплайн и пайплайн индексации. Рассмотрим каждый из этих пайплайнов подробнее.

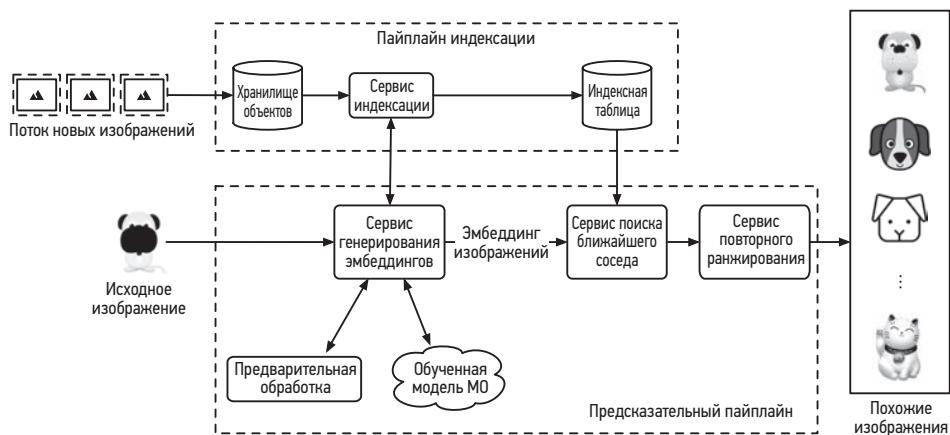


Рис. 2.19. Предсказательный пайплайн и пайплайн индексации

Предсказательный пайплайн

Сервис генерирования эмбеддингов

Сервис вычисляет эмбеддинг для исходного изображения, поступающего на вход. Как показано на рис. 2.20, он выполняет предварительную обработку изображения и использует обученную модель, чтобы вычислить эмбеддинг.

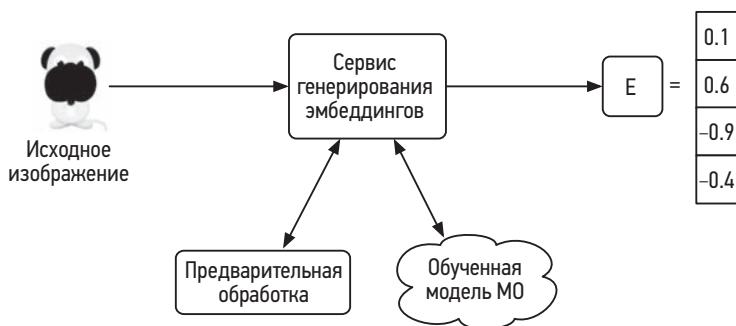


Рис. 2.20. Сервис генерирования эмбеддингов

Сервис поиска ближайшего соседа

После того как получен эмбеддинг для исходного изображения, нужно извлечь похожие изображения из пространства эмбеддингов. Этим занимается сервис поиска ближайшего соседа.

Определим поиск ближайшего соседа более формально. Для исходной точки q и множества других точек S он находит во множестве S точки, ближайшие к q . Напомним, что эмбеддинг изображения представляется точкой в N -мерном пространстве, где N – размерность вектора эмбеддинга. На рис. 2.21 изображены три ближайших соседа изображения q . Искомое изображение обозначено q , а другие изображения – X .

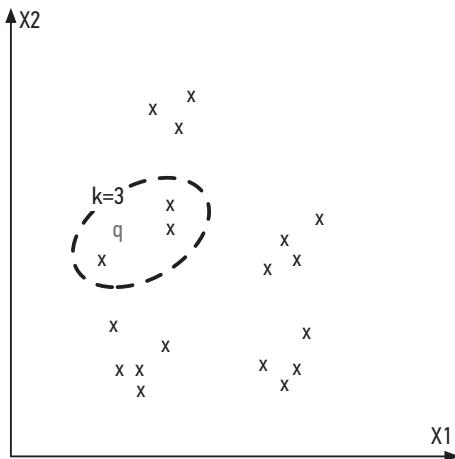


Рис. 2.21. Ближайшие три соседа изображения q в пространстве эмбеддингов

Сервис повторного ранжирования

К этому сервису относится бизнес-логика и политики. Например, он отфильтровывает неприемлемые результаты и изображения, которые нарушают конфиденциальность, удаляет дубликаты или почти идентичные результаты и обеспечивает прочую подобную логику перед тем, как выводить окончательные результаты пользователям.

Пайплайн индексации

Все изображения на платформе индексируются, что повышает эффективность поиска.

Другая задача сервиса индексации – обновлять индексную таблицу. Например, когда создатель добавляет на платформу новое изображение, служба индексирует его эмбеддинг, чтобы оно обнаруживалось при поиске ближайшего соседа.

Индексация повышает затраты памяти, потому что в индексной таблице сохраняются эмбеддинги целых изображений. Сократить затраты памяти позволяют различные методы оптимизации – такие, как векторное квантование [16] и квантование по произведению (PQ, Product Quantization) [17].

Эффективность алгоритмов поиска ближайшего соседа (NN)

Поиск ближайшего соседа (NN, Nearest Neighbor) — основной компонент систем извлечения информации, поиска и рекомендаций. Небольшие улучшения его эффективности позволяют существенно повысить общую производительность. Учитывая, насколько важен этот компонент, эксперт может предложить вам обсудить его на более глубоком уровне.

Алгоритмы NN можно разделить на две категории: точные и приближенные. Рассмотрим каждую категорию подробнее.

Точный поиск ближайшего соседа

Алгоритм точного поиска ближайшего соседа, который также называется линейным поиском, — это простейшая форма NN. Он обходит всю индексную таблицу, вычисляет расстояние от каждой точки до исходной точки q и выбирает ближайшие k точек. Временная сложность составляет $O(N \times D)$, где N — общее количество точек, а D — размерность точки.

В крупномасштабной системе, где значение N может легко достигать миллиардов, поиск с линейной сложностью работает слишком медленно.

Приближенный поиск ближайшего соседа (ANN)

Во многих приложениях бывает достаточно показать пользователю результаты с удовлетворительной степенью сходства и не нужно точно находить ближайших соседей.

В алгоритмах ANN используется специальная структура данных, которая сокращает временную сложность поиска NN до сублинейной (например, $O(D \times \log N)$). ANN обычно требует предварительной обработки или дополнительного пространства.

Алгоритмы ANN можно разделить на три типа:

- древовидный ANN;
- ANN на базе хеширования, чувствительного к местоположению (LSH, Locality-Sensitive Hashing);
- ANN на основе кластеризации.

В каждую категорию входят различные алгоритмы, и эксперты обычно не требуют, чтобы вы знали их во всех подробностях. Однако желательно понимать их хотя бы на высоком уровне. Давайте кратко рассмотрим каждую категорию.

Древовидный ANN

Древовидные алгоритмы формируют дерево, разбивая пространство на несколько областей. Затем они используют особенности дерева, чтобы ускорить

поиск. Дерево строится итеративным добавлением новых критериев в каждый узел. Например, критерием корневого узла может быть «пол = мужской». Это значит, что любая точка с атрибутом женского пола попадет в левое поддерево.

Каждый узел ветвления дерева разбивает пространство на две части по заданному критерию. Листовые узлы соответствуют отдельным областям в пространстве. На рис. 2.23 приведен пример пространства, разбитого на 7 областей. Алгоритм проводит поиск только в том разделе, где находится исходная точка.

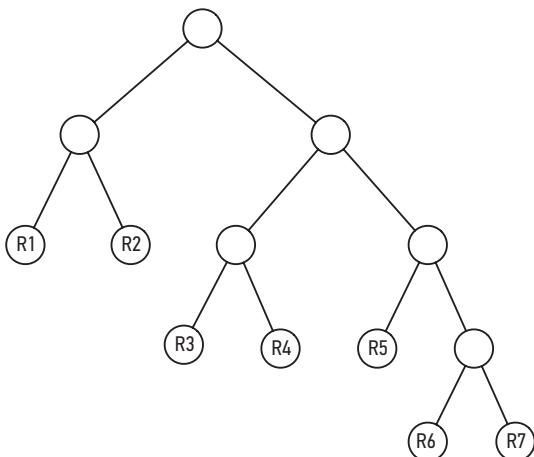


Рис. 2.22. Дерево, сформированное на основе точек

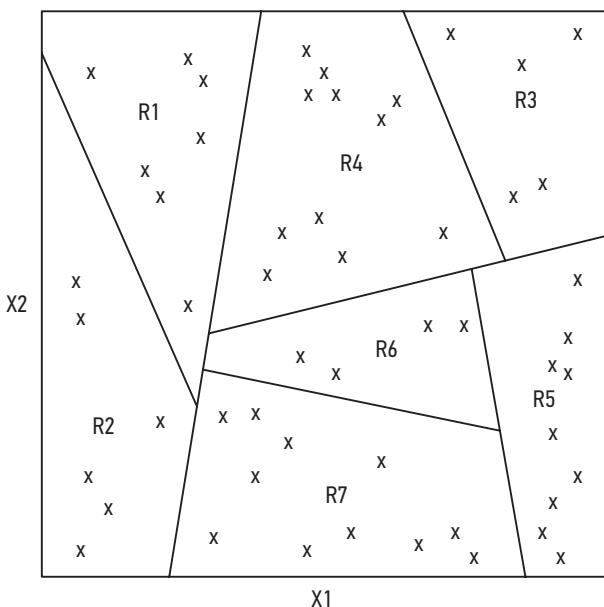


Рис. 2.23. Разбиение пространства деревом

Классические древовидные методы — R-деревья [18], K-d-деревья [19] и Annoy (Approximate Nearest Neighbor Oh Yeah) [20].

Хеширование, чувствительное к местоположению (LSH)

Поиск на базе LSH использует специальные хеш-функции, чтобы сократить раз мерность точек и сгруппировать их по сегментам (бакетам). Эти хеш-функции отображают точки, находящиеся поблизости друг от друга, на один сегмент. LSH проводит поиск только среди точек, которые находятся в одном сегменте с исходной точкой q . Подробнее узнать о LSH можно в [21].

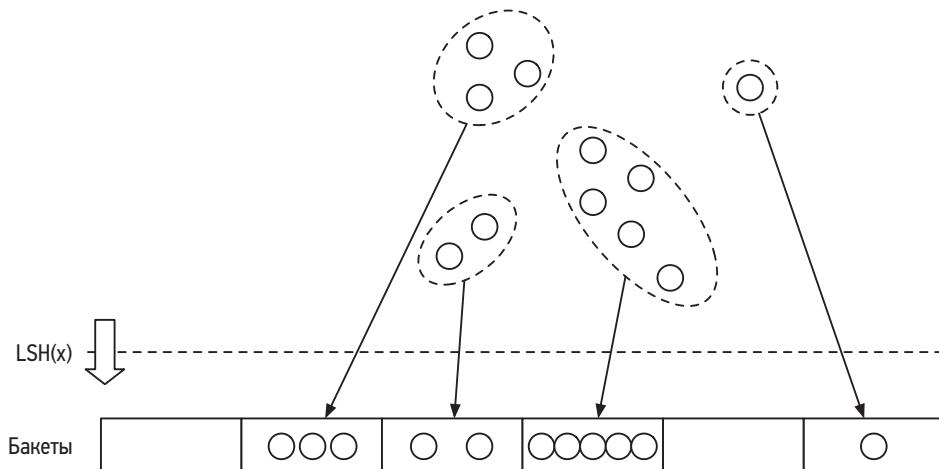


Рис. 2.24. Группировка данных по бакетам с помощью LSH

ANN на основе кластеризации

Такие алгоритмы формируют кластеры, группируя точки на основании сходства. После того как кластеры образованы, алгоритм проводит поиск только среди точек в том кластере, которому принадлежит исходная точка.

Какой алгоритм использовать?

Алгоритмы точного поиска ближайшего соседа гарантированно дают точные результаты и поэтому хорошо подходят, когда количество точек данных ограничено или когда необходимо точно найти ближайших соседей. Но при большом количестве точек практически невозможно добиться, чтобы алгоритм работал эффективно. В этом случае обычно применяются методы ANN, которые необязательно возвращают точные результаты, зато эффективнее работают на практике.

С учетом того, какие объемы данных доступны в современных системах, метод ANN становится более прагматичным решением. Мы будем использовать его

в нашей системе визуального поиска, чтобы находить эмбеддинги похожих изображений.

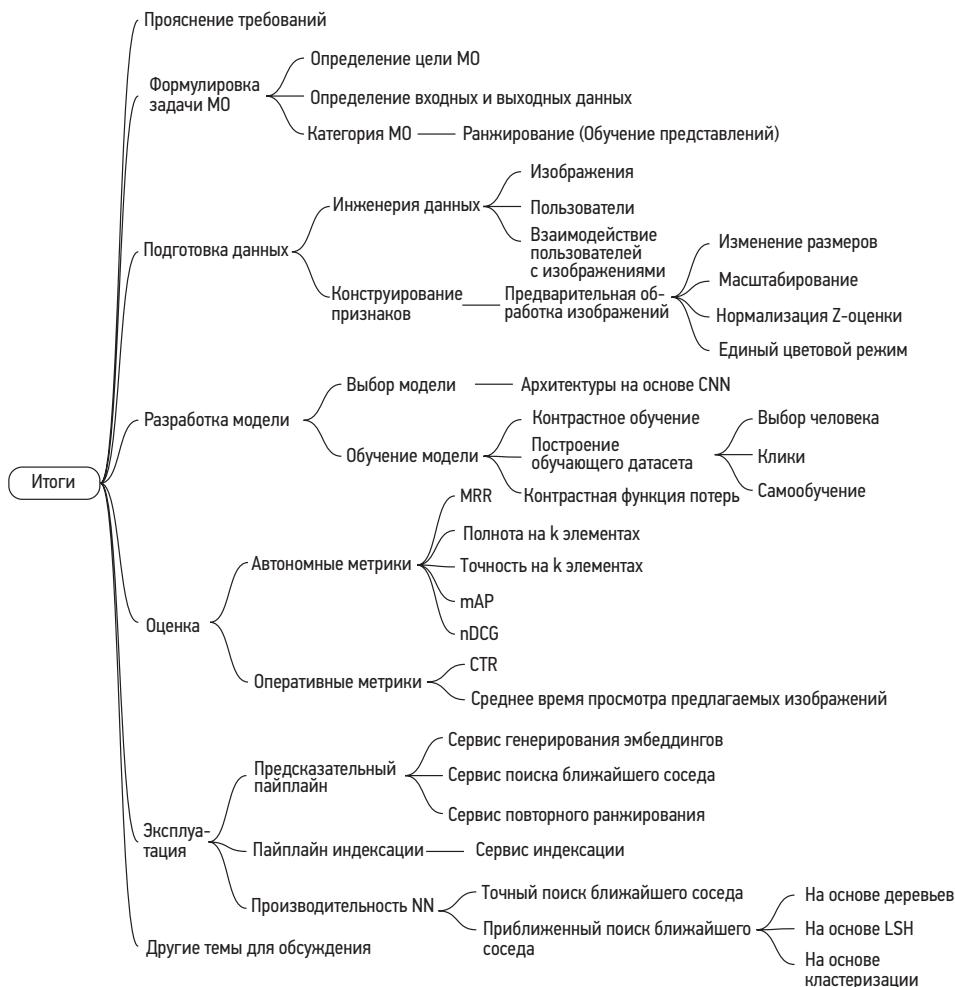
На собеседовании на вакансию в области прикладного МО эксперт может предложить вам реализовать ANN. Для этого существуют две популярные библиотеки — Faiss [22] (разработчик — Meta) и ScaNN [23] (разработчик — Google). Каждая библиотека поддерживает большинство методов, описанных в этой главе. Мы рекомендуем поближе познакомиться хотя бы с одной из них, чтобы лучше понять общие принципы и заранее подготовиться к тому, что на собеседовании по программированию МО вам предложат реализовать поиск методом ближайшего соседа.

Другие темы для обсуждения

Если в конце собеседования останется свободное время, вам могут задать дополнительные вопросы или предложить обсудить более сложные темы в зависимости от предпочтений эксперта, вашего опыта, требований к вакансии и других факторов. Ниже перечислены некоторые вопросы, к которым стоит подготовиться (особенно для вакансий старших специалистов).

- Как модерировать контент в системе, выявляя и блокируя неприемлемые изображения [24].
- Различные смещения в системе — например, позиционное [25], [26].
- Как использовать метаданные изображения (например, теги), чтобы улучшить результаты поиска. Эта тема рассматривается в главе 3.
- Умная обрезка на основе обнаружения объектов [27].
- Как использовать графовые нейронные сети, чтобы обучать модель на лучших представлениях [28].
- Поиск изображений по текстовому запросу. Мы поговорим об этом в главе 4.
- Активное обучение [29] или МО с оператором в контуре управления [30] для более эффективной разметки данных.

Итоги



Ссылки

- [1] Визуальный поиск в Pinterest. <https://arxiv.org/pdf/1505.07647.pdf>
- [2] Визуальные эмбеддинги для поиска в Pinterest. <https://medium.com/pinterest-engineering/unifying-visual-embeddings-for-visual-search-at-pinterest-74ea7ea103f0>
- [3] Обучение представлениям. https://en.wikipedia.org/wiki/Feature_learning
- [4] ResNet. <https://arxiv.org/pdf/1512.03385.pdf>
- [5] Transformer. <https://arxiv.org/pdf/1706.03762.pdf>
- [6] Transformer для компьютерного зрения. <https://arxiv.org/pdf/2010.11929.pdf>
- [7] SimCLR. <https://arxiv.org/pdf/2002.05709.pdf>
- [8] MoCo. https://openaccess.thecvf.com/content_CVPR_2020/papers/He_Momentum_Contrast_for_Unsupervised_Visual_Representation_Learning_CVPR_2020_paper.pdf
- [9] Контрастные методы обучения представлениям. <https://lilianweng.github.io/posts/2019-11-10-self-supervised/>
- [10] Скалярное произведение. https://en.wikipedia.org/wiki/Dot_product
- [11] Косинусный коэффициент. https://en.wikipedia.org/wiki/Cosine_similarity
- [12] Евклидово расстояние. https://en.wikipedia.org/wiki/Euclidean_distance
- [13] Проклятие размерности. https://en.wikipedia.org/wiki/Curse_of_dimensionality
- [14] Проклятие размерности в МО. <https://www.mygreatlearning.com/blog/understanding-curse-of-dimensionality/>
- [15] Перекрестная энтропия. https://en.wikipedia.org/wiki/Cross_entropy
- [16] Векторное квантование. [http://ws.binghamton.edu/fowler/fowler%20personal%20page/EE523_files/Ch_10_1%20VQ%20Description%20\(PPT\).pdf](http://ws.binghamton.edu/fowler/fowler%20personal%20page/EE523_files/Ch_10_1%20VQ%20Description%20(PPT).pdf)
- [17] Квантование по произведению (PQ, Product Quantization). <https://towardsdatascience.com/product-quantization-for-similarity-search-2f1f67c5fddd>
- [18] R-деревья. <https://en.wikipedia.org/wiki/R-tree>
- [19] K-d-дерево. <https://kanoki.org/2020/08/05/find-nearest-neighbor-using-kd-tree/>
- [20] Annoy. <https://towardsdatascience.com/comprehensive-guide-to-approximate-nearest-neighbors-algorithms-8b94f057d6b6>
- [21] Хеширование, чувствительное к местоположению (LSH). <https://web.stanford.edu/class/cs246/slides/03-lsh.pdf>
- [22] Библиотека Faiss. <https://github.com/facebookresearch/faiss/wiki>
- [23] Библиотека ScaNN. <https://github.com/google-research/google-research/tree/master/scann>
- [24] Модерирование контента с МО. <https://appen.com/blog/content-moderation/>
- [25] Смещение в AI и рекомендательных системах. <https://www.searchenginejournal.com/biases-search-recommender-systems/339319/#close>
- [26] Позиционное смещение. <https://eugeneyan.com/writing/position-bias/>
- [27] Умная обрезка. https://blog.twitter.com/engineering/en_us/topics/infrastructure/2018/Smart-Auto-Cropping-of-Images.
- [28] Как улучшить поиск с gnns. <https://arxiv.org/pdf/2010.01666.pdf>
- [29] Активное обучение. [https://en.wikipedia.org/wiki/Active_learning_\(machine_learning\)](https://en.wikipedia.org/wiki/Active_learning_(machine_learning))
- [30] МО с оператором в контуре управления. <https://arxiv.org/pdf/2108.00941.pdf>

3

СИСТЕМА РАЗМЫТИЯ В GOOGLE STREET VIEW

Google Street View [1] — технология платформы Google Maps, которая предоставляет интерактивные панорамы улиц по всему миру. В 2008 году компания Google создала систему, которая автоматически размывает человеческие лица и номера машин для защиты конфиденциальности. В этой главе мы спроектируем систему размытия, похожую на Google Street View.



Рис. 3.1. Изображение Street View с размытыми номерами машин

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: Правильно ли будет сказать, что бизнес-цель системы — защита конфиденциальности людей?

Эксперт: Да.

Соискатель: Мы хотим спроектировать систему, которая обнаруживает все человеческие лица и номерные знаки на изображениях Street View и размывает

их, прежде чем показывать пользователям. Все верно? Смогут ли пользователи сообщать об изображениях, которые не были правильно размыты?

Эксперт: Да, все так.

Соискатель: Доступен ли размеченный датасет для этой задачи?

Эксперт: Предположим, что у нас есть выборка из 1 миллиона изображений, где человеческие лица и номерные знаки размечены вручную.

Соискатель: В датасете может не оказаться лиц определенной расовой принадлежности, отчего может получиться смещение для некоторых признаков, описывающих человека: расы, возраста, пола и т. д. Это разумное предположение?
Эксперт: Верно замечено. Для простоты мы сейчас не будем учитывать смещения.

Соискатель: Насколько я понимаю, проблема задержки не очень критична, потому что система может обнаруживать и размывать объекты автономно. Верно?
Эксперт: Да. Пользователям можно показывать ранее обработанные изображения, пока новые изображения обрабатываются в автономном режиме.

Резюмируем описание проблемы. Требуется спроектировать систему по образцу Street View, которая автоматически размывает номерные знаки и человеческие лица. Есть обучающий набор данных, который содержит 1 миллион изображений с размеченными лицами и номерами. Бизнес-цель системы — защита конфиденциальности людей, чьи лица или автомобили попали в кадр.

Формулировка проблемы в виде задачи МО

В этом разделе мы сформулируем проблему размытия изображений в виде задачи МО.

Определение цели МО

Бизнес-цель системы — защита конфиденциальности людей за счет размытия видимых номерных знаков и человеческих лиц на изображениях Street View. Однако защита конфиденциальности не цель МО, так что ее необходимо преобразовать в цель, которую может выполнять система МО. Как вариант, цель МО может состоять в том, чтобы точно обнаруживать на изображении объекты, которые представляют интерес. Если система МО сможет точно обнаруживать эти объекты, к ним можно будет применить размытие перед тем, как показывать изображения пользователям.

В этой главе мы для краткости будем говорить «объекты» вместо «человеческие лица и номерные знаки».

Определение входных и выходных данных системы

На вход модели обнаружения объектов поступает изображение с некоторым количеством (ноль или более) объектов, которые находятся в разных местах. Модель обнаруживает эти объекты и выводит их расположение. На рис. 3.2 показаны входные и выходные данные системы обнаружения объектов.



Рис. 3.2. Входные и выходные данные системы обнаружения объектов

Выбор подходящей категории МО

В общем случае у системы обнаружения объектов две задачи:

- предсказывать расположение каждого объекта на изображении;
- предсказывать класс для каждой ограничительной рамки (например, «собака», «кошка» и т. д.).

Первая задача — это регрессия, потому что расположение можно выразить числовыми координатами (x, y). Вторую задачу можно сформулировать как многоклассовую классификацию.

Архитектуры обнаружения объектов традиционно делятся на одноступенчатые и двухступенчатые сети. В последнее время архитектуры на базе Transformer (такие, как DETR [2]) продемонстрировали перспективные результаты, но в этой главе в основном будут рассматриваться традиционные архитектуры.

Двухступенчатые сети

Как следует из названия, в двухступенчатых сетях используются две разные модели:

1. **Сеть обнаружения регионов (RPN)** анализирует изображение и предлагает регионы, которые с большой вероятностью содержат объекты.
2. **Классификатор** обрабатывает каждый предложенный регион и сопоставляет с ним класс объектов.

Эти две ступени изображены на рис. 3.3.

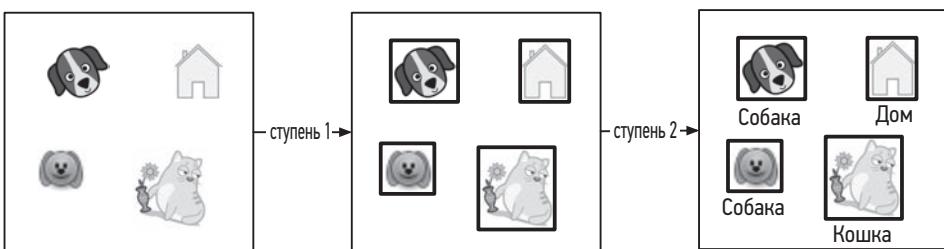


Рис. 3.3. Двухступенчатая сеть

К распространенным двухступенчатым сетям относятся R-CNN [3], Fast R-CNN [4] и Faster R-CNN [5].

Одноступенчатые сети

В таких сетях обе ступени объединены. Ограничительные рамки и классы объектов формируются одновременно, без отдельной стадии обнаружения регионов. На рис. 3.4 представлена одноступенчатая сеть.

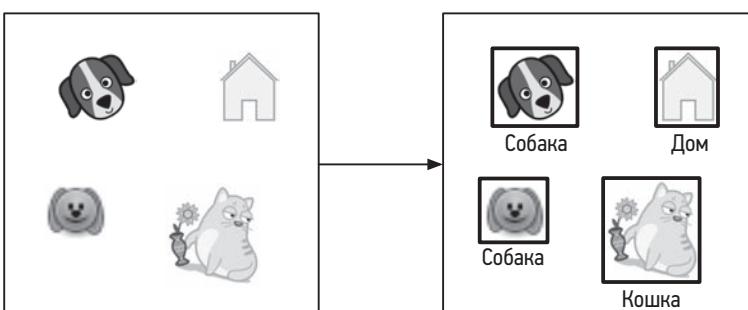


Рис. 3.4. Одноступенчатая сеть

На практике часто применяются такие архитектуры одноступенчатых сетей, как YOLO [6] и SSD [7].

Сравнение одноступенчатых и двухступенчатых сетей

Двухступенчатые сети состоят из двух компонентов, которые действуют последовательно, поэтому обычно они работают медленнее, зато обеспечивают более высокую точность.

В нашем случае датасет содержит 1 миллион изображений, что по современным стандартам не так уж много, а значит, в случае двухступенчатой сети затраты на обучение останутся приемлемыми. Поэтому здесь мы начнем с двухступенчатой сети. Если объем обучающих данных вырастет или если предсказания понадобится ускорить, можно будет переключиться на одноступенчатую сеть.

Подготовка данных

Инженерия данных

В главе 1 «Введение и общие сведения» излагались основы инженерии данных. Однако обычно желательно обсудить конкретные данные, доступные для текущей задачи. В нашем случае доступны такие данные:

- размеченный датасет;
- изображения Street View.

Рассмотрим оба пункта подробнее.

Размеченный датасет

По условию задачи, в нашем распоряжении есть 1 миллион размеченных изображений. С каждым изображением связан список ограничительных рамок и соответствующих классов объектов. В табл. 3.1 приведены примеры точек данных из датасета.

Таблица 3.1. Примеры точек данных из размеченного датасета

Путь к изображению	Объекты	Ограничительные рамки
dataset/image1.jpg	лицо	[10, 10, 25, 50]
	лицо	[120, 180, 40, 70]
	номерной знак	[80, 95, 35, 10]
dataset/image2.jpg	лицо	[170, 190, 30, 80]
	номерной знак	[25, 30, 210, 220]
dataset/image3.jpg	лицо	[30, 40, 30, 60]

Каждая ограничительная рамка (прямоугольник) представляет собой список из четырех чисел: координаты X и Y левого верхнего угла, а также ширина и высота объекта.

Изображения Street View

Изображения Street View предоставляет отдел сбора данных. Система МО обрабатывает эти изображения, чтобы обнаружить человеческие лица и номерные знаки. В табл. 3.2 представлены метаданные изображений.

Таблица 3.2. Метаданные изображений Street View

Путь к изображению	Геопозиция (широта, долгота)	Крен, тангаж, рысканье ¹	Временная метка
dataset/image1.jpg	(37.432567, −122.143993)	(0, 10, 20)	1646276421
dataset/image2.jpg	(37.387843, −122.091086)	(0, 10, −10)	1646276539
dataset/image3.jpg	(37.542081, −121.997640)	(10, −20, 45)	1646276752

Конструирование признаков

На этапе конструирования признаков мы сначала применим стандартные операции предварительной обработки изображений, такие как изменение размеров и нормализация. После этого мы увеличим размер датасета методом аугментации. Разберемся, что это такое.

Аугментация данных

Аугментация данных заключается в том, что в датасет добавляются слегка измененные копии исходных данных или новые данные, искусственно созданные из исходных. Чем больше размер датасета, тем более сложные закономерности может изучать модель. Этот метод особенно полезен, если датасет не сбалансирован, потому что аугментация увеличивает количество точек данных в монитарных классах.

Частный случай аугментации данных — аугментация изображений. Вот некоторые из ее популярных методов:

- случайная обрезка;
- случайное изменение насыщенности;
- отражение по вертикали или горизонтали;
- поворот и/или смещение;

¹ Термины из авиаобласти. Означают угол вращения самолета (а нашем случае камеры, фиксирующей изображение). Улицы снимаются сверху, а камера может быть под углом. — Примеч. ред.

- аффинные преобразования;
- изменение яркости, насыщенности или контраста.

На рис. 3.5 показано, как разные методы аугментации данных влияют на изображение.

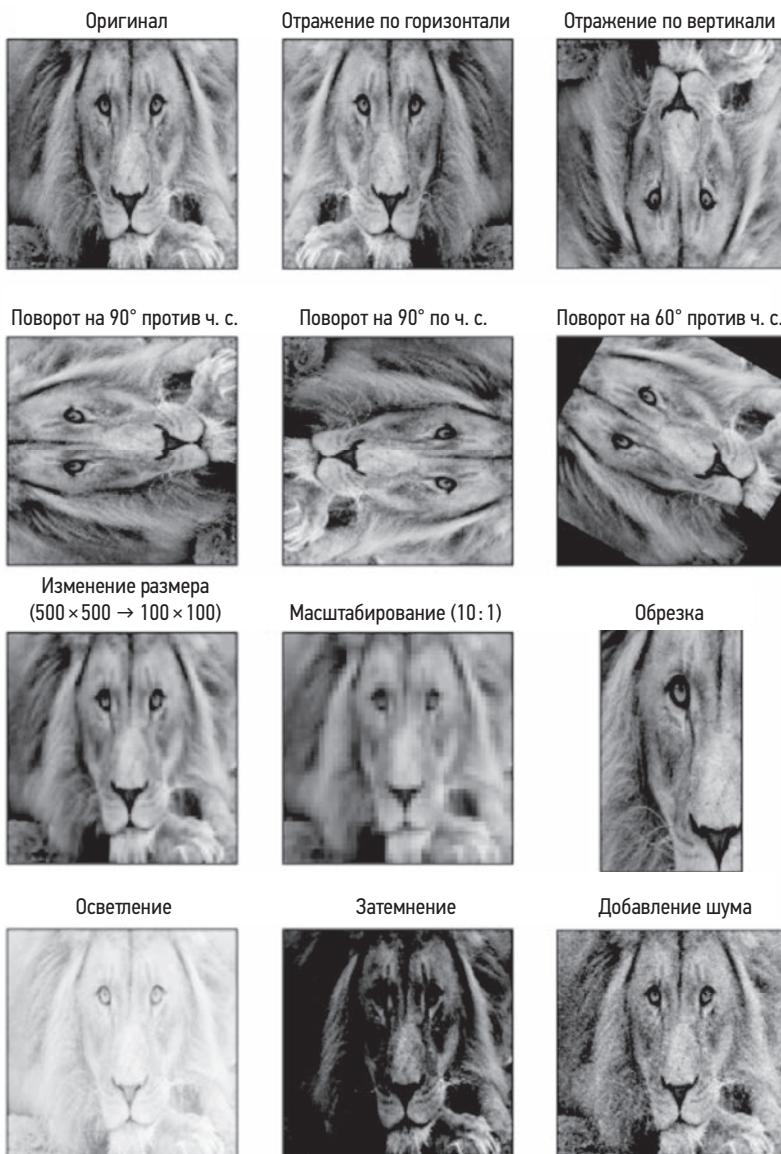


Рис. 3.5. Аугментированные изображения (из [8])

Важно заметить, что для некоторых методов аугментации — например, поворота или отражения — эталонные ограничительные прямоугольники тоже приходится преобразовывать.

Аугментация данных используется в двух формах:

- **Автономная форма:** аугментация изображений перед обучением.
- **Оперативная форма:** аугментация изображений «на ходу» во время обучения.

При автономной аугментации обучение ускоряется, потому что не требуется дополнительная аугментация, однако при этом нужно выделить память, чтобы хранить аугментированные изображения. С другой стороны, хотя оперативная аугментация замедляет обучение, она не расходует лишнюю память.

Выбор между автономной и оперативной аугментацией данных зависит от ограничений по памяти и вычислительной мощности. На собеседовании важно обсудить достоинства и недостатки разных вариантов. В нашем случае данные будут аугментироваться автономно.

На рис. 3.6 представлен процесс подготовки датасета. В ходе предварительной обработки изображения подвергаются изменению размеров, масштабированию и нормализации. В результате аугментации количество изображений увеличивается: например, 1 миллион исходных изображений превращается в 10 миллионов.

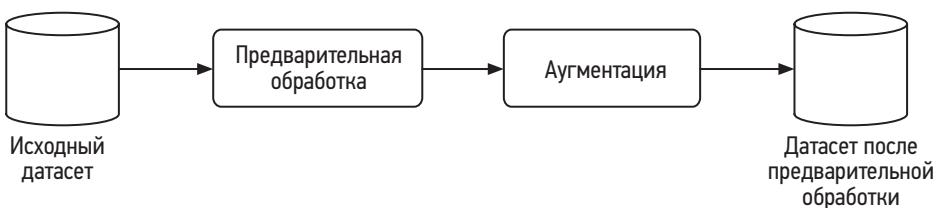


Рис. 3.6. Процесс подготовки датасета

Разработка модели

Выбор модели

Как упоминалось в разделе «Формулировка проблемы как задача МО», для этого примера мы выбрали двухступенчатую сеть. На рис. 3.7 изображена типичная двухступенчатая архитектура.

Рассмотрим каждый компонент по отдельности.

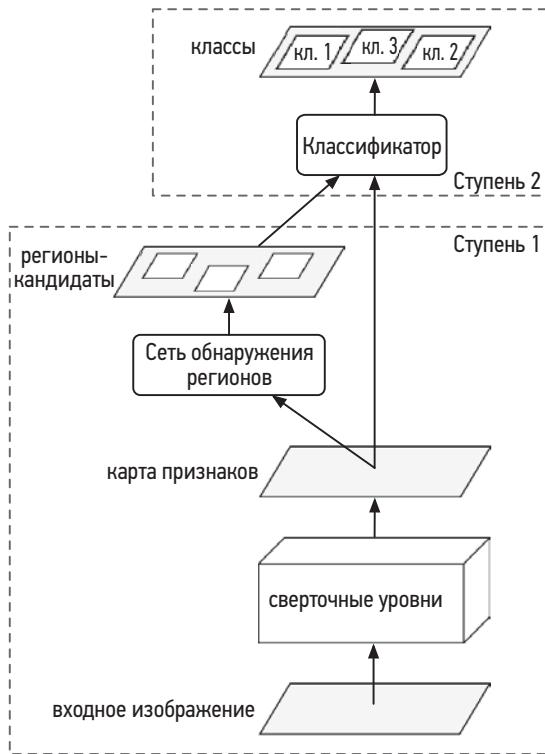


Рис. 3.7. Двухступенчатая сеть обнаружения объектов

Сверточные уровни

Сверточные уровни [9] обрабатывают входное изображение и выводят карту признаков.

Сеть обнаружения регионов (RPN)

RPN обнаруживает регионы, которые могут содержать объекты. В ее архитектуре используются нейронные сети; на вход подается карта признаков, производимая сверточными уровнями, а на выходе получаются регионы-кандидаты.

Классификатор

Классификатор принимает на вход карту признаков и предлагаемые регионы-кандидаты и назначает каждому региону определенный класс объекта. Обычно классификаторы работают на основе нейронных сетей.

Как правило, на собеседовании по проектированию систем МО не предполагается обсуждение архитектуры этих нейронных сетей. За дополнительной информацией обращайтесь к [10].

Обучение модели

Процесс обучения нейронной сети обычно состоит из трех шагов: прямого распространения, вычисления потерь и обратного распространения. Предполагается, что вы уже знакомы с этими шагами, но за дополнительной информацией обращайтесь к [11]. В этом разделе рассматриваются функции потерь, которые обычно используются для обнаружения объектов.

Модель обнаружения объектов должна хорошо решать две задачи. Во-первых, ограничительные прямоугольники предсказанных объектов должны хорошо соответствовать эталонным ограничительным прямоугольникам — это задача регрессии. Во-вторых, предсказанные вероятности всех классов объектов должны быть точными — это задача классификации. Определим функцию потерь для каждой задачи.

Функция потерь регрессии. Эта функция оценивает, насколько предсказанные ограничительные прямоугольники совмещаются с эталонными. Мы будем использовать стандартную функцию потерь регрессии, такую как среднеквадратичная ошибка (MSE), далее обозначенная L_{reg} :

$$L_{reg} = \frac{1}{M} \sum_{i=1}^M \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \right],$$

где:

- M — общее количество предсказаний;
- x_i — эталонная координата x левого верхнего угла;
- \hat{x}_i — предсказанная координата x левого верхнего угла;
- y_i — эталонная координата y левого верхнего угла;
- \hat{y}_i — предсказанная координата y левого верхнего угла;
- w_i — эталонная ширина;
- \hat{w}_i — предсказанная ширина;
- h_i — эталонная высота;
- \hat{h}_i — предсказанная высота.

Функция потерь классификации. Эта функция оценивает, насколько точно предсказаны вероятности для каждого обнаруженного объекта. Мы будем использовать стандартную функцию потерь классификации, такую как логистическая функция потерь (перекрестная энтропия) [13], далее обозначенная L_{cls} :

$$\text{log loss} = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij}),$$

где:

- N — количество предсказаний (соответствует M в книге);
- M — количество классов (соответствует C в книге);
- p_{ij} — предсказанная вероятность того, что i -е предсказание относится к j -му классу;
- y_{ij} — индикаторная функция (1, если i -е предсказание действительно относится к j -му классу, иначе 0).

Чтобы определить итоговые потери, оценивающие общее качество модели, мы суммируем потери классификации с потерями регрессии, которым назначим вес — параметр регуляризации λ :

$$L = L_{cls} + \lambda L_{reg}.$$

Оценка

На собеседовании важно обсудить, как оценивать систему МО. Эксперту обычно интересно, какие метрики вы выберете и почему. В этом разделе мы обсудим, как обычно оцениваются системы обнаружения объектов, а затем выберем важные метрики для автономной и оперативной оценки.

Модель обнаружения объектов обычно должна обнаруживать N разных объектов на изображении. Чтобы оценить общее качество модели, следует определить качество обнаружения каждого объекта по отдельности, а затем усреднить результаты.

На рис. 3.8 изображен вывод модели обнаружения объектов, где показаны как эталонные, так и обнаруженные ограничительные прямоугольники. В этом примере модель обнаружила шесть ограничительных прямоугольников, однако на изображении есть только два реальных экземпляра объекта.

Когда ограничительный прямоугольник считается правильно предсказанным? Чтобы ответить на этот вопрос, нужно понимать смысл пересечения по объединению.

Пересечение по объединению (IOU, Intersection Over Union) оценивает величину перекрытия двух ограничительных прямоугольников. На рис. 3.9 изображено наглядное представление этой метрики.

IOU показывает, правильно ли обнаружен ограничительный прямоугольник. Значение $IOU = 1$ идеально; оно говорит о том, что обнаруженный ограничительный прямоугольник в точности совпадает с эталонным. На практике это значение встречается редко. Чем выше IOU, тем точнее предсказан ограничительный прямоугольник. Обычно задается порог IOU, от которого зависит,

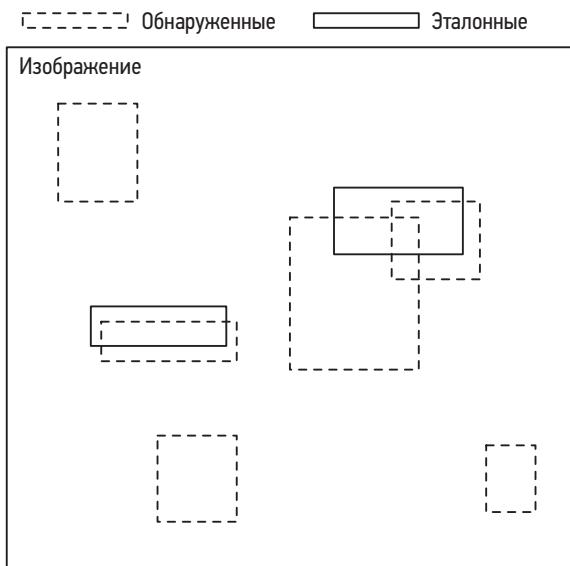


Рис. 3.8. Эталонные и обнаруженные ограничительные прямоугольники

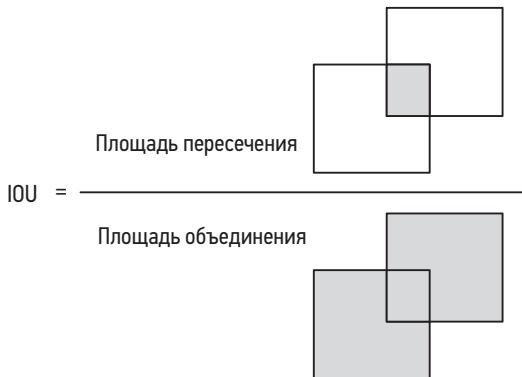


Рис. 3.9. Формула вычисления IOU

считать ли обнаруженный ограничительный прямоугольник правильным (истинно положительным) или неправильным (ложноположительным). Например, порог IOU 0.7 означает, что любой ограничительный прямоугольник с $\text{IOU} = 0.7$ и выше считается правильно обнаруженным.

Теперь, когда вы знаете, что такое IOU и как определить, правильно ли предсказан ограничительный прямоугольник, обсудим метрики автономной оценки.

Автономные метрики

Разработка модели — это итеративный процесс. Автономные метрики используются, чтобы быстро оценить качество создаваемых моделей. Вот примеры метрик, которые могут пригодиться в системе обнаружения объектов:

- точность (precision);
- средняя точность;
- усредненная средняя точность.

Точность

Доля правильных обнаружений среди всех обнаружений. Чем выше значение точности, тем надежнее обнаружения системы.

$$\text{Точность} = \frac{\text{количество правильных обнаружений}}{\text{общее количество обнаружений}}.$$

Чтобы вычислить точность, необходимо выбрать порог IOU. Чтобы лучше понять, как это делается, рассмотрим пример. На рис. 3.10 представлен набор эталонных и обнаруженных ограничительных прямоугольников с соответствующими значениями IOU.

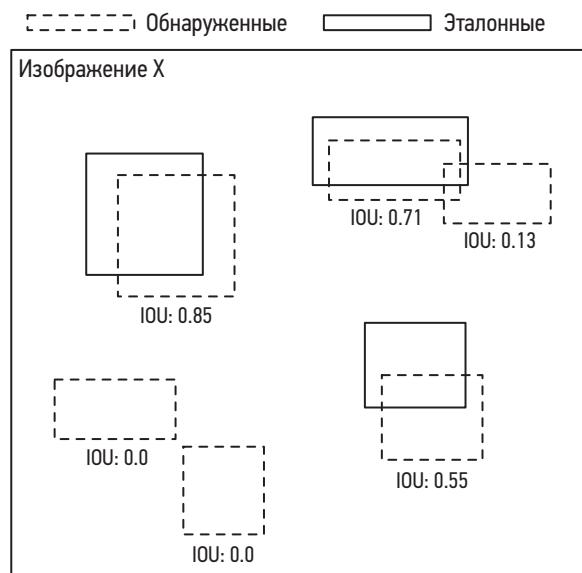


Рис. 3.10. Эталонные и обнаруженные ограничительные прямоугольники

Вычислим точность для трех разных порогов IOU: 0.7, 0.5 и 0.1.

- **Порог IOU = 0.7**

У двух обнаружений из шести IOU превышает 0.7. Следовательно, при этом пороге получается два правильных предсказания:

$$\text{Точность}_{0.7} = \frac{\text{количество правильных обнаружений}}{\text{общее количество обнаружений}} = \frac{2}{6} = 0.33.$$

- **Порог IOU = 0.5**

При этом пороге у трех обнаружений IOU превышает 0.5:

$$\text{Точность}_{0.5} = \frac{\text{количество правильных обнаружений}}{\text{общее количество обнаружений}} = \frac{3}{6} = 0.5.$$

- **Порог IOU = 0.1**

На этот раз получаем четыре правильных обнаружения:

$$\text{Точность}_{0.1} = \frac{\text{количество правильных обнаружений}}{\text{общее количество обнаружений}} = \frac{4}{6} = 0.67.$$

Как вы, возможно, заметили, главный недостаток этой метрики заключается в том, что точность изменяется в зависимости от порога IOU. Трудно понять общее качество модели по оценке точности для конкретного порога IOU. Средняя точность снимает это ограничение.

Средняя точность (AP)

Эта метрика вычисляет показатели точности для разных порогов IOU и рассчитывает их среднее значение. Формула средней точности выглядит так:

$$AP = \int_0^1 P(r)dr,$$

где $P(r)$ — точность для порога IOU, равного r .

Эту формулу можно аппроксимировать дискретным суммированием по заранее определенному списку порогов. Например, в бенчмарке Pascal VOC2008 [14] AP вычисляется по 11 пороговым значениям, разделенным равными интервалами:

$$AP = \frac{1}{11} \sum_{n=0}^{n=10} P(n).$$

AP отражает общую точность модели для конкретного класса объектов (например, человеческих лиц). Чтобы измерить общую точность модели по всем классам объектов (например, лицам и номерным знакам), следует использовать усредненную среднюю точность.

Усредненная средняя точность (mAP)

Эта метрика вычисляется как среднее значение AP по всем классам объектов и обобщает эффективность модели в целом. Формула выглядит так:

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^C \text{AP}_c,$$

где C — общее количество классов объектов, которые обнаруживает модель.

С помощью mAP обычно оцениваются системы обнаружения объектов. Чтобы узнать, какие пороги используются в стандартных бенчмарках, обращайтесь к [15], [16].

Оперативные метрики

Согласно требованиям, система должна обеспечивать конфиденциальность людей. Один из способов оценить этот показатель — подсчитывать количество обращений и жалоб пользователей. Также можно поручить операторам выборочно проверять долю некорректно размытых изображений. Важны и другие метрики, которые измеряют смещение и достоверность. Например, система должна одинаково хорошо размывать лица людей, относящихся к разным расам и возрастным группам. Но, как было зафиксировано в требованиях, измерение смещения выходит за рамки задачи.

Подведем итог раздела, посвященного оценке: в качестве автономных метрик мы выберем mAP и AP. mAP измеряет общую точность модели, тогда как AP дает представление об ее точности в конкретных классах. Главной метрикой оперативной оценки будут обращения пользователей.

Эксплуатация

В этом разделе мы сначала поговорим о типичной проблеме, которая встречается в системах обнаружения объектов, — перекрытии ограничительных прямоугольников. Затем рассмотрим общую структуру системы МО.

Перекрытие ограничительных прямоугольников

Алгоритм обнаружения объектов на изображении часто приводит к тому, что ограничительные прямоугольники перекрываются. Дело в том, что сеть RPN предлагает для каждого объекта несколько ограничительных прямоугольников с высокой степенью перекрытия. В процессе анализа важно сократить их количество до одного на объект.

Для этого часто применяется алгоритм NMS (Non-Maximum Suppression, «погашение не-максимумов») [17]. Его суть описана в следующем разделе.

NMS

NMS — алгоритм постобработки, который отбирает наиболее подходящие ограничительные прямоугольники. Он оставляет ограничительные прямоугольники с высоким уровнем достоверности и устраниет перекрытия. Пример изображен на рис. 3.11.

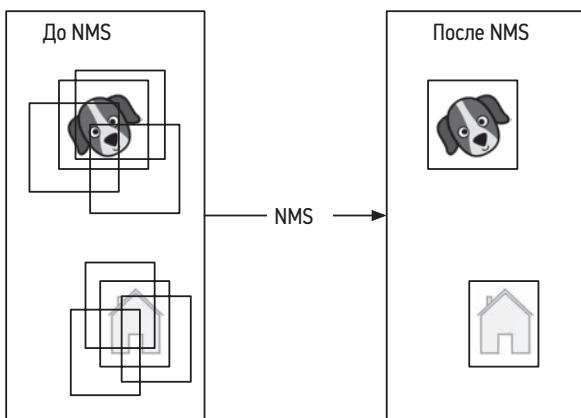


Рис. 3.11. До и после применения NMS

Об алгоритме NMS часто спрашивают на собеседованиях по проектированию систем МО, так что вам стоит хорошо понимать, как он работает [18].

Проектирование системы МО

На рис. 3.12 изображен предлагаемый дизайн системы МО для службы размытия.

Рассмотрим каждый пайплайн подробнее.

Пайплайн пакетных предсказаний

На основании собранных требований можно сделать вывод, что проблема задержки не критична, потому что пользователям можно показывать ранее обработанные изображения, пока обрабатываются новые. Поскольку результаты не требуется получать мгновенно, можно использовать пакетные предсказания и заранее вычислять результаты обнаружения объектов.

Предварительная обработка

Этот компонент предварительно обрабатывает «сырые» изображения. Сами операции предварительной обработки уже рассматривались в разделе, посвященном конструированию признаков.

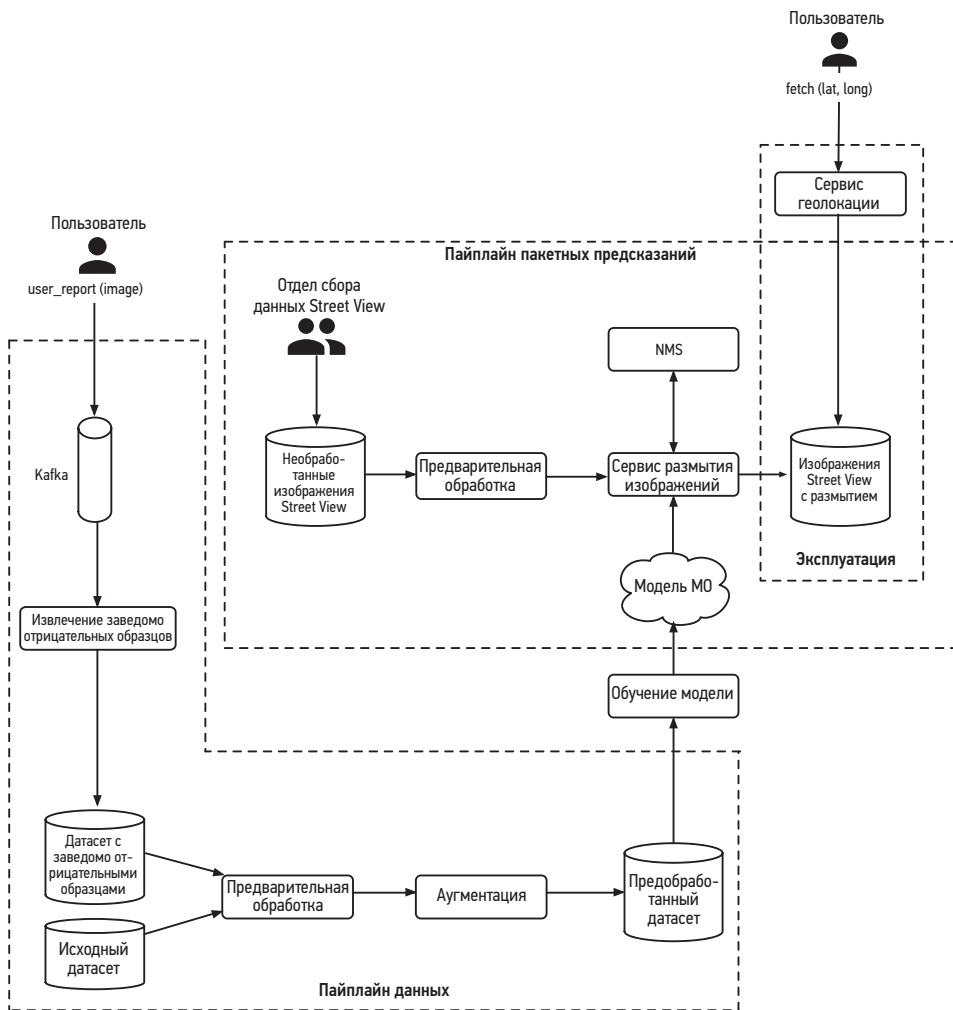


Рис. 3.12. Дизайн системы МО

Сервис размытия

Этот компонент обрабатывает каждое изображение Street View таким образом:

1. Формирует список объектов, которые обнаружены на изображении.
2. Очищает список обнаруженных объектов с помощью NMS.
3. Размывает обнаруженные объекты.
4. Сохраняет изображение с размытием в хранилище объектов (изображения Street View с размытием).

Обратите внимание, что в этой схеме сервис предварительной обработки отделен от сервиса размытия. Дело в том, что предварительная обработка изображений обычно интенсивно использует центральный процессор, а сервис размытия, как правило, задействует графический процессор. Разделение этих сервисов дает два преимущества:

- сервисы можно масштабировать независимо друг от друга в соответствии с нагрузкой каждого из них;
- ресурсы центрального и графического процессоров используются более эффективно.

Пайплайн данных

Этот пайплайн обрабатывает обращения пользователей, генерирует новые обучающие данные и подготавливает обучающие данные, которые будет использовать модель. Компоненты пайплайна данных в основном очевидны, дополнительных пояснений заслуживает только извлечение заведомо отрицательных образцов.

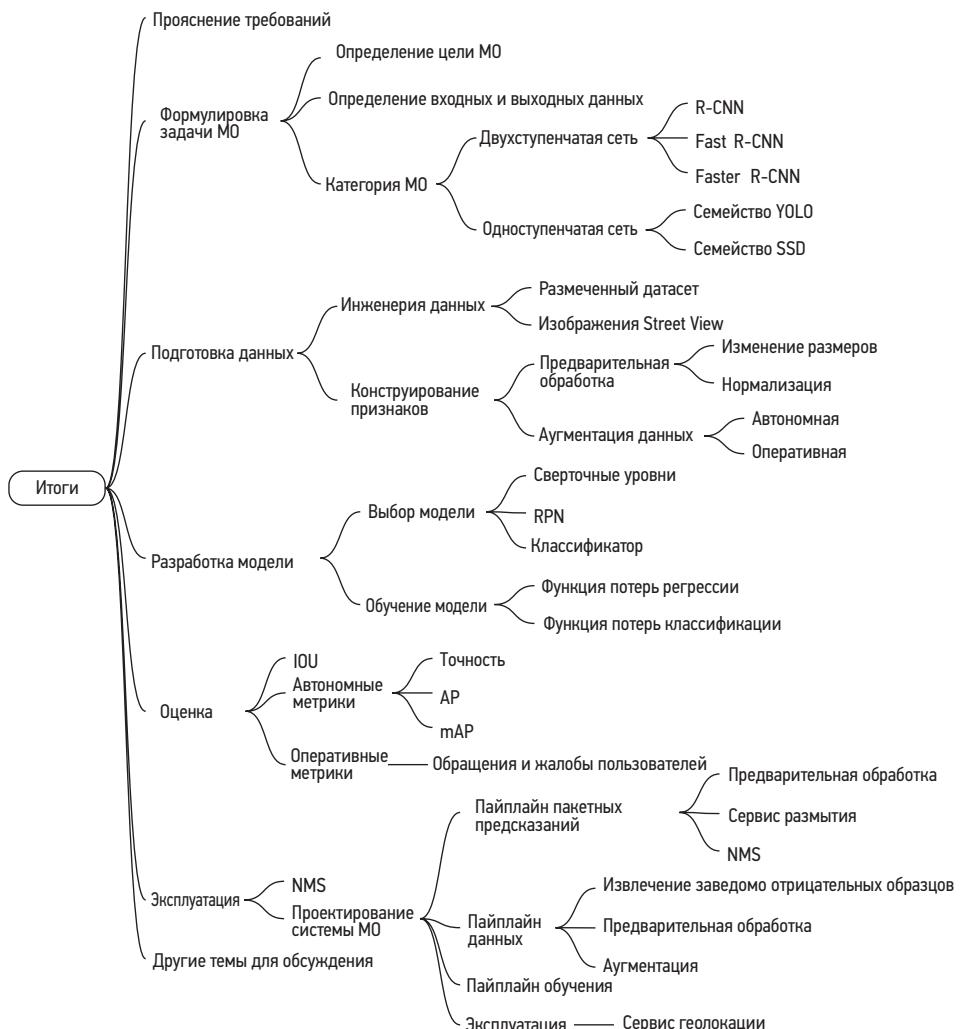
Извлечение заведомо отрицательных образцов. Заведомо отрицательными образцами называются специально подобранные примеры неправильных предсказаний, которые добавлены в обучающий датасет. Если дообучить модель на обновленном обучающем датасете, ее качество должно повыситься.

Другие темы для обсуждения

Если время позволяет, можно обсудить ряд дополнительных моментов.

- Чем архитектуры обнаружения объектов на базе Transformer отличаются от одно- и двухступенчатых моделей, каковы их достоинства и недостатки [19].
- Как методы распределенного обучения помогают улучшить обнаружение объектов на больших датасетах [20], [21].
- Как европейский Общий регламент по защите данных (GDPR) может повлиять на нашу систему [22].
- Как оценивать смещения в системах распознавания лиц [23], [24].
- Как осуществить непрерывную тонкую настройку (fine-tuning) модели [25].
- Как использовать активное обучение [26] или МО с оператором в контуре управления [27] при выборе данных для обучения.

Итоги



Ссылки

- [1] Google Street View. <https://www.google.com/streetview>
- [2] DETR. <https://github.com/facebookresearch/detr>
- [3] Семейство R-CNN. <https://lilianweng.github.io/posts/2017-12-31-object-recognition-part-3>
- [4] Fast R-CNN. <https://arxiv.org/pdf/1504.08083.pdf>
- [5] Faster R-CNN. <https://arxiv.org/pdf/1506.01497.pdf>
- [6] Семейство YOLO. <https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family>.
- [7] SSD. <https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06>
- [8] Методы аугментации данных. <https://www.kaggle.com/getting-started/190280>
- [9] CNN. https://en.wikipedia.org/wiki/Convolutional_neural_network
- [10] Обнаружение объектов. <https://dudeperf3ct.github.io/object/detection/2019/01/07/Mystery-of-Object-Detection>
- [11] Прямое и обратное распространение. <https://www.youtube.com/watch?v=qzPQ8cEsVK8>
- [12] MSE. https://en.wikipedia.org/wiki/Mean_squared_error
- [13] Логистическая функция потерь. https://en.wikipedia.org/wiki/Cross_entropy
- [14] Pascal VOC. <http://host.robots.ox.ac.uk/pascal/VOC/voc2008/index.html>
- [15] Оценка обнаружения объектов в наборе данных COCO. <https://cocodataset.org/#detection-eval>
- [16] Оценка обнаружения объектов. <https://github.com/rafaelpadilla/Object-Detection-Metrics>
- [17] NMS. <https://en.wikipedia.org/wiki/NMS>
- [18] Реализация NMS для Pytorch. <https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/>
- [19] Новые модели обнаружения объектов. <https://viso.ai/deep-learning/object-detection/>
- [20] Распределенное обучение в Tensorflow. https://www.tensorflow.org/guide/distributed_training
- [21] Распределенное обучение в Pytorch. https://pytorch.org/tutorials/beginner/dist_overview.html
- [22] GDPR и ML. <https://www.oreilly.com/radar/how-will-the-gdpr-impact-machine-learning>
- [23] Смещение и достоверность при обнаружении лиц. <http://sibgrapi.sid.inpe.br/col/sid.inpe.br/sibgrapi/2021/09.04.19.00/doc/103.pdf>
- [24] Достоверность AI. <https://www.kaggle.com/code/alexisbcook/ai-fairness>
- [25] Непрерывное обучение. <https://towardsdatascience.com/how-to-apply-continual-learning-to-your-machine-learning-models-4754adcd7f7f>
- [26] Активное обучение. [https://en.wikipedia.org/wiki/Active_learning_\(machine_learning\)](https://en.wikipedia.org/wiki/Active_learning_(machine_learning))
- [27] МО с оператором в контуре управления. <https://arxiv.org/pdf/2108.00941.pdf>

4 ПОИСК ВИДЕО НА YOUTUBE

На платформах видеохостинга, таких как YouTube, количество видеороликов быстро возрастает до миллиардов. В этой главе мы спроектируем систему поиска видео, которая может эффективно работать с таким объемом контента. Как показано на рис. 4.1, пользователь вводит текст в поле поиска, а система выводит список видеороликов, которые лучше всего соответствуют этому запросу.

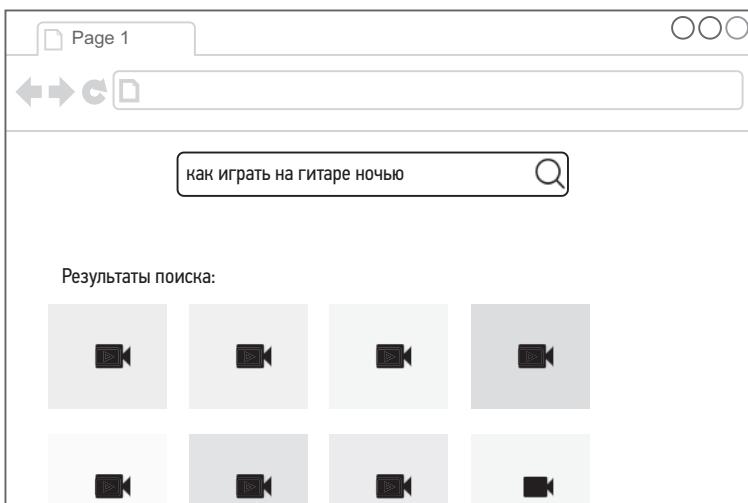


Рис. 4.1. Поиск видео по текстовому запросу

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: В поле поиска можно вводить только текст или пользователи могут искать по изображениям и видео?

Эксперт: Только текстовые запросы.

Соискатель: Контент на платформе представлен только в формате видео? Как насчет изображений или аудиофайлов?

Эксперт: На платформе размещается только видео.

Соискатель: Система поиска YouTube очень сложна. Можно ли предположить, что релевантность видео определяется исключительно его визуальным содержанием и сопутствующими текстовыми данными — например, названием и описанием?

Эксперт: Да, это справедливое допущение.

Соискатель: Доступны ли обучающие данные?

Эксперт: Да, будем считать, что доступны 10 миллионов пар типа (видео, текстовый запрос).

Соискатель: Должна ли система поиска поддерживать другие языки?

Эксперт: Для простоты будем считать, что поддерживается только один язык.

Соискатель: Сколько видеороликов доступно на платформе?

Эксперт: Один миллиард.

Соискатель: Нужно ли персонализировать результаты? Должны ли они ранжироваться по-разному для разных пользователей в зависимости от их предыдущих взаимодействий?

Эксперт: В отличие от рекомендательных систем, где персонализация абсолютно необходима, в поисковых системах персонализировать результаты не обязательно. Чтобы упростить задачу, будем считать, что персонализация не нужна.

Резюмируем описание проблемы. Требуется спроектировать систему поиска видео. На вход подается текстовый запрос, а на выходе отображается список видеороликов, которые соответствуют запросу. Чтобы найти релевантные ролики, мы будем использовать как их визуальное содержание, так и текстовые данные. Для обучения модели доступен датасет из 10 миллионов пар типа (видео, текстовый запрос).

Формулировка проблемы в виде задачи МО

Определение цели МО

Пользователи ожидают, что поисковые системы выдают релевантные и полезные результаты. Один из способов преобразовать эти ожидания в цель МО заключается в том, чтобы ранжировать видеоролики в зависимости от их релевантности текстовому запросу.

Определение входных и выходных данных системы

Как показано на рис. 4.2, система поиска принимает на вход текстовый запрос и выдает список видеороликов, отсортированных по релевантности этому запросу.

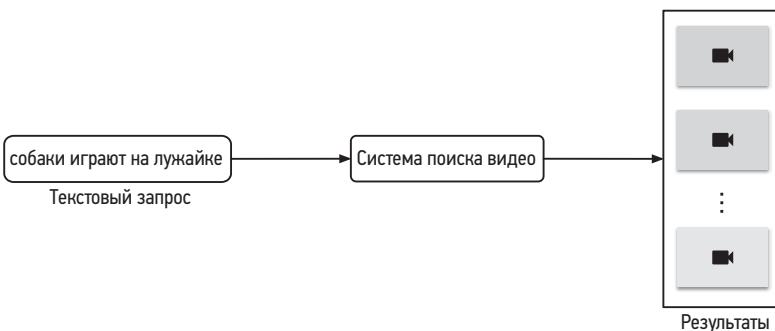


Рис. 4.2. Входные и выходные данные системы поиска видео

Выбор категории МО

Чтобы установить релевантность между видео и текстовым запросом, мы будем использовать как визуальное содержание, так и текстовые данные видео. Общий обзор архитектуры представлен на рис. 4.3.

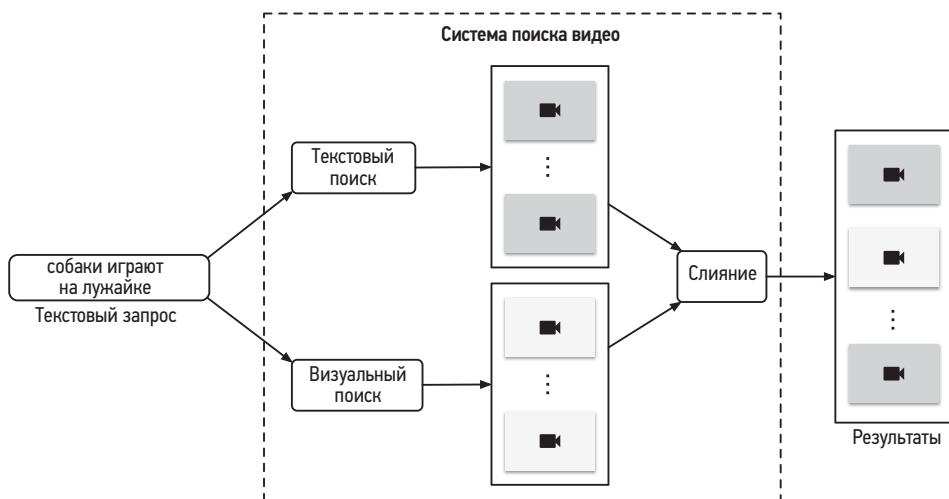


Рис. 4.3. Высокоуровневый обзор системы поиска

Кратко рассмотрим каждый из этих компонентов.

Визуальный поиск

Этот компонент принимает на вход текстовый запрос и выдает список видеороликов. Результаты ранжируются по степени сходства между текстовым запросом и визуальным содержанием.

Для поиска видеороликов по их визуальному содержанию часто применяется обучение представлений. В этом методе текстовый запрос и видео кодируются по отдельности с помощью двух кодировщиков. Как показано на рис. 4.4, модель М0 содержит кодировщик видео, который генерирует вектор эмбеддинга на основе содержания видео, а также текстовый кодировщик, генерирующий вектор эмбеддинга на основе текста. Метрика сходства между видео и текстом вычисляется через скалярное произведение их представлений.

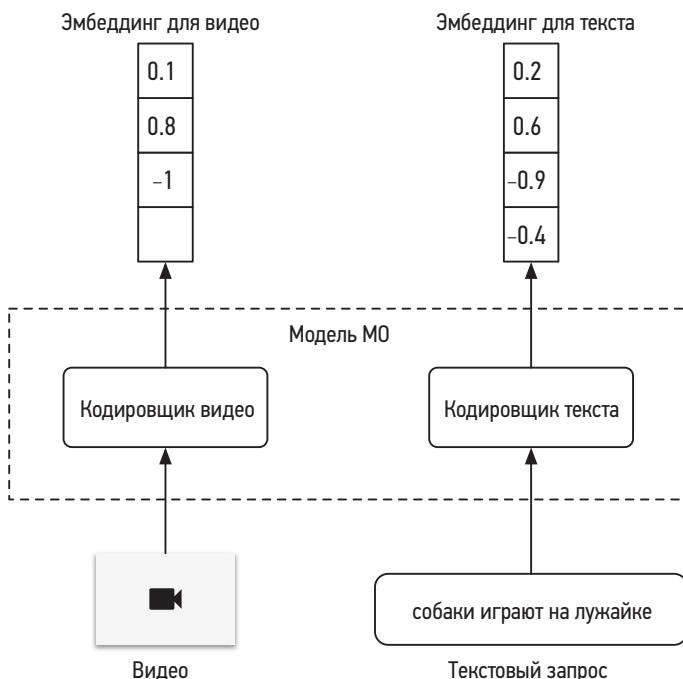


Рис. 4.4. Входные и выходные данные модели М0

Чтобы ранжировать видеоролики, которые визуально и семантически близки к текстовому запросу, мы вычисляем скалярное произведение между текстом и каждым видеороликом в пространстве эмбеддингов, а затем ранжируем видео на основании этой метрики сходства.

Текстовый поиск

На рис. 4.5 показано, как работает текстовый поиск, когда пользователь вводит текстовый запрос: «собаки играют на лужайке». В качестве результатов отображаются видеоролики, чьи названия, описания или теги наиболее похожи на текстовый запрос.

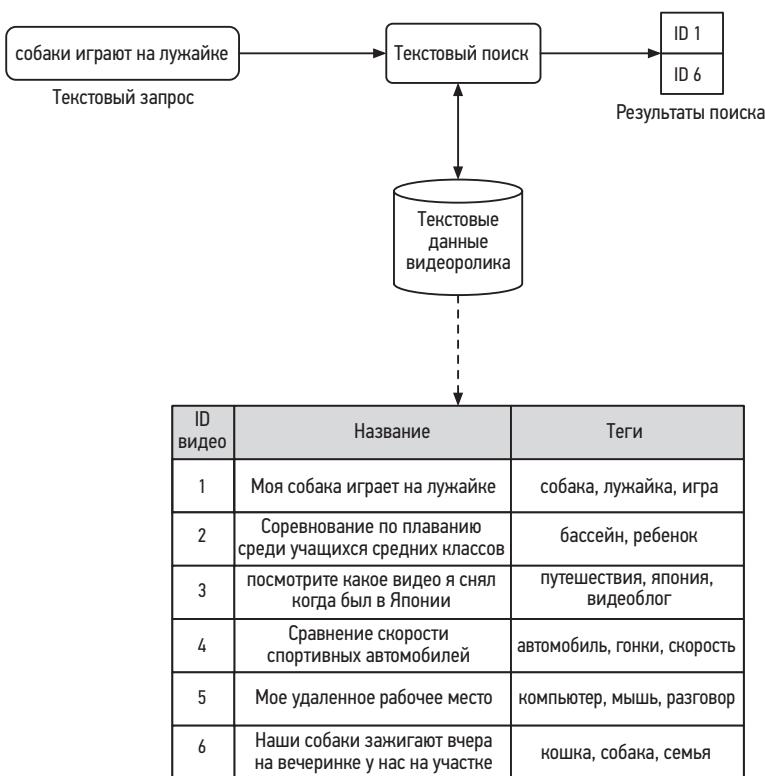


Рис. 4.5. Текстовый поиск

Стандартный метод создания текстового компонента поиска — обратный индекс, который позволяет эффективно проводить полнотекстовый поиск в базах данных. Поскольку обратная индексация не базируется на МО, она не требует затрат на обучение. Многие компании используют Elasticsearch — масштабируемый поисковый движок и хранилище документов. Чтобы подробнее узнать про Elasticsearch и глубже понять принципы его работы, обращайтесь к [1].

Подготовка данных

Инженерия данных

Поскольку у нас есть размеченный датасет для обучения и оценки модели, инженерия данных не требуется. В табл. 4.1 показано, как может выглядеть размеченный датасет.

Таблица 4.1. Размеченный датасет

Видео	Запрос	Тип выборки
76134.mp4	Дети плавают в бассейне!	Обучающая
92167.mp4	Отмечаем выпускной	Обучающая
2867.mp4	Ребята играют в футбол	Валидационная
28543.mp4	Как работает Tensorboard	Валидационная
70310.mp4	Поездка по зимней дороге	Тестовая

Конструирование признаков

Почти все алгоритмы МО принимают только числовые входные значения. На этапе конструирования признаков все неструктурированные данные, такие как текст и видео, должны быть преобразованы в числовое представление. Посмотрим, как подготовить текстовые и видеоданные для модели.

Подготовка текстовых данных

Как показано на рис. 4.6, текст обычно преобразуется в числовой вектор за три шага: нормализация текста, лексический анализ (токенизация) и преобразование токенов (лексем) в идентификаторы [2].

**Рис. 4.6.** Представление текста в виде числового вектора

Рассмотрим каждый шаг подробнее.

Нормализация текста

Нормализация текста обеспечивает единообразие слов и предложений. Например, слова «собака», «собаке» и «СОБАКА!» обозначают одно и то же, но записываются по-разному. То же можно сказать и о предложениях. Для примера возьмем такие два предложения:

- «Человек гуляет с собакой по Петербургу!»
- «человек, гуляющий со своей собакой в Санкт-Петербурге».

Оба предложения имеют одинаковый смысл, но различаются знаками препинания, формами слов и другими деталями. Вот распространенные методы нормализации текста.

- **Приведение к нижнему регистру:** все буквы переводятся в нижний регистр, потому что от этого не меняется смысл слов или предложений.
- **Удаление знаков препинания:** из текста удаляются все знаки препинания: точки, запятые, вопросительные и восклицательные знаки и т. д.
- **Усечение пробелов:** из текста удаляются начальные, конечные и сгруппированные пробельные символы.
- **Нормализация NFKD (совместимая декомпозиция)** [3]: комбинированные графемы раскладываются на комбинации простых графем.
- **Удаление диакритики:** из слов удаляются диакритические знаки: Málaga → Malaga, Noël → Noel
- **Лемматизация и выделение основ:** для набора взаимосвязанных форм слова определяется каноническая форма: *гуляет, гуляющий, гуляли* → *гулять*.

Токенизация

Лексический анализ (токенизация) — процесс разбиения текста на меньшие единицы, называемые лексемами (токенами). В общем случае применяются три разновидности токенизации.

- По словам: текст разбивается на отдельные слова по заданным ограничителям. Например, фраза вида «У меня завтра собеседование» превращается в [«У», «меня», «завтра», «собеседование»].
- По подсловам.
- По символам.

На собеседовании по проектированию систем МО обычно не вдаются в подробности алгоритмов токенизации. Если вам захочется узнать больше, обращайтесь к [4].

Преобразование лексем в идентификаторы

После того как мы получили лексемы, их надо преобразовать в числовые значения (идентификаторы). Это можно сделать двумя способами:

- таблица сопоставления;
- хеширование.

Таблица сопоставления. Каждой уникальной лексеме ставится в соответствие идентификатор, и все эти взаимно-однозначные соответствия сохраняются в таблице сопоставления. На рис. 4.7 показано, как она может выглядеть.

Слово	Иденти-фикатор
⋮	
автомобиль	18
⋮	
животные	35
⋮	
искусство	128
⋮	
путешествия	426
⋮	
страхование	1239
⋮	

Рис. 4.7. Таблица сопоставления

Хеширование. Этот метод вырабатывает идентификаторы с помощью хеш-функции, что позволяет обойтись без таблицы сопоставления и таким образом сэкономить память. На рис. 4.8 представлен пример преобразования слов в идентификаторы с помощью хеш-функции.

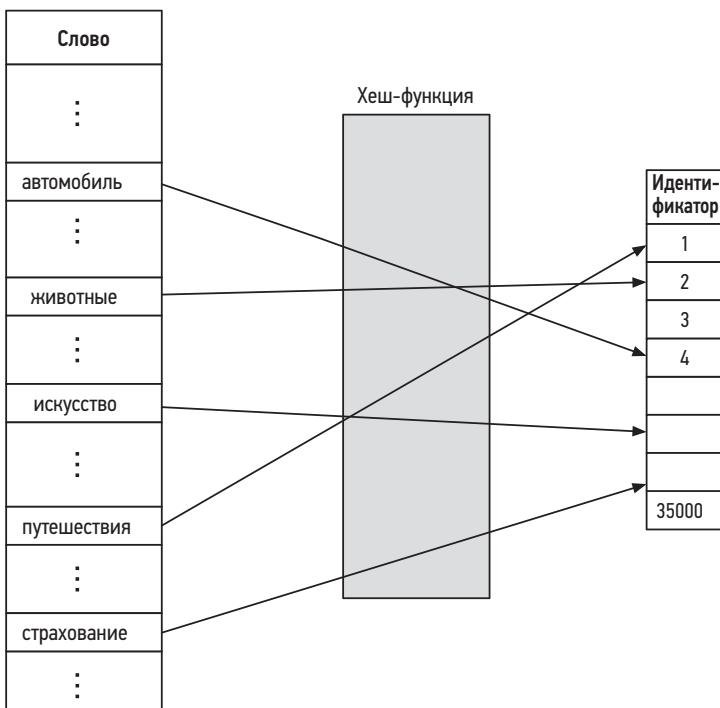


Рис. 4.8. Преобразование слов в идентификаторы с помощью хеширования

Сравним таблицу сопоставления с методом хеширования.

Таблица 4.2. Таблица сопоставления и хеширование признаков

	Таблица сопоставления	Хеширование
Скорость	Лексемы быстро преобразуются в идентификаторы	Чтобы преобразовывать лексемы в идентификаторы, нужно вычислять хеш-функцию
Преобразование идентификатора в лексемы	Идентификаторы просто преобразовать в лексемы по таблице обратного индекса	Невозможно обратно преобразовать идентификаторы в лексемы
Память	Таблица хранится в памяти. Чем больше лексем, тем больше нужно памяти	Хеш-функции достаточно, чтобы преобразовать любую лексему в ее идентификатор
Лексемы, не встречавшиеся ранее	Новые или не встречавшиеся ранее слова нельзя корректно обработать	Новые слова легко обрабатываются, потому что хеш-функцию можно применить к любому слову
Коллизии [5]	Нет проблем с коллизиями	Возможны коллизии

Подготовка видеоданных

На рис. 4.9 показан типичный процесс предварительной обработки видеоматериала.

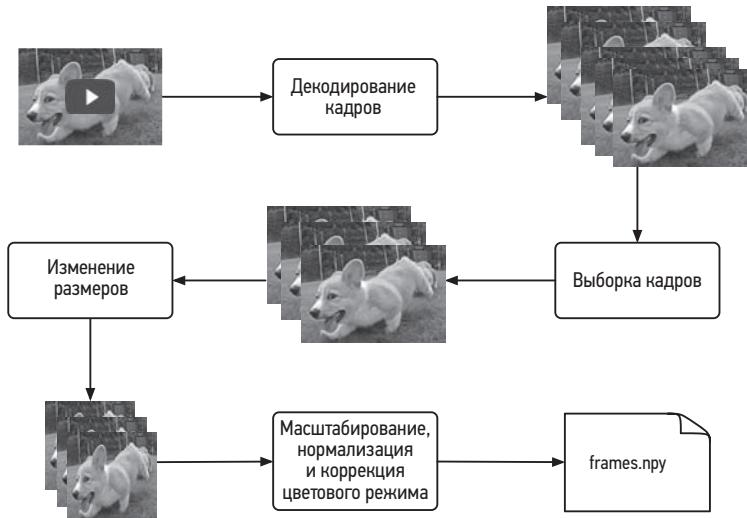


Рис. 4.9. Процесс предварительной обработки видео

Разработка модели

Выбор модели

Как упоминалось в разделе «Формулировка проблемы как задачи МО», текстовые запросы преобразуются в эмбеддинги кодировщиком текста, а видеоданные — кодировщиком видео. В этом разделе мы рассмотрим возможные архитектуры моделей для каждого кодировщика.

Кодировщик текста

Входные и выходные данные типичного кодировщика текста изображены на рис. 4.10.

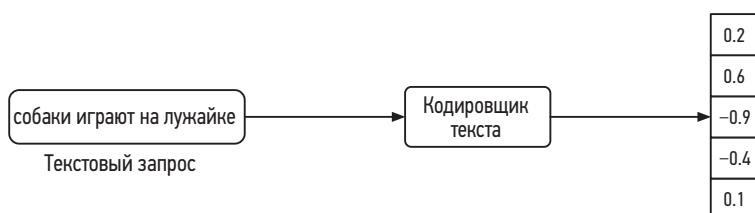


Рис. 4.10. Входные и выходные данные кодировщика текста

Кодировщик текста преобразует текст в векторное представление [6]. Например, если два предложения имеют сходный смысл, то их эмбеддинги более похожи. Кодировщики текста создаются либо статистическими методами, либо на основе МО. Рассмотрим оба варианта.

Статистические методы

Эти методы преобразуют предложения в векторы эмбеддингов с помощью статистики. Два самых популярных статистических метода:

- мешок слов (BoW, Bag of Words);
- TF-IDF (Term Frequency – Inverse Document Frequency, «частота слова – обратная частота документа»).

Мешок слов. Этот метод преобразует предложение в вектор фиксированной длины. Вхождения слов в предложения моделируются с помощью матрицы, строки которой представляют предложения, а столбцы – индексы слов. Пример BoW показан на рис. 4.11.

	гордо	доверять	звучать	очень	профессионал	хороший	человек	это
Это очень, очень хороший человек	0	0	0	2	0	1	1	1
Доверьте это хорошему профессиональному	0	1	0	0	1	1	0	1
Человек — это звучит гордо	1	0	1	0	0	0	1	1

Рис. 4.11. Представления разных предложений в мешке слов

Мешок слов – простой метод, который быстро вычисляет представления предложений, но у него есть ряд недостатков.

- Не учитывается порядок слов в предложении. Например, у предложений «утром деньги, вечером стулья» и «утром стулья, вечером деньги» в мешке слов будут одинаковые представления.
- Полученное представление не отражает семантического и контекстного смысла предложения. Например, у двух предложений с одинаковым смыслом, но разными словами будут совершенно разные представления.
- Вектор представления является разреженным. Его размерность равна общему количеству уникальных лексем. Обычно оно очень велико, так что каждое представление в основном заполнено нулями.

TF-IDF. Это числовая статистика, которая должна отражать важность слова в документе, принадлежащем коллекции или корпусу текстов. TF-IDF создает

такую же матрицу предложений и слов, как и мешок слов, но эта матрица нормализуется на основании частоты слов. Чтобы больше узнать о математических основах этого метода, обращайтесь к [7].

Поскольку TF-IDF назначает меньший вес часто встречающимся словам, представления этого метода обычно лучше, чем мешка слов. Однако здесь тоже есть свои ограничения:

- когда добавляется новое предложение, нужно выполнить нормализацию, чтобы заново вычислить частоты слов;
- порядок слов в предложении не учитывается;
- полученное представление не отражает семантический смысл предложения;
- представления получаются разреженными.

Таким образом, статистические методы обычно работают быстро, но не отражают контекстного смысла предложений, а представления разрежены. Методы на основе МО решают эти проблемы.

Методы на основе МО

В этих методах модель МО преобразует предложения в осмысленные эмбеддинги слов, так что расстояние между двумя эмбеддингами отражает семантическую близость соответствующих слов. Например, если два слова (скажем, «богатый» и «благосостояние») семантически близки, то их эмбеддинги располагаются близко в пространстве эмбеддингов. На рис. 4.12 представлена простая визуализация эмбеддингов слов в двухмерном пространстве эмбеддингов. Как видите, похожие слова группируются.

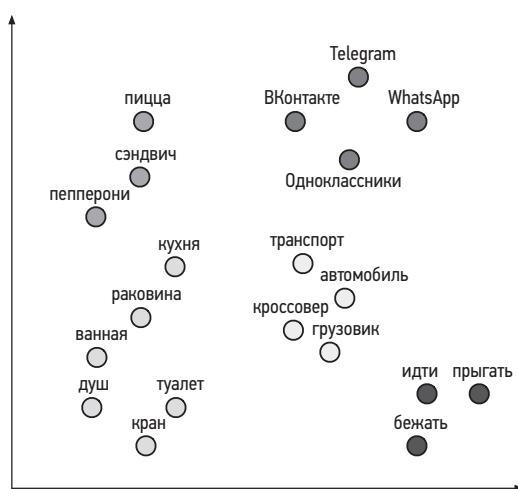


Рис. 4.12. Слова в двухмерном пространстве эмбеддингов

Существуют три распространенных метода преобразования текста в эмбеддинги, основанных на МО:

- слой сопоставления эмбеддингов;
- Word2vec;
- архитектуры на базе Transformer.

Слой сопоставления эмбеддингов

В этом методе добавляется слой, который ставит в соответствие каждому идентификатору вектор эмбеддинга. Пример показан на рис. 4.13.



Рис. 4.13. Слой сопоставления эмбеддингов

Слой сопоставления эмбеддингов — простое и эффективное решение, чтобы преобразовывать разреженные признаки (такие, как идентификаторы) в эмбеддинги фиксированной размерности. Примеры его использования еще встретятся в следующих главах.

Word2vec

Word2vec [8] — семейство взаимосвязанных моделей, с помощью которых создаются эмбеддинги слов. Эти модели базируются на архитектуре неглубокой нейронной сети и используют совместное вхождение слов в локальном контексте, чтобы обучаться на эмбеддингах слов. В частности, модель учится

предсказывать центральное слово по окружающим словам. После фазы обучения модель способна преобразовывать слова в содержательные эмбеддинги.

На Word2vec базируются две основные модели: CBOW (Continuous Bag of Words, «непрерывный мешок слов») [9] и Skip-gram [10]. На рис. 4.14 изображена высокоуровневая схема работы CBOW. Если вам захочется больше узнать об этих моделях, обращайтесь к [8].

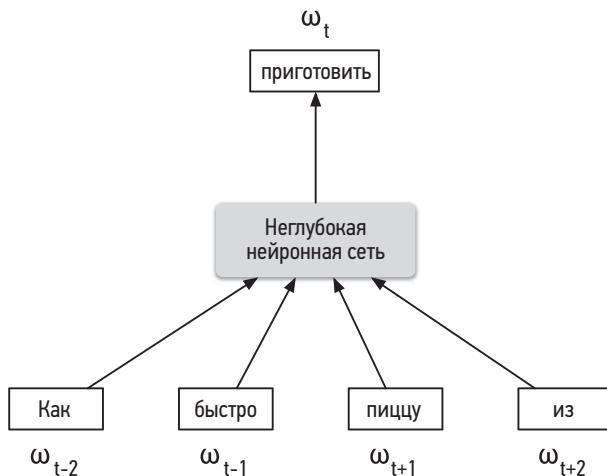


Рис. 4.14. Метод CBOW

Хотя Word2vec и слой сопоставления эмбеддингов просты и эффективны, свежие архитектуры на базе Transformer показывают многообещающие результаты.

Модели на базе Transformer

Преобразовывая слова в эмбеддинги, эти модели учитывают контекст слов в предложении. В отличие от моделей Word2vec, они производят разные эмбеддинги для одного и того же слова в зависимости от контекста.

На рис. 4.15 показана модель на базе Transformer, которая принимает на вход предложение (набор слов) и генерирует эмбеддинг для каждого слова.

Эти модели чрезвычайно эффективно распознают контекст и вырабатывают содержательные эмбеддинги. Некоторые модели, включая BERT [11], GPT3 [12] и BLOOM [13], продемонстрировали потенциал Transformer для разнообразных задач обработки естественного языка (NLP). В нашем случае для кодирования текста мы выберем архитектуру на базе Transformer — такую, как BERT.

На некоторых собеседованиях вам могут предложить рассмотреть модель на базе Transformer на более глубоком уровне. За дополнительной информацией обращайтесь к [14].

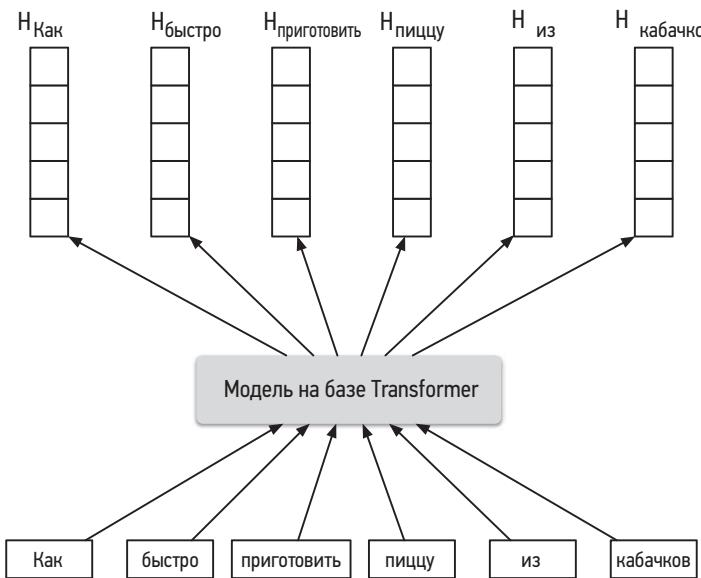


Рис. 4.15. Входные и выходные данные модели на базе Transformer

Кодировщик видео

Существует два архитектурных решения для кодирования видеоданных:

- модели на уровне видео;
- модели на уровне кадра.

Модели на уровне видео обрабатывают видеоролик целиком, чтобы создать эмбеддинг (рис. 4.16). Архитектура модели обычно базируется на 3D-свертках [15] или на Transformer. Поскольку видео обрабатывается целиком, модель требует значительных вычислительных ресурсов.

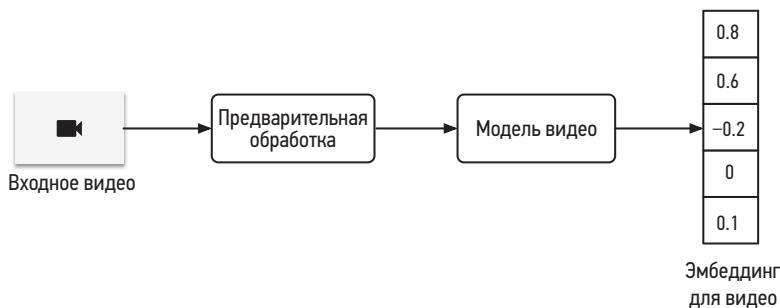


Рис. 4.16. Модель на уровне видео

Модели на уровне кадра работают иначе. В этом случае эмбеддинг для видео вырабатывается в три шага (рис. 4.17):

- выполнить предварительную обработку видео и сформировать выборку кадров;
- запустить модель на выборочных кадрах и создать их эмбеддинги;
- агрегировать (например, усреднить) эмбеддинги кадров, чтобы получить эмбеддинг для видео.

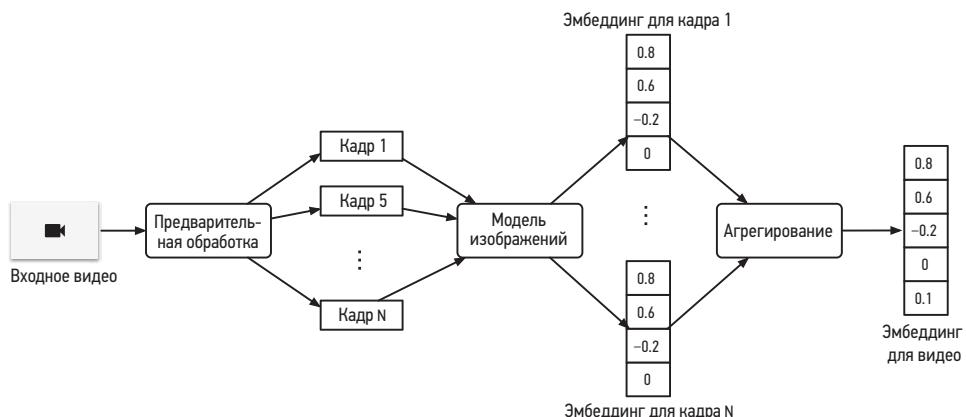


Рис. 4.17. Модель на уровне кадра

Так как эта модель действует на уровне кадров, она часто работает быстрее и требует меньших вычислительных затрат. Однако модели на уровне кадров обычно не понимают аспектов развития во времени, таких как действия и перемещения на видео. На практике модели на уровне кадра предпочтительнее во многих случаях, когда не стоит задача анализировать развитие во времени.

В данном случае мы воспользуемся моделью на уровне кадра (такой, как ViT [16]), чтобы обеспечить два преимущества:

- сократить скорость обучения и обслуживания запросов;
- сократить объем вычислений.

Обучение модели

Чтобы обучить кодировщики текста и видео, мы применим метод контрастного обучения. Если вам захочется больше узнать об этом, обратитесь к разделу «Обучение модели» в главе 2.

На рис. 4.18 показано, как вычисляются потери в ходе обучения модели.

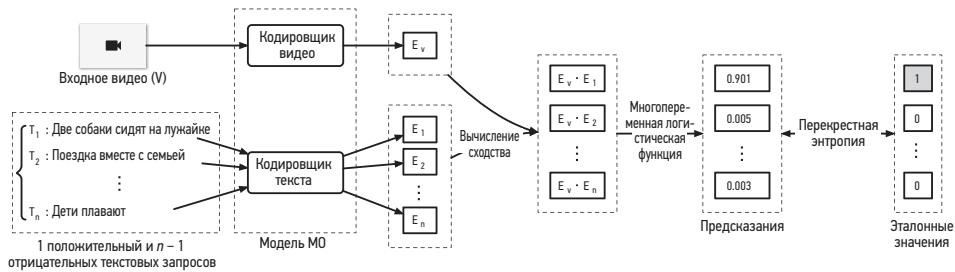


Рис. 4.18. Вычисление потерь

Оценка

Автономные метрики

Рассмотрим наиболее актуальные из автономных метрик, которые часто применяются в поисковых системах.

Точность на k (precision@k) и mAP. Вот формула:

$$\text{precision}@k = \frac{\text{количество релевантных элементов среди первых } k \text{ элементов ранжированного списка}}{k}.$$

В валидационном наборе данных каждому текстовому запросу соответствует только один видеоролик. Это значит, что числитель формулы не превышает 1, что приводит к низким значениям precision@k. Например, даже если видео, соответствующее тому или иному текстовому запросу, ранжируется в начале списка, точность на 10 элементах составит всего 0.1. Из-за этого метрики точности, такие как точность на k и mAP, оказываются не особенно полезными.

Полнота на k (recall@k). Метрика измеряет отношение количества релевантных видеороликов в результатах поиска к общему количеству релевантных видеороликов.

$$\text{recall}@k = \frac{\text{количество релевантных видеороликов среди первых } k \text{ элементов видеороликов}}{\text{общее количество релевантных видеороликов}}.$$

Как упоминалось ранее, общее количество релевантных видеороликов всегда равно 1. С учетом этого можно преобразовать формулу полноты на k в следующую форму:

Полнота на k равна 1, если релевантный видеоролик входит в первые k видеороликов, и равна 0 в противном случае.

Каковы достоинства и недостатки этой метрики?

Достоинства

- Она эффективно измеряет способность модели найти видео, которое соответствует заданному текстовому запросу.

Недостатки

- Она зависит от k , а выбрать правильное значение k может быть непросто.
- Если релевантный видеоролик не входит в первые k видеороликов в выходном списке, полнота на k всегда равна 0. Для примера возьмем случай, когда модель A ранжирует релевантное видео в позиции 15, а модель B ранжирует то же видео в позиции 50. Если оценивать качество этих двух моделей с помощью полноты на k , то в обоих случаях полнота на 10 равна 0, несмотря на то что модель A лучше модели B .

Среднеобратный ранг (MRR). Эта метрика измеряет качество модели, усредняя ранг первого релевантного элемента в каждой поисковой выдаче. Формула выглядит так:

$$\text{MRR} = \frac{1}{m} \sum_{i=1}^m \frac{1}{\text{rank}_i}.$$

MRR преодолевает недостатки полноты из k , так что в нашем примере его можно использовать как автономную метрику.

Оперативные метрики

В процессе оперативной оценки компании отслеживают широкий диапазон метрик. Рассмотрим самые важные из них:

- кликабельность (CTR);
- процент просмотров до конца;
- общее время просмотра видео из поисковой выдачи.

Кликабельность (CTR, Click-Through Rate) показывает, как часто пользователи нажимают на видеоролики из результатов поиска. Главный недостаток CTR заключается в том, что эта метрика не следит за тем, насколько открытые видеоролики релевантны для пользователя. Несмотря на это, от CTR есть польза, потому что она сообщает, сколько пользователей открыли видеоролики из поисковой выдачи.

Процент просмотров до конца — доля просмотренных до конца видеороликов из результатов поиска. Недостаток этой метрики в том, что пользователь может просмотреть видео только частично, но все равно посчитать его релевантным. Сама по себе метрика не отражает релевантности результатов поиска.

Общее время просмотра видео из поисковой выдачи — общее время, в течение которого пользователи просматривали видеоролики из результатов поиска. Пользователи склонны проводить больше времени за просмотром, если результаты поиска им интересны. Эта метрика — хороший показатель того, насколько эти результаты релевантны.

Эксплуатация

В режиме эксплуатации система выводит ранжированный список видеороликов, релевантных заданному текстовому запросу. На рис. 4.19 изображен упрощенный дизайн системы МО.

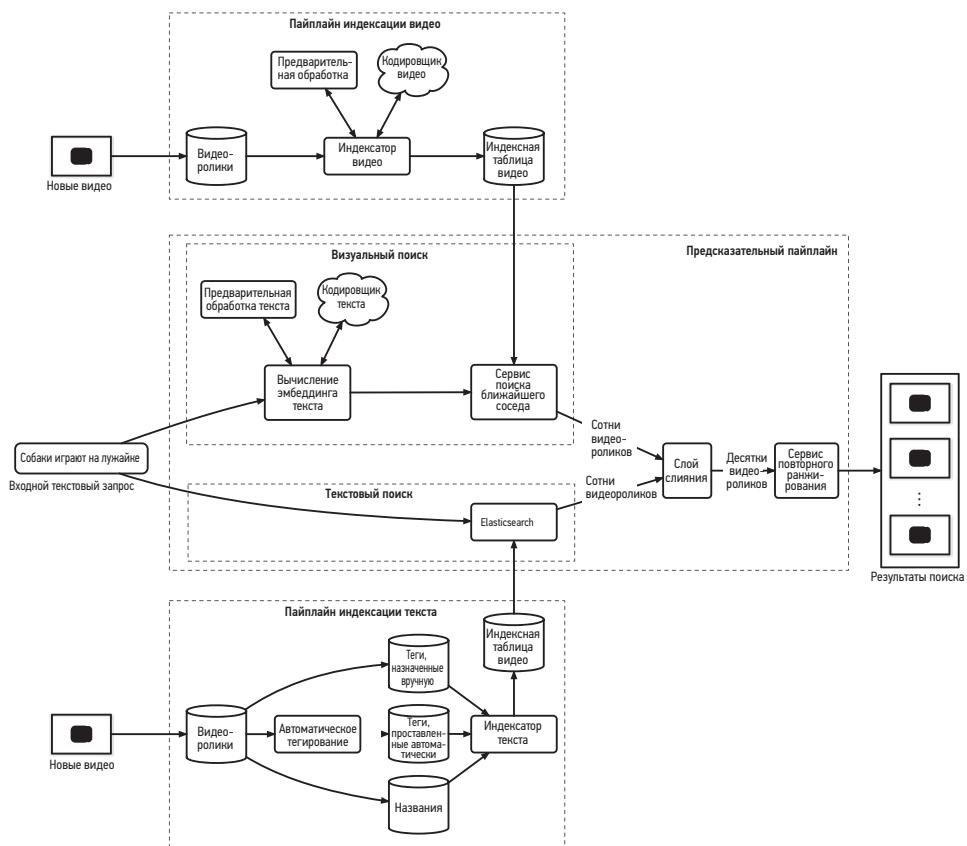


Рис. 4.19. Дизайн системы МО

Рассмотрим каждый пайпайн подробнее.

Предсказательный пайплайн

Этот пайплайн состоит из следующих компонентов:

- визуальный поиск;
- текстовый поиск;
- слой слияния;
- сервис повторного ранжирования.

Визуальный поиск кодирует текстовый запрос и использует сервис поиска ближайшего соседа, чтобы найти эмбеддинги видео, наиболее близкие к эмбеддингу текста. Поиск ускоряется с помощью алгоритмов приближенного поиска ближайшего соседа (ANN), которые упоминались в главе 2.

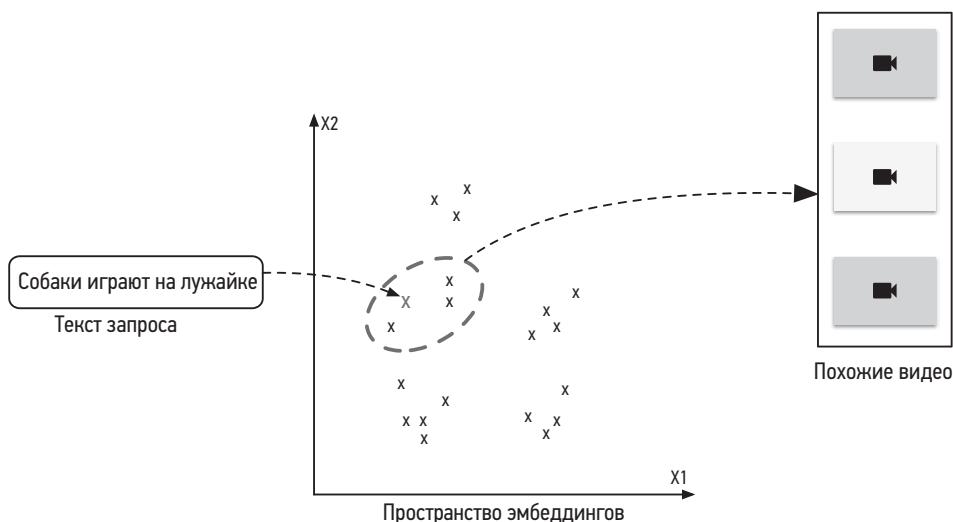


Рис. 4.20. Получение трех наиболее релевантных результатов для заданного текстового запроса

Текстовый поиск с помощью Elasticsearch ищет видео с названиями и тегами, которые подходят к текстовому запросу.

Слой слияния принимает два отдельных списка релевантных видеороликов, которые получены на предыдущих шагах, и объединяет их в новый список.

Слой слияния можно реализовать двумя способами. Простейший из них — повторно ранжировать видео на основании взвешенной суммы их предсказанных показателей релевантности. Более сложный подход состоит в том, чтобы внедрить дополнительную модель для повторного ранжирования видео; он требует

больших затрат из-за того, что модель нужно обучать. Кроме того, при этом снижается скорость обслуживания. В результате мы будем использовать первый вариант.

Сервис повторного ранжирования изменяет ранжированный список видеороликов, применяя бизнес-логику и политики.

Пайплайн индексации видео

Обученный кодировщик видео вычисляет эмбеддинги видеороликов, которые затем индексируются. Индексированные эмбеддинги видео используются сервисом поиска ближайшего соседа.

Пайплайн индексации текста

Этот пайплайн с помощью Elasticsearch индексирует названия и теги — как назначенные вручную, так и проставленные автоматически.

Загружая новые видеоролики, пользователи обычно добавляют теги, благодаря которым видео проще идентифицировать. Но что, если пользователь не добавил теги? Одно из возможных решений — генерировать теги отдельной моделью. Эта служба автоматического тегирования особенно полезна в ситуациях, когда у видео нет вручную назначенных тегов. В тегах, проставленных автоматически, может быть больше шума, чем в пользовательских, но они все равно оказываются полезны.

Другие темы для обсуждения

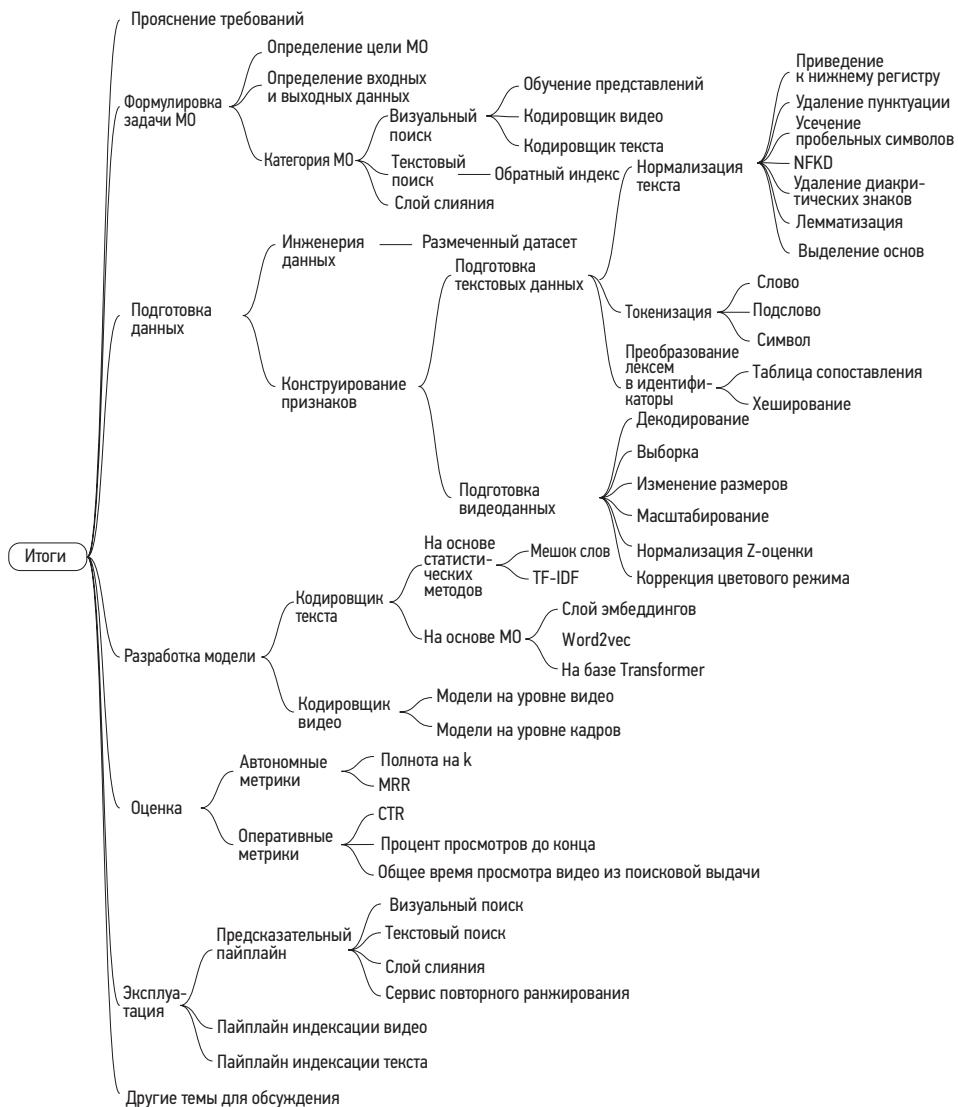
Прежде чем завершить эту главу, отметим, что в нашем примере мы основательно упростили структуру системы поиска видео. На практике такие системы устроены намного сложнее. Вот некоторые возможные усовершенствования.

- Реализовать многоэтапную структуру (отбор кандидатов и ранжирование).
- Использовать другие признаки видео: продолжительность, популярность и т. д.
- Вместо того чтобы полагаться на уже размеченные данные, можно конструировать и размечать данные в зависимости от взаимодействий (кликов, лайков и т. д.). Таким образом обеспечивается непрерывное обучение модели.
- С помощью модели МО искать названия и теги, семантически сходные с текстовыми запросами. Этую модель можно объединить с Elasticsearch, чтобы повысить качество поиска.

Если в конце собеседования останется время, можно обсудить дополнительные темы.

- Важный аспект поисковых систем — понимание запроса: система исправляет орфографию, идентифицирует категорию запроса и распознает сущности. Как создать компонент, который анализировал бы смысл запроса? [17].
- Как построить мультимодальную систему, которая обрабатывает речь и аудио, чтобы улучшить результаты поиска? [18]
- Как добавить в эту задачу поддержку других языков? [19]
- Слишком похожие видео в поисковой выдаче могут ухудшить впечатления пользователей от работы с системой. Как обнаружить слишком похожие видео, чтобы удалить их перед тем, как выводить результаты? [20]
- Текстовые запросы можно разделить на основные (head), дополнительные (torso) и шлейфовые (tail). Какие подходы обычно применяются для каждого из этих типов? [21]
- Как учитывать популярность и свежесть данных при составлении выходного списка? [22]
- Как работают реальные системы поиска видео? [23], [24], [25]

Итоги



Ссылки

- [1] Elasticsearch. https://www.tutorialspoint.com/elasticsearch/elasticsearch_query_dsl.htm
- [2] Предварительная обработка текстовых данных. <https://huggingface.co/docs/transformers/preprocessing>
- [3] Нормализация NFKD. <https://unicode.org/reports/tr15/>
- [4] Что такое лексический анализ. https://huggingface.co/docs/transformers/tokenizer_summary
- [5] Коллизии при хешировании. https://en.wikipedia.org/wiki/Hash_collision
- [6] Глубокое обучение для NLP. http://cs224d.stanford.edu/lecture_notes/notes1.pdf
- [7] TF-IDF. <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>
- [8] Модели Word2Vec. <https://www.tensorflow.org/tutorials/text/word2vec>
- [9] Непрерывный «мешок слов». <https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-cbow.html>
- [10] Модель Skip-gram. <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>
- [11] Модель BERT. <https://arxiv.org/pdf/1810.04805.pdf>
- [12] Модель GPT3. <https://arxiv.org/pdf/2005.14165.pdf>
- [13] Модель BLOOM. <https://bigscience.huggingface.co/blog/bloom>
- [14] Реализация Transformer с нуля. <https://peterbloem.nl/blog/transformers>.
- [15] 3D-свертки. <https://www.kaggle.com/code/shivamb/3d-convolutions-understanding-use-case/notebook>
- [16] Vision Transformer (ViT). <https://arxiv.org/pdf/2010.11929.pdf>
- [17] Понимание запросов в поисковых системах. <https://www.linkedin.com/pulse/ai-query-understanding-daniel-tunkelang/>.
- [18] Мультимодальное обучение представлениям для видео. <https://arxiv.org/pdf/2012.04124.pdf>
- [19] Многоязыковые модели. <https://arxiv.org/pdf/2107.00676.pdf>
- [20] Выявление слишком похожих видео. <https://arxiv.org/pdf/2005.07356.pdf>
- [21] Обобщаемая релевантность поиска. <https://livebook.manning.com/book/ai-powered-search/chapter-10/v-10/20>
- [22] Свежесть данных в поисковых и рекомендательных системах. <https://developers.google.com/machine-learning/recommendation/dnn/re-ranking>
- [23] Семантический поиск товаров в Amazon. <https://arxiv.org/pdf/1907.00937.pdf>
- [24] Релевантность ранжирования в поиске Yahoo. <https://www.kdd.org/kdd2016/papers/files/adf0361-yinA.pdf>
- [25] Семантический поиск товаров в онлайн-торговле. <https://arxiv.org/pdf/2008.08180.pdf>

5

ОБНАРУЖЕНИЕ ВРЕДОНОСНОГО КОНТЕНТА

Во многих социальных сетях — например, Facebook [1], LinkedIn [2] и Twitter [3] — действуют стандартные правила этичного поведения и принимаются соответствующие меры безопасности. Эти правила запрещают некоторые формы поведения, вредоносную деятельность и контент, который способен нанести ущерб сообществу. Подобным платформам очень важно иметь наготове технологии и ресурсы, которые позволяют выявлять опасный контент и нарушителей.

Нарушения этических принципов можно разделить на две категории.

- **Вредоносный контент:** призывы к насилию, непристойные изображения, демонстрация самоповреждения, разжигание ненависти и т. д.
- **Вредоносные действия и участники:** фиктивные учетные записи, спам, фишинг, организованный троллинг и другое опасное поведение.

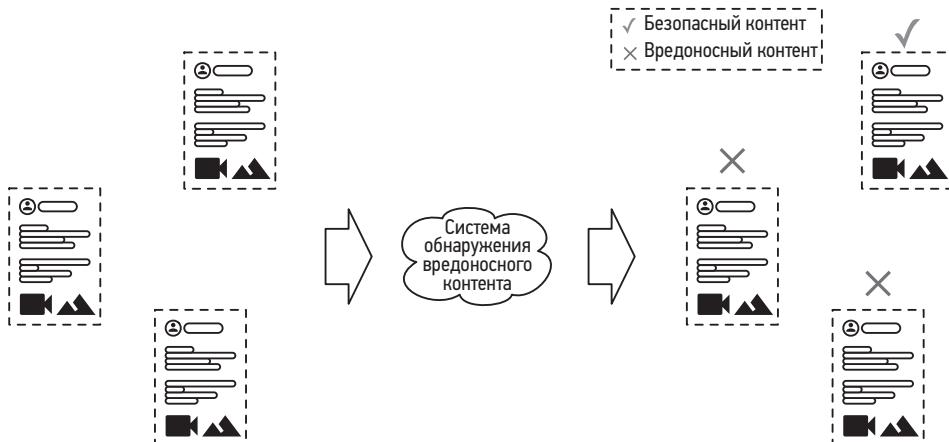


Рис. 5.1. Система обнаружения вредоносного контента

В этой главе мы сосредоточимся на обнаружении постов, которые могут содержать вредоносный контент, а именно мы спроектируем систему, которая

активно отслеживает новые посты и выявляет контент, нарушающий правила платформы, после чего удаляет его или понижает его приоритет. Чтобы узнать, как реальные компании реализуют системы обнаружения вредоносного контента, обращайтесь к [4], [5], [6].

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: Должна ли система обнаруживать не только вредоносный контент, но и участников, нарушающих правила?

Эксперт: И то и другое одинаково важно. Для простоты будем обнаруживать только вредоносный контент.

Соискатель: Мы рассматриваем только текстовые посты или система должна также работать с изображениями и видео?

Эксперт: В посте может быть текст, изображения, видео или любые их сочетания.

Соискатель: Какие языки должны поддерживаться?

Эксперт: Система должна обнаруживать вредоносный контент на разных языках. Для простоты можно считать, что у нас есть предварительно обученная многоязыковая модель, которая работает с эмбеддингами текстового контента.

Соискатель: Какие именно категории вредоносного контента должна выявлять модель? Мне приходит в голову насилие, нагота, разжигание ненависти, дезинформация... Есть ли другие категории, которые тоже надо учитывать?

Эксперт: Отлично, вы перечислили все основные категории. Дезинформация — более сложная и неоднозначная тема, так что для простоты не будем ее затрагивать.

Соискатель: Есть ли у нас люди, которые могут расставлять метки вручную?

Эксперт: На платформе размещается более 500 миллионов постов в день. Размечать их все вручную было бы слишком затратно как по средствам, так и по времени. Однако можно считать, что ручная разметка доступна для ограниченного количества постов — скажем, 10 000 в день.

Соискатель: Если пользователи могут сообщать о вредоносном контенте, это помогает выявить слабые места системы. Можно ли считать, что в системе реализована такая возможность?

Эксперт: Верно замечено. Да, пользователи могут сообщать о вредоносных постах.

Соискатель: Должны ли мы объяснять, почему система посчитала пост вредоносным и удалила?

Эксперт: Да. Крайне важно объяснять пользователям, почему мы удаляем тот или иной контент. Тогда они будут следить за тем, чтобы их будущие посты соответствовали нашим правилам.

Соискатель: Насколько быстро система должна реагировать? Нужны ли предсказания в реальном времени? То есть должна ли система немедленно обнаруживать вредоносный контент и блокировать его, или она может работать в пакетном режиме, то есть автономно выявлять вредоносный контент ежесекундно или ежедневно?

Эксперт: Это очень важный вопрос. Как вы сами думаете?

Соискатель: Мне кажется, требования могут зависеть от типа контента. Например, посты с насилием могут требовать немедленного реагирования, а для прочего контента может подойти и отложенное обнаружение.

Эксперт: Вполне разумные предположения.

Резюмируем описание проблемы. Мы проектируем систему обнаружения вредоносного контента, которая выявляет вредоносные посты, а затем удаляет их или понижает их приоритет, сообщая пользователю, почему она сочла пост вредоносным. Содержимым поста может быть текст, изображения, видео или их произвольное сочетание, и контент может создаваться на разных языках. У пользователей есть возможность сообщать о вредоносных постах.

Формулировка проблемы в виде задачи МО

Определение цели МО

Пускай целью МО будет точно предсказывать вредоносные посты. Если их удается точно обнаруживать, то их можно удалять или деприоритизировать, что улучшает безопасность платформы.

Определение входных и выходных данных системы

Система принимает на вход пост и возвращает вероятность того, что он является вредоносным (рис. 5.2).

Присмотримся повнимательнее к входному посту. Как показано на рис. 5.3, он может содержать разнородную информацию и быть мультимодальным.

Чтобы делать точные предсказания, система должна учитывать все модальности. Для объединения разнородных данных часто применяются два метода слияния: позднее слияние и раннее слияние.



Рис. 5.2. Входные и выходные данные системы обнаружения вредоносного контента

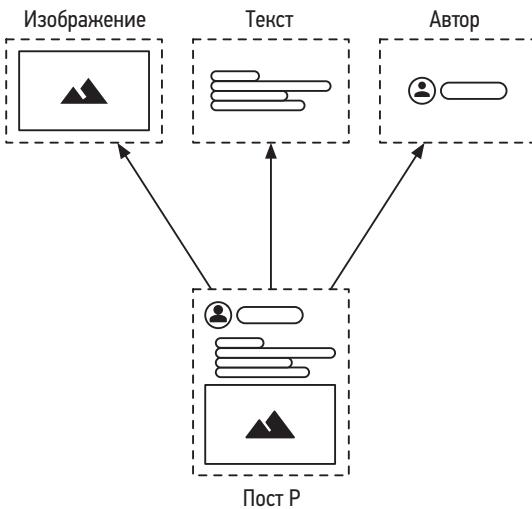


Рис. 5.3. Пост с разнородными данными

Позднее слияние

При позднем слиянии модели МО обрабатывают разные виды данных (модальности) независимо друг от друга, а затем объединяют предсказания, чтобы получить окончательное предсказание. На рис. 5.4 показано, как это работает.

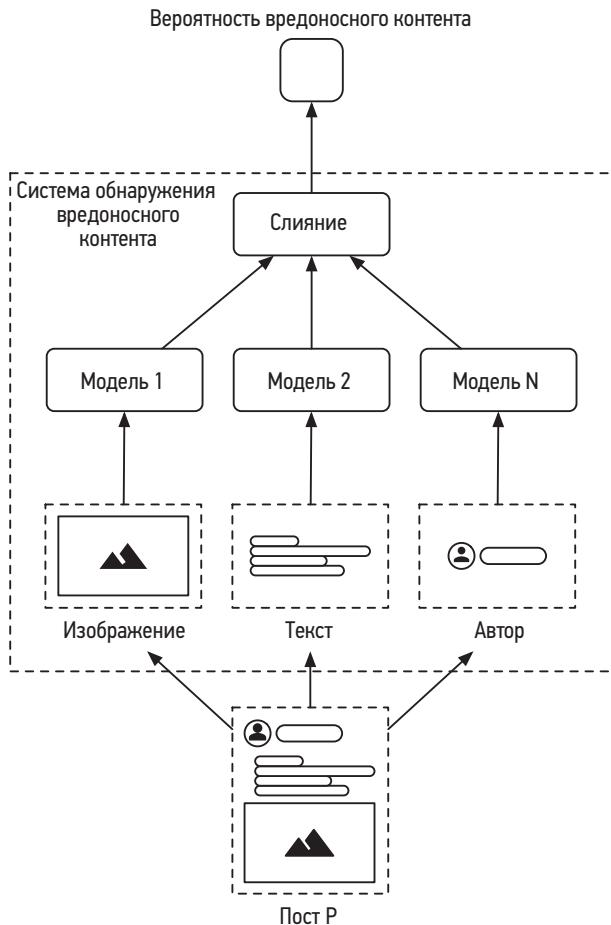


Рис. 5.4. Позднее слияние

Преимущество позднего слияния заключается в том, что каждую модель можно обучать, оценивать и улучшать независимо от других.

Однако у позднего слияния есть два серьезных недостатка. Во-первых, чтобы обучать отдельные модели, нужны разные обучающие данные для каждой модальности, а чтобы получить эти данные, могут потребоваться значительные затраты времени и средств.

Во-вторых, комбинация разных модальностей может оказаться вредоносной, при том что данные каждого вида сами по себе безобидны: часто так бывает с мемами, где изображения объединяются с текстом. В таких случаях позднее слияние не предскажет, является ли контент вредоносным. Обрабатывая каждую отдельную модальность, модели правильно предсказывают безопасность

контента, после чего уровень слияния делает вывод об общей безопасности, который оказывается неверным.

Раннее слияние

При раннем слиянии модальности сначала объединяются, а затем модель делает предсказание. На рис. 5.5 показано, как это работает.

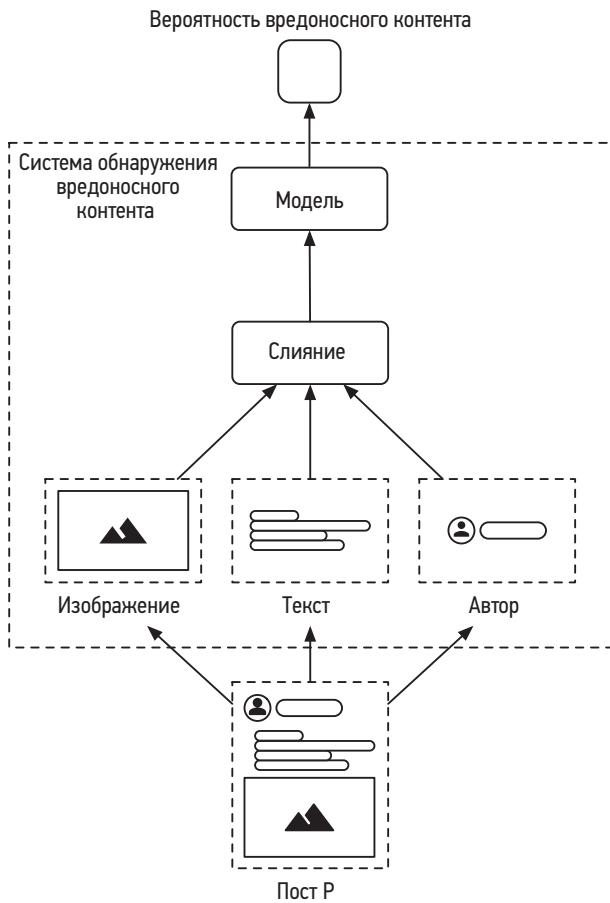


Рис. 5.5. Раннее слияние

У раннего слияния есть два основных преимущества. Во-первых, не требуется собирать обучающие данные отдельно для каждой модальности. Так как обучается всего одна модель, данные нужно собрать только для нее. Во-вторых, модель учитывает все модальности, так что если каждая модальность безопасна, а их комбинация — нет, то модель потенциально способна отразить это в объединенном векторе признака.

Тем не менее такую модель труднее обучить из-за сложных отношений между модальностями. Если обучающих данных недостаточно, модель будет плохо справляться с тем, чтобы изучать сложные отношения и вырабатывать хорошие предсказания.

Какой метод слияния использовать?

Мы выберем метод раннего слияния, потому что он позволяет обнаруживать посты, вредоносные в целом, даже при том что каждая модальность сама по себе безопасна. Кроме того, при 500 миллионах постов в день у модели будет достаточно данных для обучения.

Выбор категории МО

В этом разделе мы рассмотрим следующие возможные категории МО:

- единый бинарный классификатор (single binary classifier);
- отдельный бинарный классификатор для каждого класса вредоносного контента;
- классификатор со множественными метками (multi-label classifier);
- многозадачный классификатор (multi-task classifier).

Единый бинарный классификатор

В этом случае модель получает признаки, прошедшие слияние, и предсказывает, является ли пост вредоносным (рис. 5.6). Поскольку это результат типа «Да/Нет», модель называется бинарным классификатором.

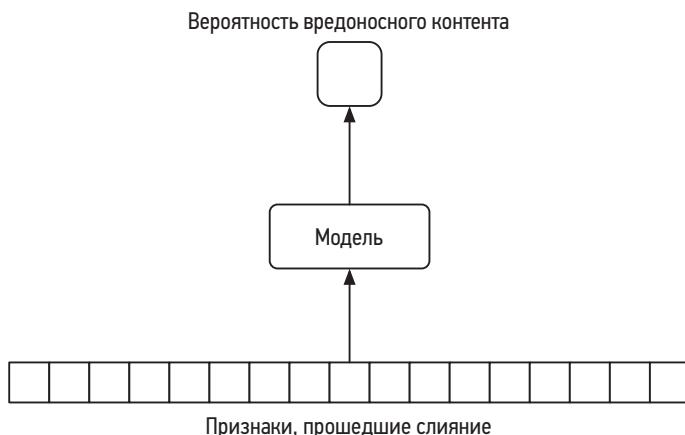


Рис. 5.6. Единый бинарный классификатор

Недостаток этого варианта в том, что трудно определить, к какому классу вредоносного контента относится пост (например, насилие или нецензурная брань). Это создает две проблемы.

- Трудно сообщить пользователям, почему пост был удален: система выдает только бинарное значение, которое указывает, является ли пост в целом вредоносным. Нет информации, почему конкретно модель его забраковала.
- Нелегко узнать, с какими именно классами вредоносного контента система работает недостаточно эффективно, а следовательно, ее труднее будет улучшить.

Поскольку очень важно объяснить пользователям, почему тот или иной пост был удален, для нашего примера единый бинарный классификатор не подойдет.

Отдельный бинарный классификатор для каждого класса вредоносного контента

В этом варианте для каждого класса вредоносного контента определяется отдельный бинарный классификатор. Как показано на рис. 5.7, каждая модель принимает на вход признаки, прошедшие слияние, и предсказывает, относится ли пост к конкретному классу.

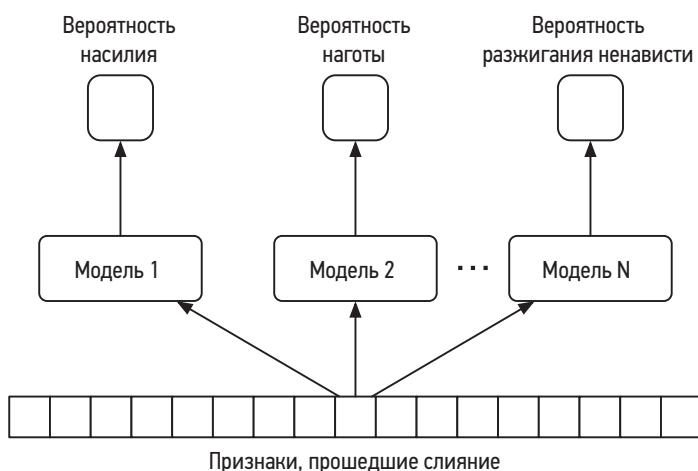


Рис. 5.7. Отдельный бинарный классификатор для каждого класса вредоносного контента

К преимуществам этого решения относится то, что система может объяснить пользователю, почему пост был отфильтрован. Кроме того, модели можно отслеживать и улучшать независимо друг от друга.

Тем не менее у этого варианта есть серьезный недостаток: так как используется несколько моделей, их приходится обучать и сопровождать по отдельности. Раздельное обучение занимает много времени и требует затрат.

Классификатор со множественными метками

В классификации со множественными метками точка данных может принадлежать произвольному количеству классов. В этом случае одна модель используется как классификатор со множественными метками. Как видно из рис. 5.8, на ее вход поступают признаки, прошедшие слияние, а модель предсказывает принадлежность к каждому классу вредоносного контента.

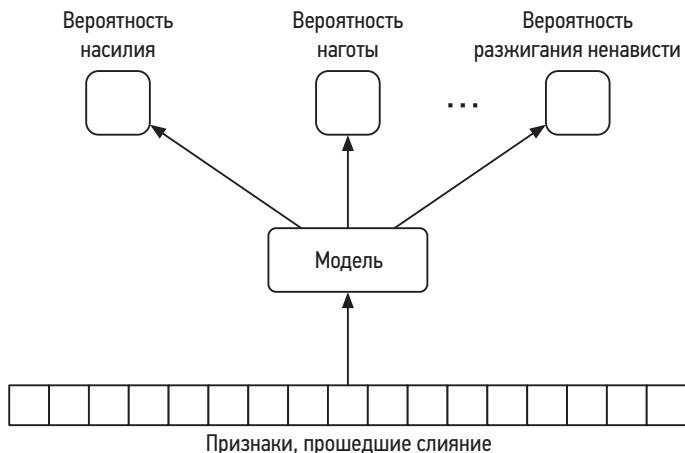


Рис. 5.8. Классификатор со множественными метками

Если использовать одну модель для всех классов вредоносного контента, ее дешевле обучать и сопровождать. Если вам захочется больше узнать об этом варианте, обратитесь к описанию метода WPIE [7].

Тем не менее для предсказания принадлежности к каждому классу вредоносного контента общая модель тоже не идеальна, потому что бывает, что входные признаки требуется преобразовывать по-разному.

Многозадачный классификатор

Многозадачным обучением называется процесс, в котором модель обучается для нескольких задач одновременно, что позволяет ей понять сходство между задачами. При этом также удается обойтись без лишних вычислений, если то или иное преобразование входных данных оказывается подходящим для нескольких задач.

В нашем случае разные классы вредоносного контента (например, насилие и нагота) рассматриваются как разные задачи, и для них используется модель многозадачной классификации. Как показано на рис. 5.9, такая классификация состоит из двух уровней: общие слои и слои отдельных задач.

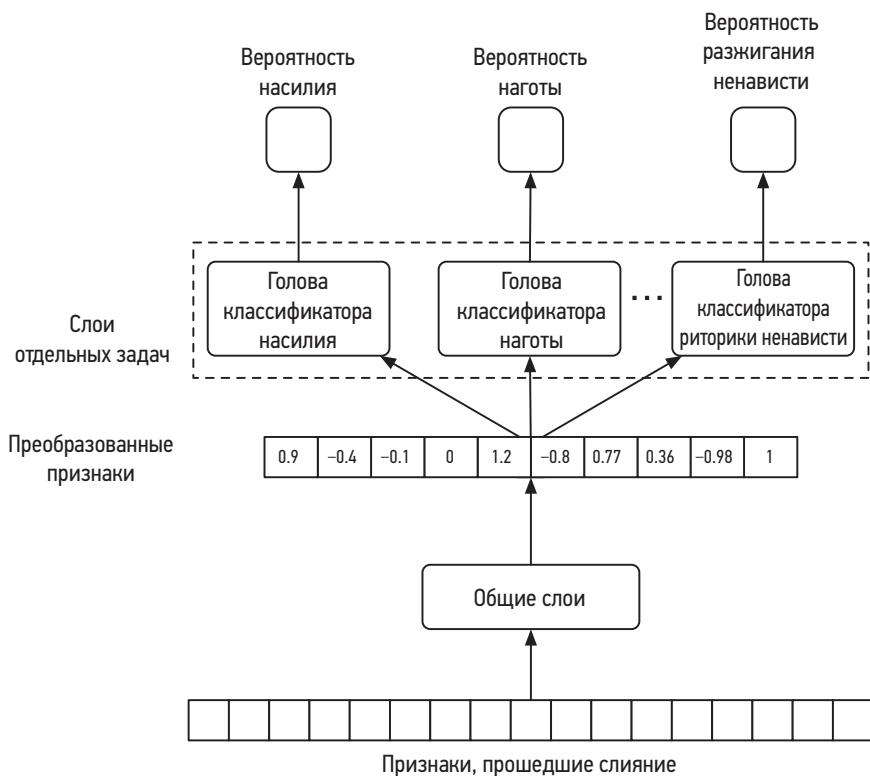
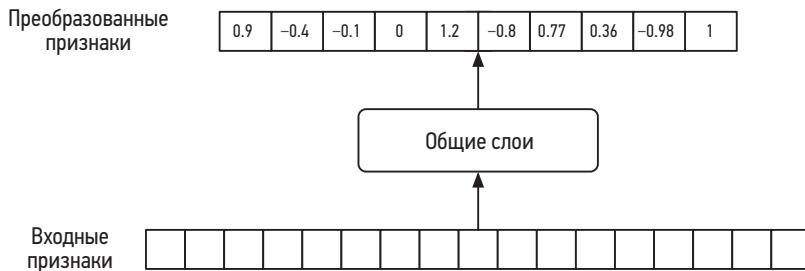


Рис. 5.9. Схема многозадачной классификации

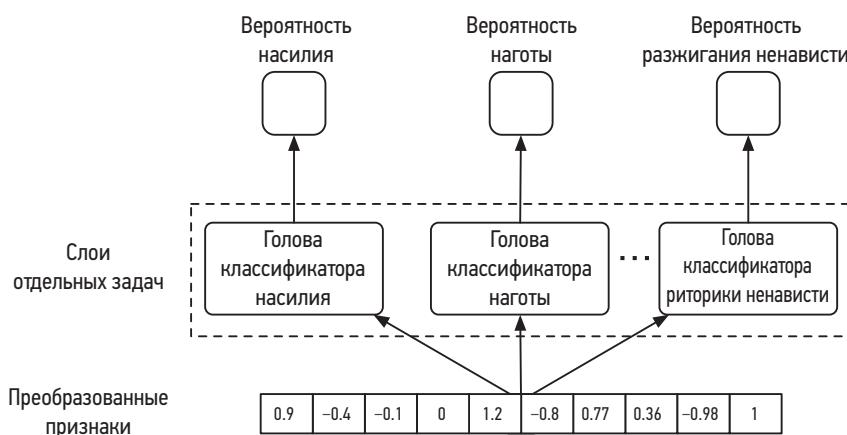
Общие слои

Общие слои, как показано на рис. 5.10, это скрытые слои, которые преобразуют входные признаки в новые, чтобы использовать их для предсказаний по каждому из классов вредоносного контента.

**Рис. 5.10.** Общие слои

Слои отдельных задач

Слои отдельных задач — это независимые слои МО (также называемые головой классификатора). Каждая голова классификатора преобразует признаки так, чтобы они наилучшим образом позволяли предсказывать вероятность конкретного вида вредоносного контента.

**Рис. 5.11.** Слои отдельных задач

У многозадачной классификации есть три основных преимущества. Во-первых, обучать и сопровождать ее менее затратно, потому что используется одна модель. Во-вторых, общие слои преобразуют признаки так, как это лучше подходит для каждой задачи. Тем самым предотвращаются избыточные вычисления, и классификация работает эффективнее. Наконец, обучающие данные каждой задачи учитываются в обучении для других задач. Это особенно полезно, если для той или иной задачи мало доступных данных.

Из-за этих преимуществ мы воспользуемся методом многозадачной классификации. На рис. 5.12 показано, как будет формулироваться задача.

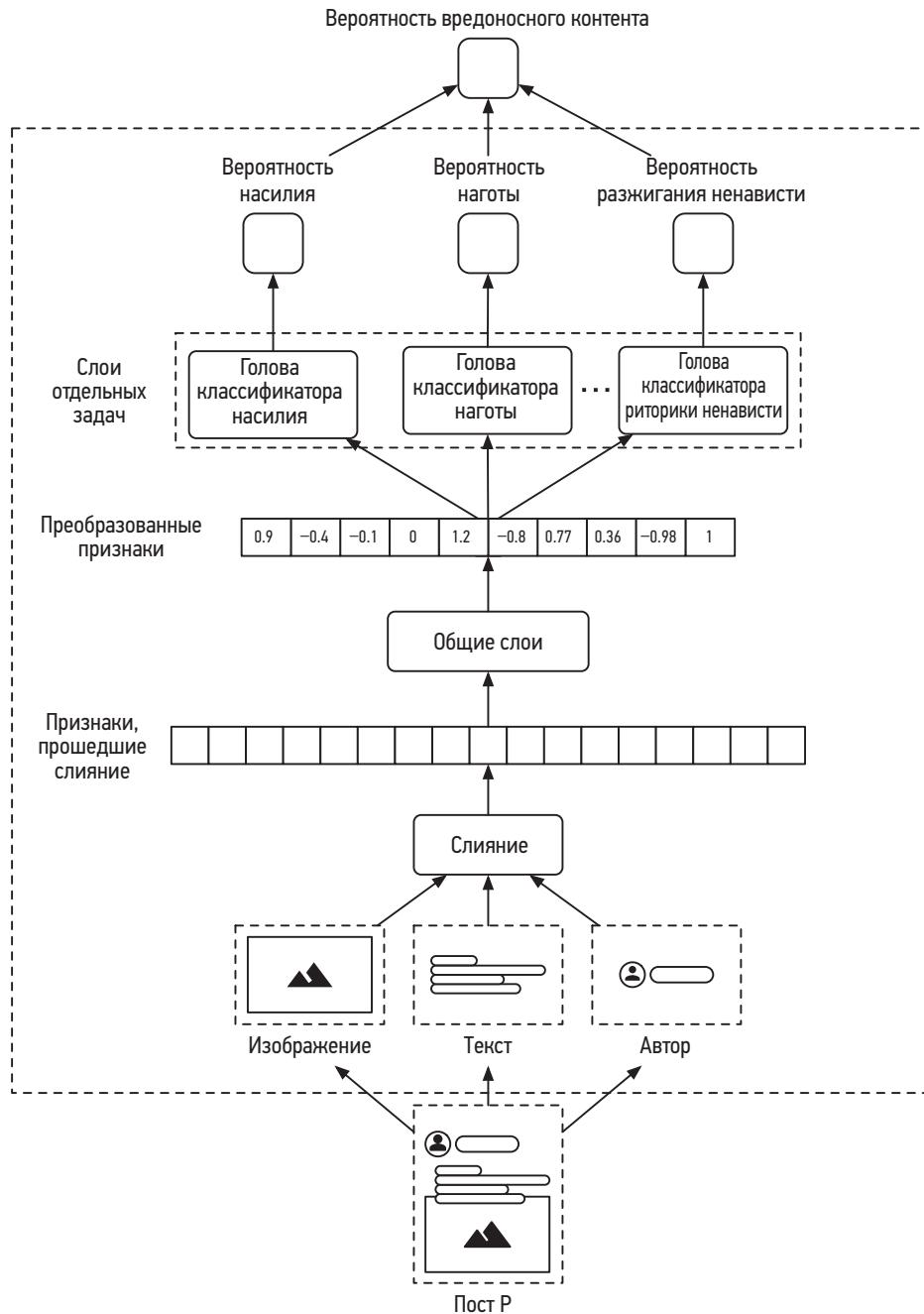


Рис. 5.12. Формулировка проблемы в виде задачи М0

Подготовка данных

Инженерия данных

Нам доступны следующие данные:

- пользователи;
- посты;
- взаимодействия пользователей с постами.

Пользователи

Схема данных о пользователях представлена в табл. 5.1.

Таблица 5.1. Схема данных о пользователях

ID пользователя	Имя пользователя	Возраст	Пол	Город	Страна	Электронная почта
-----------------	------------------	---------	-----	-------	--------	-------------------

Посты

В данных поста хранятся такие поля, как автор, время отправки и т. д. В табл. 5.2 представлены некоторые важные атрибуты. На самом деле с каждым постом обычно связываются сотни разных атрибутов.

Таблица 5.2. Данные поста

ID поста	ID автора	IP-адрес автора	Временная метка	Текстовое содержимое	Изображения или видео	Ссылки
1	1	73.93.220.240	1658469431	Сегодня сажусь на диету	http://cdn.mysite.com/u1.jpg	
2	11	89.42.110.250	1658471428	Видео просто потрясающее! Принимаются донаты	http://cdn.mysite.com/t3.mp4	gofundme.com/f3u1njd32
3	4	39.55.180.020	1658489233	Посоветуйте хороший ресторан на Петроградской стороне	http://cdn.mysite.com/t5.jpg	

Взаимодействия пользователей с постами

Данные о взаимодействиях пользователей с постами содержат реакции пользователей на посты: лайки, комментарии, пересылки и т. д. Пользователи также могут сообщать о вредоносных постах и отправлять жалобы. В табл. 5.3 показано, как могут выглядеть данные.

Таблица 5.3. Данные взаимодействия пользователей с постами

ID пользова- теля	ID поста	Тип взаимодействия	Значение взаимо- действия	Временна́я метка
11	6	Просмотр	—	1658450539
4	20	Лайк	—	1658451341
11	7	Комментарий	Отстой	1658451365
4	20	Пересылка	—	1658435948
11	7	Жалоба	насилие	1658451849

Конструирование признаков

В разделе «Формулировка проблемы как задачи МО» мы сформулировали проблему как задачу многозадачной классификации, в которой входными данными является пост. В этом разделе мы рассмотрим предсказательные признаки, которые можно сконструировать на основании поста.

Пост может содержать следующие элементы:

- текстовое содержание;
- изображение или видео;
- реакции пользователей;
- автор;
- контекстная информация.

Рассмотрим все эти элементы поочередно.

Текстовое содержание

По текстовому содержанию поста часто можно определить, является ли он вредоносным. Как рассказывалось в главе 4, подготовка текстовых данных обычно делится на две стадии:

- предварительная обработка текста (нормализация, лексический анализ и т. д.);
- векторизация: преобразование обработанного текста в содержательный вектор признаков.

Мы сосредоточимся на векторизации, потому что тема уникальна для этой главы. Чтобы извлечь вектор признаков, можно воспользоваться статистическими методами или методами на базе МО. Статистические методы (такие, как BoW или TF-IDF) легко реализуются и быстро вычисляются, однако они не способны кодировать семантику текста. Для нашей системы важно понимать семантику текстового содержания, чтобы проверять пост на вредоносность, поэтому мы выберем метод на базе МО.

Чтобы преобразовывать текст в вектор признаков, мы будем использовать предварительно обученную языковую модель на базе Transformer — такую, как BERT [8]. Однако у исходной версии BERT есть два недостатка.

- Построение эмбеддинга для текста занимает много времени из-за большого размера модели. Так как процесс медленный, его неудобно использовать для оперативных предсказаний.
- Модель BERT была обучена только на данных на английском языке. Как следствие, она не создает осмысленные эмбеддинги для текста на других языках.

DistilBERT [9], более эффективная разновидность BERT, решает обе проблемы. Если у двух предложений на разных языках одинаковый смысл, их эмбеддинги будут очень похожими. Если вам захочется больше узнать о многоязыковых моделях, обращайтесь к [10].

Изображение или видео

Обычно суть поста можно определить по включенному в него изображению или видео. Как правило, при подготовке подобных неструктурированных данных используются следующие два шага.

- **Предварительная обработка:** декодирование, изменение размеров и нормализация данных.
- **Выделение признаков:** предварительно обученная модель принимает обработанные неструктурированные данные и преобразовывает их в вектор признаков, который представляет изображение или видео. Для изображений можно воспользоваться такой предварительно обученной моделью, как визуальный кодировщик CLIP [11] или SimCLR [12]. Для видео может хорошо подойти VideoMoCo [13].

Реакции пользователей на пост

Является ли пост вредоносным, можно также определить на основании реакций пользователей, особенно если его содержимое неоднозначно. Как видно из рис. 5.13, чем больше комментариев, тем очевиднее становится, что пост относится к возможному самоубийству.



Рис. 5.13. Пост с информацией о возможном самоубийстве

Поскольку реакции пользователей эффективно помогают обнаруживать вредоносный контент, рассмотрим некоторые признаки, которые можно сконструировать на их основе.

Количество лайков, репостов, комментариев и жалоб. Обычно эти числовые значения масштабируются, чтобы модель быстрее сходилась в процессе обучения.

Комментарии. Как показано на рис. 5.13, комментарии могут помочь выявить вредоносный контент. Чтобы подготовить признаки, комментарии нужно преобразовать в числовые представления:

- с помощью предварительно обученной модели, которая уже применялась ранее, получить эмбеддинг каждого комментария;
- агрегировать (например, усреднить) эмбеддинги, чтобы получить итоговый эмбеддинг.

Сводка перечисленных признаков представлена на рис. 5.14.

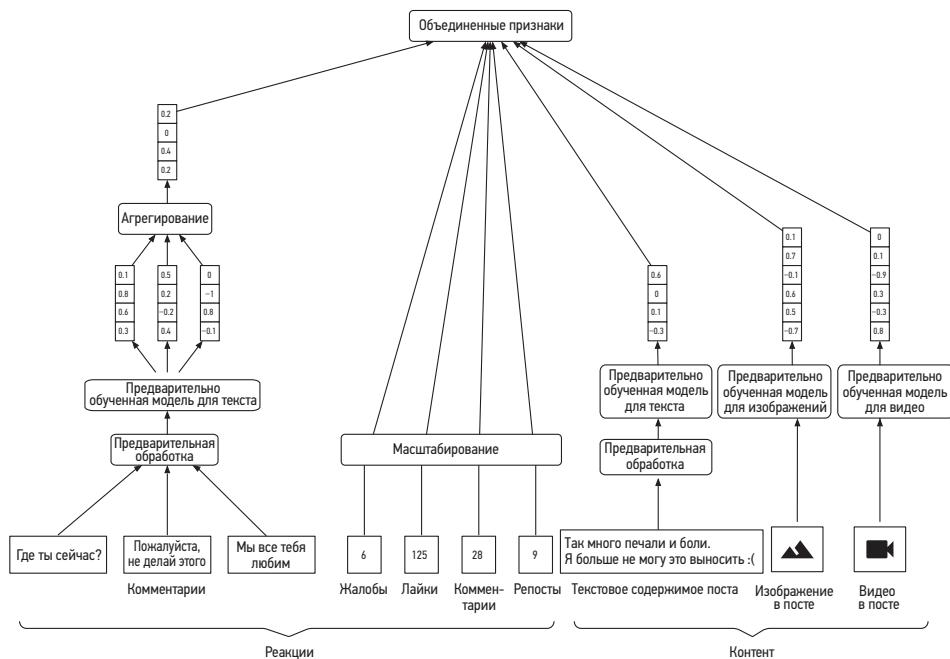


Рис. 5.14. Конструирование признаков для реакций и содержимого

Признаки автора

Прошлая активность автора тоже помогает определить, является ли пост вредоносным. Рассмотрим некоторые признаки, относящиеся к автору поста.

История нарушений автора

- **Количество нарушений.** Числовое значение: сколько раз автор раньше нарушал правила площадки.
- **Общее количество жалоб от пользователей.** Числовое значение: количество жалоб на посты автора.
- **Доля ненормативной лексики.** Числовое значение: доля нецензурных слов в предыдущих постах и комментариях автора. Чтобы выявлять такие слова, можно воспользоваться готовым списком нецензурных слов.

Демографические данные автора

- **Возраст.** Возраст пользователя — один из самых важных предсказательных признаков.
- **Пол.** Категориальный признак, представляющий пол пользователя. Чтобы представить пол, мы будем использовать унитарное кодирование.
- **Город и страна.** Эти признаки могут принимать множество разных значений. Чтобы их представить, мы воспользуемся слоем эмбеддингов, который преобразовывает город и страну в векторы признаков. Обратите внимание, что унитарное кодирование — не самый эффективный способ представить страну и город, потому что представления получаются слишком длинными и разреженными.

Информация об учетной записи

- **Количество подписчиков и подписок.**
- **Возраст учетной записи.** Числовое значение: срок существования учетной записи автора. Это предсказательный признак, потому что недавно созданные учетные записи с большей вероятностью рассылают спам или нарушают правила.

Контекстная информация

- **Время суток.** Время, в которое автор создал пост. В нашем примере время будет разбиваться на категории (утро, день, вечер или ночь), и этот признак будет представляться с помощью унитарного кодирования.
- **Устройство.** Устройство, которое использует автор, — например, смартфон или ПК. Для представления этого признака тоже будет использоваться унитарное кодирование.

На рис. 5.15 приведена сводка важнейших признаков в системе обнаружения вредоносного контента.

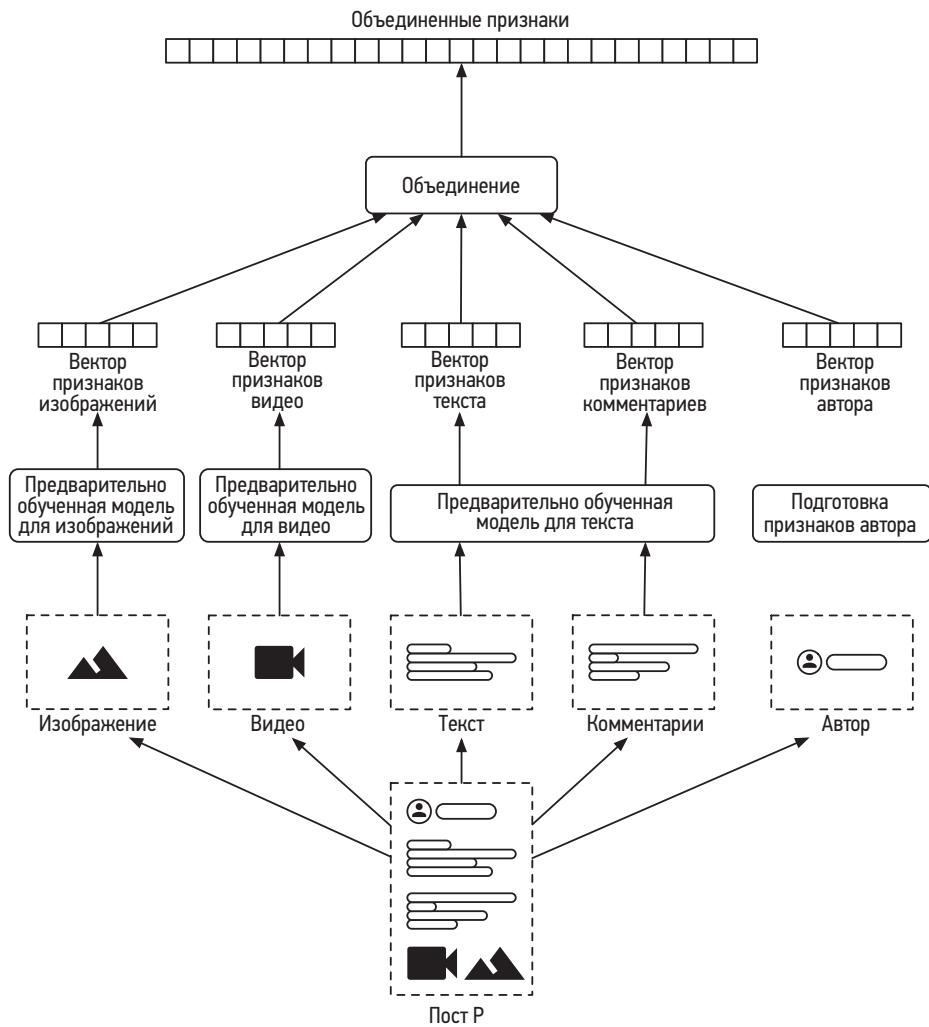


Рис. 5.15. Сводка конструирования признаков

Разработка модели

Выбор модели

Нейронная сеть — самая популярная модель для многозадачного обучения, так что в нашем примере мы тоже используем нейронные сети.

Какие факторы следует учитывать при выборе нейронной сети? Необходимо определить архитектуру нейронной сети и подобрать оптимальные гиперпараметры.

метры (скрытые уровни, функцию активации, скорость обучения и т. д.) — для этого служит регулировка гиперпараметров.

Регулировка (tuning) гиперпараметров заключается в том, что мы подбираем такие их значения, которые обеспечивают наилучшее качество модели. Для регулировки гиперпараметров обычно используется поиск по решетке (grid search). При этом новая модель обучается для каждой комбинации значений гиперпараметров, обученные модели оцениваются, и в итоге выбираются гиперпараметры, которые обеспечивают наилучшую модель. Если вам захочется больше узнать о настройке гиперпараметров, обращайтесь к [14].

Обучение модели

Построение датасета

Чтобы обучить модель многозадачной классификации, сначала нужно сконструировать датасет. В нем содержатся входные данные модели (признаки) и выходные данные (метки), которые модель должна предсказывать. Чтобы построить входные данные, мы обрабатываем посты автономно в пакетном режиме и вычисляем объединенные признаки, как описано выше. Эти признаки можно сохранять в хранилище признаков для будущего обучения. Чтобы создать метки для каждого набора входных данных, есть два способа:

- ручная разметка;
- автоматическая разметка.

В первом случае специалисты расставляют метки вручную. Так получаются наиболее точные метки, но это затратный и долгий процесс. При органической разметке метки расставляются автоматически на основании жалоб от пользователей. Хотя при этом метки сильнее зашумлены, они создаются быстрее. Для оценочного датасета мы будем использовать ручную разметку, которая приоритизирует точность меток, а для обучающего — автоматическую разметку, которая обеспечивает максимальную скорость. Точка данных из построенного набора показана на рис. 5.16.

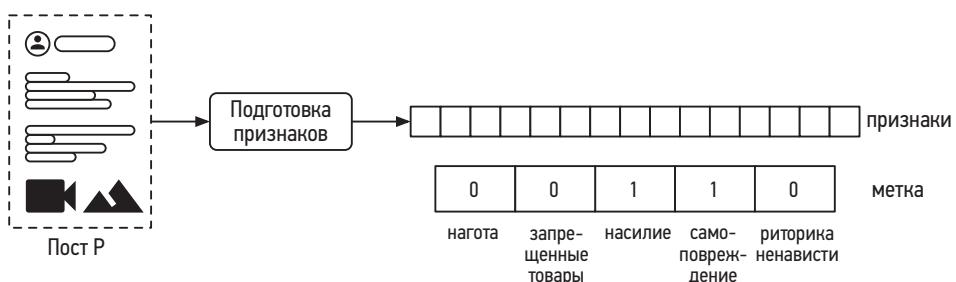


Рис. 5.16. Построенная точка данных

Выбор функции потерь

Многозадачная нейронная сеть обучается примерно так же, как обычные модели нейронных сетей. Прямое распространение выполняет расчеты, результатом которых становится предсказание, функция потерь измеряет его правильность, а обратное распространение оптимизирует параметры модели, чтобы сократить потери на следующей итерации.

Рассмотрим функцию потерь. При многозадачном обучении каждой задаче назначается функция потерь в зависимости от того, к какой категории МО относится задача. В нашем случае каждая задача формулируется как бинарная классификация, поэтому мы будем применять стандартную функцию потерь для бинарной классификации — такую, как перекрестная энтропия. Общие потери вычисляются как сумма потерь отдельных задач, как показано на рис. 5.17.

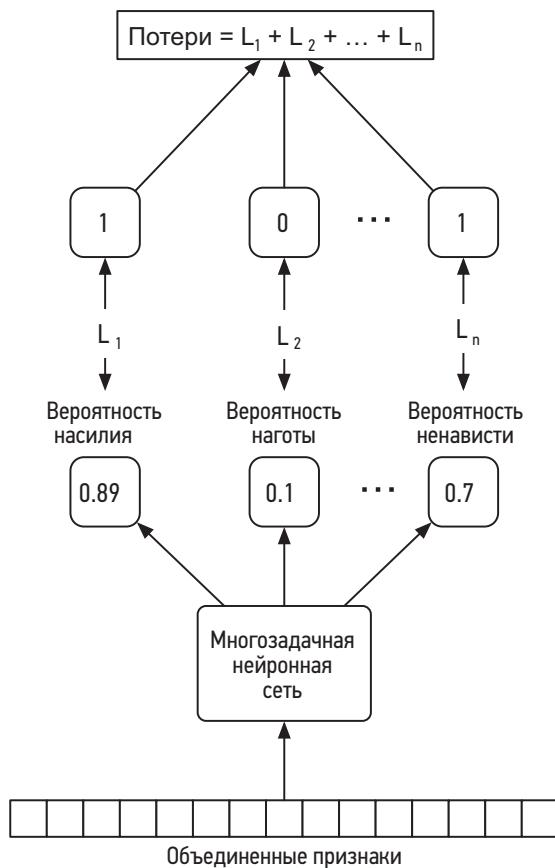


Рис. 5.17. Обучение модели

Типичная проблема при обучении мультимодальных систем — переобучение [15]. Например, если у разных модальностей разная скорость обучения, то одна модальность (например, изображение) может доминировать в процессе обучения. Для решения этой проблемы применяются два метода: градиентный блендинг и фокальная функция потерь. Чтобы больше узнать о них, обращайтесь к [16], [17].

Оценка

Автономные метрики

Качество модели бинарной классификации обычно оценивается такими автономными метриками, как точность, полнота и F_1 -мера. Тем не менее одна точность или одна полнота не дает общего представления о качестве. Например, у модели с высокой точностью может быть очень низкий показатель полноты. Чтобы преодолеть этот недостаток, применяются кривая точности — полноты (PR) и кривая рабочей характеристики приемника (ROC). Рассмотрим эти методы подробнее.

PR-кривая показывает баланс между точностью и полнотой модели. Как демонстрируется на рис. 5.18, PR-кривая строится как график точности модели при разных вероятностных порогах от 0 до 1. Площадь под PR-кривой, которая обозначается PR-AUC (AUC — Area Under the Curve, «площадь под кривой»), характеризует соотношение между точностью и полнотой. В общем случае чем выше PR-AUC, тем точнее модель.

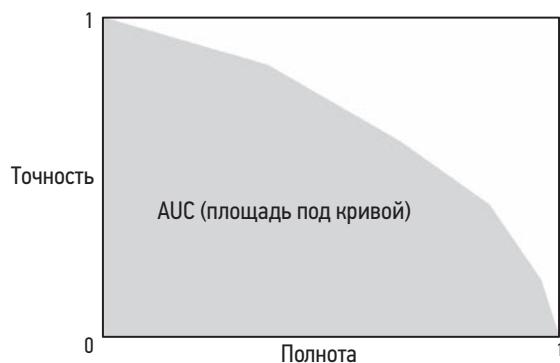


Рис. 5.18. PR-кривая

ROC-кривая показывает соотношение частот истинно положительных результатов (полнота) и ложноположительных. По аналогии с PR-кривой, ROC-AUC характеризует качество модели на основании площади под ROC-кривой.

PR- и ROC-кривые — это два разных способа описать качество модели классификации. Чтобы узнать о различиях между ними, обращайтесь к [18].

В нашем случае в качестве автономных метрик будут использоваться как ROC-AUC, так и PR-AUC.

Оперативные метрики

Вот некоторые важные метрики, отражающие безопасность платформы.

Распространенность. Отношение количества вредоносных постов, которые система не обнаружила, к общему количеству постов на платформе.

$$\text{Распространенность} = \frac{\text{количество вредоносных постов, не обнаруженных системой}}{\text{общее количество постов на платформе}}.$$

Недостаток этой метрики в том, что она учитывает все вредоносные посты одинаково, хотя один вредоносный пост со 100 тыс. просмотров причиняет больше вреда, чем два поста с 10 просмотрами каждый.

Показы вредоносного контента. Мы отдаём предпочтение этой метрике перед распространённостью. Дело в том, что количество вредоносных постов на платформе не показывает, скольких людей затронули эти посты, тогда как количество показов вредоносного контента отражает эту информацию.

Доля успешных обжалований. Доля постов, которые были сочтены вредоносными, но разблокированы в результате обжалования.

$$\text{Доля успешных обжалований} = \frac{\text{количество успешно обжалованных постов, сочтенных вредоносными}}{\text{количество вредоносных постов, обнаруженных системой}}.$$

Доля профилактических вмешательств. Доля вредоносных постов, которые система обнаружила и удалила до того, как пользователи пожаловались на них.

$$\text{Доля профилактических вмешательств} = \frac{\text{количество вредоносных постов, обнаруженных системой}}{\text{количество вредоносных постов, обнаруженных системой или пользователями}}.$$

Количество жалоб пользователей на вредоносные сообщения отдельных классов. Эта метрика оценивает качество системы по количеству жалоб пользователей на сообщения, относящиеся к каждому отдельному классу вредоносного контента.

Эксплуатация

На рис. 5.19 изображен высокоуровневый дизайн системы МО. Далее рассмотрим каждый из его компонентов подробнее.

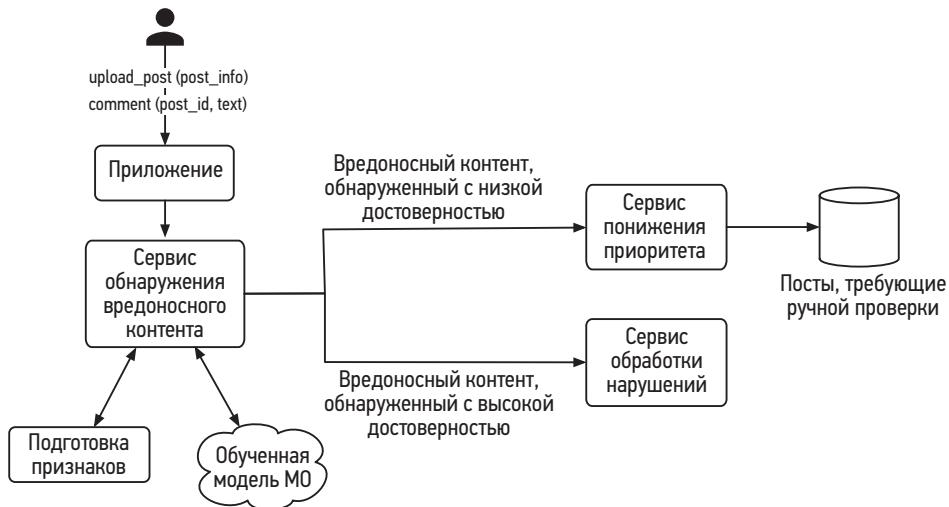


Рис. 5.19. Дизайн системы МО

Сервис обнаружения вредоносного контента

Для нового поста этот сервис предсказывает вероятность того, что пост является вредоносным. Согласно требованиям, некоторые особо опасные типы вредоносного контента нужно обрабатывать немедленно. В таких случаях сервис обработки нарушений безотлагательно удаляет пост.

Сервис обработки нарушений

Сервис обработки нарушений немедленно удаляет пост, если сервис обнаружения вредоносного контента предсказывает вредоносность с высокой достоверностью. Он также сообщает пользователю, почему пост был удален.

Сервис понижения приоритета

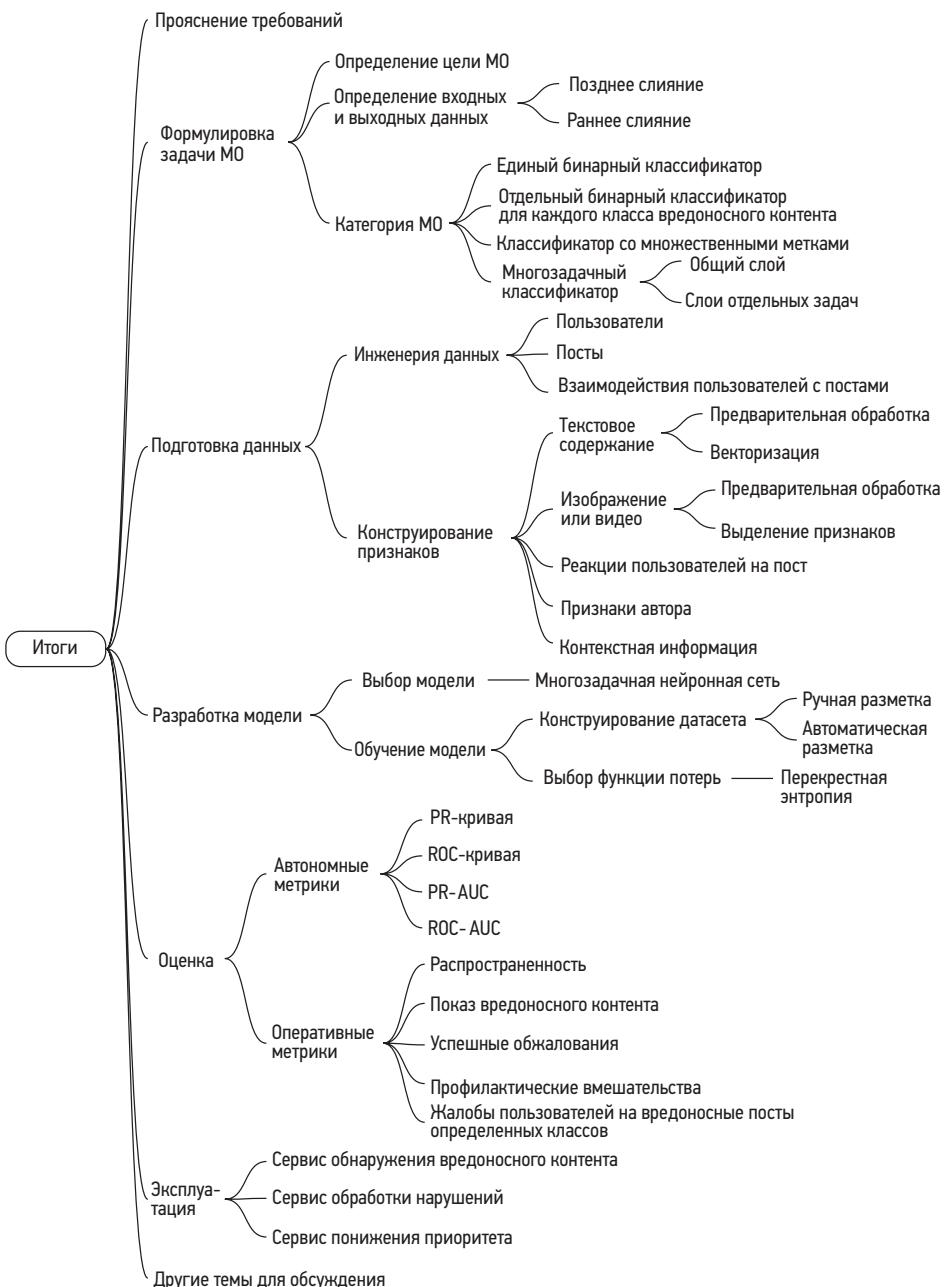
Когда сервис обнаружения вредоносного контента предсказывает вредоносность с низкой достоверностью, сервис понижения приоритета временно деприоритизирует пост, чтобы он менее активно распространялся среди пользователей.

Затем пост помещается в хранилище для ручной обработки. Модераторы вручную проверяют пост и помечают его одним из предварительно определенных классов вредоносного контента. Эти посты с метками будут использоваться на последующих итерациях обучения, чтобы улучшить модель.

Другие темы для обсуждения

- Как обрабатывать смещения, возникающие из-за ручной разметки [19].
- Как адаптировать систему для обнаружения злободневных классов вредоносного контента (ковид, выборы и т. д.) [20].
- Как построить систему обнаружения вредоносного контента, которая использует временную информацию — например, последовательность действий пользователей [21], [22].
- Как эффективно выбрать примеры постов для ручного анализа [23].
- Как обнаруживать подлинные и фиктивные учетные записи [24].
- Как поступать с «пограничным» контентом [25], то есть таким, который не запрещен правилами, но очень близок к их нарушению.
- Как сделать систему обнаружения вредоносного контента, которую можно развернуть на пользовательском устройстве [26].
- Как заменить архитектуры на базе Transformer линейными компонентами Transformer, чтобы создать более эффективную систему [27], [28].

Итоги



Ссылки

- [1] Неаутентичное поведение на Facebook. <https://transparency.fb.com/policies/community-standards/inauthentic-behavior/>
- [2] Политики профессионального сообщества LinkedIn. <https://www.linkedin.com/legal/professional-community-policies>
- [3] Политики добросовестного поведения в Twitter. <https://help.twitter.com/en/rules-and-policies/election-integrity-policy>
- [4] Исследование по этике на Facebook. <https://arxiv.org/pdf/2009.10311.pdf>
- [5] Система обнаружения нарушений правил Pinterest. <https://medium.com/pinterest-engineering/how-pinterest-fights-misinformation-hate-speech-and-self-harm-content-with-machine-learning-1806b73b40ef>
- [6] Обнаружение оскорбительного поведения в LinkedIn. <https://engineering.linkedin.com/blog/2019/isolation-forest>
- [7] Метод WPIE. <https://ai.facebook.com/blog/community-standards-report/>
- [8] BERT. <https://arxiv.org/pdf/1810.04805.pdf>
- [9] Многоязыковая модель DistilBERT. <https://huggingface.co/distilbert-base-multilingual-cased>
- [10] Многоязыковые модели. <https://arxiv.org/pdf/2107.00676.pdf>
- [11] Модель CLIP. <https://openai.com/blog/clip/>
- [12] SimCLR. <https://arxiv.org/pdf/2002.05709.pdf>
- [13] VideoMoCo. <https://arxiv.org/pdf/2103.05905.pdf>
- [14] Регулировка гиперпараметров. <https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview>
- [15] Переобучение. <https://en.wikipedia.org/wiki/Overfitting>
- [16] Фокальная функция потерь. <https://amaarora.github.io/2020/06/29/FocalLoss.html>
- [17] Градиентный блендинг в мультимодальных системах. <https://arxiv.org/pdf/1905.12681.pdf>
- [18] ROC-кривая и кривая точности — полноты. <https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/>
- [19] Смещение из-за ручной разметки. <https://labelyourdata.com/articles/bias-in-machine-learning>
- [20] Подход Facebook к быстрой обработке злободневного вредоносного контента. <https://ai.facebook.com/blog/harmful-content-can-evolve-quickly-our-new-ai-system-adapts-to-tackle-it/>.
- [21] Подход Facebook к методу TIES. <https://arxiv.org/pdf/2002.07917.pdf>
- [22] Эмбеддинги временных взаимодействий. <https://www.facebook.com/atscaleevents/videos/730968530723238/>
- [23] Построение и масштабирование систем ручного рецензирования. <https://www.facebook.com/atscaleevents/videos/1201751883328695/>
- [24] Фреймворк для обнаружения учетных записей, нарушающих правила. <https://www.youtube.com/watch?v=YeX4MdU0JNk>

- [25] Пограничный контент. <https://transparency.fb.com/features/approach-to-ranking/content-distribution-guidelines/content-borderline-to-the-community-standards/>
- [26] Эффективное обнаружение вредоносного контента. <https://about.fb.com/news/2021/12/metas-new-ai-system-tackles-harmful-content/>
- [27] Линейная реализация Transformer. <https://arxiv.org/pdf/2006.04768.pdf>
- [28] Эффективные модели ИИ для обнаружения риторики ненависти. <https://ai.facebook.com/blog/how-facebook-uses-super-efficient-ai-models-to-detect-hate-speech/>

6

СИСТЕМА РЕКОМЕНДАЦИИ ВИДЕО

Рекомендательные системы играют ключевую роль в службах потокового видео и музыки. Например, YouTube рекомендует видеоролики, которые могут понравиться пользователям, Netflix рекомендует фильмы, а Spotify — музыку.

В этой главе мы спроектируем рекомендательную систему для видеоплатформы вроде той, которая используется на YouTube [1]. Система рекомендует видеоролики на главной странице, персонализируя подборку для каждого пользователя на основании его профиля, предыдущих взаимодействий и т. д.

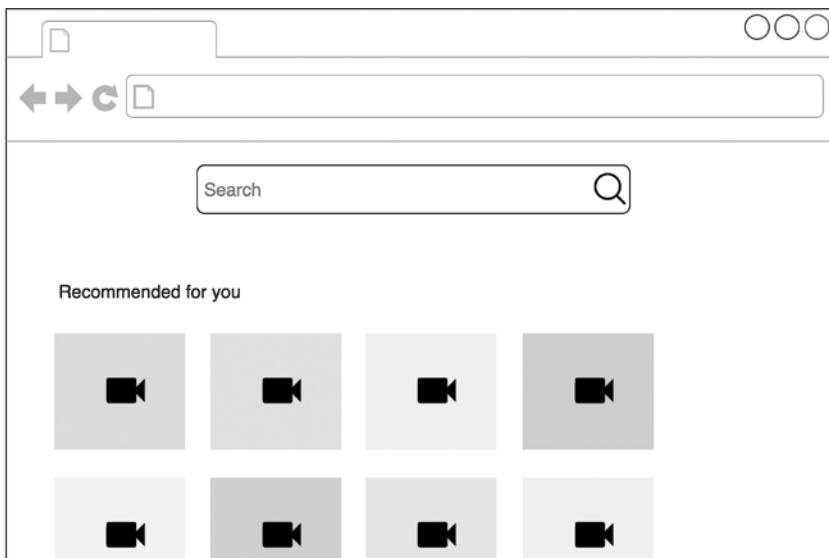


Рис. 6.1. Рекомендации видеороликов на главной странице

Рекомендательные системы часто устроены очень сложно, и разработчики прикладывают значительные усилия, чтобы создать эффективную и масштабируемую систему. Впрочем, не беспокойтесь: никто не ожидает, что вы спроектируете идеальную систему в течение 45-минутного собеседования. Экспертов прежде всего интересует ваш ход мыслей, коммуникационные навыки, а также умение

проектировать системы МО и рассматривать сильные и слабые стороны разных решений.

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: Можно ли считать, что бизнес-цель системы рекомендации видео — повысить степень вовлеченности пользователей?

Эксперт: Верно.

Соискатель: Будет ли система рекомендовать видеоролики, похожие на тот, который пользователь смотрит в настоящий момент? Или она должна выводить персонализированную подборку видео на главной странице?

Эксперт: Эта система предлагает персонализированную подборку видео для пользователей, которые заходят на главную страницу.

Соискатель: Поскольку YouTube — глобальный сервис, можно ли считать, что пользователи живут в разных странах, а видеоролики бывают на разных языках?

Эксперт: Это разумное предположение.

Соискатель: Можно ли рассчитывать на то, что получится построить датасет на основании взаимодействий пользователей с видеоконтентом?

Эксперт: Да, и это тоже разумно.

Соискатель: Могут ли пользователи группировать видео по плейлистам? Плейлисты могут содержать полезную информацию для модели МО на этапе обучения.

Эксперт: Ради простоты будем считать, что плейлистов нет.

Соискатель: Сколько видеороликов доступно на платформе?

Эксперт: Около 10 миллиардов.

Соискатель: Насколько быстро система должна рекомендовать видеоролики пользователям? Можно ли считать, что рекомендация должна занимать не более 200 миллисекунд?

Эксперт: Пожалуй, так.

Резюмируем описание проблемы. Требуется спроектировать систему, которая рекомендует видеоролики для каждого пользователя на главной странице. Бизнес-цель состоит в том, чтобы увеличить вовлеченность пользователей. Каждый раз, когда пользователь заходит на главную страницу, система реко-

мендует ему самые «вовлекающие» видеоролики. Пользователи живут в разных странах, и видеоролики могут быть на разных языках. На платформе доступно около 10 миллиардов роликов, и рекомендации должны предоставляться достаточно быстро.

Формулировка проблемы в виде задачи МО

Определение цели МО

Бизнес-цель системы — повысить степень вовлеченности пользователей. Преобразовать бизнес-цели в четко определенные цели МО можно несколькими способами. Мы изучим некоторые из них и обсудим их достоинства и недостатки.

Максимизировать количество кликов. Систему рекомендации видео можно спроектировать так, чтобы достичь максимального количества кликов. Однако у такого подхода есть серьезный недостаток: система может рекомендовать видеоролики из категории так называемого «кликбейта»: название и значок выглядят заманчиво, но содержимое видео оказывается скучным, нерелевантным и даже мошенническим. Кликбейтные видеоролики портят впечатление пользователей и со временем снижают их вовлеченность.

Максимизировать количество роликов, просмотренных до конца. Система также может рекомендовать видеоролики, которые пользователь с большой вероятностью досмотрит до конца. Главный недостаток этого решения — модель может рекомендовать более короткие видео, просмотр которых занимает меньше времени.

Максимизировать общее время просмотра. В этом случае рекомендуются видеоролики, за просмотром которых пользователи в сумме проведут больше времени.

Максимизировать количество релевантных видеороликов. В этом случае рекомендуются видеоролики, релевантные для пользователя. Разработчики или менеджеры продуктов могут определять релевантность по некоторым правилам, которые опираются на явные и неявные реакции пользователей. Например, можно считать видео релевантным, если пользователь нажал кнопку «лайк» или просмотрел не менее половины. Определив релевантность, можно построить набор данных и обучить модель предсказывать показатель релевантности для заданного пользователя и видео.

В нашей системе в качестве цели МО будет выбран последний критерий, потому что он позволяет лучше контролировать, какие сигналы использовать. Кроме того, этой цели не присущи недостатки других вариантов, о которых говорилось выше.

Определение входных и выходных данных системы

Как показано на рис. 6.2, система рекомендации видео принимает на вход пользователя и возвращает ранжированный список видеороликов, отсортированных по релевантности.

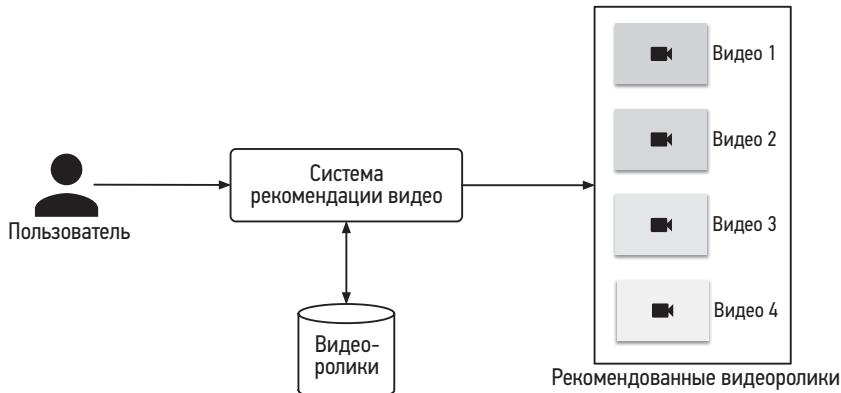


Рис. 6.2. Входные и выходные данные системы рекомендации видео

Выбор категории МО

В этом разделе мы рассмотрим три распространенные разновидности персонализированных рекомендательных систем:

- фильтрация на основе содержимого;
- коллаборативная фильтрация;
- гибридная фильтрация.

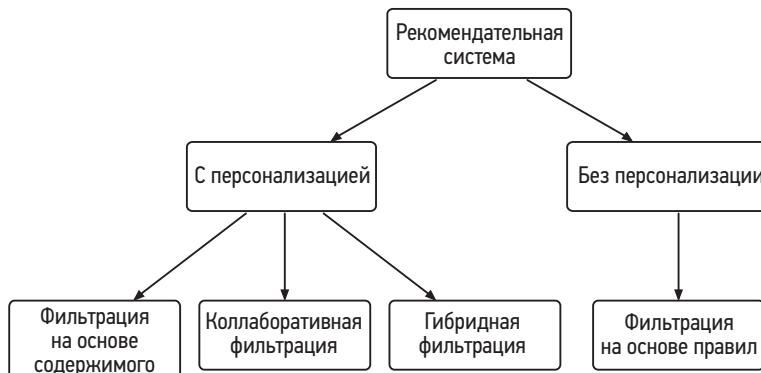


Рис. 6.3. Распространенные виды рекомендательных систем

Рассмотрим каждую разновидность подробнее.

Фильтрация на основе содержимого

Этот метод использует различные признаки видео, чтобы рекомендовать новые видеоролики, похожие на те, которые пользователь ранее посчитал релевантными. Например, если пользователь часто смотрел видео о лыжном спорте, этот метод может предсказать новые ролики на эту же тему. Пример показан на рис. 6.4.

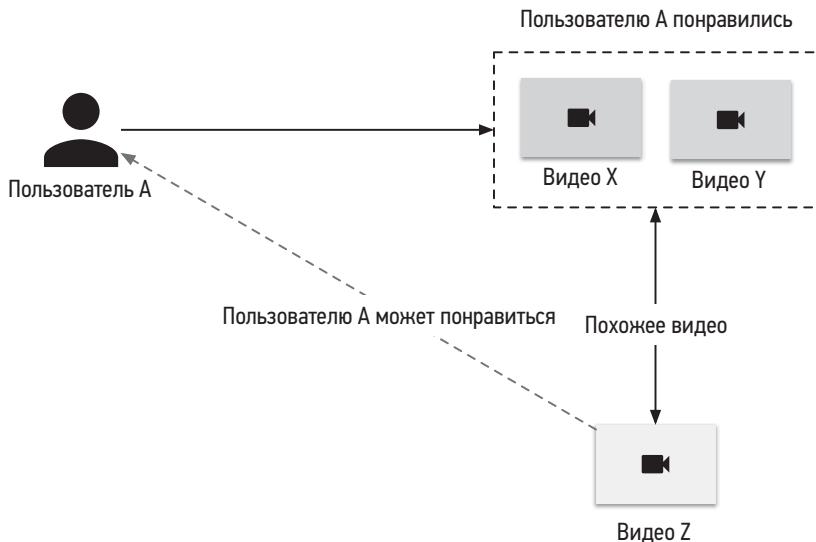


Рис. 6.4. Фильтрация на основе содержимого

Вот что происходит на диаграмме:

1. Пользователю А в прошлом понравились видео X и Y.
2. Видео Z похоже на видео X и Y.
3. Система рекомендует пользователю А видео Z.

У фильтрации на основе содержимого есть и достоинства, и недостатки.

Достоинства

- **Можно рекомендовать новые видео.** С этим методом не нужно ждать данных о пользовательских взаимодействиях, чтобы формировать профили для новых видеороликов. Профиль видео зависит исключительно от признаков самого видео.

- **Учитываются уникальные интересы пользователей.** Это связано с тем, что видеоролики рекомендуются на основании прошлых взаимодействий пользователей.

Недостатки

- Трудно выявлять новые интересы пользователя.
- Метод требует знания предметной области. Признаки видео часто приходится конструировать вручную.

Коллаборативная фильтрация (*Collaborative Filtering, CF*)

Чтобы рекомендовать новые видео, этот метод использует сходство между пользователями (CF на основе пользователей) или сходство между видеороликами (CF на основе единиц контента). Коллаборативная фильтрация опирается на интуитивно понятную идею о том, что похожих пользователей интересуют похожие видеоролики. Пример CF на основе пользователей показан на рис. 6.5.

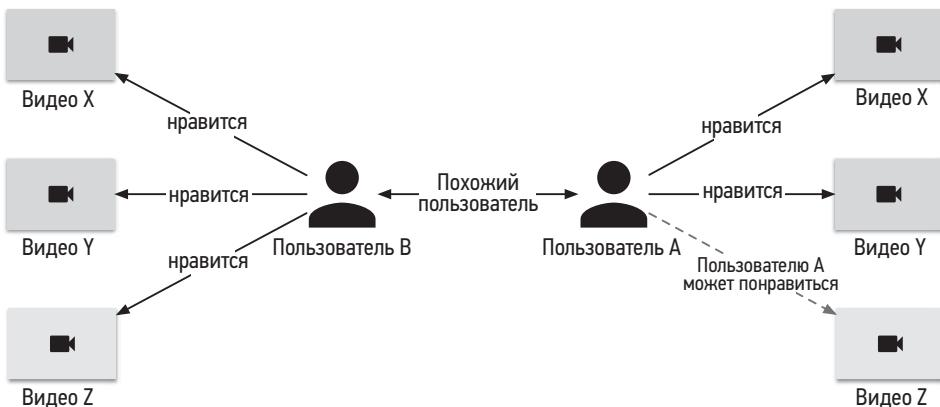


Рис. 6.5. Коллаборативная фильтрация на основе пользователей

Поясним смысл диаграммы. Задача системы — порекомендовать новое видео пользователю А.

1. Найти пользователя, похожего на А по предыдущим взаимодействиям; допустим, это пользователь В.

2. Найти видео, которое просматривал пользователь В, но еще не видел пользователь А; допустим, это видео Z.
3. Рекомендовать видео Z пользователю А.

Главное различие между фильтрацией на основе содержимого и коллаборативной фильтрацией заключается в том, что СF, формируя рекомендации, не использует признаки видео, а полагается исключительно на историю пользовательских взаимодействий. У этого метода тоже есть как достоинства, так и недостатки.

Достоинства

- **Не требуется знания предметной области.** Коллаборативная фильтрация не зависит от признаков видео, поэтому не требуется конструировать признаки на основе видео с учетом предметной области.
- **Легко выявлять новые интересы пользователей.** Система может рекомендовать видео на новые темы, которыми похожие пользователи интересовались в прошлом.
- **Эффективность.** Модели на базе СF обычно быстрее работают и менее требовательны к вычислительным ресурсам, чем фильтрация на основании содержимого, потому что они не зависят от признаков видео.

Недостатки

- **Проблема «холодного старта».** Так называется ситуация, когда для нового видео или пользователя доступно мало данных, отчего система не может выработать точные рекомендации. СF страдает от проблемы «холодного старта», когда нет исторических данных о взаимодействиях для новых пользователей или новых видео; это мешает находить похожих пользователей или похожие видео. Позднее в разделе «Эксплуатация» мы расскажем, как наша система будет решать проблему «холодного старта».
- **Трудно учитывать узкие интересы.** Коллаборативная фильтрация плохо работает с пользователями, которые обладают специфическими или нишевыми интересами. Чтобы вырабатывать рекомендации, СF полагается на данные похожих пользователей, а найти похожего пользователя с такими же узкими интересами может быть трудно.

В табл. 6.1 сравниваются два типа фильтрации. Как видите, они дополняют друг друга.

Таблица 6.1. Сравнение фильтрации на основе содержимого и коллаборативной фильтрации

	Фильтрация на основе содержимого	Коллаборативная фильтрация
Обработка новых видео	✓	✗
Обнаружение новых областей интересов	✗	✓
Не требуется знание предметной области	✗	✓
Эффективность	✗	✓

Гибридная фильтрация

Гибридная фильтрация совмещает CF и фильтрацию на основе содержимого. Как показано на рис. 6.6, гибридная фильтрация объединяет рекомендательные системы двух перечисленных типов последовательно или параллельно. В реальных системах обычно применяется последовательная гибридная фильтрация [2].

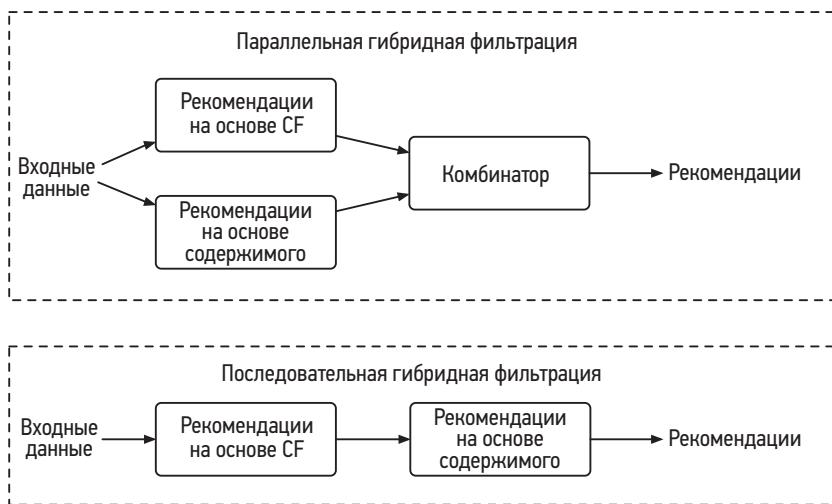


Рис. 6.6. Метод гибридной фильтрации

Гибридный подход повышает качество рекомендаций, потому что использует два источника данных: историю пользовательских взаимодействий и признаки видео. Признаки видео позволяют системе рекомендовать релевантные видео на основании того, какие ролики пользователь просматривал в прошлом, а кол-

лаборативная фильтрация помогает пользователям обнаруживать новые сферы интересов.

Какой метод лучше использовать?

Чтобы повысить качество рекомендаций, многие компании применяют гибридную фильтрацию. Например, в статье, опубликованной Google [2], рассказано, как YouTube, формируя подборку рекомендаций видео, на первой стадии применяет модель на основе CF (генератор кандидатов), а на второй стадии — модель на основе содержимого. Из-за преимуществ гибридной фильтрации мы выберем именно этот вариант.

Подготовка данных

Инженерия данных

Нам доступны такие данные:

- видео;
- пользователи;
- взаимодействие пользователей с видео.

Видео

Видеоданные — это необработанные видеофайлы и связанные с ними метаданные: идентификатор, продолжительность, название и т. д. Некоторые из этих атрибутов явно предоставляют отправители видео, а другие (например, продолжительность) система может определять неявно.

Таблица 6.2. Метаданные видео

ID видео	Продолжительность	Теги, назначенные вручную	Название, введенное вручную	Лайки	Просмотры	Язык
1	28	Собака, Семья	Наша очаровательная собака играет!	138	5300	Английский
2	300	Машина, Масло	Как заменить масло в машине?	5	250	Испанский
3	3600	Бали, Влог	Наш медовый месяц на Бали	2200	255 тыс.	Арабский

Пользователи

Следующая простая схема представляет данные о пользователях.

Таблица 6.3. Схема данных о пользователях

ID пользователя	Имя пользователя	Возраст	Пол	Город	Страна	Язык	Часовой пояс

Взаимодействия пользователей с видео

Сюда относятся различные взаимодействия пользователей с видео, включая лайки, клики, показы и прошлые поиски. Взаимодействия регистрируются вместе с прочей контекстной информацией, включая местонахождение и временную метку. Таблица 6.4 показывает, как могут храниться взаимодействия пользователей с видео.

Таблица 6.4. Данные о взаимодействиях пользователей с видео

ID пользователя	ID видео	Тип взаимодействия	Значение взаимодействия	Геопозиция (широта, долгота)	Временная метка
4	18	Лайк	–	38.8951 -77.0364	1658451361
2	18	Показ	8 секунд	38.8951 -77.0364	1658451841
2	6	Просмотр	46 минут	41.9241 -89.0389	1658822820
6	9	Клик	–	22.7531 47.9642	1658832118
9	–	Поиск	Основы кластеризации	22.7531 47.9642	1659259402
8	6	Комментарий	Отличное видео. Спасибо	37.5189 122.6405	1659244197

Конструирование признаков

Система МО должна находить видео, релевантные для пользователей. Сконструируем признаки, которые помогут делать обоснованные предсказания.

Признаки видео

Некоторые важные признаки видео:

- идентификатор;
- продолжительность;
- язык;
- названия и теги.

Идентификатор видео

Идентификаторы — это категориальные данные. Чтобы представить их числовыми векторами, мы используем слой эмбеддингов, который будет анализироваться во время обучения модели.

Продолжительность

Это приблизительная длительность видео от начала до конца. Эта информация важна, потому что одни пользователи могут предпочитать более короткие видеоролики, а других могут интересовать длинные материалы.

Язык

Язык видео — тоже важный признак, потому что пользователи естественным образом предпочитают те или иные языки. Так как язык — категориальная переменная, которая принимает конечное множество дискретных значений, мы будем представлять ее с помощью слоя эмбеддингов.

Названия и теги

Названия и теги используются, чтобы описать видео. Их либо вводят вручную пользователь, который загружает видео на платформу, либо неявно предсказывают автономные модели МО. Названия и теги видео — ценные предикторы. Например, название «Как приготовить пиццу» означает, что видео имеет отношение к пицце и к кулинарии.

Как подготовить данные. Мы воспользуемся облегченной предварительно обученной моделью (такой, как CBOW [3]), чтобы отобразить теги на векторы признаков.

Название будет отображаться на вектор признаков с применением модели контекстно-зависимых эмбеддингов слов — например, предварительно обученной модели BERT [4].

На рис. 6.7 представлена подготовка признаков видео.

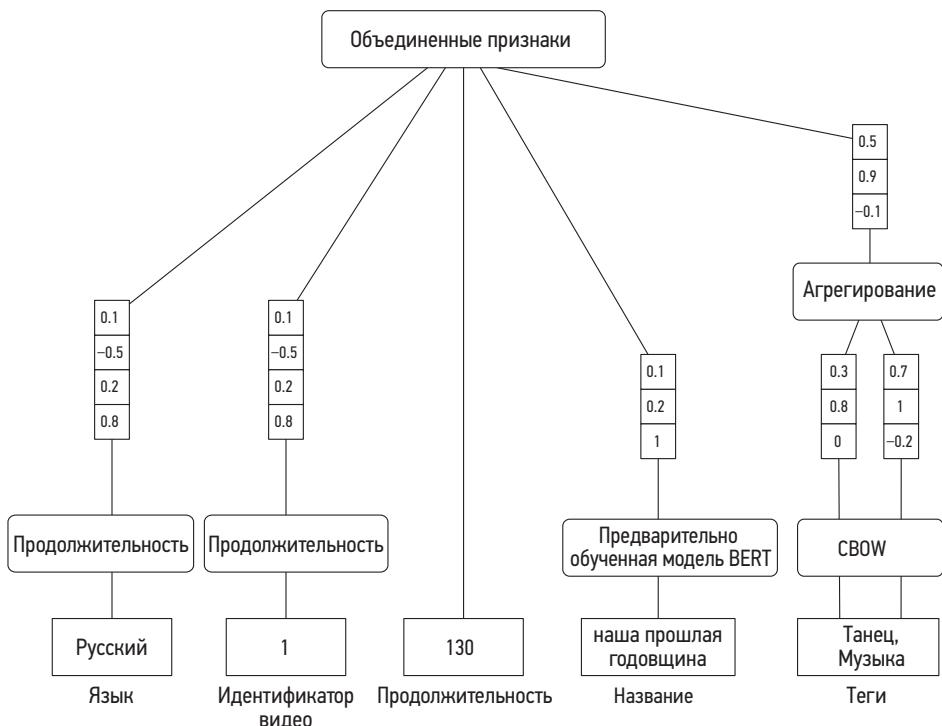


Рис. 6.7. Подготовка признаков видео

Признаки пользователей

Признаки пользователей мы разделим на следующие категории:

- демографические признаки;
- контекстная информация;
- история взаимодействий пользователя.

Демографические признаки

Сводка основных демографических признаков представлена на рис. 6.8.

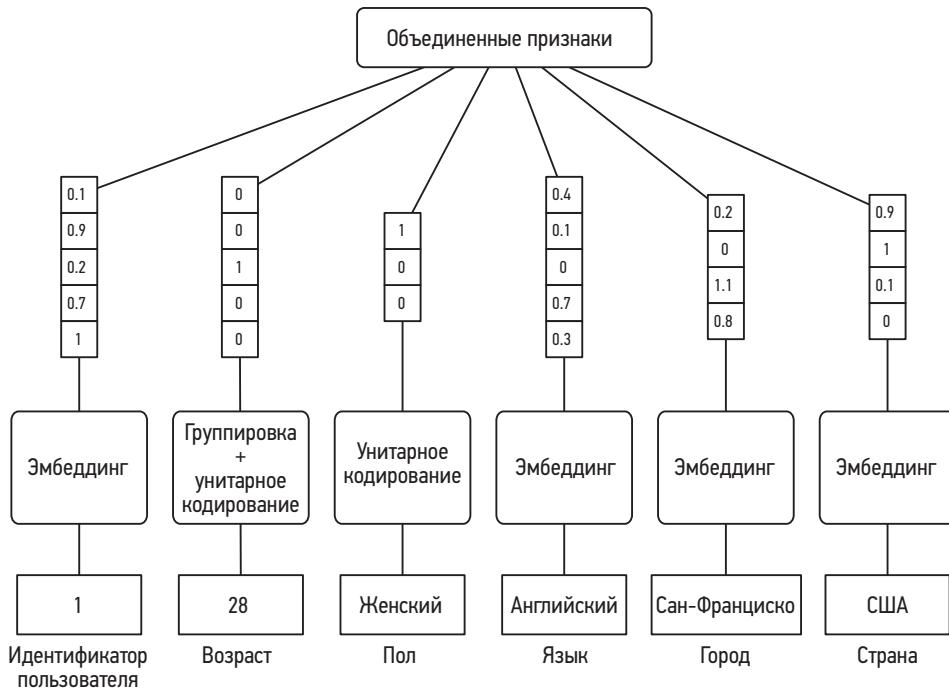


Рис. 6.8. Признаки, основанные на демографических данных пользователя

Контекстная информация

Вот некоторые важные признаки, относящиеся к контекстной информации.

- **Время суток.** Пользователь может смотреть разные видеоролики в разное время суток. Например, программист может вечером просматривать больше обучающих роликов.
- **Устройство.** На мобильных устройствах пользователи могут предпочитать более короткие видео.
- **День недели.** Пользователи могут предпочитать разные видео в зависимости от дня недели.

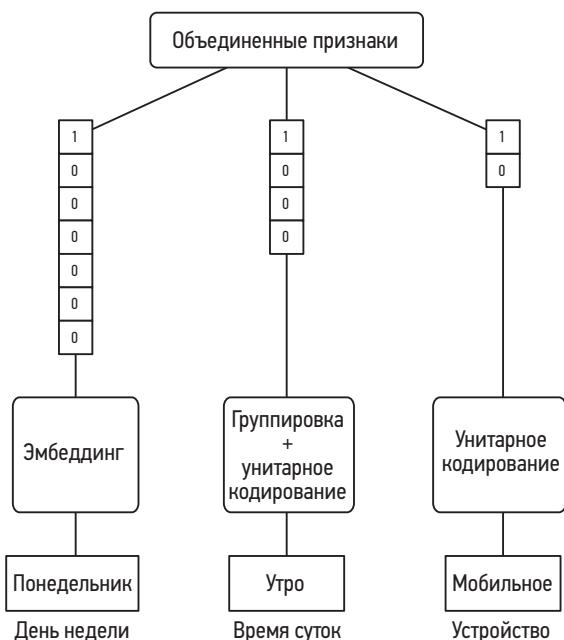


Рис. 6.9. Признаки, относящиеся к контекстной информации

История взаимодействий пользователя

История взаимодействий пользователя существенно помогает понимать его интересы. Вот некоторые из соответствующих признаков:

- история поиска;
- понравившиеся видео;
- просмотры и показы видео.

История поиска

Почему это важно. Предыдущие поисковые запросы показывают, какие видеоролики пользователь смотрел раньше, а прошлое поведение нередко является индикатором будущего.

Как подготовить данные. С помощью предварительно обученной модели эмбеддингов (например, BERT) отобразите каждый поисковый запрос на вектор эмбеддинга. Обратите внимание, что история поиска пользователя — это список текстовых запросов переменного размера. Чтобы создать вектор признаков фиксированного размера, обобщающий все поисковые запросы, мы будем усреднять эмбеддинги запросов.

Понравившиеся видео

Почему это важно. По тому, каким видеороликам пользователь ранее ставил лайки, можно судить о том, какой тип контента его интересует.

Как подготовить данные. Идентификаторы видео отображаются на векторы эмбеддингов через слой эмбеддингов. Как и в случае с историей поиска, мы усредняем эмбеддинги понравившихся роликов, чтобы получить вектор понравившихся видео фиксированного размера.

Просмотры и показы видео

Признаки для просмотров и показов видео конструируются почти так же, как для понравившихся видео, так что не будем повторяться.

На рис. 6.10 представлена краткая сводка признаков, которые относятся к взаимодействиям пользователей с видео.

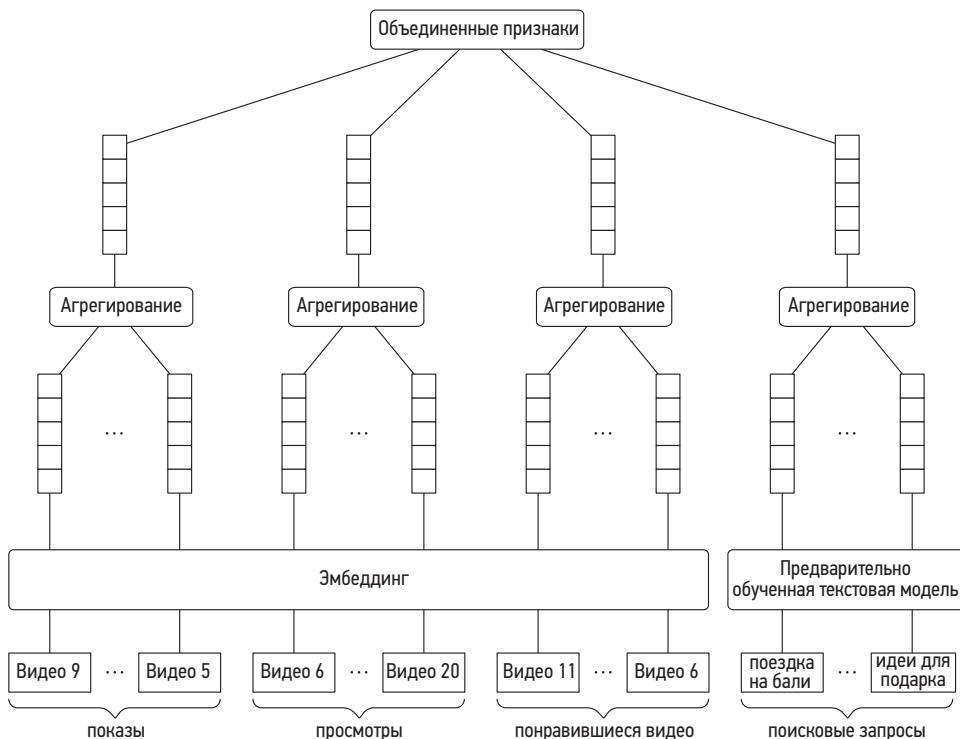


Рис. 6.10. Признаки, относящиеся к взаимодействиям пользователей с видео

Разработка модели

В этом разделе мы рассмотрим две модели, которые основаны на эмбеддингах и обычно применяются в рекомендательных системах на основе CF или содержимого:

- матричная факторизация;
- двухбашенная (two-tower) нейронная сеть.

Матричная факторизация

Чтобы понять модель матричной факторизации, необходимо знать, что такое матрица обратной связи.

Матрица обратной связи

Эта матрица представляет мнения пользователей о видео. На рис. 6.11 представлена двоичная матрица обратной связи «пользователь — видео», в которой каждая строка соответствует пользователю, а каждый столбец — видео. Элементы матрицы представляют мнение пользователя о видео. В этой главе мы будем называть пары {пользователь, видео} со значениями 1 «наблюдаемыми» или «положительными».

	Видео 1	Видео 2	Видео 3	Видео 4	Видео 5
Пользователь 1	1	1			
Пользователь 2			1	1	
Пользователь 3		1			1

Рис. 6.11. Матрица обратной связи «пользователь — видео»

Как узнать, считает ли пользователь то или иное видео релевантным? Возможны три варианта:

- явная обратная связь;
- неявная обратная связь;
- комбинация явной и неявной обратной связи.

Явная обратная связь. Матрица обратной связи строится на основе взаимодействий, которые явно выражают мнение пользователя о видео, таких как лайки или пересылки. Явная обратная связь точно передает мнение пользователей, потому что она опирается на конкретные действия, в которых выражается их интерес к видео. Однако у этого варианта есть серьезный недостаток: матрица оказывается разреженной, потому что лишь небольшая часть пользователей предоставляет явную обратную связь. С разреженными данными модели МО труднее обучаются.

Неявная обратная связь. В этом варианте используются взаимодействия, которые неявно указывают на мнение пользователя о видео, — например, клики или время просмотра. Неявная обратная связь дает больше точек данных, и обученная модель получается лучше. Главный недостаток в том, что неявная обратная связь не отражает мнения пользователей напрямую и может вносить много шума.

Комбинация явной и неявной обратной связи. Этот вариант объединяет явную и неявную обратную связь с помощью эвристик.

Какой вариант лучше всего подойдет для матрицы обратной связи?

Поскольку модель должна обучиться на значениях матрицы обратной связи, важно построить матрицу, которая хорошо соответствует ранее выбранной цели МО.

В нашем примере цель МО — максимизировать релевантность, при этом релевантность определяется как комбинация явной и неявной обратной связи. Следовательно, лучше всего подойдет вариант с этой комбинацией.

Модель матричной факторизации

Матричная факторизация — это простая модель эмбеддингов. Алгоритм раскладывает матрицу обратной связи «пользователь — видео» на произведение двух матриц меньшей размерности. Одна из этих матриц представляет эмбеддинги пользователей, а другая — эмбеддинги видео. Иначе говоря, модель обучается отображать каждого пользователя на вектор эмбеддинга и каждый видеоролик на вектор эмбеддинга так, чтобы расстояние между этими векторами представляло релевантность видео для пользователя. На рис. 6.12 показано, как можно разложить матрицу обратной связи на эмбеддинги пользователей и видео.

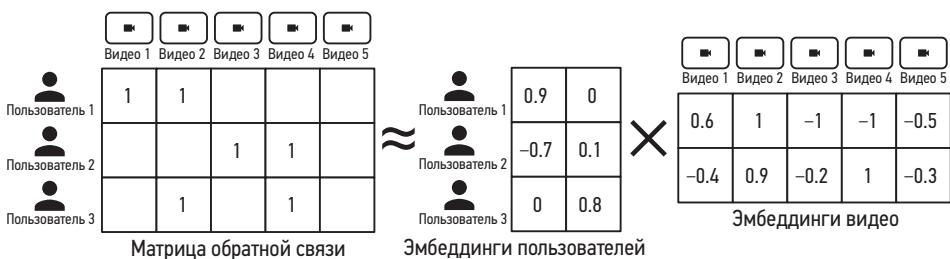


Рис. 6.12. Разложение матрицы обратной связи на две матрицы

Обучение матричной факторизации

В процессе обучения мы стараемся построить матрицы эмбеддингов пользователей и видео так, чтобы их произведение хорошо аппроксимировало матрицу обратной связи (рис. 6.13).

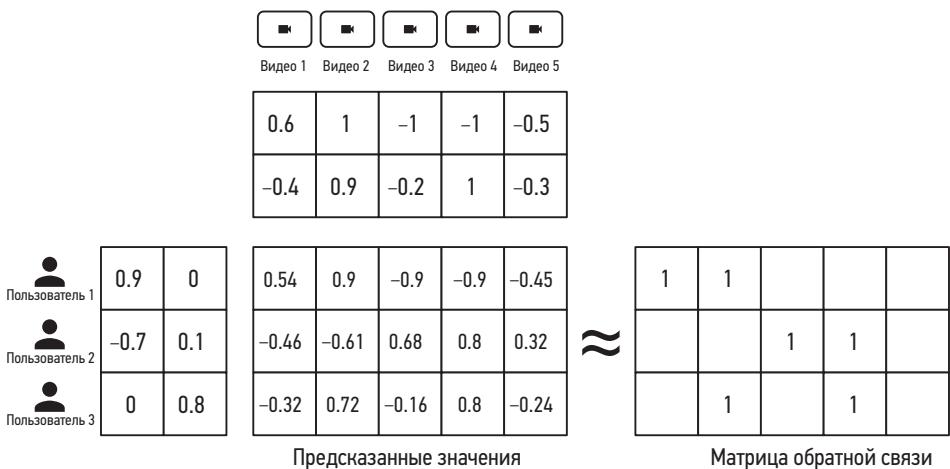


Рис. 6.13. Произведение эмбеддингов должно аппроксимировать матрицу обратной связи

В процессе обучения матричная факторизация сначала случайным образом инициализирует две матрицы эмбеддингов, а затем итеративно оптимизирует эмбеддинги, сокращая потери между матрицей предсказанных значений и матрицей обратной связи. Важным фактором обучения становится выбор функции потерь. Рассмотрим несколько вариантов:

- квадраты расстояний по наблюдаемым парам (пользователь, видео);
- квадраты расстояний по наблюдаемым и ненаблюдаемым парам (пользователь, видео);

- взвешенная комбинация квадратов расстояний по наблюдаемым и ненаблюдаемым парам.

Квадраты расстояний по наблюдаемым парам (пользователь, видео)

Эта функция потерь оценивает сумму квадратов расстояний по всем парам наблюдаемых (то есть ненулевых) элементов матрицы обратной связи. Это показано на рис. 6.14.

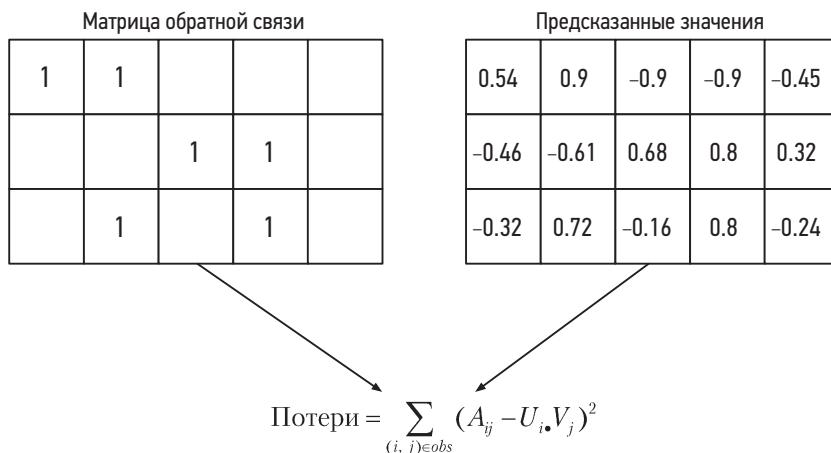


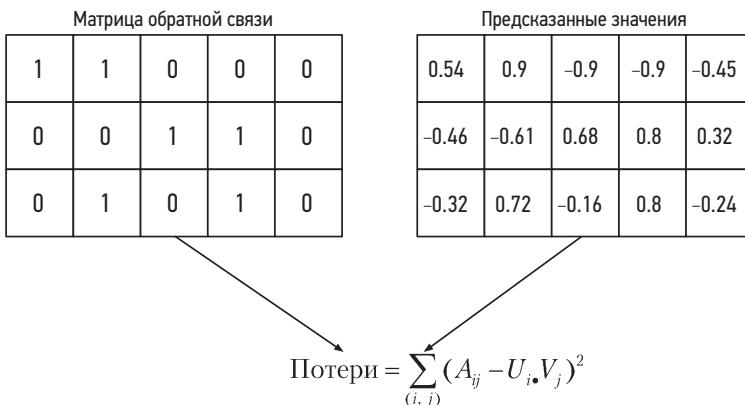
Рис. 6.14. Квадраты расстояний по наблюдаемым парам (пользователь, видео)

A_{ij} обозначает элемент на пересечении i -й строки и j -го столбца в матрице обратной связи, U_i — эмбеддинг i -го пользователя, V_j — эмбеддинг j -го видео, а суммирование производится только по наблюдаемым парам (obs).

Суммирование только по наблюдаемым парам приводит к неудачным эмбеддингам, потому что функция потерь не штрафует модель за плохие предсказания по ненаблюдаемым парам. Например, если матрица эмбеддингов состоит из одних единиц, на обучающих данных у нее будут нулевые потери. Тем не менее такие эмбеддинги будут плохо работать для неизвестных ранее пар (пользователь, видео).

Квадраты расстояний по наблюдаемым и ненаблюдаемым парам (пользователь, видео)

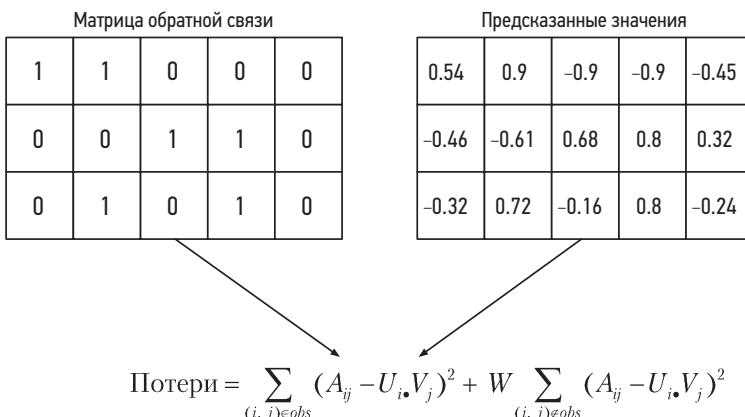
Эта функция потерь рассматривает ненаблюдаемые пары как отрицательные точки данных и присваивает им нули в матрице обратной связи. Как показано на рис. 6.15, функция потерь вычисляет сумму квадратов расстояний по всем элементам матрицы обратной связи.

**Рис. 6.15.** Квадраты расстояний по всем парам (пользователь, видео)

Эта функция потерь решает упомянутую выше проблему, назначая штрафы за плохие предсказания по ненаблюдаемым элементам. Однако у нее тоже есть серьезный недостаток. Матрица обратной связи обычно разрежена (в ней много ненаблюдаемых пар), так что в процессе обучения ненаблюдаемые пары доминируют над наблюдаемыми. В результате получаются предсказания, которые в основном близки к нулю. Это нежелательное явление, которое снижает эффективность обобщений для ненаблюдаемых пар (пользователь, видео).

Взвешенная комбинация квадратов расстояний по наблюдаемым и ненаблюдаемым парам

Чтобы преодолеть недостатки функций потерь, описанных выше, мы выберем вариант со взвешенной комбинацией обеих функций.

**Рис. 6.16.** Комбинированная функция потерь

Первая сумма в этой формуле вычисляет потери по наблюдаемым парам, а вторая — по ненаблюдаемым. W — гиперпараметр, который задает весовой коэффициент для двух сумм и обеспечивает, чтобы ни одна сумма не доминировала над другой на этапе обучения. Эта функция потерь с правильно отрегулированным значением W хорошо работает на практике [5], так что мы выберем ее для своей системы.

Оптимизация матричной факторизации

Чтобы обучить модель МО, необходим алгоритм оптимизации. В матричной факторизации часто применяются два алгоритма:

- Стохастический градиентный спуск (SGD, Stochastic Gradient Descent). Этот алгоритм используется для минимизации потерь [6].
- Метод взвешенных чередующихся наименьших квадратов (WALS, Weighted Alternating Least Squares). Этот алгоритм специфичен для матричной факторизации и работает так:
 - 1) зафиксировать одну матрицу эмбеддингов (U) и оптимизировать другую матрицу (V);
 - 2) зафиксировать другую матрицу эмбеддингов (V) и оптимизировать первую (U);
 - 3) повторить.

WALS обычно сходится быстрее, и его можно распараллелить. Чтобы больше узнать о WALS, обращайтесь к [7]. Этот алгоритм будет использован в нашем примере.

Вычисление матричной факторизации

Чтобы предсказать релевантность между произвольным пользователем и видеороликом-кандидатом, мы вычисляем степень сходства между их эмбеддингами с помощью такой метрики, как скалярное произведение. Например, как показано на рис. 6.17, показатель релевантности между пользователем 2 и видео 5 равен 0.32.

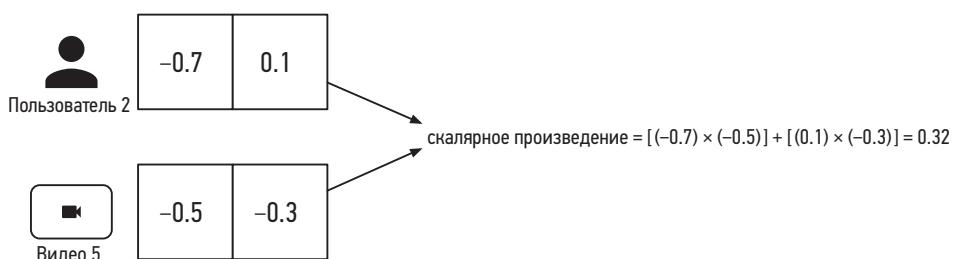


Рис. 6.17. Показатель релевантности для пары (пользователь 2, видео 5)

На рис. 6.18 представлены предсказанные значения для всех пар (пользователь, видео). Система возвращает рекомендуемые видео на основании показателей релевантности.



	Видео 1	Видео 2	Видео 3	Видео 4	Видео 5
Пользователь 1	0.6	1	-1	-1	-0.5
Пользователь 2	-0.4	0.9	-0.2	1	-0.3
Пользователь 3	0.9	0	0.54	0.9	-0.9
	-0.7	0.1	-0.46	-0.61	0.68
	0	0.8	-0.32	0.72	-0.16
			0.8	0.8	-0.24

Рис. 6.18. Предсказанные показатели релевантности для пар

НАПОМИНАНИЕ

Так как в матричной факторизации используются только взаимодействия пользователей с видео, она обычно применяется в коллаборативной фильтрации.

Прежде чем завершить тему матричной факторизации, рассмотрим достоинства и недостатки этой модели.

Достоинства

- Скорость обучения: матричная факторизация эффективна на этапе обучения, потому что нужно сформировать всего две матрицы.
- Скорость обслуживания: матричная факторизация быстро работает на стадии эксплуатации. При обучении получаются статичные эмбеддинги; это значит, что после обучения их можно использовать повторно, не преобразуя входные данные во время запроса.

Недостатки

- Матричная факторизация учитывает только взаимодействия пользователей с видео и не действует другие признаки — например, возраст или язык пользователей. Это ограничивает предсказательную способность модели, потому что такие признаки, как язык, помогают повысить качество рекомендаций.

- Трудно обрабатывать новых пользователей, у которых еще нет достаточного количества взаимодействий для того, чтобы модель могла вычислить осмысленные эмбеддинги. В этом случае матричная факторизация не может определить, релевантен ли видеоролик для пользователя, вычисляя скалярное произведение их эмбеддингов.

Посмотрим, как двухбашенные нейронные сети устраниют недостатки матричной факторизации.

Двухбашенная нейронная сеть

Двухбашенная нейронная сеть состоит из двух линий-кодировщиков («башен»): башня пользователей и башня видео. Кодировщик пользователей принимает на вход признаки пользователя и отображает их на вектор эмбеддинга (эмбеддинг пользователя). Кодировщик видео поступает аналогично с признаками видео, чтобы получить эмбеддинг видео. Расстояние между эмбеддингами в общем пространстве эмбеддингов представляет релевантность.

Двухбашенная архитектура изображена на рис. 6.19. В отличие от матричной факторизации, гибкость таких архитектур позволяет включать разнообразные признаки, чтобы лучше отражать специфические интересы пользователя.

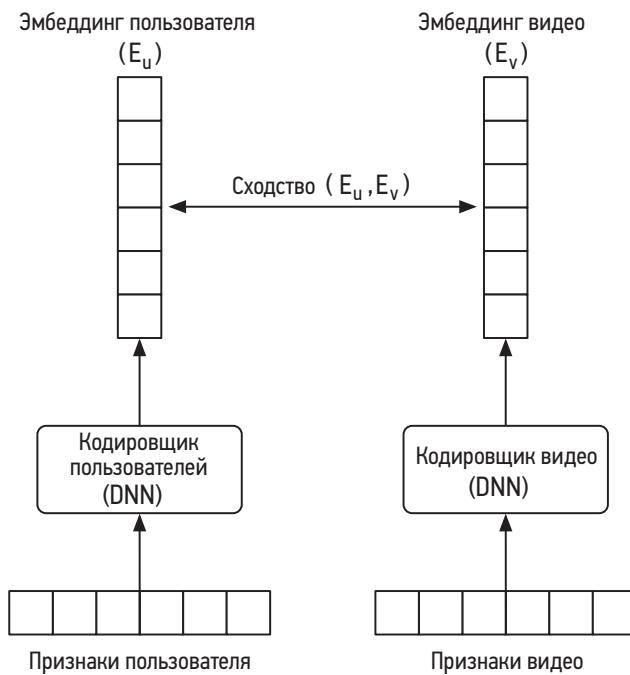


Рис. 6.19. Двухбашенная нейронная сеть

Построение датасета

Чтобы построить датасет, мы будем выделять признаки из разных пар \langle пользователь, видео \rangle и помечать их как положительные или отрицательные в зависимости от обратной связи пользователя. Например, пара будет считаться положительной, если пользователь поставил лайк видеоролику или просмотрел его хотя бы наполовину.

Чтобы построить отрицательные точки данных, можно выбрать либо случайные видеоролики, которые не должны быть релевантными, либо те, которым пользователь явно поставил дизлайк. На рис. 6.20 представлен пример построенных точек данных.

№	Признаки, относящиеся к пользователю	Признаки, относящиеся к видео	Метка													
1	<table border="1"><tr><td>0</td><td>0</td><td>1</td><td>0.7</td><td>-0.6</td><td>0</td><td>0</td></tr></table>	0	0	1	0.7	-0.6	0	0	<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>0.9</td><td>0.9</td><td>1</td></tr></table>	0	1	0	0.9	0.9	1	1 (положительная)
0	0	1	0.7	-0.6	0	0										
0	1	0	0.9	0.9	1											
2	<table border="1"><tr><td>0</td><td>1</td><td>1</td><td>0.2</td><td>0.1</td><td>1</td><td>0</td></tr></table>	0	1	1	0.2	0.1	1	0	<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>-0.1</td><td>0.3</td><td>1</td></tr></table>	0	1	0	-0.1	0.3	1	2 (отрицательная)
0	1	1	0.2	0.1	1	0										
0	1	0	-0.1	0.3	1											

Рис. 6.20. Две построенные точки данных

Следует заметить, что пользователи обычно считают релевантными только небольшую долю всех видеороликов. При построении обучающих данных это приводит к несбалансированному набору данных, в котором отрицательных пар намного больше, чем положительных. Обучить модель на таком наборе достаточно сложно. Чтобы решить проблему дисбаланса, можно воспользоваться методами из главы 1 «Введение и общие сведения».

Выбор функции потерь

Так как двухбашенная нейронная сеть обучается прогнозировать бинарные метки, проблему можно отнести к категории задач классификации. Чтобы оптимизировать кодировщики на этапе обучения, мы воспользуемся типичной классификационной функцией потерь, такой как перекрестная энтропия. Этот процесс показан на рис. 6.21.

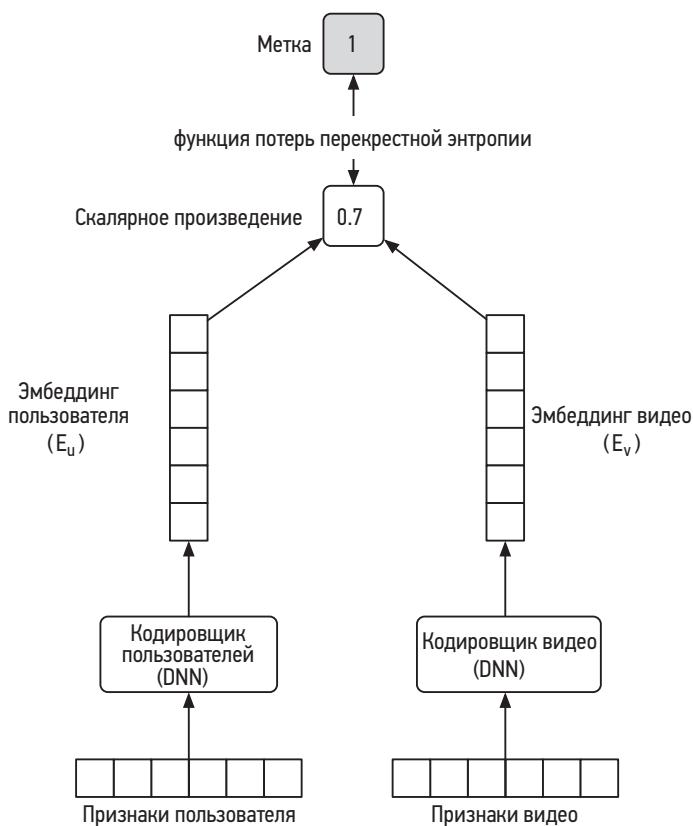


Рис. 6.21. Процесс обучения двухбашенной нейронной сети

Работа двухбашенной нейронной сети

В процессе эксплуатации система использует эмбеддинги, чтобы найти наиболее релевантные видео для заданного пользователя. Это классический пример задачи поиска ближайшего соседа. Чтобы эффективно находить первые k наиболее похожих эмбеддингов видео, мы воспользуемся методом приближенного поиска ближайшего соседа.

Двухбашенные нейронные сети используются как для фильтрации на основе содержимого, так и для коллаборативной фильтрации. Во втором случае, как по-

казано на рис. 6.22, кодировщик видео представляет собой не что иное, как слой эмбеддингов, который преобразует идентификатор видео в вектор эмбеддинга. При таком подходе модель не зависит от других признаков видео.

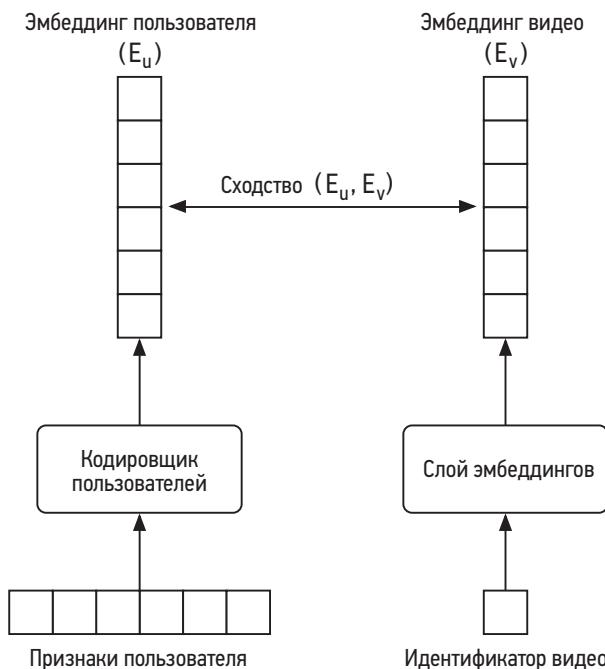


Рис. 6.22. Двухбашенная нейронная сеть для коллаборативной фильтрации

Рассмотрим достоинства и недостатки двухбашенной нейронной сети.

Достоинства

- **Задействуются признаки пользователей.** Модель принимает на вход признаки пользователей (возраст, пол и т. д.), и эти предсказательные признаки помогают формировать лучшие рекомендации.
- **Учитываются новые пользователи.** Модель легко обрабатывает новых пользователей, потому что она полагается на признаки пользователей (возраст, пол и т. д.).

Недостатки

- **Низкая скорость обслуживания.** Модель должна вычислять эмбеддинг пользователя во время запроса, поэтому запросы обрабатываются медленнее. Кроме того, если модель используется для фильтрации на основе содержи-

мого, она должна преобразовывать признаки видео в эмбеддинги видео, что замедляет обслуживание.

- **Обучение более затратно.** У двухбашенных нейронных сетей больше параметров для обучения, чем у матричной факторизации, а следовательно, обучение сильнее нагружает вычислительные ресурсы.

Матричная факторизация или двухбашенная нейронная сеть?

В табл. 6.5 приведена сводка различий между матричной факторизацией и архитектурой двухбашенной нейронной сети.

Таблица 6.5. Матричная факторизация и двухбашенная нейронная сеть

	Матричная факторизация	Двухбашенная нейронная сеть
Затраты на обучение	Более эффективное обучение	Более затратное обучение
Скорость обработки	Быстрее, потому что эмбеддинги являются статическими и могут вычисляться заранее	Признаки пользователей преобразуются в эмбеддинги во время запроса
Проблема «холодного старта»	Трудно обрабатывать новых пользователей	Новые пользователи обрабатываются легко, потому что используются признаки пользователей
Качество рекомендаций	Не идеально, потому что модель не использует признаки пользователей и видео	Более качественные рекомендации, потому что задействовано больше признаков

Оценка

Качество системы можно оценивать автономными и оперативными метриками.

Автономные метрики

Следующие автономные метрики широко используются в рекомендательных системах.

Точность на k оценивает долю релевантных видео среди первых k рекомендованных видео. Можно использовать несколько значений k (например, 1, 5, 10).

mAP оценивает качество ранжирования рекомендуемых видео. Она хорошо подходит для нашего примера, потому что здесь используются бинарные показатели релевантности.

Разнообразие (diversity) оценивает, насколько рекомендованные видео отличаются друг от друга. Важно отслеживать эту характеристику, потому что пользователей больше интересуют разнообразные видео. Разнообразие вычисляется как среднее попарное сходство между видеороликами в списке (например, косинусный коэффициент или скалярное произведение). Низкое среднее попарное сходство указывает на то, что список разнообразен.

Не стоит использовать разнообразие как единственную метрику качества, потому что это может привести к неверным интерпретациям. Например, если рекомендуемые видео разнообразны, но не релевантны для пользователя, он вряд ли сочтет такие рекомендации полезными. Таким образом, имеет смысл использовать разные автономные метрики, чтобы контролировать и релевантность, и разнообразие.

Оперативные метрики

В реальных системах в ходе оперативной оценки отслеживается множество метрик. Самые важные из них:

- кликабельность (CTR);
- количество просмотров до конца;
- общее время просмотра;
- явная обратная связь от пользователей.

Кликабельность (CTR). Отношение количества видеороликов, на которые нажали пользователи, к общему количеству рекомендованных видео. Формула для вычисления:

$$\text{CTR} = \frac{\text{количество видеороликов, на которые нажали пользователи}}{\text{общее количество рекомендованных видео}}.$$

CTR хорошо помогает отслеживать вовлеченность пользователей, но эта метрика не позволяет выявлять или учитывать кликбейтные видео.

Количество просмотров до конца. Общее количество рекомендованных видео, которые пользователи просмотрели до конца. По этой метрике можно понять, насколько часто система рекомендует видео, которые просматриваются пользователями.

Общее время просмотра. Общее время, которое пользователи провели за просмотром рекомендованных видео. Когда рекомендации интересны пользователям, они проводят в целом больше времени за просмотром.

Явная обратная связь от пользователей. Общее количество видеороликов, которым пользователи явно поставили лайки или дизлайки. Эта метрика точно отражает мнение пользователей о рекомендованных видео.

Эксплуатация

На стадии эксплуатации система рекомендует каждому пользователю самые релевантные видео, чтобы сузить выборку с миллиардов видеороликов до обозримого количества. В этом разделе мы предложим предсказательный пайплайн, который обслуживает запросы эффективно и точно.

Если при миллиардах доступных видеороликов мы выберем тяжеловесную модель с большим количеством входных признаков, обслуживание замедлится. С другой стороны, если выбрать облегченную модель, то может пострадать качество рекомендаций. Что же делать? Естественное решение — использовать сразу несколько моделей в многоступенчатой архитектуре. Например, в двухступенчатой архитектуре на первой ступени облегченная модель быстро сужает подборку видео — это называется генерацией кандидатов. На второй ступени, называемой скорингом, используется более сложная модель, которая точнее формирует оценки и ранжирует видео. На рис. 6.23 показано, как генерация кандидатов и скоринг сочетаются, чтобы выработать список релевантных видео.

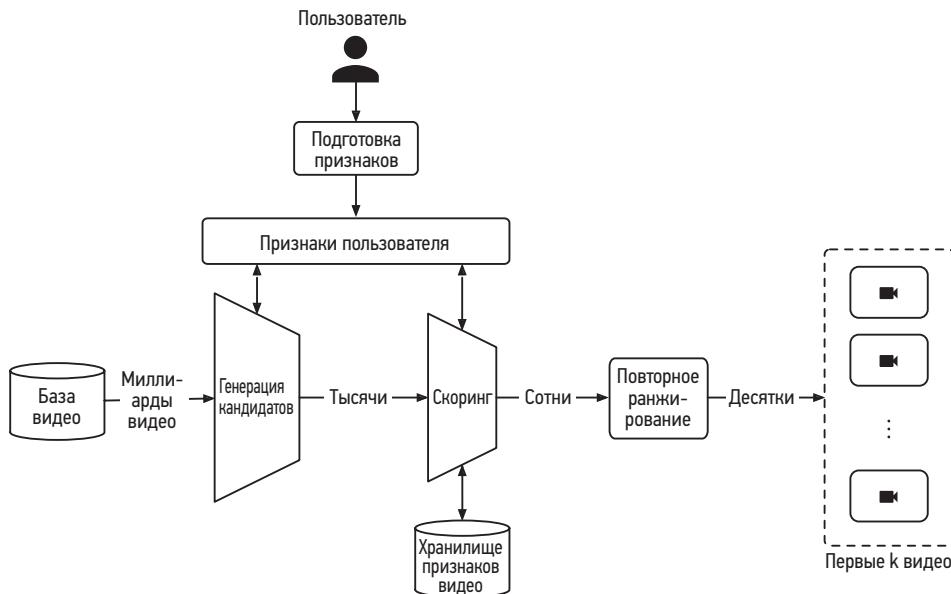


Рис. 6.23. Предсказательный пайплайн

Рассмотрим компоненты предсказательного пайплайна:

- генерация кандидатов;
 - скоринг;
 - повторное ранжирование.

Генерация кандидатов

Цель генерации кандидатов — сократить выборку видео от потенциальных миллиардов до тысяч. На этой ступени производительность важнее точности, а на ложноположительные результаты можно не обращать внимания.

Чтобы кандидаты генерировались быстро, мы выбираем модель, которая не зависит от признаков видео. Кроме того, модель должна поддерживать предсказания для новых пользователей. Для этой ступени хорошо подойдет двухбашенная нейронная сеть.

На рис. 6.24 показана схема генерации кандидатов. Генератор получает эмбеддинг пользователя от кодировщика пользователей. После того как вычисления завершены, он получает наиболее похожие видеоролики от сервиса приближенного поиска ближайшего соседа. Эти видео ранжируются по сходству в пространстве эмбеддингов и возвращаются в качестве выходных данных.

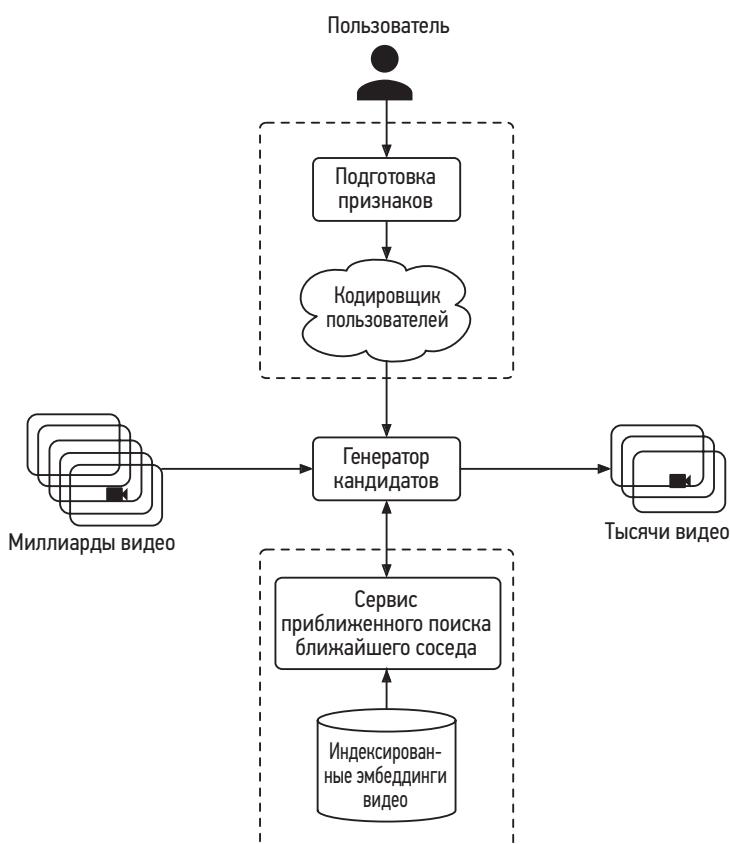


Рис. 6.24. Генерация кандидатов

В реальных системах может использоваться несколько генераторов кандидатов, потому что это нередко повышает эффективность рекомендаций. Посмотрим, почему так происходит.

Видео может интересовать пользователей по многим причинам. Например, тот или иной пользователь хочет просмотреть видео, потому что оно популярно, злободневно или относится к его местоположению. Чтобы такие видео попадали в рекомендации, обычно используется несколько генераторов кандидатов, как показано на рис. 6.25.

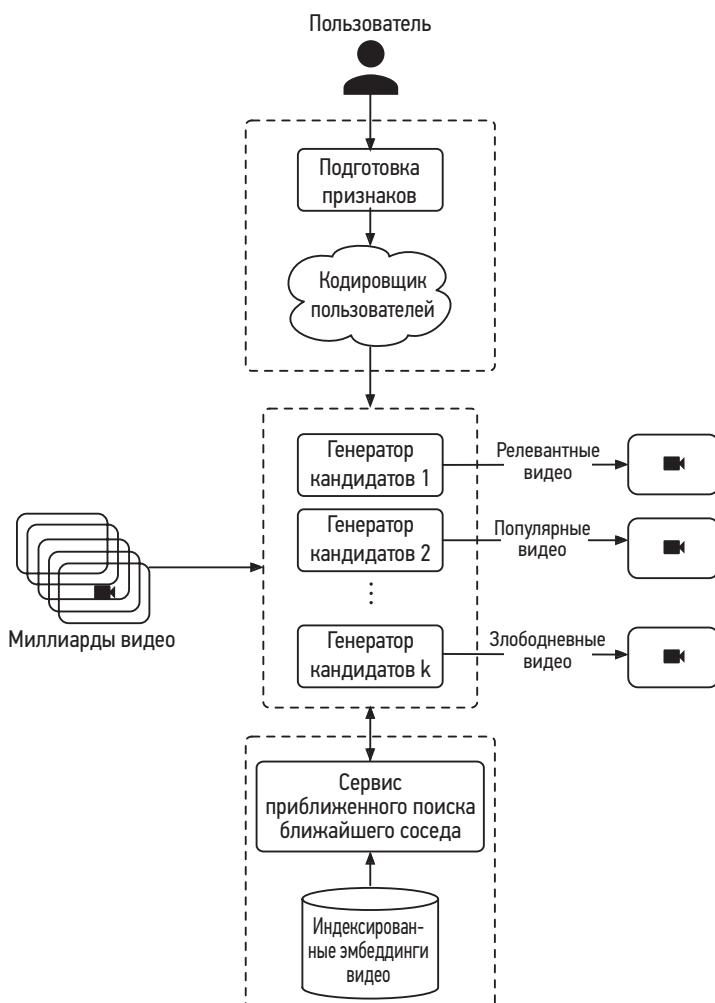


Рис. 6.25. Использование k генераторов кандидатов обеспечивает разнообразие рекомендуемых видео

Как только выборку потенциально подходящих видео удастся сократить с миллиардов до тысяч, эти видео можно будет ранжировать с помощью скоринга, прежде чем выводить.

Скоринг

Скоринг, который также называется ранжированием, принимает на вход данные пользователя и видеоролики-кандидаты, присваивает каждому видео оценку и выводит ранжированный список видео.

На этой ступени точность важнее производительности. Из этих соображений мы выбираем фильтрацию на основе содержимого и модель, основанную на признаках видео. Обычно для этой цели используется двухбашенная нейронная сеть. Поскольку на этапе скоринга обычно ранжируются только небольшая выборка видео, можно применить более тяжеловесную модель с большим количеством параметров. На рис. 6.26 представлена общая схема скоринга.

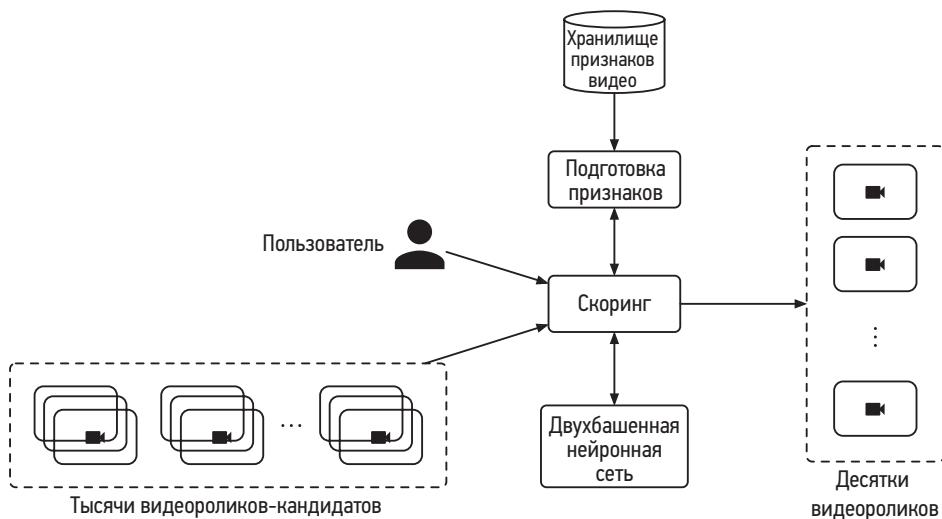


Рис. 6.26. Общая схема скоринга

Повторное ранжирование

Этот компонент повторно ранжирует результаты, добавляя дополнительные критерии или ограничения. Например, с помощью отдельных моделей МО можно выявлять кликбайтные видео. Внедряя повторное ранжирование, стоит учитывать несколько важных факторов:

- региональные ограничения для видео;
- свежесть видео;

- видео, содержащее дезинформацию;
- дубликаты или почти идентичные видео;
- объективность (fairness) и смещение.

Трудности при разработке систем рекомендации видео

Прежде чем завершить эту главу, посмотрим, как наше решение преодолевает типичные трудности, присущие системам рекомендации видео.

Скорость обслуживания

Очень важно, чтобы рекомендации видео работали быстро. Но, поскольку в системе хранятся миллиарды видеороликов, оказывается непросто оперативно вырабатывать точные рекомендации.

Чтобы справиться с этой проблемой, используется двухступенчатая архитектура. На первой ступени применяется облегченная модель, которая быстро сокращает количество кандидатов с миллиардов до тысяч. YouTube использует аналогичное решение [2], а в Instagram внедрена многоступенчатая архитектура [8].

Точность

Точность обеспечивается благодаря скорингу: видео ранжируется с помощью мощной модели, которая основана на большем количестве признаков, включая признаки видео. Использование более мощной модели не ухудшает скорость обслуживания, потому что в результате генерации кандидатов выбирается относительно небольшое подмножество видео.

Разнообразие

Многие пользователи предпочитают, чтобы рекомендации были разнообразными. Чтобы система создавала разнообразные подборки видео, мы применяем несколько генераторов кандидатов, как объясняется в разделе «Генерация кандидатов».

Проблема «холодного старта»

Как наша система решает проблему «холодного старта»?

Для новых пользователей. Когда пользователь только начинает работать с нашей платформой, данные о его взаимодействиях недоступны.

В таком случае предсказания делаются с помощью двухбашенных нейронных сетей на основе таких признаков, как возраст, пол, язык, местонахождение и т. д. Рекомендованные видео до определенной степени персонализируются даже для новых пользователей. По мере того как пользователь взаимодействует с большим количеством видео, мы можем делать более точные предсказания на основе новых взаимодействий.

Для новых видео. Когда в систему добавляется новый видеоролик, его метаданные и содержимое доступны, но взаимодействий еще нет. Проблему можно решить с помощью эвристики. Мы предлагаем видео случайным пользователям и собираем статистику взаимодействий. Накопив достаточно этой информации, можно отрегулировать двухбашенную нейронную сеть на основе новых взаимодействий.

Масштабируемость обучения

Непросто обучать модели на больших датасетах без серьезных затрат. В рекомендательных системах постоянно добавляются новые взаимодействия, а модели должны быстро адаптироваться, чтобы рекомендации получались точнее. Для этого модели должны легко регулироваться.

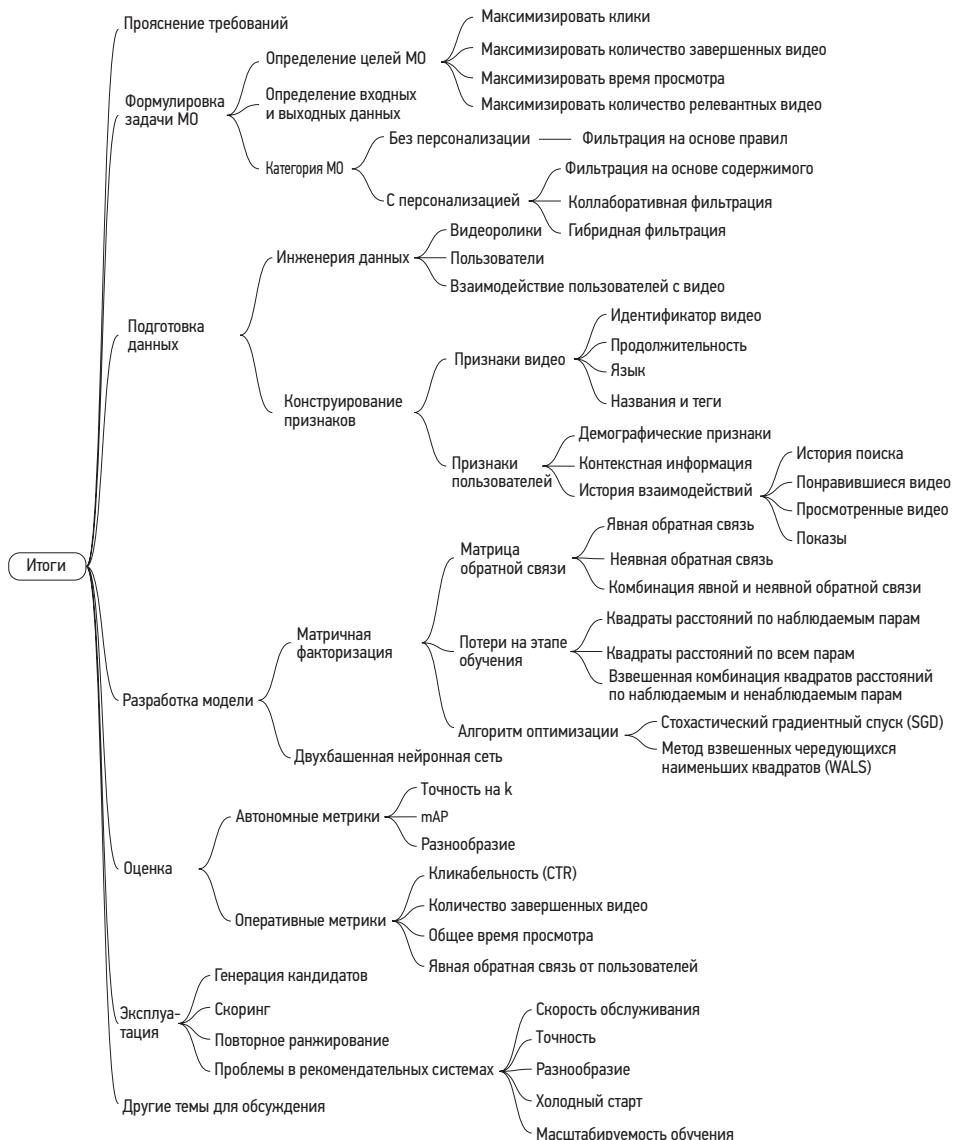
В нашем случае модели строятся на основе нейронных сетей и проектируются с расчетом на дальнейшую регулировку.

Другие темы для обсуждения

Если в конце собеседования еще остается время, можно обсудить дополнительные темы.

- Баланс между использованием готовых схем и поиском новых решений в рекомендательных системах [9].
- Различные виды смещений в рекомендательных системах [10].
- Важные этические аспекты разработки рекомендательных систем [11].
- Эффект сезонности в рекомендательных системах: изменения в поведении пользователей в разное время года [12].
- Как оптимизировать систему для нескольких целей вместо одной [13].
- Как извлекать пользу из отрицательной обратной связи — например, дизлайков [14].
- Как учитывать последовательность видео в истории поиска или просмотра пользователей [2].

Итоги



Ссылки

- [1] Рекомендательная система YouTube. <https://blog.youtube/inside-youtube/on-youtubes-recommendation-system>
- [2] Глубокие нейронные сети (DNN) для рекомендаций YouTube. <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45530.pdf>.
- [3] CBOW. <https://arxiv.org/pdf/1301.3781.pdf>
- [4] BERT. <https://arxiv.org/pdf/1810.04805.pdf>
- [5] Матричная факторизация. <https://developers.google.com/machine-learning/recognition/collaborative/matrix>
- [6] Стохастический градиентный спуск. https://en.wikipedia.org/wiki/Stochastic_gradient_descent
- [7] Оптимизация методом WALS. <https://fairyonice.github.io/Learn-about-collaborative-filtering-and-weighted-alternating-least-square-with-tensorflow.html>
- [8] Многоступенчатая рекомендательная система Instagram. <https://ai.facebook.com/blog/power-by-ai-instagram-explore-recommender-system/>
- [9] Баланс между использованием готовых схем и поиском новых решений («проблема многорукого бандита»). https://en.wikipedia.org/wiki/Multi-armed_bandit
- [10] Смещения в ИИ и рекомендательных системах. <https://www.searchenginejournal.com/biases-search-recommender-systems/339319/#close>
- [11] Этические аспекты разработки рекомендательных систем. <https://link.springer.com/article/10.1007/s00146-020-00950-y>.
- [12] Сезонность в рекомендательных системах. <https://www.computer.org/csdl/proceedings/article/big-data/2019/09005954/1hJsfT0qL6>
- [13] Многозадачная система ранжирования. <https://daiwk.github.io/assets/youtube-multitask.pdf>
- [14] Польза от отрицательной обратной связи. <https://arxiv.org/abs/1607.04228?context=cs>

7

СИСТЕМА РЕКОМЕНДАЦИИ СОБЫТИЙ

В этой главе мы спроектируем систему для рекомендации событий (мероприятий), сходную с используемой на Eventbrite. Это популярный портал для управления событиями и покупки билетов, на котором пользователи могут создавать и просматривать мероприятия и записываться на них. Рекомендательная система учитывает личные предпочтения пользователей и выводит события, актуальные для них.

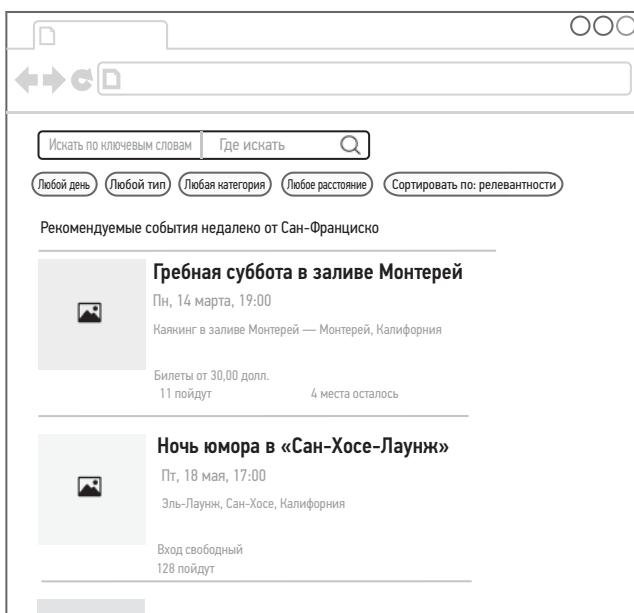


Рис. 7.1. Рекомендованные события

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: В чем состоит бизнес-цель? Можно ли считать, что главная цель — увеличить продажи билетов?

Эксперт: Да, вполне.

Соискатель: Кроме посещения событий (мероприятий), смогут ли пользователи бронировать номера в отелях или столики в ресторанах?

Эксперт: Для простоты будем считать, что поддерживаются только события.

Соискатель: Событием можно считать однократное явление, которое происходит в течение некоторого времени, а затем перестает существовать. Это так?

Эксперт: Это отличная трактовка.

Соискатель: Какие атрибуты бывают у события? Можно ли считать, что нам доступно его текстовое описание, диапазон цен, местонахождение, дата, время и т. д.?

Эксперт: Безусловно, это разумные предположения.

Соискатель: Доступны ли размеченные данные?

Эксперт: У нас нет набора данных с ручной разметкой. Чтобы построить обучающий набор, можно использовать данные о взаимодействиях пользователей с событиями.

Соискатель: Доступна ли информация о текущем местонахождении пользователя?

Эксперт: Да. Поскольку мы разрабатываем рекомендательную систему с учетом местонахождения, будем считать, что пользователи согласны делиться своей геопозицией.

Соискатель: Могут ли пользователи становиться друзьями на платформе? Информация о друзьях поможет создать персонализированную систему рекомендации событий.

Эксперт: Хороший вопрос. Да, будем считать, что пользователи могут «подружиться» на платформе. Дружба является взаимной: то есть если пользователь А является другом В, то В также является другом А.

Соискатель: Могут ли одни пользователи приглашать других на события?

Эксперт: Да.

Соискатель: Могут ли пользователи отвечать на приглашения?

Эксперт: Для простоты будем считать, что можно только записываться на события.

Соискатель: События будут платными или бесплатными?

Эксперт: Система должна поддерживать и те и другие.

Соискатель: Сколько всего пользователей и событий?

Эксперт: На каждый месяц назначается около 1 миллиона событий.

Соискатель: Сколько активных пользователей в день открывают сайт или приложение?

Эксперт: Около миллиона уникальных пользователей в день.

Соискатель: Так как мы разрабатываем рекомендательную систему событий с учетом геолокации, важно иметь возможность эффективно вычислять расстояние и время перемещения между двумя точками. Можно ли получать такие данные с помощью внешних API — например, Google Maps API или других картографических служб?

Эксперт: Точно подмечено! Считайте, что можно использовать сторонние службы для получения картографических данных.

Резюмируем описание проблемы. Требуется спроектировать систему рекомендации событий, которая предоставляет каждому пользователю персонализированный список событий. После того как событие завершилось, пользователи не могут на него записаться. Кроме записи на события, пользователи могут приглашать на них друг друга и устанавливать отношения дружбы. Обучающие данные должны строиться в процессе эксплуатации на основе взаимодействий пользователей. Главная цель системы — увеличить продажи билетов.

Формулировка проблемы в виде задачи МО

Определение цели МО

Согласно требованиям, бизнес-цель — увеличить продажи билетов. Один из способов преобразовать эту формулировку в четко определенную цель МО — поставить задачу максимизировать количество регистраций на события.

Определение входных и выходных данных системы

На вход система принимает данные пользователя, а на выходе возвращает первые k событий, ранжированных по релевантности для пользователя.

Выбор подходящей категории МО

Задачу формирования рекомендаций можно решить разными способами:

- использовать простые правила — например, рекомендовать популярные события;
- применять модели с эмбеддингами, опирающиеся на фильтрацию (на основе содержимого или коллаборативную);
- переформулировать проблему в виде задачи ранжирования.

Методы, основанные на правилах, могут послужить хорошей отправной точкой для того, чтобы выработать базовое решение. Однако методы на основе МО обычно приводят к более качественным результатам. В этой главе мы переформулируем проблему в виде задачи ранжирования и решим ее с помощью метода LTR (Learning To Rank, «обучение ранжированию»).

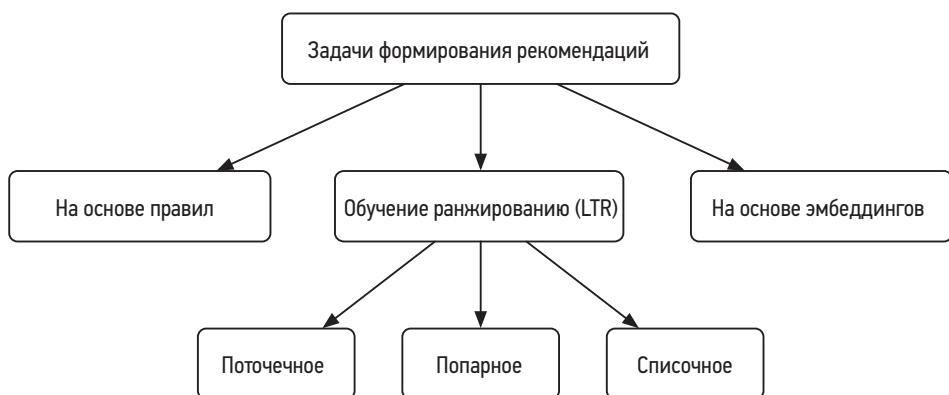


Рис. 7.2. Разные подходы к задачам формирования рекомендаций

LTR — класс алгоритмических методов, в которых применяется машинное обучение с учителем, чтобы решать задачи ранжирования. Задачу ранжирования можно формально определить так: «Есть запрос и список элементов; как лучше всего упорядочить элементы по убыванию релевантности?» Обычно различают три категории методов LTR: поточечные (pointwise), попарные (pairwise) и списочные (listwise). Кратко рассмотрим каждую из этих категорий. Обратите внимание, что подробное объяснение этих методов выходит за рамки книги. Если вы захотите больше узнать об LTR, обращайтесь к [1].

Поточечные методы LTR

Эти методы перебирают все элементы один за другим и предсказывают релевантность между запросом и элементом, используя методы классификации

или регрессии. В этом случае предсказание для каждого элемента не зависит от других элементов.



Рис. 7.3. Поточная модель ранжирования

Итоговое ранжирование определяется сортировкой предсказанных оценок релевантности.

Попарные методы LTR

Эти методы принимают два элемента и предсказывают, какой из них более релевантен для запроса.

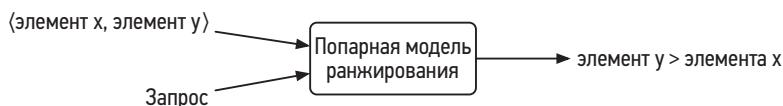


Рис. 7.4. Попарная модель ранжирования

Самые популярные попарные алгоритмы LTR — RankNet [2], LambdaRank [3] и LambdaMART [4].

Списочные методы LTR

Эти методы предсказывают оптимальный порядок для целого списка элементов в соответствии с запросом.

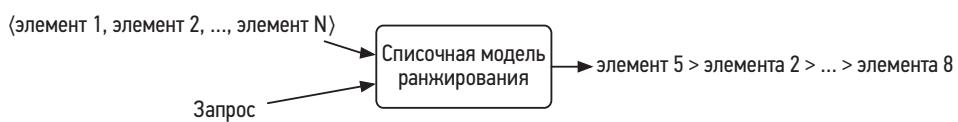


Рис. 7.5. Списочная модель ранжирования

Самые популярные списочные алгоритмы LTR — SoftRank [5], ListNet [6] и AdaRank [7].

В общем случае попарные и списочные методы обеспечивают более точные результаты, но создают больше трудностей с реализацией и обучением. Для простоты мы будем использовать в этом примере поточечный метод. В частности,

мы применим модель бинарной классификации, которая получает единичное событие и предсказывает вероятность того, что пользователь запишется на него. Этот подход представлен на рис. 7.6.

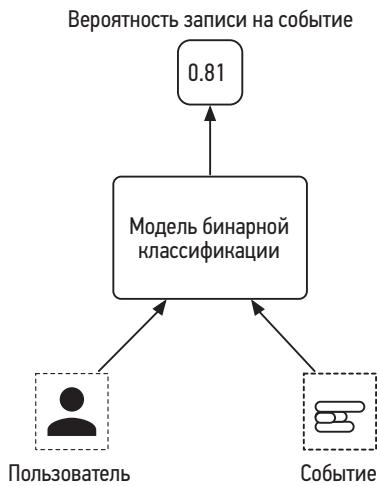


Рис. 7.6. Модель бинарной классификации

Подготовка данных

Инженерия данных

Чтобы сконструировать хорошие признаки, надо сначала понять, какие данные доступны в системе. На платформе управления событиями центральное место занимают пользователи и события, поэтому будем считать, что доступны такие данные:

- пользователи;
- события;
- отношения дружбы;
- взаимодействия.

Пользователи

Ниже изображена простая схема данных о пользователе.

Таблица 7.1. Схема данных о пользователе

ID пользователя	Имя пользователя	Возраст	Пол	Город	Страна	Язык	Часовой пояс
-----------------	------------------	---------	-----	-------	--------	------	--------------

События

В табл. 7.2 показано, как могут выглядеть данные о событии.

Таблица 7.2. Данные о событии

ID события	ID организатора	Категория	Подкатегория	Описание	Цена	Местонахождение	Дата и время
1	5	Музыка	Концерт	Тур Dua Lipa в Майами	200–900	American Airlines Arena, Майами, Флорида	18.09.2022 19:00–24:00
2	11	Спорт	Баскетбол	Матч Golden State Warriors — Milwaukee Bucks	140–2500	Chase Center, Сан-Франциско, Калифорния	22.09.2022 17:00–19:00
3	7	Искусство	Театр	Юмор и магия Роберта Холла	Бесплатно	San Jose Improv, Сан-Хосе, Калифорния	06.09.2022 18:00–19:30

Друзья

В табл. 7.3 каждая строка представляет отношение дружбы между двумя пользователями и момент времени, когда это отношение было установлено.

Таблица 7.3. Данные о друзьях

ID пользователя 1	ID пользователя 2	Временная метка
28	3	1658451341
7	39	1659281720
11	25	1659312942

Взаимодействия

В табл. 7.4 хранятся данные о взаимодействиях пользователей: записи на события, приглашения и показы. На практике эта информация может храниться в разных базах данных, но для простоты мы объединим ее в одну таблицу.

Таблица 7.4. Данные о взаимодействиях

ID пользо-ватель	ID собы-тия	Тип взаимодей-ствия	Значение взаи-модействия	Геопозиция (широта, долгота)	Временнаá метка
4	18	Показ	—	38.8951 -77.0364	1658450539
4	18	Запись	Код подтверж-дения	38.8951 -77.0364	1658451341
4	18	Приглашение	Пользователь 9	41.9241 -89.0389	1658451365

Конструирование признаков

Рекомендации для событий создают больше проблем, чем другие рекомендации. Событие принципиально отличается от книги или фильма, потому что оно становится недоступным после того, как завершилось. Как правило, события существуют недолго — то есть между созданием события и его завершением проходит относительно немного времени. В результате в истории события накапливается довольно мало взаимодействий. Поэтому рекомендации событий по своей природе страдают от проблемы «холодного старта» и характеризуются постоянным притоком новых элементов.

Чтобы справиться с этими проблемами, мы будем тщательнее конструировать признаки, пытаясь создать как можно больше осмысленных признаков. Для экономии места мы обсудим только некоторые важнейшие признаки, хотя на практике предсказательных признаков может быть намного больше.

В этом разделе мы создадим признаки, относящиеся к каждой из следующих категорий:

- местонахождение;
- время;
- социальный фактор;
- пользователь;
- событие.

Признаки местонахождения

Легко ли добраться до места, где проводится событие?

Доступность места проведения события — это важный фактор. Например, если событие устраивается где-то в горах вдали от дорожной сети, далеко не

все будут готовы в нем участвовать. Создадим следующие признаки, которые характеризуют доступность.

- **Уровень пешей доступности.** Число от 0 до 100, которое оценивает, насколько удобно добираться до места пешком в зависимости от расстояния до инфраструктурных объектов. Чтобы вычислить это значение, анализируются различные факторы — такие, как расстояние до ближайших инфраструктурных объектов, наличие пешеходных маршрутов, плотность населения и т. д. Предполагается, что уровень пешей доступности можно получить из внешних источников — таких, как Google Maps, OpenStreetMap и т. д. В табл. 7.5 приведены уровни пешей доступности, разделенные на пять категорий.

Таблица 7.5. Категории уровней пешей доступности

Категория	Уровень пешей доступности	Описание
1	90–100	Транспорт не требуется
2	70–89	Просто добраться пешком
3	50–69	Возможно добраться пешком
4	25–49	Желателен транспорт
5	0–24	Необходим транспорт

- **Сходство уровней пешей доступности.** Насколько уровень пешей доступности события близок к среднему уровню пешей доступности событий, на которые пользователь записывался до этого.
- Уровень транспортной доступности, сходство уровней транспортной доступности, уровень велосипедной доступности, сходство уровней велосипедной доступности.

Насколько далеко находится место проведения события от места, где живет пользователь?

Решение пользователя о том, участвовать ли в событии, во многом зависит от того, проводится ли оно в той же стране и городе, где он живет. Чтобы это учесть, можно создать два признака: **совпадение страны** и **совпадение города**. Если страна (город), где живет пользователь, совпадает со страной (городом) проведения события, то соответствующий признак равен 1; в противном случае он равен 0.

Устраивает ли расстояние пользователя?

Некоторые пользователи предпочитают события вблизи от своего дома, тогда как другие любят более удаленные. Чтобы отразить эти предпочтения, будем использовать следующие признаки.

- **Расстояние от дома пользователя до места проведения события.** Это значение можно получить из внешних API и разделить на несколько категорий, например:
 - 0: меньше мили;
 - 1: 1–5 миль;
 - 2: 5–20 миль;
 - 3: 20–50 миль;
 - 4: 50–100 миль;
 - 5: больше 100 миль.
- **Сходство расстояний.** Насколько расстояние до места проведения события близко к среднему расстоянию до мест проведения событий, на которые пользователь записывался ранее. (На практике в качестве среднего значения можно использовать медианное расстояние или процентильный диапазон.)

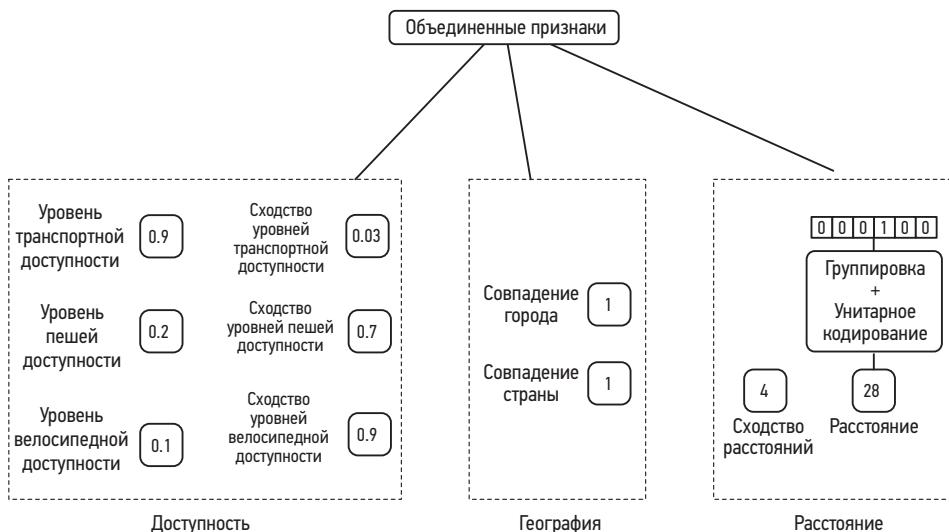


Рис. 7.7. Признаки местонахождения

Признаки времени

Насколько удобно время, оставшееся до события?

Некоторые пользователи планируют посещение событий за несколько дней, а другие этого не делают. Чтобы отразить эти предпочтения, создадим следующие признаки.

- **Время, оставшееся до начала события.** Этот признак можно разделить на несколько категорий и закодировать унитарным кодированием, например:
 - 0: до начала события осталось менее 1 часа;
 - 1: 1–2 часа;
 - 2: 2–4 часа;
 - 3: 4–6 часов;
 - 4: 6–12 часов;
 - 5: 12–24 часа;
 - 6: 1–3 дня;
 - 7: 3–7 дней;
 - 8: более 7 дней.
- **Сходство оставшегося времени.** Насколько оставшееся время близко к среднему оставшемуся времени для событий, на которые пользователь записывался ранее.
- **Оценочное время на дорогу** от местоположения пользователя к месту проведения события. Это значение можно получить от внешних сервисов и разделить на категории.
- **Сходство оценочного времени на дорогу.** Насколько оценочное время на дорогу к месту проведения события близко к среднему оценочному времени на дорогу для событий, на которые пользователь записывался ранее.

Насколько дата и время удобны для пользователя?

Одни пользователи предпочитают события по выходным, а другие — по будним дням. Одним удобнее утренние события, другим — вечерние. Чтобы отразить историю предпочтений пользователя по дням недели, мы создадим профиль пользователя — вектор длины 7, каждый элемент которого соответствует определенному дню недели. Значение элемента — количество событий, которые пользователь посетил в соответствующий день недели, деленное на общее количество посещенных событий, то есть историческая частота посещения событий по дням недели. На рис. 7.8 изображено соответствующее распределение. Как видно из диаграммы, этот пользователь никогда не посещал события по понедельникам и средам, поэтому, возможно, ему не стоит рекомендовать события, назначенные на среду. Аналогичным образом можно построить почасовые профили пользователей, а также добавить метрики сходства по дням и часам.

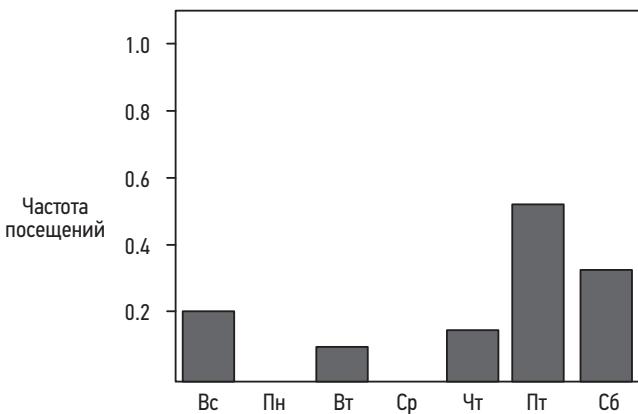


Рис. 7.8. Распределение посещаемости событий по дням недели

Сводка признаков, относящихся ко времени, приведена на рис. 7.9.

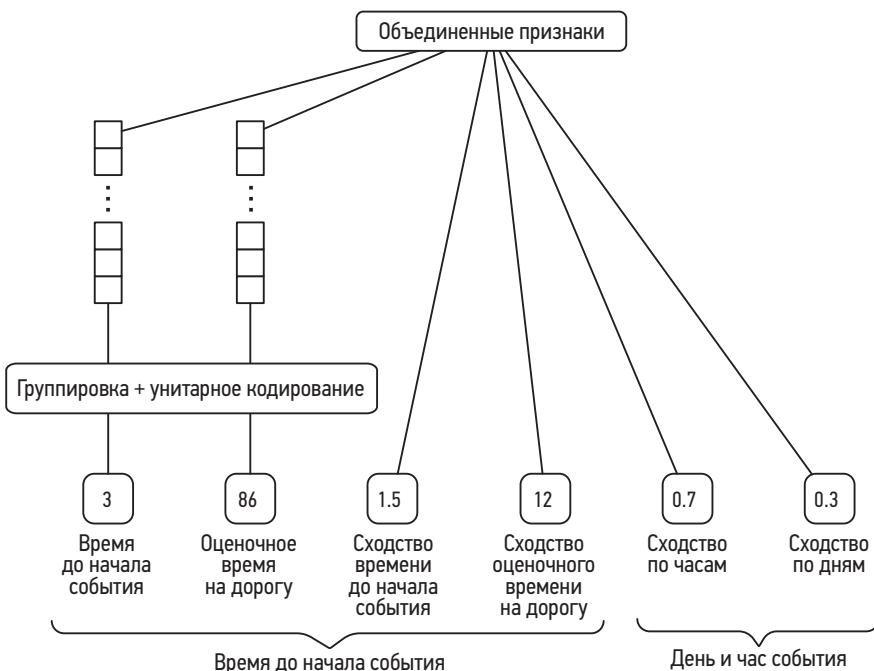


Рис. 7.9. Признаки, относящиеся ко времени

Признаки, относящиеся к социальному фактору

Сколько посетителей ожидается на событии?

В общем случае пользователи более склонны записываться на событие, если на него записалось много других посетителей. Чтобы отразить эти предпочтения, создадим следующие признаки:

- количество пользователей, которые записались на событие;
- отношение общего количества записавшихся пользователей к количеству показов;
- сходство по записи. Насколько количество пользователей, записавшихся на данное событие, близко к среднему количеству пользователей, которые ранее записывались на те же события, что и данный пользователь.

События, которые посещают друзья

Пользователь с большей вероятностью запишется на событие, если это событие собираются посетить его друзья. Примеры признаков, которые можно использовать в этом контексте:

- количество друзей пользователя, записавшихся на событие;
- отношение количества записавшихся друзей к общему количеству друзей;
- сходство по записавшимся друзьям. Насколько количество друзей, записавшихся на данное событие, близко к этому показателю для предыдущих событий, на которые ранее записывался данный пользователь.

Был ли пользователь приглашен на событие другими пользователями?

Пользователи охотнее посещают события, на которые их пригласили. Примеры полезных признаков — количество друзей или количество других пользователей, пригласивших данного пользователя на событие.

Является ли организатор события другом пользователя?

Пользователи склонны посещать события, которые организовывают их друзья. Чтобы отразить эту особенность, мы создадим бинарный признак: если организатор события — друг пользователя, то значение признака равно 1, а в противном случае оно равно 0.

Часто ли пользователь посещал предыдущие события этого организатора?

Некоторые пользователи склонны следить за событиями конкретных организаторов.

Признаки пользователя

Возраст и пол

Некоторые события ориентированы на тот или иной возраст и/или пол. Например, события «Женщины в технической отрасли» и «Уроки жизни: как преуспеть, когда вам за 30» — примеры событий, которые нацелены на определенные демографические группы. Чтобы отразить эту особенность, создадим два признака, закодированных с помощью унитарного кодирования:

- пол пользователя;
- возраст пользователя, разделенный на категории.

Признаки события

Стоимость участия

Расходы на участие в событии могут повлиять на решение пользователя о том, записываться ли на него. Вот некоторые возможные признаки:

- Стоимость участия, разделенная на категории, например:
 - 0: бесплатно;
 - 1: 1–99 долл.;
 - 2: 100–499 долл.;
 - 3: 500–1999 долл.;
 - 4: 2000 долл. и более.
- Сходство по стоимости. Насколько стоимость участия в данном событии близка к средней стоимости участия в событиях, на которые ранее записывался пользователь.

Насколько похоже описание события на описания прежних событий, на которые записывался пользователь?

Этот признак характеризует интересы пользователя на основании событий, на которые он записывался ранее. Например, если в описаниях предыдущих событий часто встречается слово «концерт», это может указывать на то, что пользователя интересуют концерты. Чтобы отразить это, создадим признак, который представляет сходство между описанием данного события и описаниями прежних событий, на которые записывался пользователь. Чтобы вычислить сходство, описание преобразуется в числовую вектор методом TF-ID и рассчитывается косинусное расстояние.

Обратите внимание, что этот признак может быть зашумленным, потому что организаторы составляют описания вручную. Чтобы оценить важность этого признака, можно поэкспериментировать, обучая модель с ним и без него.

На рис. 7.10 представлена диаграмма признаков, которые относятся к пользователям и событиям, а также к записям и приглашениям.

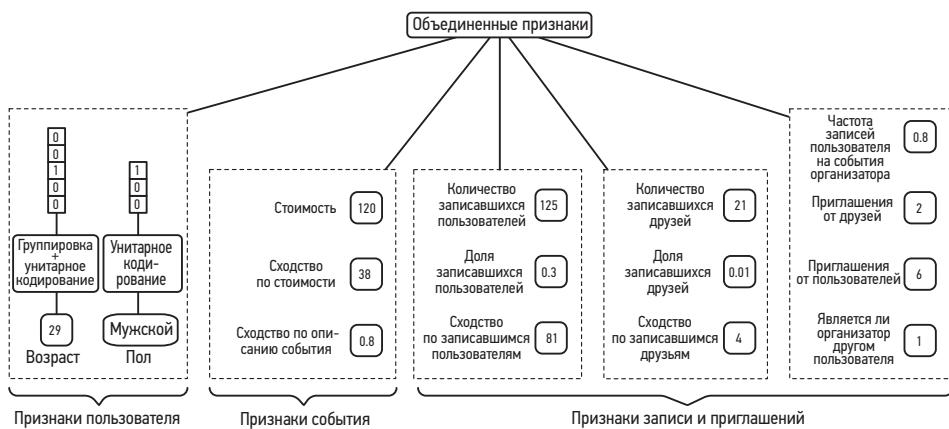


Рис. 7.10. Признаки, относящиеся к пользователям, событиям, записям и приглашениям

Это не исчерпывающий список. На практике можно конструировать много других предсказательных признаков — например, популярность организатора, историю поиска пользователя, категорию события, автоматически сгенерированные теги события и т. д. На собеседовании необязательно строго следовать структуре этого раздела: можно выбрать ее как отправную точку, а затем переключиться на темы, которые более актуальны для эксперта. Вот несколько вопросов, которые потенциально заслуживают особого внимания.

- **Статические и динамические признаки.** К статическим признакам относятся те, которые изменяются относительно редко, — например, возраст, пол и описание события. Эти признаки можно периодически вычислять в пакетном режиме и сохранять в хранилище признаков. С другой стороны, динамические признаки быстро изменяются: например, к этой категории относится количество пользователей, которые записались на событие, и время, оставшееся до события. Возможно, эксперт захочет, чтобы вы глубже осветили эту тему и обсудили различия между пакетной и оперативной обработкой в МО. За дополнительной информацией обращайтесь к [8].
- **Эффективное вычисление признаков.** Вычислять признаки в реальном времени неэффективно. Возможно, имеет смысл поговорить об этой проблеме и о том, как ее избежать. Например, вместо того чтобы вычислять расстояние между текущим местонахождением пользователя и местом проведения события как отдельный признак, можно передать модели обе геопозиции в двух разных признаках и положиться на то, что модель неявно вычислит полезную информацию из этих данных. Если вы захотите больше узнать о подготовке географических данных для моделей МО, обращайтесь к [9].
- **Коэффициент затухания** для признаков, зависящих от последних X взаимодействий пользователя. Этот коэффициент назначает больший вес более свежим взаимодействиям.

- **Обучение с эмбеддингами** для преобразования каждого события и пользователя в вектор эмбеддинга. Эти векторы используются как признаки, представляющие события и пользователей.
- **Конструирование признаков на основе свойств пользователя может создавать смещение.** Например, если в модели пригодность соискателя для работы зависит от его возраста или пола, это может привести к дискриминации.

Разработка модели

Выбор модели

Задачи бинарной классификации можно решать различными методами МО. Вот некоторые примеры:

- логистическая регрессия;
- дерево решений;
- дерево решений с градиентным бустингом (GBDT);
- нейронная сеть.

Логистическая регрессия (LR)

Логистическая регрессия моделирует вероятность бинарного исхода с помощью линейной комбинации одного или нескольких признаков. За дополнительной информацией о LR обращайтесь к [10].

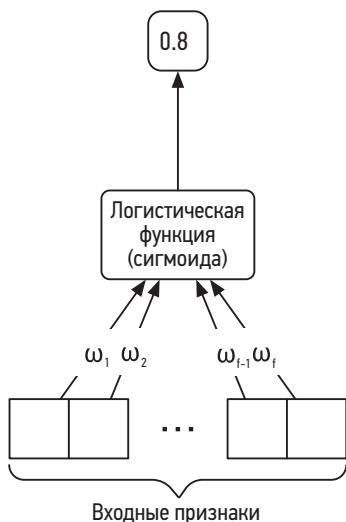


Рис. 7.11. Логистическая регрессия

Рассмотрим достоинства и недостатки логистической регрессии.

Достоинства

- **Высокая скорость вычисления.** Взвешенная комбинация входных признаков вычисляется быстро.
- **Эффективное обучение.** Модель с простой архитектурой легче реализовать и интерпретировать, и она быстрее обучается.
- Хорошо работает с линейно разделимыми данными (рис. 7.12).
- **Просто интерпретировать и понимать.** Веса, присвоенные каждому признаку, обозначают относительную важность разных признаков, а это помогает понять, почему было принято то или иное решение.

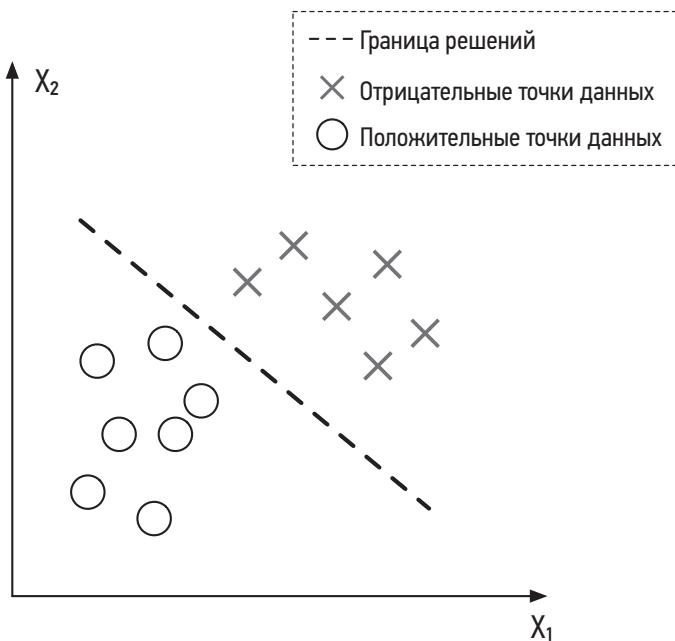


Рис. 7.12. Линейно разделимые данные с границей решений LR

Недостатки

- С помощью LR нельзя решать **нелинейные задачи**, потому что этот метод использует линейную комбинацию входных признаков.
- При высокой корреляции двух и более признаков возникает **мультиколлинеарность**. Одно из известных ограничений LR состоит в том, что модель плохо обучается при мультиколлинеарности во входных признаках.

В нашей системе может быть очень много входных признаков. Часто эти признаки связаны сложными нелинейными отношениями с целевой переменной (бинарным результатом). Модели логистической регрессии могут не справиться с подобной сложностью.

Дерево решений

Деревья решений — еще один класс методов обучения, которые используют для предсказаний древовидную модель решений и их возможных последствий. На рис. 7.13 показано простое дерево решений с двумя признаками: возрастом и полом, а также соответствующая граница решений. Каждый листовой узел дерева соответствует бинарному результату: «+» означает, что заданное входное значение классифицируется как положительное, а «-» — как отрицательное. За дополнительной информацией о деревьях решений обращайтесь к [11].

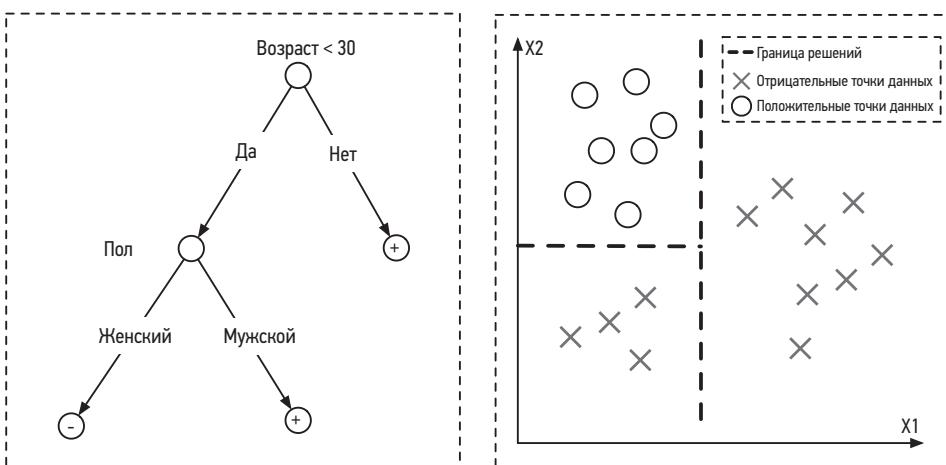


Рис. 7.13. Дерево решений (слева) и граница решений, полученная в результате обучения (справа)

Достоинства

- Деревья решений **быстро обучаются**.
- **Быстрый результат.** Деревья решений выдают предсказания.
- **Мало затрат на подготовку данных.** Для деревьев решений не нужно нормализовывать или масштабировать данные, потому что алгоритм не зависит от распределения входных признаков.
- **Интерпретируемость и простота понимания.** Визуализация дерева показывает, почему было принято решение и какие факторы оказались наиболее важными.

Недостатки

- **Неоптимальная граница решений.** Деревья решений создают границы решений, параллельные осям в пространстве признаков (см. рис. 7.13). Для некоторых распределений данных такой способ задания границы может быть неоптимальным.
- **Переобучение.** Деревья решений очень чувствительны: небольшое изменение во входных данных может привести к тому, что на этапе эксплуатации модель будет вырабатывать другие результаты. Аналогично незначительные изменения в обучающих данных могут привести к совершенно иной структуре дерева. Это серьезная проблема, из-за которой предсказания получаются менее надежными.

На практике наивные деревья решений применяются редко, потому что они слишком чувствительны к вариациям входных данных. Чтобы снизить эту чувствительность, обычно применяются два метода:

- бэггинг (от bootstrap aggregating, «бутстреп-агрегирование»);
- бустинг.

Эти методы широко применяются, поэтому важно понимать, как они работают. Рассмотрим подробнее.

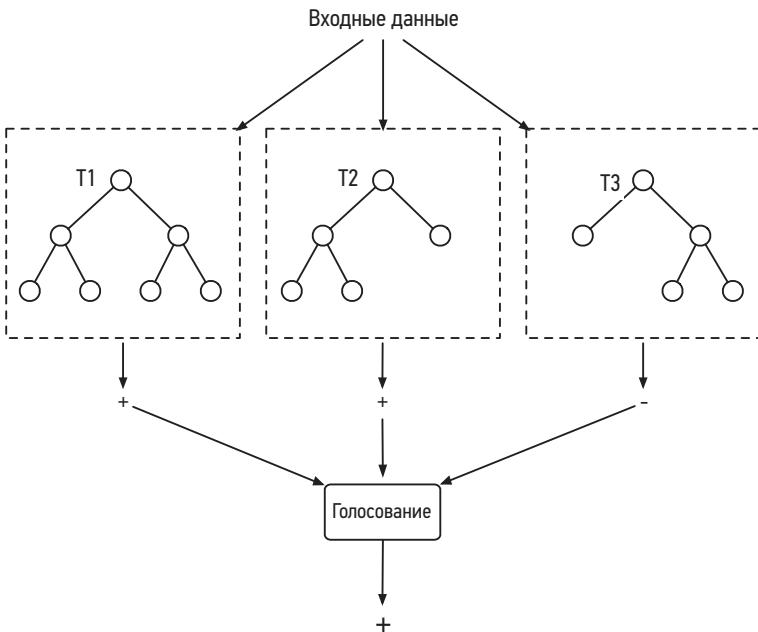
Бэггинг

Бэггинг — ансамблевый метод обучения, при котором несколько моделей МО обучаются параллельно на разных подмножествах обучающих данных. Чтобы получить итоговое предсказание, предсказания всех этих обученных моделей объединяются. Тем самым модель становится гораздо менее чувствительной к вариациям данных.

Один из примеров бэггинга — распространенная модель случайного леса [12]. В процессе обучения случайный лес параллельно строит несколько деревьев решений. Каждое дерево независимо предсказывает выходной класс (положительный или отрицательный) для заданных входных данных, после чего механизм голосования объединяет эти результаты в итоговое предсказание. На рис. 7.14 показан случайный лес с тремя деревьями решений.

Основные **преимущества** бэггинга:

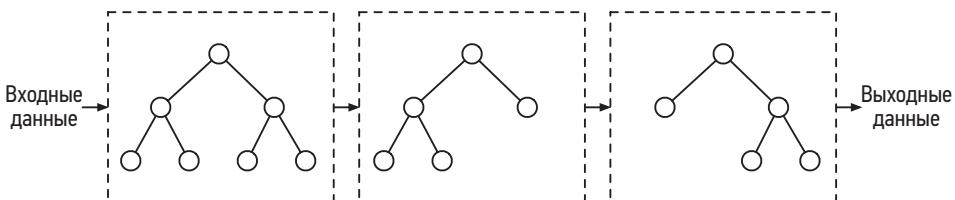
- снижается эффект переобучения (разброс);
- время обучения существенно не увеличивается, потому что деревья решений могут обучаться параллельно;
- во время формирования результата нет значительной задержки, потому что деревья решений могут обрабатывать входные данные параллельно.

**Рис. 7.14.** Случайный лес

При всех достоинствах бэггинг оказывается бесполезным, когда модель сталкивается с недообучением (высоким смещением). Справиться с недостатками бэггинга помогает другой метод, который называется бустингом.

Бустинг

В машинном обучении бустинг заключается в том, что несколько слабых классификаторов обучаются последовательно, чтобы снизить ошибки предсказаний. Слабый классификатор — это простой классификатор, который работает лишь ненамного эффективнее случайных предположений. При бустинге несколько слабых классификаторов преобразуются в одну сильную модель обучения. Пример бустинга показан на рис. 7.15.

**Рис. 7.15.** Пример бустинга

Достоинства

- Бустинг снижает смещение и разброс. Объединяя слабые классификаторы, можно получить сильную модель, которая менее чувствительна к изменениям в данных. Чтобы больше узнать о балансе смещения и разброса, обращайтесь к [13].

Недостатки

- Медленное обучение и выработка результатов. Поскольку классификаторы обучаются на ошибках предыдущих классификаторов, они работают последовательно, а значит, время обслуживания увеличивается.

На практике бустинг обычно предпочтительнее бэггинга, потому что бэггинг не помогает против смещения, а бустинг ослабляет эффект как смещения, так и разброса.

Типичные деревья решений на базе бустинга — AdaBoost [14], XGBoost [15] и GradientBoost [16]. Они часто применяются при обучении классификационных моделей.

Дерево решений с градиентным бустингом (GBDT)

GBDT — часто применяемая модель на основе деревьев решений, которые улучшаются с помощью градиентного бустинга. Некоторые разновидности GBDT, такие как XGBoost [15], продемонстрировали значительную эффективность в различных соревнованиях по МО [17]. Если вы захотите больше узнать о GBDT, обращайтесь к [18], [19].

Этот пользователь запишется на событие

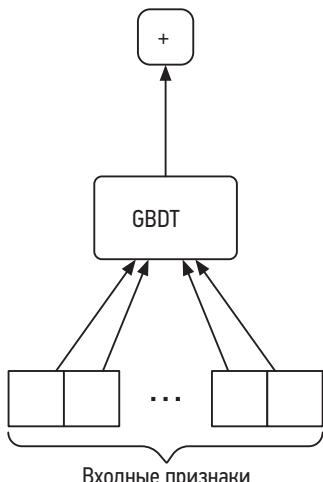


Рис. 7.16. Модель GBDT с бинарным выводом

Ниже перечислены достоинства и недостатки модели GBDT.

Достоинства

- **Простота подготовки данных.** Как и деревья решений, модель не требует специальной подготовки данных.
- **Снижается разброс.** GBDT снижает разброс благодаря бустингу.
- **Снижается смещение.** GBDT сокращает ошибку предсказания благодаря тому, что используются несколько слабых классификаторов, каждый из которых итеративно улучшает точки данных, неправильно классифицированные предыдущими классификаторами.
- Хорошо обрабатываются **структурированные данные**.

Недостатки

- **Нужно настраивать многочисленные гиперпараметры:** количество итераций, глубину деревьев, параметры регуляризации и т. д.
- GBDT плохо работает с **неструктуризованными данными** – изображениями, видео, аудио и т. д.
- GBDT **непригоден для непрерывного обучения** на потоковых данных.

Поскольку признаки, которые мы сконструировали в этом примере, являются структуризованными данными, GBDT или одна из его разновидностей – например, XGBoost – хорошо подойдет для экспериментов.

Главный недостаток GBDT состоит в том, что этот метод не годится для непрерывного обучения. В системе рекомендации событий постоянно появляются новые данные: свежие взаимодействия пользователей, записи на события, новые события и даже новые пользователи. Кроме того, вкусы и интересы пользователей тоже могут меняться со временем. Для хорошей системы рекомендаций важно непрерывно адаптироваться к новым данным. Без непрерывного обучения придется время от времени заново обучать GBDT с нуля, а это очень затратно. В следующем разделе рассматриваются нейронные сети, которые преодолевают это ограничение.

Нейронная сеть (NN)

В системах рекомендации событий обычно есть много признаков, которые необязательно линейно связаны с результатом, и нелегко обучать модель на этих сложных отношениях. Кроме того, непрерывное обучение необходимо, чтобы адаптировать модель к новым данным.

Нейронные сети хорошо справляются с этими проблемами. Они способны изучать сложные задачи с нелинейными границами решений. Кроме того, модели нейронных сетей легко регулируются для новых данных, благодаря чему они идеально подходят для непрерывного обучения.

Если вы еще не знакомы с устройством нейронных сетей, обратитесь к [20].

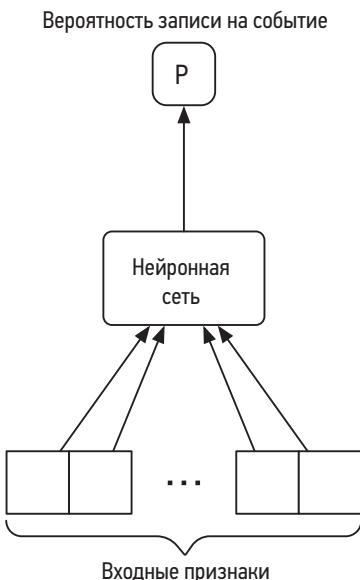


Рис. 7.17. Входные и выходные данные нейронной сети

Рассмотрим достоинства и недостатки нейронных сетей.

Достоинства

- **Непрерывное обучение.** Нейронные сети рассчитаны на то, чтобы непрерывно обучаться на данных и совершенствоваться.
- **Эффективная работа с неструктурированными данными:** текстом, изображениями, видео, аудио и т. д.
- **Функциональная выразительность.** Нейронные сети обладают большой выразительной силой благодаря большому количеству параметров обучения. Они могут изучать очень сложные задачи и нелинейные границы решений.

Недостатки

- **Вычислительно-затратное обучение.**
- **Качество входных данных сильно влияет на результат.** Нейронные сети чувствительны к входным данным. Например, если входные признаки лежат в очень разных диапазонах, модель может медленно сходить на этапе обучения. Важным шагом для нейронных сетей является подготовка данных: нормализация, логарифмирование, унитарное кодирование и т. д.

- Для обучения нейронной сети нужно **много обучающих данных**.
- **Эффект «черного ящика».** Нейронные сети не интерпретируемые: трудно понять, как каждый отдельный признак влияет на результат, потому что входные данные проходят через многие уровни нелинейных преобразований.

Какую модель выбрать?

Выбрать подходящую модель достаточно сложно. Часто приходится экспериментировать с разными моделями, чтобы выяснить, какая работает лучше. Модель можно выбирать на основании таких факторов:

- сложность задачи;
- распределение и тип данных;
- требования и ограничения продукта: затраты на обучение, скорость, размер модели и т. д.

В нашем примере и GBDT, и нейронные сети — хорошие кандидаты для экспериментов. Мы начнем с разновидности GBDT — XGBoost, потому что она быстро реализуется и обучается. Результат можно использовать в качестве исходного базового решения.

Получив это решение, мы посмотрим, можно ли построить лучшую модель с нейронными сетями. Нейронные сети могут хорошо подойти по ряду причин.

- В нашей системе доступен большой объем обучающих данных. Пользователи постоянно взаимодействуют с системой: записываются на события, приглашают друзей, публикуют новые события и т. д. С учетом количества пользователей возникает большой объем данных, доступных для обучения.
- Данные могут не быть линейно разделимыми, но нейронные сети способны обучаться на нелинейных данных.
- При проектировании архитектур нейронных сетей нужно регулировать ряд гиперпараметров, включая количество скрытых слоев и нейронов на каждом слое, функцию активации и т. д. Подробности архитектуры нейронных сетей обычно не выходят на первый план на собеседованиях по проектированию систем МО, потому что нет общих правил того, как выбрать корректную архитектуру.

Обучение модели

Построение датасета

Построение обучающих и оценочных датасетов — важнейший шаг разработки модели. Для примера посмотрим, как вычисляются признаки и их метки.

Чтобы построить отдельную точку данных, мы извлекаем из данных о взаимодействиях пару $(\text{пользователь}, \text{событие})$ и вычисляем входные признаки из этой

пары. Затем точка данных помечается меткой 1, если пользователь записался на событие, или 0, если не записался.

№	Выделение признаков для пары (пользователь, событие)								Метка
1		1	0	1	1	0	1		1
2		0	0	0	1	1	0		0

Рис. 7.18. Построение датасета

После того как набор данных построен, может возникнуть проблема несбалансированности классов. Дело в том, что пользователи могут просмотреть десятки и сотни событий, прежде чем записаться на одно из них. В результате пар (пользователь, событие) с отрицательной меткой оказывается значительно больше, чем положительных точек данных. Чтобы бороться с несбалансированностью классов, можно воспользоваться одним из следующих приемов:

- для обучения классификатора использовать фокальную или сбалансированную по классам функцию потерь;
- сузить выборку из мажоритарного класса.

Выбор функции потерь

Поскольку мы имеем дело с моделью бинарной классификации, будем оптимизировать модель нейронной сети с помощью типичной классификационной функции потерь — например, бинарной перекрестной энтропии.

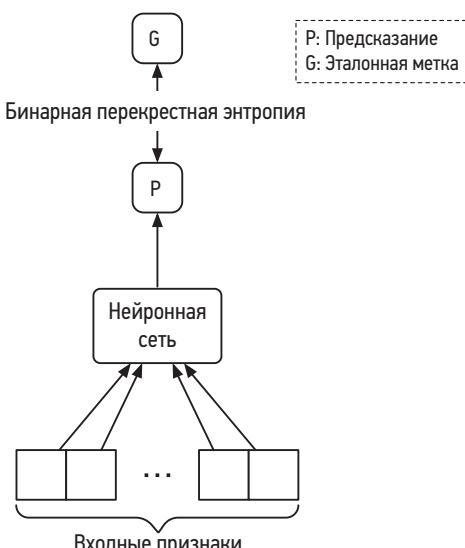


Рис. 7.19. Вычисление потерь между предсказанием и эталонной меткой

Оценка

Автономные метрики

Рассмотрим следующие метрики, с помощью которых можно оценивать систему ранжирования.

Полнота на k или точность на k . Эти метрики не подходят, потому что не учитывают качество ранжирования вывода.

MRR, nDCG или mAP. Эти три метрики часто используются, чтобы оценивать качество ранжирования. Но какая из них лучше?

MRR хорошо помогает выбрать первый релевантный элемент в списке, что годится для систем, которые должны возвращать только один подходящий элемент. Однако в системах рекомендаций событий бывает так, что для пользователя релевантны сразу несколько событий. Для такой ситуации MRR не подойдет.

nDCG хорошо работает в ситуациях, когда показатель релевантности между пользователем и элементом не бинарен.

С другой стороны, mAP подходит только при бинарных показателях релевантности. В нашем случае исход бинарен: пользователь либо записался на событие, либо нет. Поэтому mAP будет наилучшей метрикой.

Оперативные метрики

Наша бизнес-цель — повысить выручку за счет увеличения продаж билетов. Чтобы выяснить, как система влияет на выручку, исследуем следующие метрики:

- кликабельность (CTR);
- конверсия;
- закладочность;
- рост выручки.

Кликабельность (CTR). Показывает, как часто пользователи щелкают по рекомендованным событиям.

$$\text{CTR} = \frac{\text{количество событий, по которым кликнули пользователи}}{\text{общее количество показов}}$$

Чем выше CTR, тем лучше система рекомендует события, которые интересны пользователям. А чем больше кликов, тем обычно активнее пользователи записываются на события.

Однако полагаться только на CTR как на оперативную метрику может быть недостаточно, потому что некоторые объявления о событиях могут оказаться кликбейтными. В идеале нам хотелось бы измерить, насколько рекомендуемые события релевантны для пользователя. Соответствующая метрика, описанная ниже, называется конверсией.

Конверсия показывает, насколько часто пользователи записываются на рекомендованные события. Формула выглядит так:

$$\text{Конверсия} = \frac{\text{количество записей на события}}{\text{общее количество показов}}.$$

Высокая конверсия говорит о том, что пользователи чаще записываются на рекомендованные события. Например, конверсия 0.3 означает, что пользователи в среднем записываются на три события из каждого десяти рекомендованных.

Закладочность показывает, насколько часто пользователи заносят рекомендуемые события в избранные (сохраняют в закладки). Эта метрика актуальна, если платформа поддерживает соответствующую функциональность.

Рост выручки показывает, как увеличились продажи билетов в результате рекомендаций событий.

Эксплуатация

В этом разделе мы представим дизайн системы МО, с помощью которой можно обслуживать запросы. Как показано на рис. 7.20, в дизайне два основных пайплайна:

- пайpline оперативного обучения;
- предсказательный пайpline.

Пайpline оперативного обучения

Как упоминалось ранее, рекомендации событий по своей природе осложняются «холодным стартом» и постоянным притоком новых элементов. В результате приходится постоянно регулировать модель, адаптируя ее к новым данным. Этот пайpline отвечает за непрерывное обучение новых моделей: он интегрирует новые данные, оценивает обученные модели и развертывает их.

Предсказательный пайплайн

Предсказательный пайплайн отвечает за предсказание первых k самых релевантных событий для заданного пользователя. Рассмотрим некоторые важные компоненты этого пайплайна.

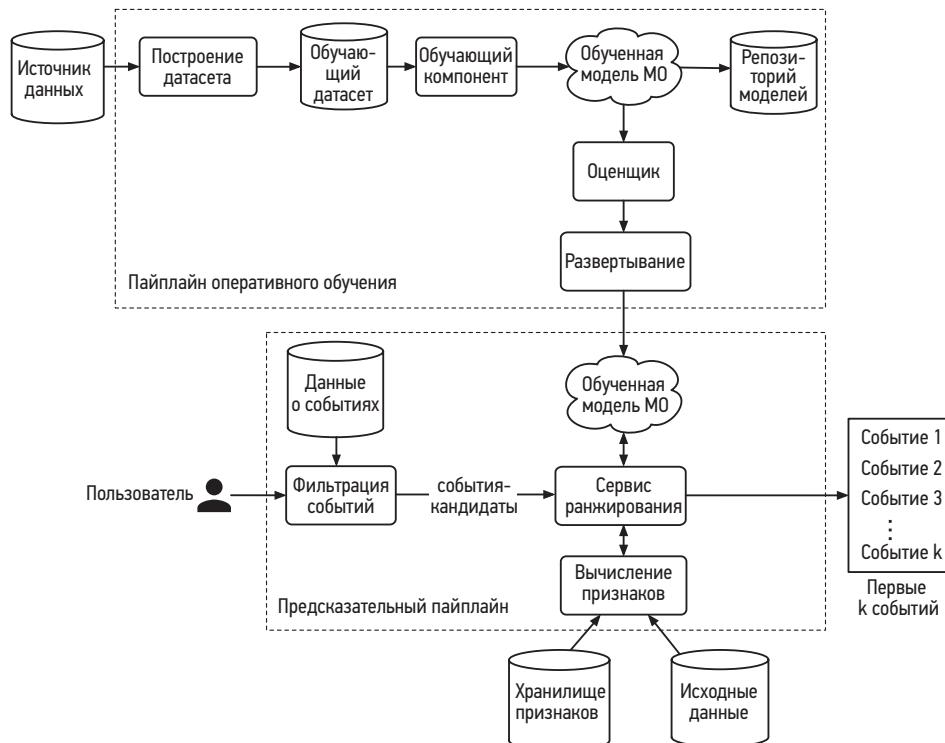


Рис. 7.20. Дизайн системы МО

Фильтрация событий

Компонент фильтрации событий принимает на вход данные о пользователе и сокращает количество событий с 1 миллиона до небольшой подборки. Его работа основана на простых правилах вроде места проведения событий или различных пользовательских фильтров. Например, если пользователь добавил фильтр «только концерты», компонент быстро сузит список до небольшого подмножества событий-кандидатов.

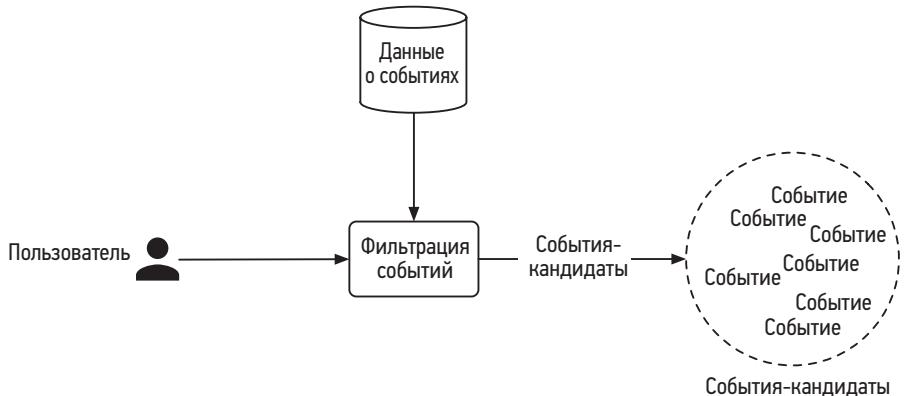


Рис. 7.21. Входные и выходные данные компонента фильтрации событий

Сервис ранжирования

Этот сервис принимает на вход пользователя и события-кандидаты, которые выработал компонент фильтрации, вычисляет признаки для каждой пары {пользователь, событие}, сортирует события на основании вероятностей, предсказанных моделью, и выводит ранжированный список k самых релевантных событий для пользователя.



Рис. 7.22. Схема работы сервиса ранжирования

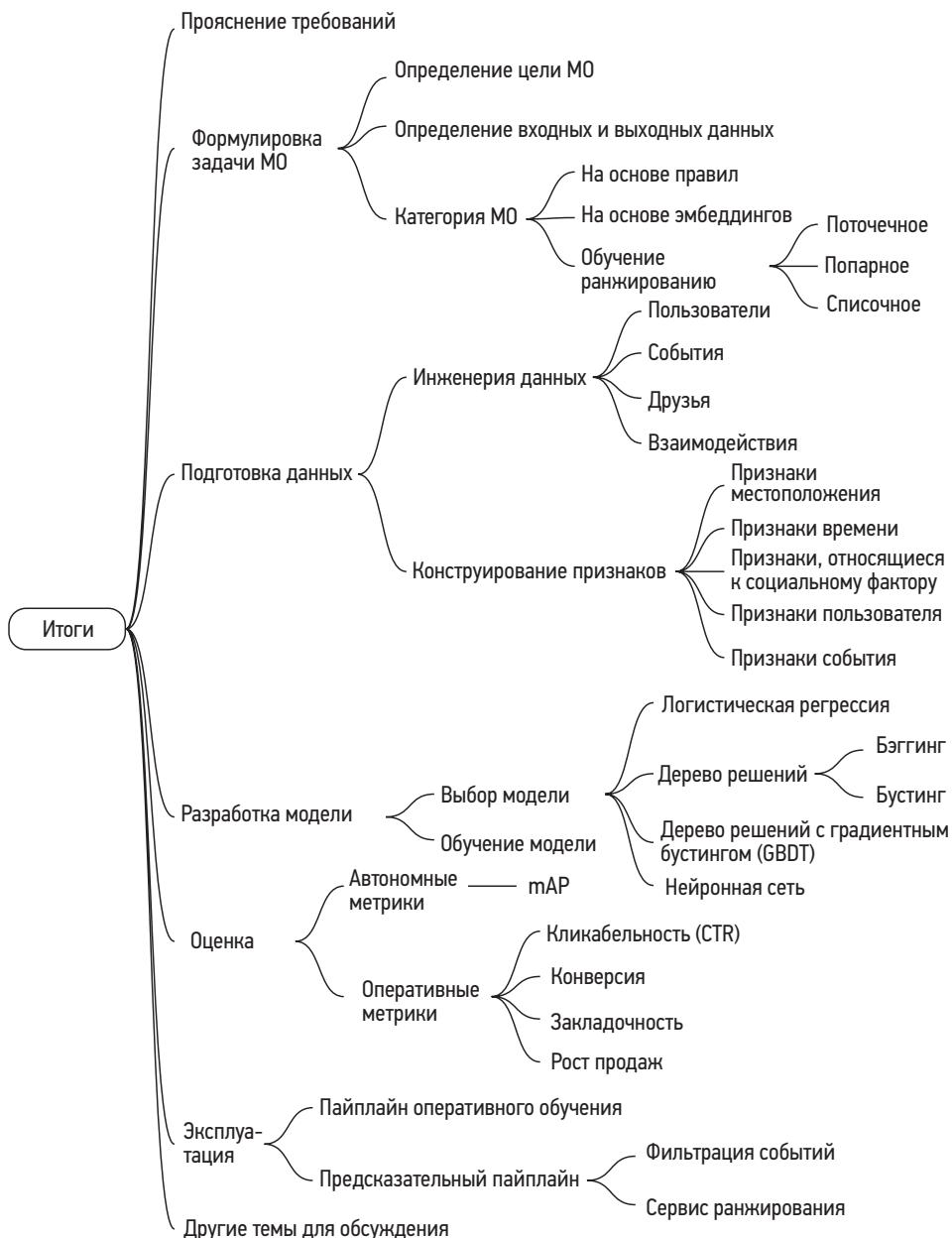
Сервис ранжирования взаимодействует с компонентом, отвечающим за вычисление признаков, которые нужны модели. Статические признаки загружаются из хранилища признаков, а динамические вычисляются в реальном времени по исходным данным.

Другие темы для обсуждения

Если в конце собеседования еще остается время, можно обсудить дополнительные темы.

- Какие типы смещений могут наблюдаться в системе [21].
- Как пересечение признаков помогает усилить функциональную выразительность [22].
- Некоторые пользователи предпочитают более разнообразные списки событий. Как обеспечить, чтобы рекомендуемые события были разнообразными и актуальными [23]?
- Для обучения модели мы задействовали свойства пользователя, а также его геопозицию в реальном времени. Какие соображения конфиденциальности и безопасности стоит учитывать при этом [24]?
- Платформы управления событиями — это обычно двусторонний рынок, где организаторы событий обеспечивают предложение, а пользователи — спрос. Как добиться того, чтобы система не оказалась оптимизирована только для одной стороны? Кроме того, как обеспечить справедливые условия для всех организаторов? Чтобы больше узнать об особенностях разработки двусторонних рынков, обращайтесь к [25].
- Как избежать утечки данных при построении датасета [26]?
- Как определить нужную частоту обновления модели [27]?

Итоги



Ссылки

- [1] Методы обучения ранжированию. <https://livebook.manning.com/book/practical-recommender-systems/chapter-13/53>
- [2] RankNet. https://icml.cc/2015/wp-content/uploads/2015/06/icml_ranking.pdf
- [3] LambdaRank. <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/lambdarank.pdf>
- [4] LambdaMART. <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf>
- [5] SoftRank. <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/SoftRankWsdm08Submitted.pdf>
- [6] ListNet. <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2007-40.pdf>
- [7] AdaRank. <https://dl.acm.org/doi/10.1145/1277741.1277809>
- [8] Пакетная и потоковая обработка. <https://www.confluent.io/learn/batch-vs-real-time-data-processing/>
- [9] Географические данные в системах МО. <https://towardsdatascience.com/leveraging-geolocation-data-for-machine-learning-essential-techniques-192ce3a969bc>
- [10] Логистическая регрессия. <https://www.youtube.com/watch?v=yIYKR4sgzI8>
- [11] Дерево решений. <https://careerfoundry.com/en/blog/data-analytics/what-is-a-decision-tree/>
- [12] Случайные леса. https://en.wikipedia.org/wiki/Random_forest
- [13] Баланс смещения и разброса. http://www.cs.cornell.edu/courses/cs578/2005fa/CS578_bagging.boosting.lecture.pdf
- [14] AdaBoost. <https://en.wikipedia.org/wiki/AdaBoost>
- [15] XGBoost. <https://xgboost.readthedocs.io/en/stable/>
- [16] Градиентный бустинг. <https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>
- [17] XGBoost в конкурсах Kaggle. <https://www.kaggle.com/getting-started/145362>
- [18] GBDT. <https://blog.paperspace.com/gradient-boosting-for-classification/>
- [19] Введение в GBDT. <https://www.machinelearningplus.com/machine-learning/an-introduction-to-gradient-boosting-decision-trees/>
- [20] Введение в нейронные сети. <https://www.youtube.com/watch?v=0twSSFZN9Mc>
- [21] Проблемы смещения и их решение в рекомендательных системах. <https://www.youtube.com/watch?v=pPq9iyGIZZ8>
- [22] Пересечение признаков для кодирования нелинейности. <https://developers.google.com/machine-learning/crash-course/feature-crosses/encoding-nonlinearity>
- [23] Свежесть и разнообразие рекомендаций в рекомендательных системах. <https://developers.google.com/machine-learning/recommendation/dnn/re-ranking>
- [24] Конфиденциальность и безопасность в МО. <https://www.microsoft.com/en-us/research/blog/privacy-preserving-machine-learning-maintaining-confidentiality-and-preserving-trust/>
- [25] Специфика двусторонних рынков. <https://www.uber.com/blog/uber-eats-recommending-marketplace/>
- [26] Утечка данных. <https://machinelearningmastery.com/data-leakage-machine-learning/>
- [27] Частота оперативного обучения. <https://huyenchip.com/2022/01/02/real-time-machine-learning-challenges-and-solutions.html#towards-continual-learning>

8

ПРЕДСКАЗАНИЕ КЛИКОВ ПО РЕКЛАМЕ НА СОЦИАЛЬНЫХ ПЛАТФОРМАХ

Введение

Онлайн-реклама позволяет рекламодателям размещать объявления с помощью специализированных платформ и получать такие объективно измеряемые показатели, как показы, клики и конверсию. Для многих онлайн-платформ, таких как Google, Facebook и Instagram, чрезвычайно важно предоставлять рекламу, релевантную для пользователей.

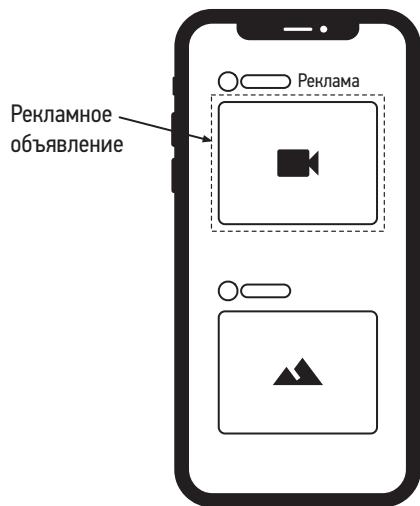


Рис. 8.1. Реклама в ленте пользователя

В этой главе мы спроектируем систему предсказания кликов по рекламе (также называемую системой предсказания CTR), сходную с той, которая используется на популярных платформах социальных сетей.

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: Можно ли считать, что бизнес-цель системы предсказания кликов по рекламе — максимизировать выручку?

Эксперт: Да, верно.

Соискатель: Существуют разные виды рекламы, включая видео и изображения. Кроме того, реклама может отображаться в разных размерах и форматах: в ленте пользователя, во всплывающих окнах и т. д. Можно ли для простоты считать, что реклама размещается только в ленте пользователя, а все клики генерируют одинаковую выручку?

Эксперт: Это можно допустить.

Соискатель: Может ли система показывать пользователю одну и ту же рекламу несколько раз?

Эксперт: Да, многократный показ рекламы возможен. Иногда реклама превращается в клик после нескольких показов. В реальных системах задается «период усталости»: одна и та же реклама не будет показываться одному и тому же пользователю в течение X дней, если он ее игнорирует. Для простоты будем считать, что у нас нет периода усталости.

Соискатель: Будем ли мы поддерживать функцию «Скрыть рекламу»? А как насчет «Блокировать этого рекламодателя»? Негативная обратная связь помогает выявлять нерелевантную рекламу.

Эксперт: Хороший вопрос. Будем считать, что пользователи могут скрыть рекламу, которая им не нравится. Блокировать рекламодателя — интересная функция, но мы пока не будем ее внедрять.

Соискатель: Можно ли предполагать, что обучающий датасет должен строиться на основе данных о пользователях и о рекламных объявлениях, а метки должны отражать взаимодействия пользователей с объявлениями?

Эксперт: Конечно.

Соискатель: На основе кликов можно построить положительные точки данных обучающего датасета, но откуда взять отрицательные? Можно ли считать, что любой показ рекламы, по которой пользователь не кликнул, — это отрицательная точка данных? Что, если пользователь быстро прокручивает страницу и не тратит время на просмотр рекламы? А если мы засчитаем показ как отрицательный, а потом пользователь кликнет по объявлению?

Эксперт: Хорошие вопросы. А вы что думаете?

Соискатель: Если реклама видна на экране пользователя в течение некоторого времени, но пользователь по ней не кликает, ее можно считать отрицательной точкой. Или можно считать все показы отрицательными, пока не произойдет клик. Кроме того, чтобы помечать отрицательные точки, можно положиться на негативную обратную связь — например, нажатие на «Скрыть рекламу».

Эксперт: Логично! На практике для пометки отрицательных точек данных можно пользоваться другими сложными методами [1]. Но в рамках этого собеседования давайте опираться на ваши предложения.

Соискатель: В системе предсказания кликов по рекламе важно, чтобы модель непрерывно обучалась на новых взаимодействиях. Можно ли предположить, что здесь необходимо непрерывное обучение?

Эксперт: Верно замечено. Эксперименты показали, что даже пятиминутная задержка в обновлении моделей может повредить эффективности [1].

Резюмируем описание проблемы. Требуется спроектировать систему предсказания кликов по рекламе. Бизнес-цель системы — максимизировать выручку. Реклама размещается только в ленте пользователя, а все клики генерируют одинаковую выручку. Модель должна непрерывно обучаться на новых взаимодействиях. Датасет строится на основе данных о пользователях и о рекламных объявлениях, после чего размечается на основе взаимодействий. В этой главе мы не будем обсуждать специфические вопросы из области AdTech, потому что они не актуальны для собеседования по машинному обучению. За дополнительной информацией об AdTech обращайтесь к [2].

Формулировка проблемы в виде задачи МО

Определение цели МО

Цель системы предсказания кликов по рекламе — увеличить выручку за счет того, чтобы показывать пользователям рекламу, по которой они с большей вероятностью захотят кликнуть. Это можно сформулировать как следующую цель МО: предсказывать, захочет ли пользователь щелкнуть по рекламе. Это обусловлено тем, что если система будет правильно предсказывать вероятности кликов, то она сможет показывать пользователям релевантную рекламу, а это приведет к повышению выручки.

Определение входных и выходных данных системы

Система предсказания кликов по рекламе принимает на вход данные пользователя и выводит ранжированный список рекламных объявлений, упорядоченный по вероятности клика.



Рис. 8.2. Входные и выходные данные системы предсказания кликов по рекламе

Выбор категории МО

На рис. 8.2 показано, как предсказание кликов по рекламе можно сформулировать в виде задачи ранжирования. Как упоминалось в главе 7, поточечная модель LTR — отличная отправная точка для решения задач ранжирования. В поточечном LTR используется модель бинарной классификации, которая принимает на вход пару \langle пользователь, объявление \rangle и предсказывает, захочет ли пользователь кликнуть по объявлению. На рис. 8.3 представлены входные и выходные данные модели.

Вероятность того, что пользователь кликнет по объявлению

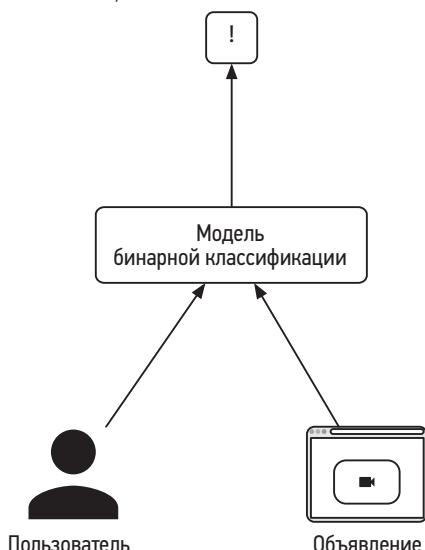


Рис. 8.3. Входные
и выходные данные
модели бинарной
классификации

Подготовка данных

Инженерия данных

В системе доступны такие исходные данные:

- рекламные объявления;
- пользователи;
- взаимодействия пользователей с объявлениями.

Объявления

Данные рекламных объявлений приведены в табл. 8.1. На практике с каждым объявлением могут быть связаны сотни атрибутов. Для простоты мы перечисляем лишь самые важные из них.

Таблица 8.1. Данные рекламных объявлений

ID объявления	ID рекламодателя	ID рекламной группы	ID кампании	Категория	Подкатегория	Изображения или видео
1	1	4	7	путешествия	отель	http://cdn.mysite.com/u1.jpg
2	7	2	9	страхование	автомобили	http://cdn.mysite.com/t3.mp4
3	9	6	28	путешествия	воздушный транспорт	http://cdn.mysite.com/t5.jpg

Пользователи

Схема данных о пользователях представлена в табл. 8.2.

Таблица 8.2. Схема данных о пользователях

ID пользователя	Имя пользователя	Возраст	Пол	Город	Страна	Язык	Часовой пояс
-----------------	------------------	---------	-----	-------	--------	------	--------------

Взаимодействия пользователей с объявлениями

В этой таблице хранятся данные о взаимодействиях пользователей с объявлениями — таких, как показы, клики и конверсия.

Таблица 8.3. Данные взаимодействий пользователей с объявлениями

ID пользователя	ID объявления	Тип взаимодействия	Продолжительность показа ¹	Геопозиция (широта, долгота)	Временная метка
11	6	Показ	5 с	38.8951 –77.0364	165845053
11	7	Показ	0,4 с	41.9241 –89.0389	1658451365
4	20	Клик	–	22.7531 47.9642	1658435948
11	6	Конверсия	–	22.7531 47.9642	1658451849

Конструирование признаков

Цель этого раздела — создать признаки, которые помогут предсказывать пользовательские клики.

Признаки объявлений

К признакам рекламных объявлений относятся:

- идентификаторы (ID);
- изображения и видео;
- категория и подкатегория;
- количество показов и кликов.

Рассмотрим каждый признак подробнее.

Идентификаторы

Идентификатор рекламодателя, идентификатор кампании, идентификатор рекламной группы, идентификатор объявления и т. д.

¹ Общее время, в течение которого объявление остается на экране пользователя.

Почему это важно. Идентификаторы используются как предсказательные признаки, которые отражают уникальные характеристики разных рекламодателей, кампаний, рекламных групп и объявлений.

Как подготовить данные. Слой эмбеддингов преобразует разреженные признаки (например, идентификаторы) в плотные векторы признаков. Для каждого типа идентификаторов предусмотрен отдельный слой эмбеддингов.

Изображения и видео

Почему это важно. Видео или изображение в объявлении — еще один сигнал, который помогает понять, о чем реклама. Например, изображение самолета может указывать на то, что реклама связана с путешествиями.

Как подготовить данные. Сначала изображения или видео проходят предварительную обработку. После этого используется предобученная модель (например, SimCLR [3]), которая преобразовывает неструктурированные данные в вектор признаков.

Категория и подкатегория рекламы

Категория и подкатегория, в которых рекламодатель разместил объявление. Примеры возможных категорий: «Искусство и развлечения», «Автомобили и транспорт», «Красота и здоровье».

Почему это важно. Эти признаки помогают модели понять, к какой категории относится реклама.

Как подготовить данные. Данные вручную предоставляет рекламодатель в соответствии с заранее определенным списком категорий и подкатегорий. За дополнительной информацией о подготовке текстовых данных обращайтесь к главе 4.

Показы и клики

- Общее количество показов и кликов по объявлению.
- Общее количество показов и кликов по всем объявлениям отдельного рекламодателя.
- Общее количество показов для рекламной кампании.

Почему это важно. Эти признаки показывают, как пользователи реагируют на объявление. Например, пользователи с большей вероятностью щелкнут по объявлению с высокой кликабельностью (CTR).



Рис. 8.4. Сводка подготовки признаков, относящихся к объявлениям

Признаки пользователя

По аналогии с предыдущими главами мы выберем следующие признаки:

- демографические признаки: возраст, пол, город, страна и т. д.;
- контекстная информация: устройство, время суток и т. д.;
- признаки, относящиеся к взаимодействиям: объявления, по которым кликали, историческая статистика вовлеченности пользователя и т. д.

Давайте внимательнее рассмотрим признаки взаимодействий.

Объявления, по которым кликали

Почему это важно. Предыдущие клики показывают заинтересованность пользователей. Например, если пользователь часто щелкает по объявлениям на тему страхования, можно предположить, что он с большой вероятностью щелкнет по похожему объявлению снова.

Как подготовить данные. Так же, как было описано в разделе «Признаки объявлений».

Историческая статистика вовлеченности пользователя

Сюда относятся числовые показатели истории вовлеченности пользователя — например, общее количество просмотров объявлений и доля объявлений, по которым он щелкнул.

Почему это важно. Исторические данные вовлеченности — хороший предсказательный признак вовлеченности пользователя в будущем. В общем случае пользователь с большей вероятностью будет щелкать по объявлениям, если он часто щелкал по подобным объявлениям в прошлом.

Как подготовить данные. Статистика вовлеченности представляется числовыми значениями. Чтобы подготовить их, мы масштабируем значения, приведя их к сходным диапазонам.

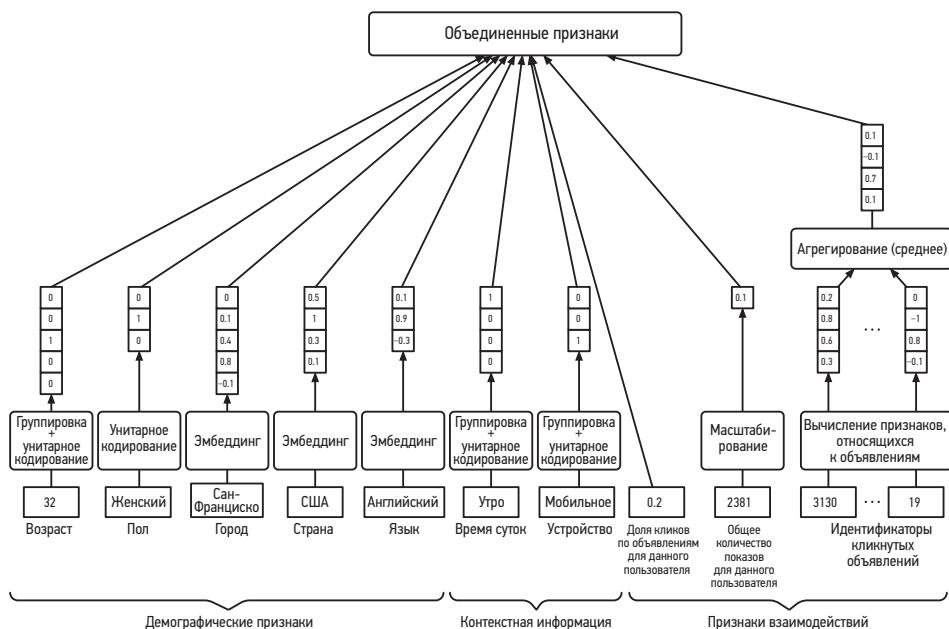


Рис. 8.5. Подготовка признаков для метаданных пользователей и взаимодействий

Прежде чем завершать раздел о подготовке данных, рассмотрим типичную проблему в системах предсказания кликов по рекламе. Чаще всего в таких системах много категориальных признаков с высокой кардинальностью. Например, признак «Категория рекламы» принимает значения из огромного списка всех возможных категорий. Похожим образом признаки «Идентификатор рекламодателя» и «Идентификатор пользователя» потенциально могут принимать миллионы уникальных значений в соответствии с количеством активных пользователей или рекламодателей на платформе. С учетом того, насколько велико пространство признаков, может оказаться, что тысячи и миллионы из них в основном заполнены нулями. В разделе о выборе модели мы посмотрим, как решать эти специфические проблемы.

Разработка модели

Выбор модели

Как упоминалось в разделе «Формулировка проблемы в виде задачи МО», для задачи ранжирования мы выбираем бинарную классификацию. Бывают разные модели бинарной классификации. В системах предсказания кликов по рекламе часто применяются следующие решения:

- логистическая регрессия;
- пересечение признаков + логистическая регрессия;
- деревья решений с градиентным бустингом (GBDT);
- деревья решений с градиентным бустингом + логистическая регрессия;
- нейронная сеть;
- глубокие перекрестные сети (DCN, Deep & Cross Networks);
- факторизационные машины;
- глубокие факторизационные машины.

Логистическая регрессия (LR)

Логистическая регрессия моделирует вероятность бинарного результата на основе линейной комбинации одного или нескольких признаков. LR быстро обучается и легко реализуется. Однако система предсказания кликов по рекламе, основанная на LR, обладает рядом недостатков.

- **LR не подходит для нелинейных проблем.** LR решает задачу на основе линейной комбинации входных признаков, что приводит к линейной границе решений. В системах предсказания кликов по рекламе данные обычно не являются линейно разделимыми, так что LR может показать плохие результаты.
- **Нельзя отразить взаимодействия признаков.** В системах предсказания кликов на рекламе часто встречаются взаимодействия между признаками. Если признаки взаимодействуют друг с другом, то выходную вероятность нельзя выразить как сумму эффектов признаков, потому что эффект одного признака зависит от значения другого признака.

Из-за этих двух недостатков логистическая регрессия — не лучший вариант для системы предсказания кликов по рекламе. Тем не менее, поскольку она быстро реализуется и легко обучается, с ее помощью часто создают базовую модель.

Пересечение признаков + логистическая регрессия

Чтобы эффективнее отразить взаимодействия признаков, используется так называемое пересечение признаков.

Что такое пересечение признаков

Пересечение признаков – метод, с помощью которого в МО создаются новые признаки из существующих. Для этого два или более существующих признака объединяются в новый перекрестный признак, который вычисляется путем умножения, сложения или другой комбинирующей операции. При этом появляется возможность отражать нелинейные взаимодействия между исходными признаками, отчего качество моделей МО может существенно повыситься. Например, такие взаимодействия, как «молодежь и баскетбол» или «США и футбол», могут усиливать способность модели предсказывать вероятность кликов.

Как создавать перекрестные признаки

Для пересечения признаков мы вручную добавляем новые признаки к уже существующим, опираясь на предшествующие знания. Как показано на рис. 8.6, пересечение двух признаков (например, «страна» и «язык») добавляет в существующее пространство признаков шесть новых признаков. Чтобы больше узнать о пересечении признаков, обращайтесь к [4].



Рис. 8.6. Пересечение двух признаков: страны и языка

Как использовать пересечение признаков вместе с логистической регрессией

На рис. 8.7 показано, как пересечение признаков сочетается с логистической регрессией:

1. Пересечение применяется к исходному набору признаков, чтобы выделить новые перекрестные признаки.
2. Исходные и перекрестные признаки используются в качестве входных данных для модели LR.

Этот метод позволяет модели отразить некоторые попарные взаимодействия признаков (взаимодействия второго порядка), однако у него есть четыре недостатка.

- **Ручная работа.** Чтобы выбрать признаки для пересечения, необходимо участие человека, что требует времени и затрат.

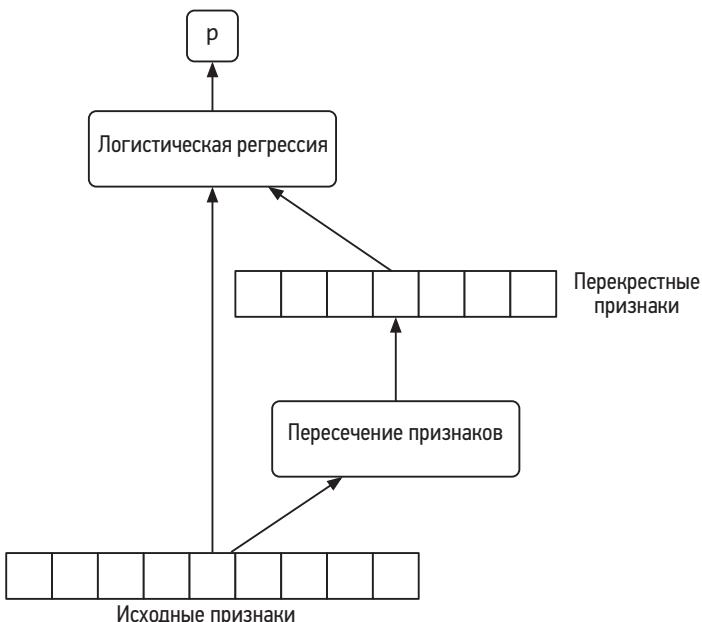


Рис. 8.7. Пересечение признаков в сочетании с логистической регрессией

- **Нужно знать предметную область.** Эффективное пересечение признаков опирается на опыт в предметной области. Чтобы определить, какие взаимодействия между признаками станут предсказательными сигналами для модели, нужно заранее понимать суть проблемы и ориентироваться в пространстве признаков.
- **Не отражаются сложные взаимодействия.** Пересечения признаков может оказаться недостаточно, чтобы отразить все сложные взаимодействия между тысячами разреженных признаков.
- **Разреженность.** Исходные признаки могут быть разреженными. После пересечения кардинальность перекрестных признаков может оказаться очень высокой, что приведет к еще большей разреженности.

Из-за перечисленных недостатков этот метод нельзя назвать идеальным решением для системы предсказания кликов по рекламе.

Деревья решений с градиентным бустингом (GBDT)

Метод GBDT рассматривался в главе 7 «Система рекомендации событий». Здесь мы перечислим его достоинства и недостатки только в применении к системе предсказаний кликов по рекламе.

Достоинства

- GBDT легко понять и интерпретировать.

Недостатки

- **GBDT плохо подходит для непрерывного обучения.** В системах предсказания кликов по рекламе непрерывно собираются новые данные о пользователях, объявлениях и взаимодействиях. Чтобы модель продолжала обучаться на новых данных, в общем случае возможны два варианта: 1) обучение с нуля или 2) регулировка модели на новых данных, на которую метод GBDT не рассчитан. Таким образом, модель обычно приходится обучать с нуля, а в больших масштабах это неэффективно.
- **Нельзя обучать слои эмбеддингов.** В системах предсказания кликов по рекламе часто используется множество разреженных категориальных признаков, которые можно эффективно представить слоем эмбеддингов. Однако GBDT не позволяет извлечь пользу из слоя эмбеддингов.

GBDT + логистическая регрессия

Этот метод состоит из двух шагов.

1. Обучить модель GBDT для текущей задачи.
2. Вместо того чтобы использовать обученную модель для предсказаний, используйте ее для того, чтобы выбирать и выделять новые предсказательные признаки. Свежесозданные признаки вместе с исходными используются как входные данные модели LR для предсказания кликов.

GBDT для выбора признаков

Выбор признаков сокращает набор входных признаков до самых полезных и информативных. С помощью деревьев решений можно отобрать подмножество самых важных признаков. Чтобы лучше понять, как деревья принятия решений помогают генерировать признаки, обращайтесь к [5].

GBDT для выделения признаков

Выделение признаков сокращает их количество, создавая новые признаки из уже существующих так, чтобы выделенные признаки обладали большей предсказательной силой. Рисунок 8.8 поясняет, как выделяются признаки с помощью GBDT.

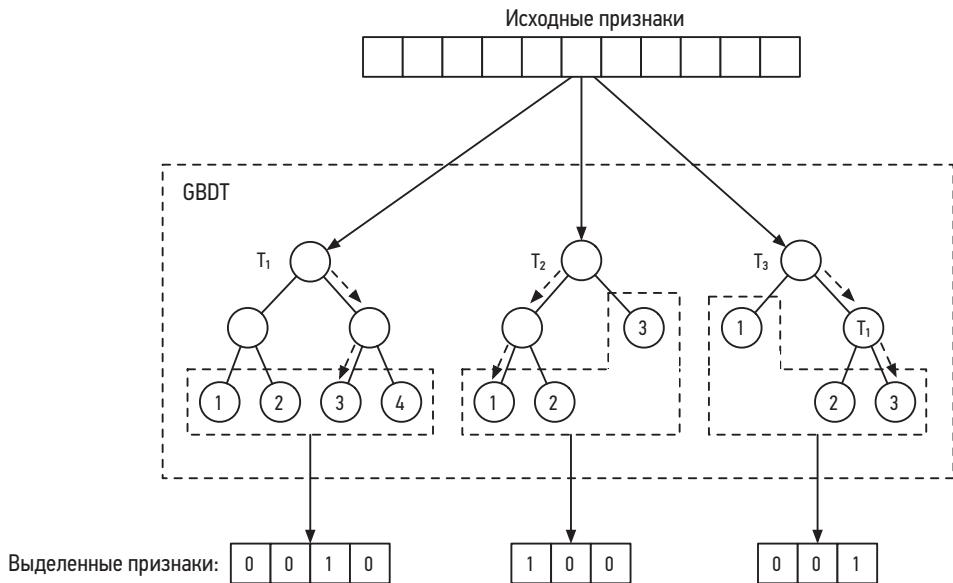


Рис. 8.8. Выделение признаков с помощью GBDT

После GBDT применяется LR, как показано на рис. 8.9.

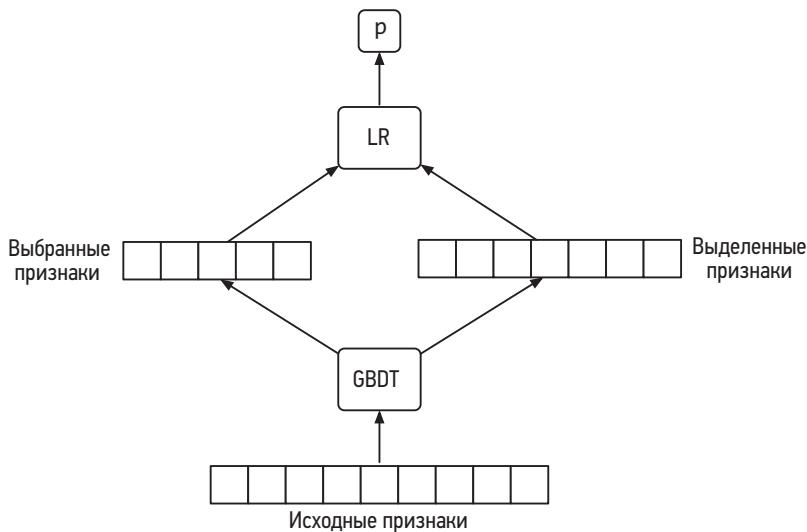


Рис. 8.9. Общая схема GBDT с логистической регрессией

Рассмотрим достоинства и недостатки этого метода.

Достоинства

- В отличие от существующих признаков, новые признаки, создаваемые GBDT, обладают большей предсказательной силой, что упрощает обучение модели LR.

Недостатки

- **Не отражаются сложные взаимодействия.** Как и в случае с LR, этот метод не может обучаться попарным взаимодействиям признаков.
- **Низкая скорость непрерывного обучения.** Регулировка моделей GBDT на новых данных требует времени, что в целом снижает скорость непрерывного обучения.

Нейронная сеть (NN)

Нейронная сеть — еще одна потенциальная основа для системы предсказания кликов по рекламе. Есть два варианта архитектуры нейронной сети, которые позволяют предсказывать вероятность кликов:

- одиночная нейронная сеть;
- двухбашенная архитектура.

Одиночная нейронная сеть. Используя исходные признаки в качестве входных данных, нейронная сеть выводит вероятности кликов (рис. 8.10).

Вероятность того, что пользователь кликнет по объявлению

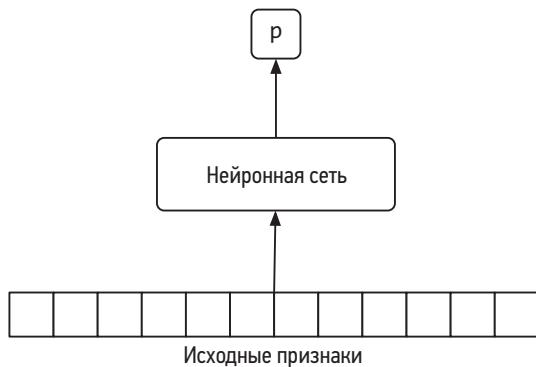


Рис. 8.10. Архитектура одиночной нейронной сети

Двухбашенная архитектура. Здесь используются два кодировщика: кодировщик пользователей и кодировщик объявлений. По степени сходства между эмбеддингами объявления и пользователя определяется релевантность, то есть вероятность клика. На рис. 8.11 представлена общая схема этой архитектуры.

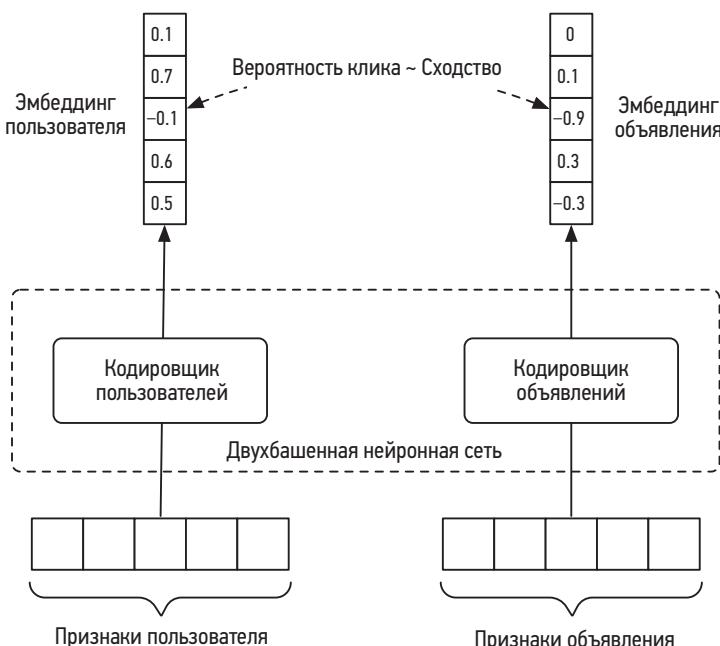


Рис. 8.11. Двухбашенная нейронная сеть на основе эмбеддингов

Несмотря на многочисленные достоинства, нейронные сети оказываются не лучшим вариантом для систем прогнозирования кликов по рекламе по следующим причинам.

- **Разреженность.** Так как пространство признаков обычно оказывается большим и разреженным, большинство признаков заполнено нулями. Иногда нейронная сеть не может эффективно обучиться задаче, потому что в ее распоряжении нет достаточного количества точек данных.
- **Трудно отразить все попарные взаимодействия признаков** из-за их большого количества.

Из-за этих ограничений мы не будем использовать нейронные сети в этой задаче.

Глубокая перекрестная сеть (DCN)

В 2017 году компания Google предложила архитектуру DCN (Deep & Cross Network), которая способна автоматически выявлять взаимодействия признаков, так что вручную выполнять пересечение признаков не требуется. В этом методе используются две параллельные сети.

- **Глубокая сеть** (DNN, Deep Neural Network) изучает сложные и обобщаемые признаки.

- **Перекрестная сеть** автоматически выявляет взаимодействия между признаками и обучается хорошим пересечениям признаков.

Чтобы сформировать итоговое предсказание, результаты глубокой и перекрестной сетей объединяются.

Существуют две разновидности архитектур DCN: многоступенчатая и параллельная. На рис. 8.12 изображена архитектура параллельной DCN.

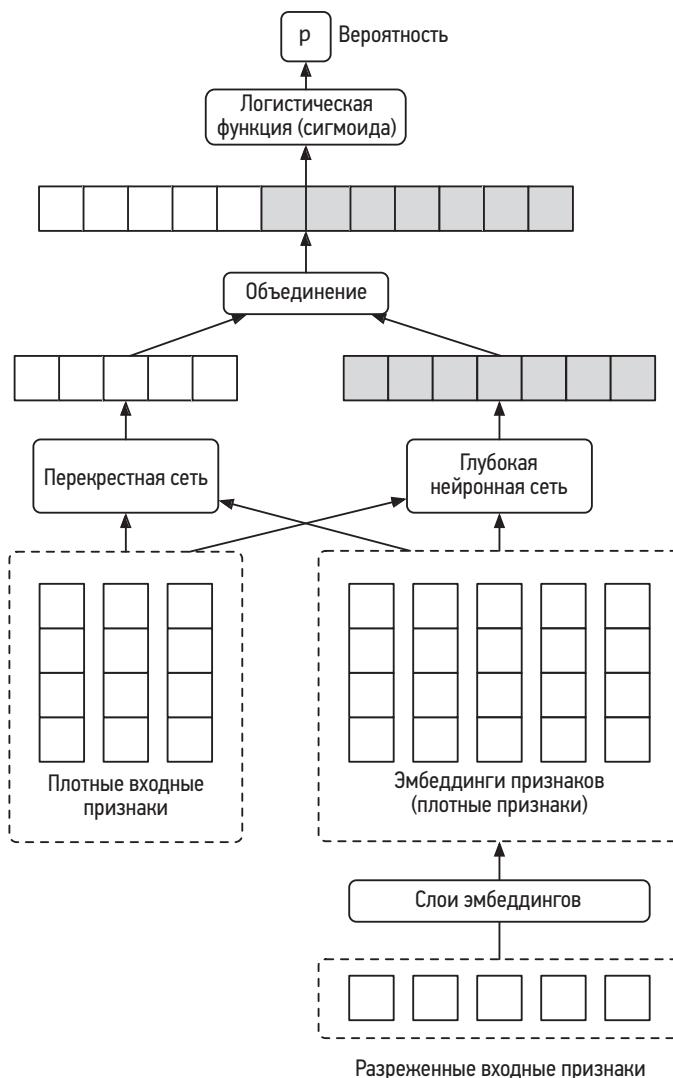


Рис. 8.12. Архитектура DCN

За дополнительной информацией о многоступенчатой архитектуре обращайтесь к [7]. На собеседовании по проектированию систем МО от вас, скорее всего, не потребуют подробно описывать DCN. Если вы захотите больше узнать о глубоких перекрестных сетях, обращайтесь к [7], [8].

Архитектура DCN эффективнее нейронных сетей, потому что она неявно обуивается пересечениям признаков. Однако перекрестная сеть моделирует только некоторые взаимодействия признаков, что может снизить качество модели.

Факторизационные машины (FM)

Факторизационная машина — модель на основе эмбеддингов, которая улучшает логистическую регрессию, автоматически моделируя все попарные взаимодействия признаков. Эта модель широко применяется в системах предсказания кликов по рекламе, потому что она эффективно отражает сложные взаимодействия между признаками.

Как работает факторизационная машина? Она автоматически моделирует все попарные пересечения признаков, изучая вектор эмбеддингов для каждого признака. Взаимодействие между двумя признаками выражается как скалярное произведение их эмбеддингов. Чтобы лучше понять этот принцип, взгляните на формулу:

$$\hat{y}(x) = w_0 + \sum_i w_i x_i + \sum_i \sum_j \langle v_i, v_j \rangle x_i x_j.$$

Здесь x_i — i -й признак, w_i — его вес, полученный в результате обучения, v_i — эмбеддинг для i -го признака, а $\langle v_i, v_j \rangle$ — скалярное произведение двух эмбеддингов.

Формула может показаться сложной, но на самом деле ее легко понять. Первые два слагаемых образуют линейную комбинацию признаков по аналогии с тем, как это происходит в логистической регрессии. Третье слагаемое моделирует попарные взаимодействия признаков. На рис. 8.13 представлена общая схема факторизационной машины. За подробной информацией о FM обращайтесь к [9].

$$\hat{y}(x) = w_0 + \underbrace{\sum_i w_i x_i}_{\text{Логистическая регрессия}} + \underbrace{\sum_i \sum_j \langle v_i, v_j \rangle x_i x_j}_{\text{Попарные взаимодействия}}.$$

Модель факторизационной машины и ее разновидности (такие, как FFM — факторизационная машина с поддержкой полей) эффективно отражают попарные взаимодействия между признаками. В отличие от нейронных сетей, FM не может изучать сложные взаимодействия высших порядков на основе признаков. Чтобы преодолеть этот недостаток, в следующем методе мы объединим FM с DNN.

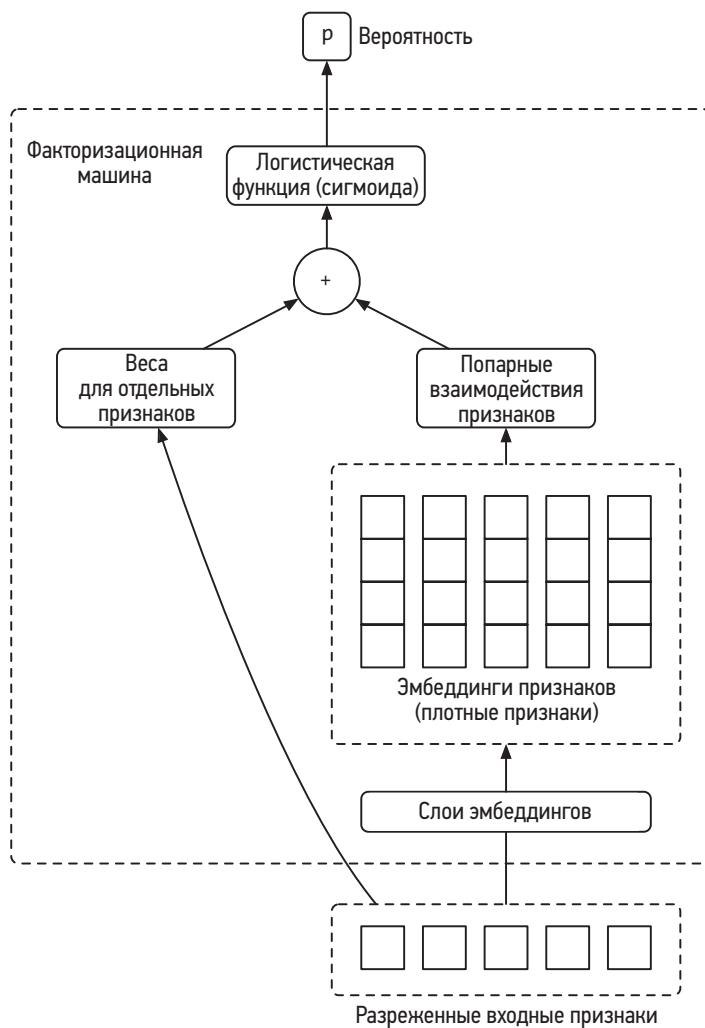


Рис. 8.13. Архитектура факторизационной машины

Глубокие факторизационные машины (DeepFM)

Модель машинного обучения DeepFM объединяет сильные стороны нейронных сетей и факторизационных машин. Сеть DNN выявляет сложные признаки высших порядков, а FM – низкоуровневые попарные взаимодействия признаков. На рис. 8.14 представлена общая архитектура DeepFM. Если вам захочется больше узнать об этой модели, обращайтесь к [10].

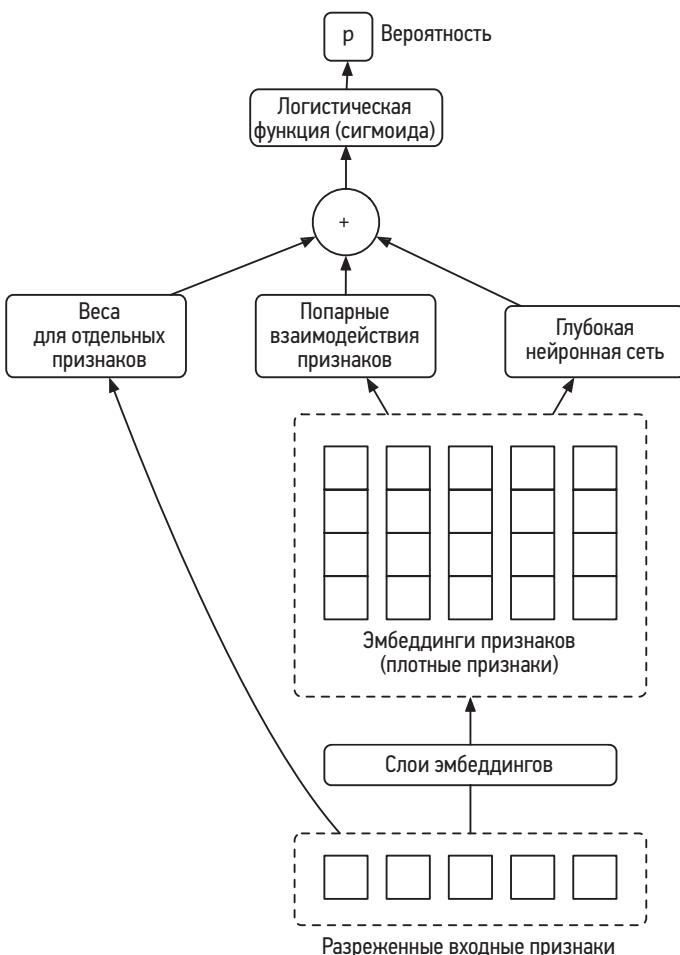


Рис. 8.14. Архитектура глубокой факторизационной машины (DeepFM)

Одно из возможных улучшений заключается в том, чтобы объединить GBDT с DeepFM таким образом, чтобы модель GBDT преобразовывала исходные признаки в признаки с большей предсказательной силой, а DeepFM работала с новыми признаками. Этот метод получал награды на различных соревнованиях по системам предсказания кликов по рекламе [11]. С другой стороны, если добавить GBDT к DeepFM, то скорость обучения и вычислений снижается, а процесс непрерывного обучения замедляется.

На практике наиболее подходящую модель обычно выбирают по результатам экспериментов. В этом примере мы начнем с простой модели LR в качестве ба-

зового решения. Затем мы поэкспериментируем с DCN и DeepFM, поскольку обе модели широко применяются на практике.

Обучение модели

Построение датасета

Для каждого показа объявления будет строиться новая точка данных. Входные признаки вычисляются на основе данных о пользователе и объявлении. Точкам данных присваиваются метки по такому принципу.

- **Положительная метка.** Если пользователь щелкает по объявлению менее чем через t секунд после того, как оно показано, точка данных помечается как положительная. Здесь t — гиперпараметр, который можно регулировать в ходе экспериментов.
- **Отрицательная метка.** Если пользователь не щелкает по объявлению в пределах t секунд, то точка данных помечается как отрицательная.

Чтобы найти оптимальную стратегию разметки отрицательных точек данных, в реальных системах используются более сложные методы. За дополнительной информацией обращайтесь к [1].

Nº	Признаки пользователя и взаимодействий	Признаки объявления	Метка													
1	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>0.8</td><td>0.1</td><td>1</td><td>0</td></tr></table>	1	0	1	0.8	0.1	1	0	<table border="1"><tr><td>0</td><td>1</td><td>1</td><td>0.4</td><td>0.9</td><td>0</td></tr></table>	0	1	1	0.4	0.9	0	Положительная
1	0	1	0.8	0.1	1	0										
0	1	1	0.4	0.9	0											
2	<table border="1"><tr><td>1</td><td>1</td><td>0</td><td>-0.6</td><td>0.9</td><td>1</td><td>1</td></tr></table>	1	1	0	-0.6	0.9	1	1	<table border="1"><tr><td>1</td><td>1</td><td>0</td><td>0.2</td><td>0.7</td><td>1</td></tr></table>	1	1	0	0.2	0.7	1	Отрицательная
1	1	0	-0.6	0.9	1	1										
1	1	0	0.2	0.7	1											

Рис. 8.15. Построенный датасет

Чтобы модель не переставала адаптироваться к новым данным, она должна постоянно обучаться. А значит, на основе новых взаимодействий нужно непрерывно генерировать новые обучающие точки данных. Мы рассмотрим непрерывное обучение в разделе «Эксплуатация».

Выбор функции потерь

Поскольку мы обучаем модель бинарной классификации, в качестве функции потерь выберем перекрестную энтропию.

Оценка

Автономные метрики

Системы предсказания кликов по рекламе обычно оцениваются по двум метрикам:

- перекрестная энтропия (CE, Cross-Entropy);
- нормализованная перекрестная энтропия (NCE, Normalized Cross-Entropy).

Перекрестная энтропия оценивает, насколько предсказанные вероятности близки к эталонным меткам. Эта метрика равна 0 в идеальной системе, которая предсказывает 0 для отрицательных классов и 1 для положительных. Чем ниже CE, тем точнее предсказания. Формула выглядит так:

$$H(p, q) = -\sum_{c=1}^C p_c \log q_c,$$

где p — эталонная вероятность, q — предсказанное значение, а C — общее количество классов.

Для бинарной классификации формулу CE можно переписать в следующем виде:

$$H(p, q) = -\sum_i p_i \log q_i = -\sum_i (y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)),$$

где y_i — эталонная метка i -й точки данных, а \hat{y}_i — предсказанная вероятность этой точки.

Рассмотрим конкретный пример, представленный на рис. 8.16.

Заметим, что перекрестная энтропия часто используется не только в качестве метрики, но и как стандартная функция потерь в задачах классификации во время обучения модели.

Нормализованная перекрестная энтропия (NCE). NCE — это отношение CE нашей модели к CE фоновой CTR (средней CTR в обучающих данных). Иначе говоря, NCE сравнивает модель с простым базовым решением, которое всегда предсказывает фоновое значение CTR.

Низкая NCE означает, что модель эффективнее базового решения. $NCE \geq 1$ указывает на то, что модель работает не лучше, чем базовое решение.

$$\text{Нормализованная перекрестная энтропия} = \frac{\text{CE (модель МО)}}{\text{CE (базовые решения)}}.$$

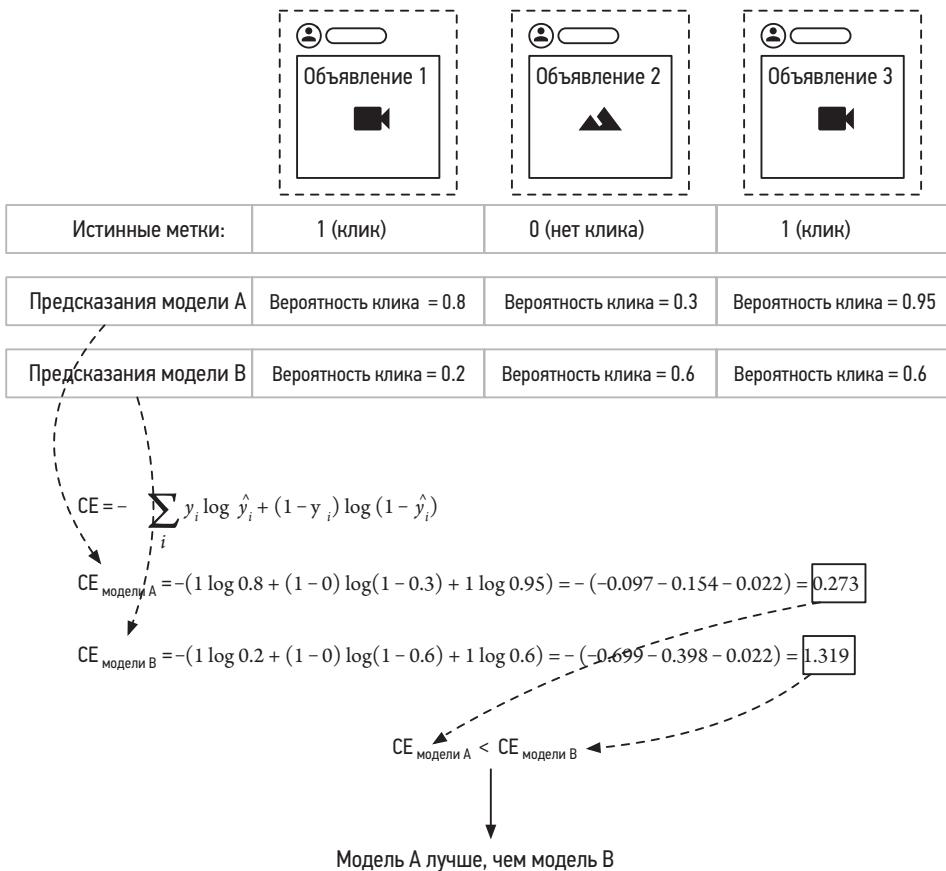


Рис. 8.16. Перекрестная энтропия двух моделей МО

Чтобы разобраться, как вычисляется NCE, рассмотрим конкретный пример на рис. 8.17. В этом примере простая базовая модель всегда предсказывает 0.6 (CTR обучающих данных). В данном случае NCE равна 0.324 (менее 1), то есть модель A эффективнее простой базовой модели.

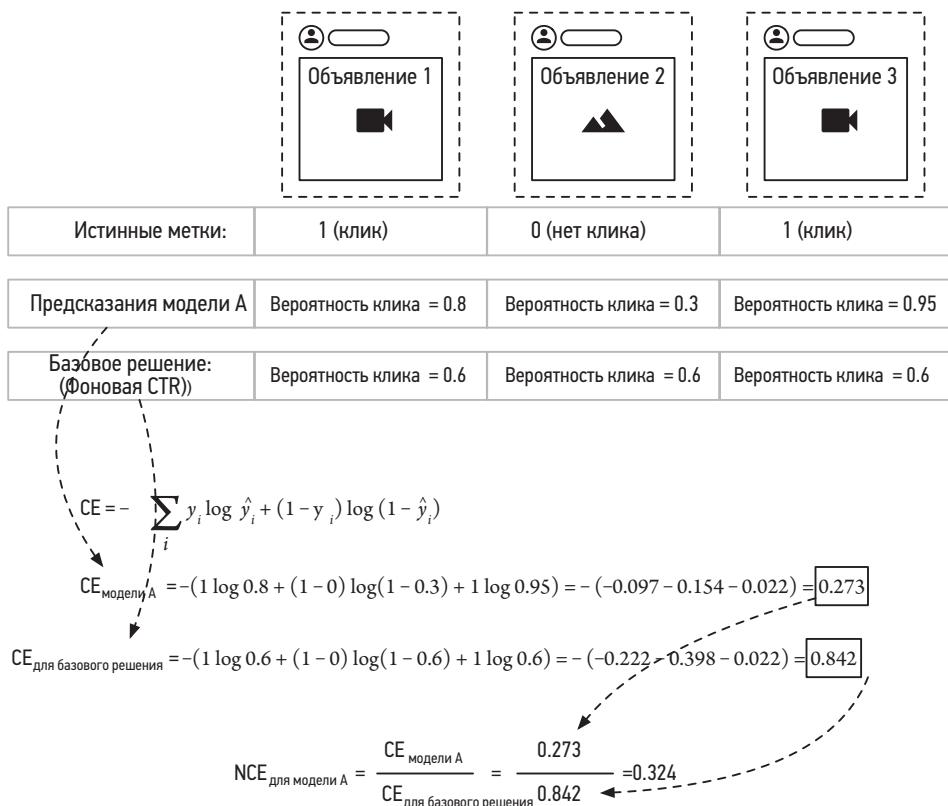


Рис. 8.17. Вычисление NCE для модели А

Оперативные метрики

Рассмотрим некоторые метрики, которые можно использовать в ходе оперативной оценки:

- кликабельность (CTR);
- конверсия;
- рост выручки;
- доля скрываемых объявлений.

CTR. Отношение количества объявлений, по которым щелкнули пользователи, к общему количеству показанных объявлений.

$$\text{CTR} = \frac{\text{количество объявлений, по которым щелкнули пользователи}}{\text{количество показанных объявлений}}.$$

CTR — отличная оперативная метрика для систем предсказания кликов по рекламе, потому что максимизация кликов обычно напрямую связана с ростом выручки.

Конверсия. Отношение количества пользователей, ставших клиентами, к общему количеству показов рекламы.

$$\text{Конверсия} = \frac{\text{количество пользователей, ставших клиентами}}{\text{количество показанных объявлений}}.$$

Эту метрику важно отслеживать, потому что она показывает, сколько раз рекламодатели получили реальную пользу от системы. Если рекламодатели увидят, что реклама не приводит к конверсии, они в конце концов потеряют интерес и перестанут тратиться на рекламу.

Рост выручки. Относительный рост выручки со временем (в процентах).

Доля скрываемых объявлений. Отношение количества объявлений, которые скрыли пользователи, к общему количеству показанных объявлений.

$$\text{Доля скрываемых} = \frac{\text{количество объявлений, которые скрыли пользователи}}{\text{количество показанных объявлений}}.$$

вмешательств

По этой метрике можно судить о том, сколько нерелевантных объявлений система показывает пользователям; такие показы также называются ложноположительными.

Эксплуатация

В режиме эксплуатации система выводит список рекламных объявлений, ранжированный по вероятности клика. Предложенный дизайн МО представлен на рис. 8.18. Рассмотрим каждый из пайплайнов:

- пайpline подготовки данных;
- пайpline непрерывного обучения;
- предсказательный пайpline.

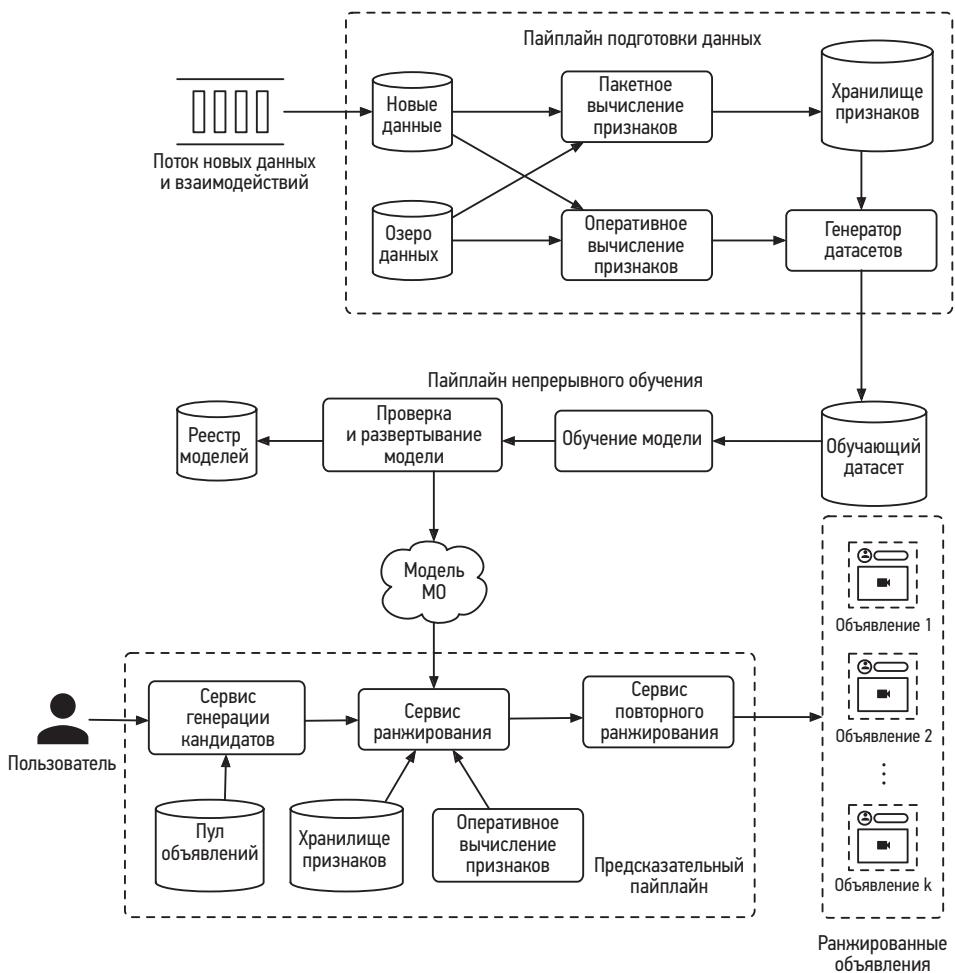


Рис. 8.18. Дизайн системы МО

Пайплайн подготовки данных

Пайплайн подготовки данных решает две задачи.

1. Вычисляет признаки в оперативном и пакетном режиме.
2. Непрерывно генерирует обучающие данные на основе новых объявлений и взаимодействий.

Признаки вычисляются в двух режимах: пакетном и оперативном. Посмотрим, чем они отличаются друг от друга.

Пакетное вычисление признаков

Некоторые из выбранных признаков — статические, то есть такие, которые изменяются очень редко. Например, сюда относятся изображения в объявлениях и категория рекламы. Этот компонент периодически (например, раз в несколько дней или недель) вычисляет статические признаки в пакетных заданиях и сохраняет их в хранилище признаков. Из-за того что признаки вычисляются заранее, улучшается производительность системы в режиме эксплуатации.

Оперативное вычисление признаков

Некоторые признаки — динамические, потому что они часто изменяются: например, количество показов рекламы и количество кликов. Этот компонент вычисляет динамические признаки в момент запроса.

Пайплайн непрерывного обучения

Согласно требованиям, для задачи критично непрерывное обучение модели. Этот пайплайн отвечает за тонкую настройку модели на новых обучающих данных, оценивает новую модель и развертывает ее, если она улучшает метрики. Это гарантирует, чтобы в предсказательном пайплайне всегда используется модель, адаптированная для самых свежих данных.

Предсказательный пайплайн

Предсказательный пайплайн принимает на вход данные о пользователе и выводит список рекламных объявлений, ранжированный по вероятности кликов. Поскольку некоторые признаки, от которых зависит модель, являются динамическими, использовать пакетное предсказание не удастся. Вместо этого запросы обслуживаются по мере поступления путем оперативного предсказания.

Как было показано в предыдущих главах, в предсказательном пайплайне используется двухбашенная архитектура. Сначала сервис генерации кандидатов эффективно сужает пул доступных объявлений до небольшого подмножества. При этом используются критерии таргетированного показа рекламы, которые часто предоставляют рекламодатели: возраст, пол, страна и т. д.

Затем применяется модель ранжирования, которая получает объявления-кандидаты от сервиса генерации кандидатов, ранжирует их по вероятности кликов и выводит самые релевантные результаты. Этот компонент взаимодействует с тем же хранилищем признаков и компонентом оперативного вычисления признаков. После того как получены и статические, и динамические признаки, сервис ранжирования использует модель, чтобы предсказать вероятность клика для каждого кандидата. На основе этой вероятности объявления ранжируются и выводятся те из них, вероятность клика по которым максимальна.

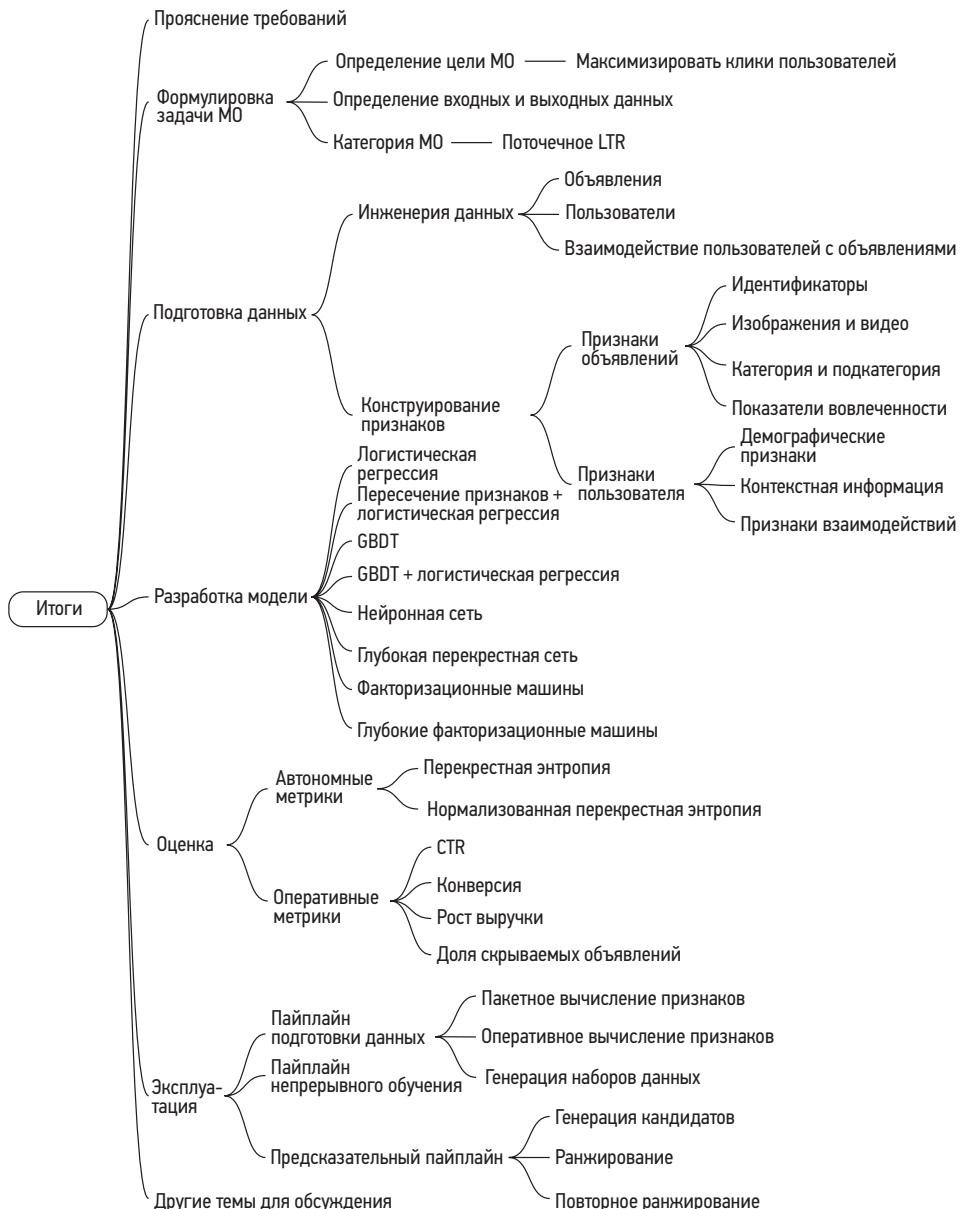
Наконец, сервис повторного ранжирования модифицирует список объявлений, применяя дополнительную логику и эвристики. Например, рекламу можно сделать разнообразнее, удалив из списка слишком похожие объявления.

Другие темы для обсуждения

Если в конце собеседования еще остается время, можно обсудить дополнительные темы.

- В системах ранжирования и рекомендаций важно предотвратить утечку данных [12], [13].
- В системах предсказания кликов по рекламе модель нужно откалибровать. Обсудите методы калибровки модели [14].
- Распространенной разновидностью факторизационной машины является машина с поддержкой полей (FFM). Полезно обсудить FFM и ее отличия от FM [15].
- Распространенная разновидность DeepFM – XDeepFM. Обсудите XDeepFM и ее отличия от DeepFM [10].
- В этой главе вы узнали, почему для систем предсказания кликов по рекламе необходимо непрерывное обучение. Однако непрерывное обучение на новых данных может привести к эффекту «катастрофического забывания». Обсудите, что это такое и как с ним бороться [16].

Итоги



Ссылки

- [1] Как бороться с задержкой обратной связи. <https://arxiv.org/pdf/1907.06558.pdf>
- [2] Основы AdTech. <https://advertising.amazon.com/library/guides/what-is-adtech>
- [3] SimCLR. <https://arxiv.org/pdf/2002.05709.pdf>
- [4] Пересечение признаков. <https://developers.google.com/machine-learning/crash-course/feature-crosses/video-lecture>
- [5] Извлечение признаков с помощью GBDT. <https://towardsdatascience.com/feature-generation-with-gradient-boosted-decision-trees-21d4946d6ab5>
- [6] DCN. <https://arxiv.org/pdf/1708.05123.pdf>
- [7] DCN V2. <https://arxiv.org/pdf/2008.13535.pdf>
- [8] Статья Microsoft о DCN. <https://www.kdd.org/kdd2016/papers/files/adf0975-shanA.pdf>
- [9] Факторизационные машины. <https://www.jefkine.com/recsys/2017/03/27/factorization-machines/>
- [10] Глубокие факторизационные машины. https://d2l.ai/chapter_recommender-systems/deepfm.html
- [11] Решение, выигравшее соревнование Kaggle по предсказанию кликов по рекламе. <https://www.youtube.com/watch?v=4Go5crRVyuU>
- [12] Утечка данных в системах МО. <https://machinelearningmastery.com/data-leakage-machine-learning/>
- [13] Разбиение датасетов по временным интервалам. <https://www.linkedin.com/pulse/time-based-splitting-determining-train-test-data-come-manraj-chalokia/>
- [14] Калибровка модели. <https://machinelearningmastery.com/calibrated-classification-model-in-scikit-learn/>
- [15] Факторизационные машины с поддержкой полей. <https://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf>
- [16] Проблема катастрофического забывания в непрерывном обучении. <https://www.cs.uic.edu/~liub/lifelong-learning/continual-learning.pdf>

9

ПОХОЖИЕ ОБЪЕКТЫ НА ПЛАТФОРМАХ КРАТКОСРОЧНОЙ АРЕНДЫ ЖИЛЬЯ

Многие системы рекомендуют пользователям, которые просматривают какой-либо объект, подборку похожих объектов. Это ключевая технология, которая позволяет находить потенциально релевантный контент на больших платформах. Например, Airbnb рекомендует похожие предложения по аренде жилья, Amazon — похожие товары, а Expedia — похожие туры.

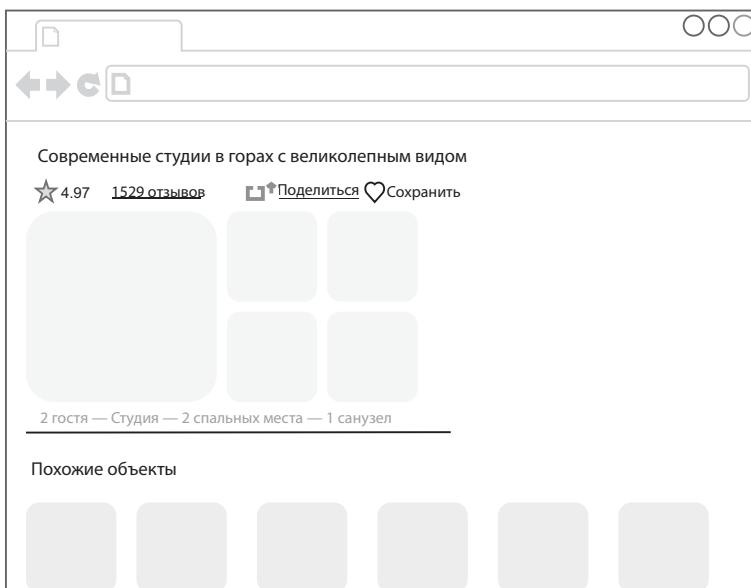


Рис. 9.1. Рекомендуемые похожие объекты

В этой главе мы спроектируем функциональность похожих объектов вроде той, которую предлагают сайты краткосрочной аренды жилья — например, Airbnb

и Vrbo. Когда пользователь открывает конкретный объект, ему предлагается список похожих объектов.

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: Можно ли считать, что бизнес-цель — увеличить количество бронирований?

Эксперт: Да.

Соискатель: Как определяется сходство? Должны ли рекомендуемые объекты быть в чем-то похожи на текущий объект, который просматривает пользователь?

Эксперт: Да, все верно. Два объекта считаются похожими, если они находятся недалеко друг от друга, принадлежат одному диапазону цен и т. д.

Соискатель: Персонализируются ли рекомендуемые объекты для пользователей?

Эксперт: Мы хотим, чтобы эта функциональность работала как для авторизованных, так и для анонимных пользователей. На практике мы работаем с этими двумя группами по-разному и применяем персонализацию только для авторизованных пользователей. Однако для простоты будем считать, что мы одинаково поступаем с авторизованными и анонимными пользователями.

Соискатель: Сколько объектов доступно на платформе?

Эксперт: Пять миллионов.

Соискатель: Как построить обучающий датасет?

Эксперт: Хороший вопрос. В рамках собеседования будем считать, что для этого будут использоваться только взаимодействия пользователей с объектами. Пускай модель не использует ни атрибуты пользователей (такие, как возраст или местонахождение), ни атрибуты объекта (например, цену и местонахождение).

Соискатель: Как скоро после размещения новые объекты должны начинать появляться в списке похожих?

Эксперт: Допустим, будет приемлемо, если новые объекты начнут появляться в рекомендациях через день после размещения. За это время система собирает данные о взаимодействиях для новых объектов.

Резюмируем описание проблемы. Требуется спроектировать функциональность похожих объектов для платформы кратковременной аренды жилья. На вход принимается конкретный объект, который сейчас просматривает пользователь, а на выходе генерируется ранжированный список похожих объектов, по которым

пользователь с большой вероятностью будет кликать. Система должна рекомендовать одни и те же объекты и анонимным, и авторизованным пользователям. На платформе доступно 5 миллионов предложений, и новые предложения могут начать появляться в рекомендациях через день после размещения. Бизнес-цель системы — повысить количество бронирований.

Формулировка проблемы в виде задачи МО

Определение цели МО

Пользователь обычно просматривает серию объектов с похожими характеристиками: например, они находятся в одном городе или в одном ценовом диапазоне. Опираясь на это наблюдение, мы определим цель МО так: точно предсказывать, по какому следующему объекту пользователь кликнет, исходя из того, какой объект он просматривает сейчас.

Определение входных и выходных данных системы

Как показано на рис. 9.2, система похожих объектов принимает объект, который пользователь просматривает сейчас, а затем выдает ранжированный список объектов, отсортированный по вероятности того, что пользователь их откроет.

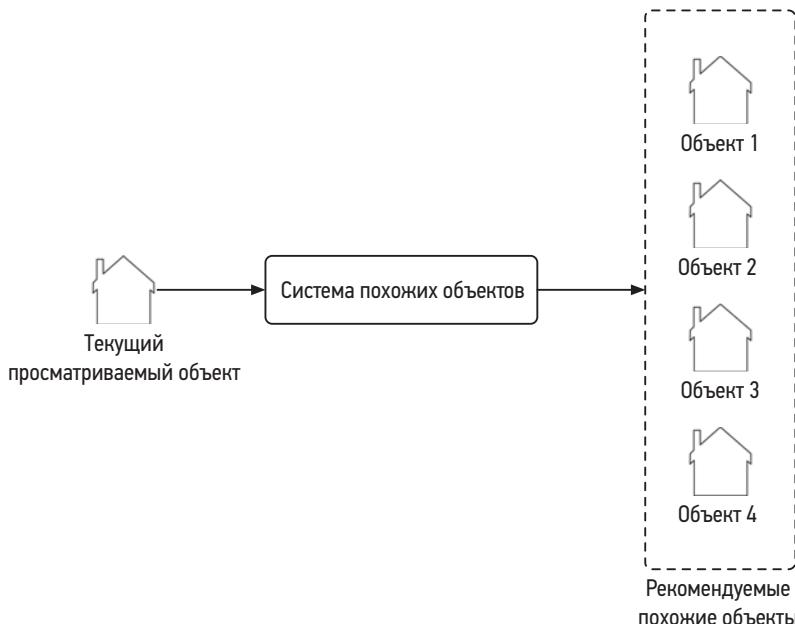


Рис. 9.2. Входные и выходные данные системы похожих объектов

Выбор категории МО

Многие рекомендательные системы полагаются на историю взаимодействий пользователей, чтобы понять их долгосрочные интересы. Однако такой подход может быть неэффективным для системы похожих объектов. В нашей ситуации недавно просмотренные объекты информативнее, чем те, которые просматривались давно. В таких случаях обычно применяются сессионные рекомендательные системы.

Сессионные рекомендательные системы

Сессионная (session-based) рекомендательная система стремится предсказать следующий элемент по серии последних элементов, которые просмотрел пользователь. В нашей системе интересы пользователя зависят от контекста и быстро меняются. Хорошие рекомендации больше опираются на последние взаимодействия пользователя, чем на его общие интересы.

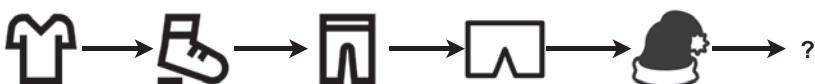


Рис. 9.3. Сеанс (сессия) просмотра товаров

Сессионные и традиционные рекомендательные системы

В традиционных рекомендательных системах интересы пользователей не зависят от контекста и меняются не очень часто. В сессионных системах интересы динамичны и быстро меняются. Традиционная система изучает общие интересы пользователей, а сессионная пытается понять их краткосрочные интересы на основе недавней истории просмотра.

Чтобы разработать сессионную рекомендательную систему, обычно рассматривают эмбеддинги элементов и анализируют, какие элементы встречаются совместно в истории просмотра пользователей. Например, Instagram анализирует эмбеддинги учетных записей пользователей для функции Explore [1], Airbnb анализирует эмбеддинги объектов для поиска похожих объектов [2], а Word2vec [3] использует аналогичный подход, чтобы обучаться содержательным эмбеддингам слов.

В этой главе подбор похожих объектов будет переформулирован как задача построения сессионной рекомендательной системы. Чтобы ее построить, мы обучим модель, которая отображает каждый объект на вектор эмбеддинга так, что если два объекта часто встречаются вместе в истории просмотра пользователей, то их векторы эмбеддингов будут близки друг к другу в пространстве эмбеддингов.

Чтобы рекомендовать похожие объекты, мы ищем в пространстве эмбеддингов объекты, ближайшие к просматриваемому. Рассмотрим пример: на рис. 9.4 каждый объект отображается в двухмерное пространство. Чтобы порекомендовать объекты, похожие на объект X , мы выбираем первые три объекта с ближайшими эмбеддингами.

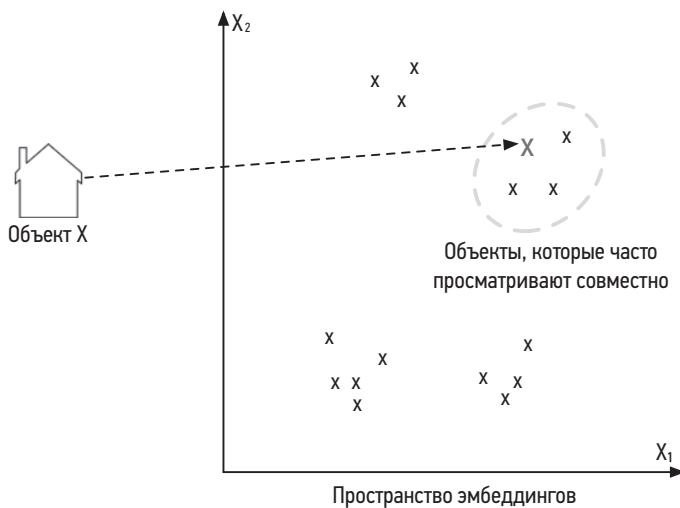


Рис. 9.4. Похожие объекты в пространстве эмбеддингов

Подготовка данных

Инженерия данных

В системе доступны следующие данные:

- пользователи;
- объекты;
- взаимодействия пользователей с объектами.

Пользователи

Ниже приведена упрощенная схема данных о пользователе.

Таблица 9.1. Схема данных о пользователе

ID пользователя	Имя пользователя	Возраст	Пол	Город	Страна	Язык	Часовой пояс
-----------------	------------------	---------	-----	-------	--------	------	--------------

Объекты

В данных об объектах содержатся атрибуты каждого объекта: цена, количество спальных мест, идентификатор собственника и т. д. В табл. 9.2 приведен простой пример того, как могут выглядеть данные объектов.

Таблица 9.2. Данные объектов

ID объекта	ID арендодателя	Цена	Площадь (кв. м)	Рейтинг	Тип	Город	Спальные места	Макс. кол-во гостей
1	135	135	320	4.97	Целый дом	Нью-Йорк	3	4
2	81	80	250	4.6	Отдельная комната	Сан-Франциско	1	2
3	64	65	770	5.0	Общая комната	Бостон	4	6

Взаимодействия пользователей с объектами

В табл. 9.3 хранятся данные о взаимодействиях пользователей с объектами — например, показы, клики и бронирования.

Таблица 9.3. Взаимодействия пользователей с объектами

ID взаимодействия	ID пользователя	ID объекта	Позиция объекта в списке	Тип взаимодействия	Источник	Временная метка
2	18	26	2	Клик	Поиск	1655121925
3	5	18	5	Бронирование	Похожие объекты	1655135257

Конструирование признаков

Как описано в разделе «Формулировка проблемы в виде задачи МО», в процессе обучения модель использует только историю просмотра пользователей и не за- действует другую информацию — цену, возраст пользователя и т. д.

В этой главе история просмотра называется сеансом поиска. Сеанс (или сессия) поиска — это серия идентификаторов объектов, по которым щелкает пользователь; серия продолжается непрерывно и завершается бронированием. На рис. 9.5 показан пример сеанса поиска, который начинается с того, что пользователь щелкнул на O_1 , а заканчивается бронированием O_{20} .



Рис. 9.5. Сеанс поиска

На шаге конструирования признаков мы извлекаем сеансы поиска из данных о взаимодействиях. В табл. 9.4 приведен простой пример сеансов поиска.

Таблица 9.4. Данные сеансов поиска

ID сеанса	ID просмотренных объектов	ID забронированного объекта
1	1, 5, 4, 9	26
2	6, 8, 9, 21, 6, 13, 6	5
3	5, 9	11

Разработка модели

Выбор модели

Нейронная сеть — популярный метод обучения с эмбеддингами. Чтобы выбрать хорошую архитектуру для нейронной сети, нужно учесть ряд факторов: сложность задачи, объем обучающих данных и т. д. Один из стандартных способов выбрать гиперпараметры нейронной сети (количество нейронов и слоев, функцию активации и т. д.) заключается в том, чтобы экспериментировать и остановиться на архитектуре, которая покажет наилучшие результаты. Здесь мы выберем архитектуру неглубокой нейронной сети, которая будет обучаться на эмбеддингах объектов.

Обучение модели

Как показано на рис. 9.6, для заданного входного объекта модель должна предсказывать объекты в пределах его контекста.

Обучение начинается с того, что эмбеддинги инициализируются случайными векторами. Модель постепенно обучается на этих векторах, просматривая сеансы поиска методом скользящего окна. При скольжении эмбеддинг центрального объекта в окне обновляется так, чтобы он стал похож на другие эмбеддинги в этом же окне и не был похож на эмбеддинги за его пределами. Затем модель использует эти эмбеддинги, чтобы предсказывать контекст заданного объекта.

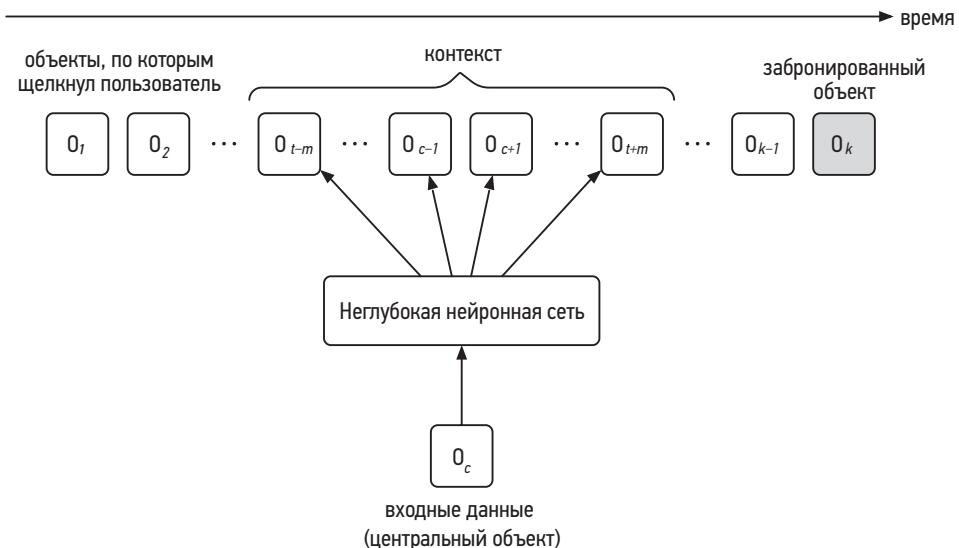


Рис. 9.6. Предсказание похожих объектов

Чтобы адаптировать модель к новым объектам, мы ежедневно обучаем ее на вновь сконструированных обучающих данных.

Построение датасета

Датасет можно построить разными способами. В нашем примере будет выбран метод отрицательной выборки [4], который часто применяют для обучения с эмбеддингами.

Чтобы построить обучающие данные, мы создаем положительные и отрицательные пары на основе сеансов поиска. Положительные пары — это объекты, у которых должны быть похожие эмбеддинги, тогда как в отрицательных парах эмбеддинги должны различаться.

Выражаясь точнее, для каждого сеанса модель считывает объекты методом скользящего окна. По мере скольжения создаются положительные пары из центрального объекта в окне и объектов из его контекста. Отрицательные пары образуются из центрального объекта и случайно выбранных объектов вне его контекста. Положительным парам присваивается эталонная метка 1, а отрицательным — 0.

На рис. 9.7 показано, как формируются положительные и отрицательные пары при скольжении по сеансу поиска.

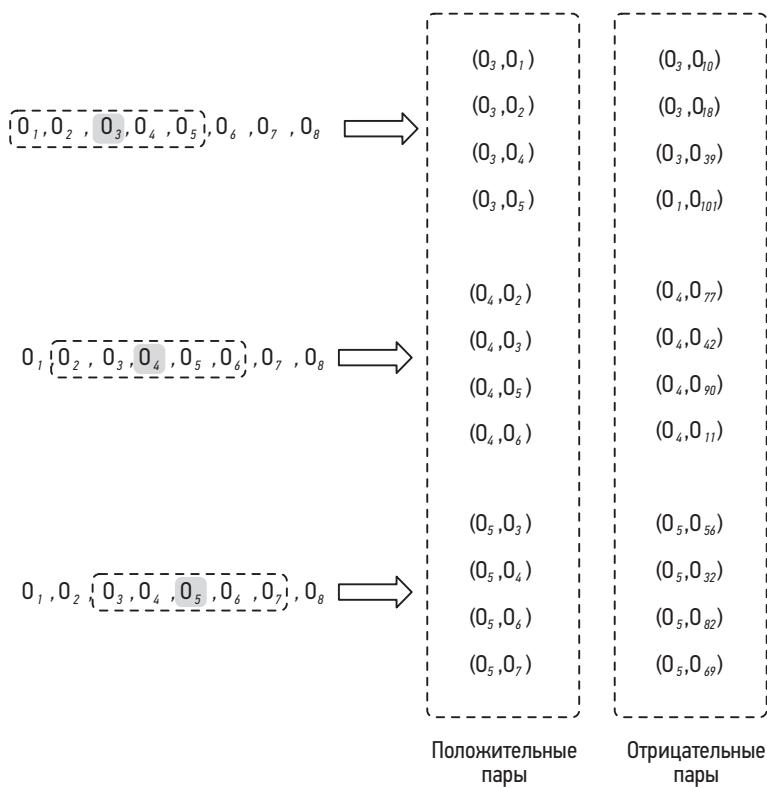


Рис. 9.7. Построение положительных и отрицательных пар объектов

Выбор функции потерь

Функция потерь оценивает соответствие между эталонной меткой и предсказанной вероятностью. Если два объекта образуют положительную пару, то их эмбеддинги должны быть близкими, а если отрицательную, то эмбеддинги должны находиться далеко друг от друга. Говоря более формально, потери вычисляются так:

1. Вычислить расстояние (например, скалярное произведение) между двумя эмбеддингами.
2. С помощью сигмоидной функции преобразовать расстояние в значение вероятности в диапазоне от 0 до 1.
3. С помощью перекрестной энтропии (стандартной классификационной функции потерь) измерить расхождение между предсказанной вероятностью и эталонной меткой.

Порядок действий при вычислении потерь представлен на рис. 9.8.

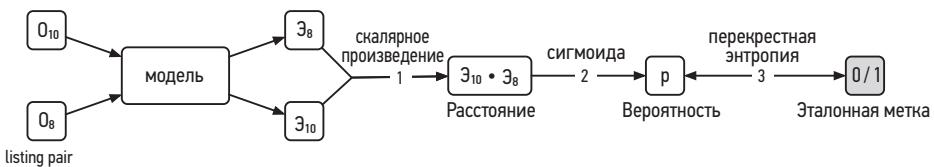


Рис. 9.8. Порядок действий при вычислении потерь

Функцию потерь *loss* можно представить следующей формулой:

$$\text{loss} = \sum_{(c, p) \in D_p} \log \frac{1}{1 + e^{-E_p \cdot E_c}} + \sum_{(c, n) \in D_n} \log \frac{1}{1 + e^{E_n \cdot E_c}},$$

где:

- c — центральный объект, p — положительный объект (который встречается в контексте вместе с c), а n — отрицательный объект (который не встречается вместе с c);
- E_c, E_n, E_p — векторы эмбеддингов объектов c, n и p соответственно;
- D_p — положительное множество пар $\langle c, p \rangle$, которое представляет кортежи \langle центральный объект, объект из контекста \rangle , чьи векторы эмбеддингов близки друг к другу;
- D_n — отрицательное множество пар $\langle c, n \rangle$, которое представляет кортежи \langle центральный объект, случайный объект \rangle , чьи векторы эмбеддингов удалены друг от друга.

Первое суммирование вычисляет потери по положительным парам, а второе — по отрицательным.

Можно ли улучшить функцию потерь, чтобы обучение с эмбеддингами было эффективнее?

Описанная выше функция потерь — хорошая отправная точка, но у нее есть два недостатка. Во-первых, во время обучения эмбеддинг центрального объекта сдвигается к эмбеддингам в своем контексте, а не к эмбеддингу того объекта, который в итоге будет забронирован. Поэтому эмбеддинги хорошо помогают предсказывать соседние объекты, по которым кликает пользователь, а не тот объект, который он в конце концов выберет.

Во-вторых, в ранее сгенерированные отрицательные пары в основном входят объекты из разных регионов, потому что они выбираются случайнym образом.

Однако пользователи чаще ищут жилье для краткосрочной аренды в пределах определенного региона — например, Сан-Франциско. В результате появляются эмбеддинги, плохо работающие с объектами, которые относятся к тому же региону, но не встречались совместно в контексте.

Посмотрим, как избавиться от этих недостатков.

Использование забронированного объекта в качестве глобального контекста

Чтобы обучить эмбеддинги, которые хорошо предсказывают забронированные объекты, мы будем на этапе обучения рассматривать объект, который в итоге забронировал пользователь, как глобальный контекст. При скольжении окна некоторые объекты добавляются в контекстное множество или удаляются из него, но забронированный объект всегда остается в глобальном контексте и участвует в обновлении вектора центрального объекта.

Чтобы на этапе обучения использовать итоговый забронированный объект в качестве глобального контекста, мы добавляем пары \langle центральное предложение, выбранное предложение \rangle в обучающие данные и помечаем их как положительные. Это заставляет модель размещать эмбеддинг забронированного объекта ближе к эмбеддингу каждого из объектов, по которым кликал пользователь во время сеанса поиска, как показано на рис. 9.9.

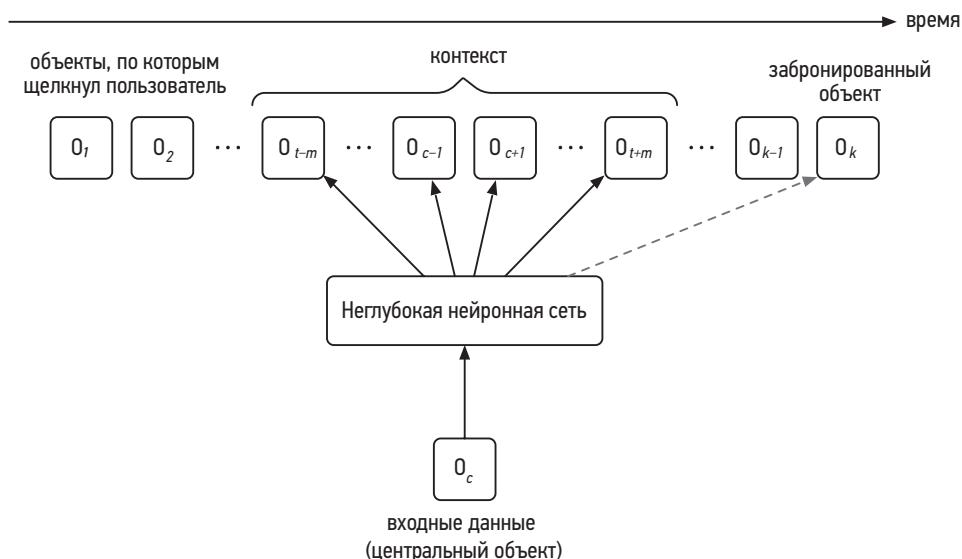


Рис. 9.9. Добавление забронированного объекта в положительные пары

Добавление отрицательных пар из того же региона в обучающие данные

При скольжении окна мы выбираем объект, который находится в одном регионе с центральным объектом, но не входит в контекст этого объекта. Мы помечаем эту пару как отрицательную и добавляем в обучающие данные.

Вот как будет выглядеть обновленная функция потерь с учетом добавленных обучающих данных:

$$\begin{aligned} loss = & \sum_{(c, p) \in D_p} \log \frac{1}{1 + e^{-E_c \cdot E_p}} + \sum_{(c, n) \in D_n} \log \frac{1}{1 + e^{E_c \cdot E_n}} + \\ & + \sum_{(c, b) \in D_{\text{брон}}} \log \frac{1}{1 + e^{-E_c \cdot E_b}} + \sum_{(c, n) \in D_{\text{фикс}}} \log \frac{1}{1 + e^{E_c \cdot E_n}}, \end{aligned}$$

где:

- E_b — вектор эмбеддинга забронированного объекта;
- $D_{\text{брон}}$ — пары (c, b) , представляющие кортежи {центральный объект, забронированный объект}, которые были приближены друг к другу;
- $D_{\text{фикс}}$ — фиксированные отрицательные пары (c, n) , представляющие кортежи {центральный объект, отрицательный объект из того же региона}, которые были отдалены друг от друга.

Первые две суммы разъяснялись выше. Третья сумма вычисляет потери по вновь добавленным положительным парам, которые содержат глобальный контекст. Она помогает модели смещать эмбеддинги центральных объектов ближе к эмбеддингам тех объектов, которые в итоге были забронированы.

Четвертая сумма вычисляет потери по добавленным отрицательным парам из того же региона. Она заставляет модель смещать их эмбеддинги дальше друг от друга.

Оценка

Автономные метрики

На этапе разработки модели мы используем автономные метрики, чтобы измерять качество выходных данных модели и сравнивать только что созданные модели со старыми. Чтобы оценивать эмбеддинги, полученные в результате обучения, можно проверять, насколько эффективно они на основе последнего клика пользователя предсказывают объект, который он в результате забронировал. Создадим метрику «средний ранг забронированного объекта» и обсудим ее подробнее.

Средний ранг забронированного объекта. Чтобы понять смысл этой метрики, рассмотрим пример. На рис. 9.10 представлен пользовательский сеанс поиска. Как видно из диаграммы, он состоит из 7 объектов. Сеанс начинается с объекта, который пользователь просмотрел первым (O_0), а затем идут пять объектов, по которым он последовательно щелкал. Последний объект (O_6) был в итоге забронирован.

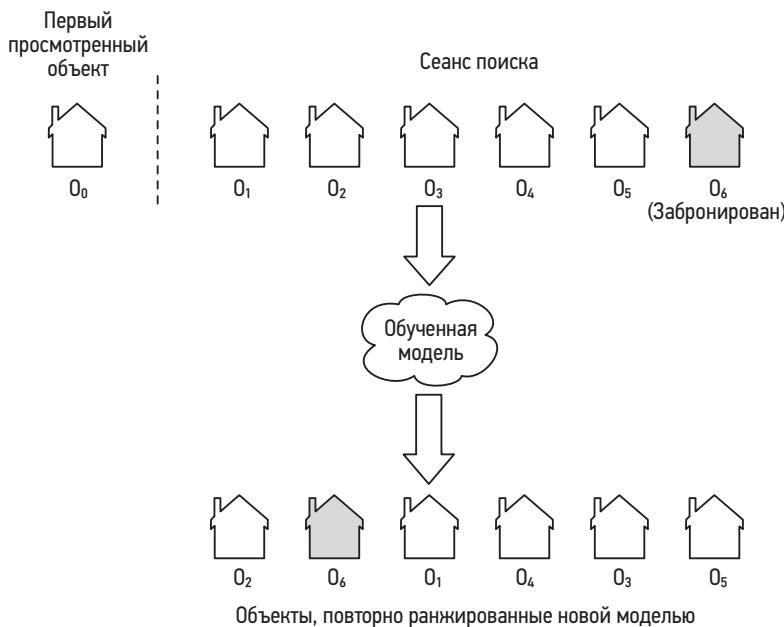


Рис. 9.10. Повторное ранжирование сеанса моделью

С помощью модели вычисляется сходство между первым объектом, по которому щелкнул пользователь, и другими объектами в пространстве эмбеддингов. После того как степени сходства вычислены, объекты ранжируются. Позиция забронированного объекта показывает, насколько высоко в рейтинге рекомендаций он находился бы при использовании новой модели. Как видно из рис. 9.10, новой модели (в нижнем ряду) удалось поместить забронированный объект (O_6) на второе место.

Если модель присваивает забронированному объекту высокий ранг, это значит, что изученные эмбеддинги смогут поместить этот объект выше в списке рекомендаций. Чтобы вычислить итоговую метрику, мы усредняем ранги забронированных объектов по всем сеансам в валидационном датасете.

Оперативные метрики

Согласно требованиям, бизнес-цель заключается в том, чтобы повысить количество бронирований. Вот две возможные оперативные метрики:

- кликабельность (CTR);
- сессионная доля бронирований.

CTR показывает, насколько часто пользователи, которые видят рекомендованные объекты, щелкают по ним.

$$\text{CTR} = \frac{\text{количество рекомендованных объектов, по которым щелкнули пользователи}}{\text{общее количество рекомендованных объектов}}.$$

Эта метрика характеризует степень вовлеченности пользователя. Например, если пользователи чаще кликают по объектам, то растет вероятность того, что какой-нибудь из них конвертируется в бронирование. Но поскольку CTR не измеряет реальное количество бронирований на платформе, мы дополнительно используем метрику сессионной доли бронирований.

Сессионная доля бронирований показывает, сколько сеансов поиска (поисковых сессий) завершаются бронированием.

$$\text{Сессионная доля бронирований} = \frac{\text{количество сеансов, закончившихся бронированием}}{\text{общее количество сеансов бронирований}}.$$

Эта метрика напрямую связана с нашей бизнес-целью, а именно с повышением количества бронирований. Чем выше сессионная доля бронирований, тем большую выручку генерирует платформа.

Эксплуатация

В режиме эксплуатации система рекомендует объекты, похожие на тот, который сейчас просматривает пользователь. На рис. 9.11 представлен общий дизайн системы МО.

Рассмотрим основные компоненты подробнее.

Обучающий пайплайн

Обучающий пайплайн регулирует модель на основе новых объектов и взаимодействий пользователей с объектами. В результате модель остается адаптированной к новым взаимодействиям и объектам.

Индексирующий пайплайн

Имея обученную модель, можно заранее вычислить эмбеддинги всех предложений на платформе и сохранить их в индексной таблице. От этого предсказательный пайплайн работает значительно быстрее.

Индексирующий пайплайн создает и сопровождает индексную таблицу. Например, когда становится доступным эмбеддинг нового объекта, пайплайн добавляет его в эту таблицу. Кроме того, когда становится доступной вновь обученная модель, пайплайн с ее помощью заново вычисляет все эмбеддинги и обновляет таблицу.

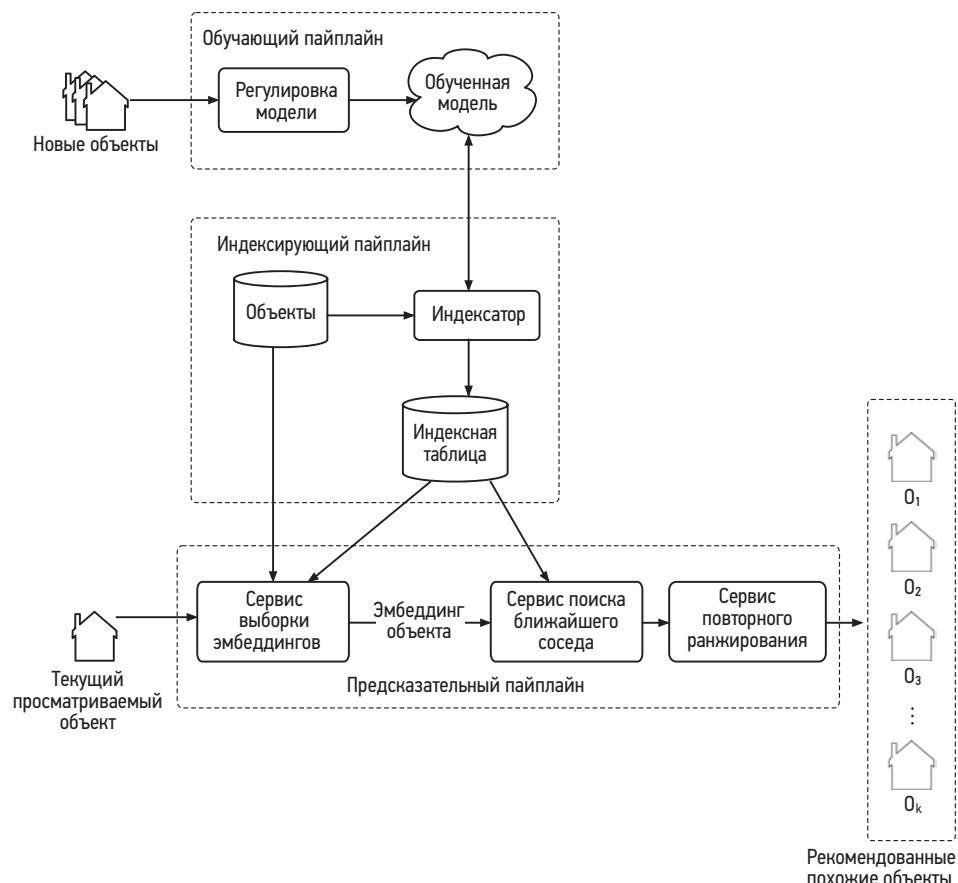


Рис. 9.11. Дизайн системы МО

Предсказательный пайплайн

Предсказательный пайплайн рекомендует объекты, похожие на тот, который сейчас просматривает пользователь. Как показано на рис. 9.11, пайплайн состоит из следующих компонентов:

- сервис выборки эмбеддингов;
- сервис поиска ближайшего соседа;
- сервис повторного ранжирования.

Рассмотрим каждый компонент.

Сервис выборки эмбеддингов

Этот сервис принимает на вход текущий просматриваемый объект и действует по-разному в зависимости от того, обработала ли его модель во время обучения.

Входной объект был обработан моделью во время обучения

Если модель обработала объект при обучении, то его вектор эмбеддинга уже вычислен и доступен в индексной таблице. В таком случае сервис выборки эмбеддингов напрямую загружает эмбеддинг из таблицы.

Входной объект не был обработан моделью во время обучения

Если на вход поступает новый объект, значит, модель еще не встречала его в процессе обучения. Это создает проблемы, потому что невозможно найти похожие объекты, если нет эмбеддинга для этого объекта.

Чтобы решить эту проблему, сервис выборки эмбеддингов обрабатывает новые объекты с помощью эвристик. Например, для нового объекта можно использовать эмбеддинг другого географически близкого объекта. Когда для нового объекта наберется достаточно данных о взаимодействиях, обучающий пайплайн изучает эмбеддинг через тонкую настройку модели.

Сервис поиска ближайшего соседа

Чтобы рекомендовать похожие объекты, нужно вычислить степень сходства между эмбеддингом текущего просматриваемого объекта и эмбеддингами других объектов на платформе. Именно этим занимается сервис поиска ближайшего соседа. Этот сервис вычисляет степени сходства и выводит объекты, которые являются ближайшими соседями в пространстве эмбеддингов.

Не забывайте, что на платформе доступно пять миллионов объектов. Чтобы вычислить степень сходства для такого количества объектов, понадобится время, и эта процедура может замедлить обслуживание. Следовательно, чтобы ускорить обслуживание, нам понадобится приближенный поиск ближайшего соседа.

Сервис повторного ранжирования

Этот сервис модифицирует подборку объектов, применяя пользовательские фильтры и некоторые ограничения. Например, она исключает объект, если он выходит за границу пользовательского фильтра цен. Кроме того, из подборки также могут быть удалены объекты в городах, которые не совпадают с городом текущего просматриваемого предложения.

Другие темы для обсуждения

Если в конце собеседования еще остается время, можно обсудить дополнительные темы.

- Что такое позиционное смещение и что с ним делать [5].
- Какие результаты сессионный подход показывает по сравнению со случайным блужданием [6], и как случайное блуждание с перезапуском (RWR) можно использовать для рекомендации похожих объектов [7].
- Как персонализировать результаты сессионной рекомендательной системы с учетом долгосрочных интересов пользователей (внутрисессионная персонализация) [2]
- Учитывая, что сезонность существенно влияет на рынок краткосрочной аренды жилья, как можно интегрировать ее в нашу систему похожих объектов [8].

Итоги



Ссылки

- [1] Рекомендательная система Explore в Instagram. <https://ai.facebook.com/blog/powered-by-ai-instagram-explore-recommender-system>
- [2] Эмбеддинги объектов при ранжировании результатов поиска. <https://medium.com/airbnb-engineering/listing-embeddings-for-similar-listing-recommendations-and-real-time-personalization-in-search-601172f7603e>.
- [3] Word2vec. <https://en.wikipedia.org/wiki/Word2vec>
- [4] Метод отрицательной выборки. <https://www.baeldung.com/cs/nlps-word2vec-negative-sampling>.
- [5] Позиционное смещение. <https://eugeneyan.com/writing/position-bias/>
- [6] Случайное блуждание. https://en.wikipedia.org/wiki/Random_walk
- [7] Случайное блуждание с перезапуском. https://www.youtube.com/watch?v=HbzQzUaJ_9I
- [8] Сезонность в рекомендательных системах. <https://www.computer.org/csdl/proceedings-article/big-data/2019/09005954/1hJsfT0qL6>

10 ПЕРСОНАЛИЗИРОВАННАЯ ЛЕНТА НОВОСТЕЙ

Введение

Лента новостей — функция социальных сетей, которая подкрепляет вовлеченность пользователей, показывая хронологию активности их друзей и других интересных пользователей и сообществ. Большинство социальных сетей — в том числе Facebook [1], Twiter [2] и LinkedIn [3] — персонализируют ленты новостей, чтобы поддерживать вовлеченность.

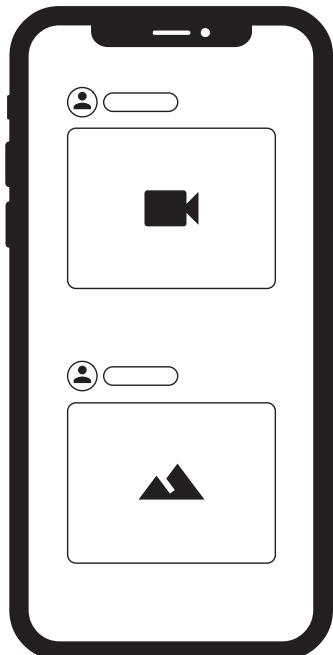


Рис. 10.1. Персонализированная лента пользователя

В этой главе мы спроектируем систему персонализации ленты новостей.

Прояснение требований

Вот типичный диалог между соискателем и экспертом.

Соискатель: Можно ли считать, что персонализированная лента новостей служит для того, чтобы поддерживать уровень вовлеченности пользователей на платформе?

Эксперт: Да, мы выводим рекламу между постами, и чем больше вовлеченность, тем выше выручка.

Соискатель: Когда пользователь обновляет свою ленту, мы отображаем посты с новой активностью. Можно ли считать, что сюда относятся как еще не просмотренные посты, так и посты с еще не просмотренными комментариями?

Эксперт: Да, это разумное предположение.

Соискатель: Что содержится в постах: текст, изображения, видео или любая их комбинация?

Эксперт: Это может быть их произвольная комбинация.

Соискатель: Для лучшей вовлеченности система должна размещать самый привлекательный контент в начале ленты, потому что пользователи, вероятнее всего, будут взаимодействовать с несколькими верхними постами. Это так?

Эксперт: Да, все верно.

Соискатель: Оптимизируется ли система для какого-то конкретного типа вовлеченности? Ведь вовлеченность бывает разная: клики, лайки, репосты и т. д.

Эксперт: Отличный вопрос. У разных реакций на нашей платформе разная ценность: например, если посту поставили лайк, это ценнее, чем простой клик по нему. В идеале при ранжировании постов система должна учитывать основные реакции. При этом я оставляю за вами право решить, что считать вовлеченностью, и выбрать, для чего оптимизировать модель.

Соискатель: Какие основные реакции доступны на платформе? Думаю, что пользователи могут кликать, лайкать, делиться контентом, комментировать, скрывать контент, блокировать других пользователей и отправлять заявки в друзья. Что-нибудь еще?

Эксперт: Вы перечислили все основные варианты. Давайте сосредоточимся на них.

Соискатель: Насколько быстро должна работать система?

Эксперт: Предполагается, что система должна быстро отображать ранжированные посты после того, как пользователь обновил свою ленту или открыл приложение. Если это будет занимать слишком много времени, пользователю надоест ждать, и он уйдет. Будем считать, что система должна выводить ранжированные посты менее чем за 200 миллисекунд.

Соискатель: Сколько в системе активных пользователей в день? Сколько обновлений ленты предполагается за день?

Эксперт: У нас почти три миллиарда пользователей. Около двух миллиардов из них — ежедневные активные пользователи, которые проверяют свои ленты не менее двух раз в день.

Резюмируем описание проблемы. Требуется спроектировать систему персонализации ленты новостей. Система получает посты, которые пользователь ранее не просматривал, а также посты с непросмотренными комментариями, и ранжирует их на основании того, насколько они способствуют вовлеченности пользователя. Получение результата должно занимать не более 200 миллисекунд. Цель системы — повысить вовлеченность пользователей.

Формулировка проблемы в виде задачи МО

Определение цели МО

Рассмотрим три возможные цели МО:

- максимизировать количество тех или иных неявных реакций — например, времени пребывания или кликов;
- максимизировать количество тех или иных явных реакций — например, лайков или репостов;
- максимизировать взвешенную оценку, основанную как на неявных, так и на явных реакциях.

Обсудим каждый вариант подробнее.

Вариант 1. Максимизировать количество неявных реакций — например, времени пребывания или кликов.

В этом случае неявные сигналы интерпретируются как опосредованные свидетельства вовлеченности пользователя. Например, можно оптимизировать систему МО так, чтобы максимизировать количество кликов.

Преимущество этого варианта заключается в том, что о неявных реакциях доступно больше данных, чем о явных. А чем больше обучающих данных, тем обычно точнее модель.

К недостаткам относится то, что неявные реакции не всегда отражают настоящее отношение пользователей к посту. Например, пользователь может щелкнуть по посту и обнаружить, что его не хочется читать.

Вариант 2. Максимизировать количество явных реакций — например, лайков или репостов.

В этом случае отношение пользователей к посту оценивается по явным реакциям.

Преимущество такого подхода в том, что явные сигналы обычно более весомы, чем неявные. Например, если посту поставили лайк — это более сильный сигнал вовлеченности, чем простой клик.

Главный недостаток в том, что лишь небольшое число пользователей выражает свое отношение явными реакциями. Например, пользователь может посчитать пост интересным, но не отреагировать на него. В такой ситуации модели трудно сделать точное предсказание из-за того, что обучающие данные ограничены.

Вариант 3. Максимизировать взвешенную оценку, основанную как на неявных, так и на явных реакциях.

В этом случае, чтобы узнать отношение пользователя к посту, используются как неявные, так и явные реакции. При этом каждой реакции назначается вес в зависимости от того, какую ценность для нас она представляет. Затем система МО оптимизируется, чтобы максимизировать взвешенную оценку реакций.

В табл. 10.1 показано, какие веса можно назначить разным реакциям. Как видите, нажатие кнопки «Нравится» имеет больший вес, чем клик, а репост ценнее лайка. Кроме того, негативным реакциям, таким как сокрытие и блокировка, присвоены отрицательные веса. Обратите внимание, что эти веса могут задаваться, исходя из потребностей бизнеса.

Таблица 10.1. Веса разных реакций

Реакция	Клик	Лайк	Комментарий	Репост	Заявка в друзья	Сокрытие	Блокировка
Вес	1	5	10	20	30	-20	-50

Какой вариант выбрать?

Мы выберем последний комбинированный вариант, потому что он позволяет назначить разные веса разным реакциям. Этот аспект важен, потому что мы можем оптимизировать систему в зависимости от того, что ценно для бизнеса.

Определение входных и выходных данных системы

Как показано на рис. 10.2, персонализированная лента новостей принимает на вход данные о пользователе, а на выходе возвращает ранжированный список непросмотренных постов, а также постов с непросмотренными комментариями, отсортированный по показателю вовлеченности.

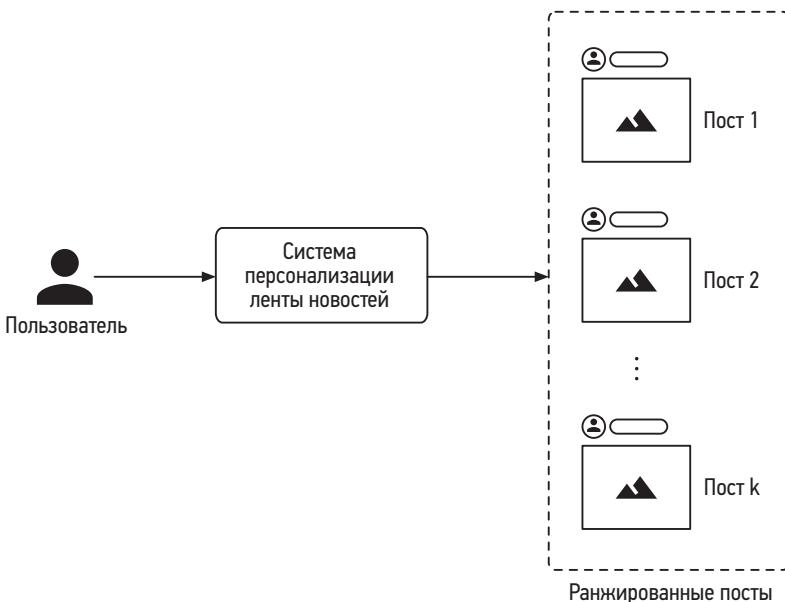


Рис. 10.2. Входные и выходные данные системы персонализации ленты новостей

Выбор категории МО

Система персонализации ленты новостей вырабатывает список постов, ранжированный по тому, насколько они способствуют вовлеченности пользователя. Поточечная модель обучения ранжированию (LTR) — простой, но эффективный метод, который персонализирует ленту, ранжируя посты на основании показателей вовлеченности. Чтобы понять, как вычислять эти показатели, рассмотрим конкретный пример.

Как демонстрируется на рис. 10.3, мы применяем несколько бинарных классификаторов, чтобы прогнозировать вероятности разных явных и неявных реакций для пары \langle пользователь, пост \rangle .

После того как вероятности спрогнозированы, мы вычисляем показатель вовлеченности. Соответствующий пример показан на рис. 10.4.

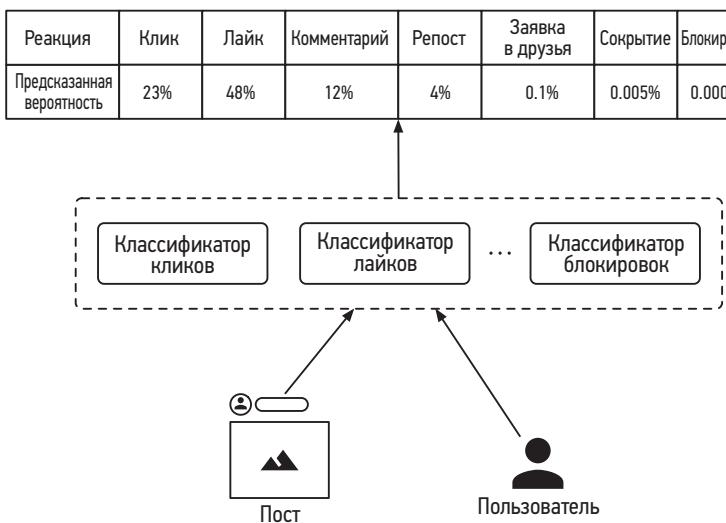


Рис. 10.3. Предсказанные вероятности различных реакций

Реакция	Клик	Лайк	Комментарий	Репост	Заявка в друзья	Сокрытие	Блокировка
Предсказанная вероятность	23%	48%	12%	4%	0.1%	0.005%	0.0003%
Вес	1	5	10	20	30	-20	-50
Значение	0.23	2.4	1.2	0.8	0.03	-0.001	-0.00015
Показатель вовлеченности = 4.65885							

Рис. 10.4. Вычисление показателя вовлеченности

Подготовка данных

Инженерия данных

Перед тем как конструировать предсказательные признаки, бывает полезно понять, какие исходные данные есть в вашем распоряжении. В нашем примере предполагается, что доступны следующие данные:

- пользователи;
- посты;

- взаимодействия пользователей с постами;
- дружеские отношения.

Пользователи

Ниже представлена схема данных о пользователе.

Таблица 10.2. Схема данных о пользователе

ID поль- зователя	Имя поль- зователя	Возраст	Пол	Город	Страна	Язык	Часовой пояс
----------------------	-----------------------	---------	-----	-------	--------	------	-----------------

Посты

В табл. 10.3 приведены данные постов.

Таблица 10.3. Данные постов

ID ав- тора	Текстовое содержимое	Хештеги	Упоминания	Изображения или видео	Временна́я метка
5	Сегодня в нашем лю- бимом месте с лучшим другом	жизнь_пре- красна, счастье	hs2008	-	1658450539
1	Это была лучшая поезд- ка в моей жизни	Путешествия, Мальдивы	Alexish, shan.tony	http://cdn.mysite.com/ maldives.jpg	1658451341
29	Сегодня со мной случи- лось нечто неприятное, о чем я хотел бы расска- зать. Я отправился...	-	-	-	1658451365

Взаимодействия пользователей с постами

В табл. 10.4 приведены данные о взаимодействиях пользователей с постами.

Таблица 10.4. Данные о взаимодействиях пользователей с постами

ID поль- зователя	ID поста	Тип взаимо- действия	Значение взаи- модействия	Геопозиция (широта, долгота)	Временна́я метка
4	18	Лайк	-	38.8951 -77.0364	1658450539

ID поль- зователя	ID поста	Тип взаимо- действия	Значение взаи- модействия	Геопозиция (широта, долгота)	Временная метка
4	18	Репост	Пользователь 9	41.9241 -89.0389	1658451365
9	18	Комментарий	Выглядишь по- трясающе	22.7531 47.9642	1658435948
9	18	Блокировка	-	22.531 47.9642	1658451849
6	9	Показ	-	37.5189 122.6405	1658821820

Дружеские отношения

В таблице дружеских отношений хранятся данные о связях между пользователями. Предполагается, что пользователи могут указывать своих близких друзей и членов семьи. Примеры данных о дружеских отношениях приведены в табл. 10.5.

Таблица 10.5. Данные о дружеских отношениях

ID пользователя 1	ID пользователя 2	Временная метка формирования связи	Близкий друг	Член семьи
28	3	1558451341	Да	Нет
7	39	1559281720	Нет	Да
11	25	1559312942	Нет	Нет

Конструирование признаков

В этом разделе мы сконструируем предсказательные признаки и подготовим их для модели. В частности, будут сконструированы признаки в следующих категориях:

- посты;
- пользователи;
- связи между пользователями и авторами.

Признаки постов

На практике у каждого поста есть множество атрибутов. Рассмотреть все атрибуты не удастся, поэтому обсудим лишь самые важные:

- текстовое содержимое;
- изображения или видео;
- реакции;
- хештеги;
- возраст поста.

Текстовое содержимое

Что это такое. Текстовый контент, то есть тело поста.

Почему эти данные важны. Текстовое содержимое дает понять, о чем написан пост.

Как подготовить данные. Мы предварительно обрабатываем текстовое содержимое, после чего с помощью предварительно обученной языковой модели преобразуем текст в числовой вектор. Так как текстовое содержимое обычно состоит из предложений, а не из отдельных слов, мы используем контекстно-зависимую языковую модель, такую как BERT [4].

Изображения или видео

Что это такое. В посте могут быть изображения или видео.

Почему эти данные важны. Из изображений можно извлечь важные сигналы. Например, изображение оружия может указывать на то, что пост небезопасен для детей.

Как подготовить данные. Мы сначала обрабатываем изображения или видео, а затем предварительно обученная модель преобразует неструктурированные медиаданные в вектор эмбеддинга. В качестве такой модели можно выбрать, например, ResNet [5] или недавно появившуюся CLIP [6].

Реакции

Что это такое. К этой категории относится количество лайков, репостов, ответов и прочих подобных атрибутов поста.

Почему эти данные важны. Количество лайков, репостов, сокрытий и т. д. показывает, насколько вовлекающим может быть пост. Пользователь скорее прочитает пост с тысячей лайков, чем с десятком.

Как подготовить данные. Значения представляются числами. Эти числа масштабируются и приводятся к сходным диапазонам.

Хештеги

Почему эти данные важны. Хештеги описывают темы, к которым относится пост, и помогают группировать контент, связанный с одной темой. Например, пост с хештегом `#женщины_в_IT` указывает, что контент связан с технологиями и женщинами, так что модель может ранжировать его выше для пользователей, которые интересуются технологиями.

Как подготовить данные. Подробная процедура предварительной обработки текста уже объяснялась в главе 4 «Поиск видео на YouTube», так что здесь мы ограничимся специфическими операциями, которые касаются подготовки хештегов.

- **Токенизация.** Хештеги вроде «жизньпрекрасна» или «библиотека_программиста» состоят из нескольких слов. Чтобы выделять лексемы из хештегов, используются такие методы, как алгоритм Витерби [7]. Например, «жизньпрекрасна» преобразуется в два слова: «жизнь» и «прекрасный».
- **Преобразование лексем в идентификаторы.** Хештеги в социальных сетях быстро эволюционируют и изменяются по мере того, как появляются и исчезают актуальные темы. Хеширование признаков хорошо тем, что оно позволяет присвоить индексы ранее не встречавшимся тегам.
- **Векторизация.** Мы будем векторизовать хештеги с помощью простых методов текстового представления (таких, как TF-IDF [8] или Word2vec [9]) вместо моделей на базе Transformer. Дело в том, что модели на базе Transformer имеют смысл тогда, когда важен контекст данных. Каждый хештег состоит из одного слова или фразы, и, чтобы понять его смысл, контекст обычно не нужен. Поэтому мы предпочтем более быстрые и легкие методы текстового представления.

Возраст поста

Что это такое. Этот признак показывает, сколько времени прошло с момента, когда автор опубликовал контент.

Почему эти данные важны. Пользователи склонны интересоваться новым контентом.

Как подготовить данные. Возраст поста разбивается на категории, которые затем представляются с помощью унитарного кодирования. Например, можно выбрать следующие категории:

- 0: менее 1 часа;
- 1: 1–5 часов;
- 2: 5–24 часа;
- 3: 1–7 дней;
- 4: 7–30 дней;
- 5: более 1 месяца.

Признаки, относящиеся к постам, представлены на рис. 10.5.

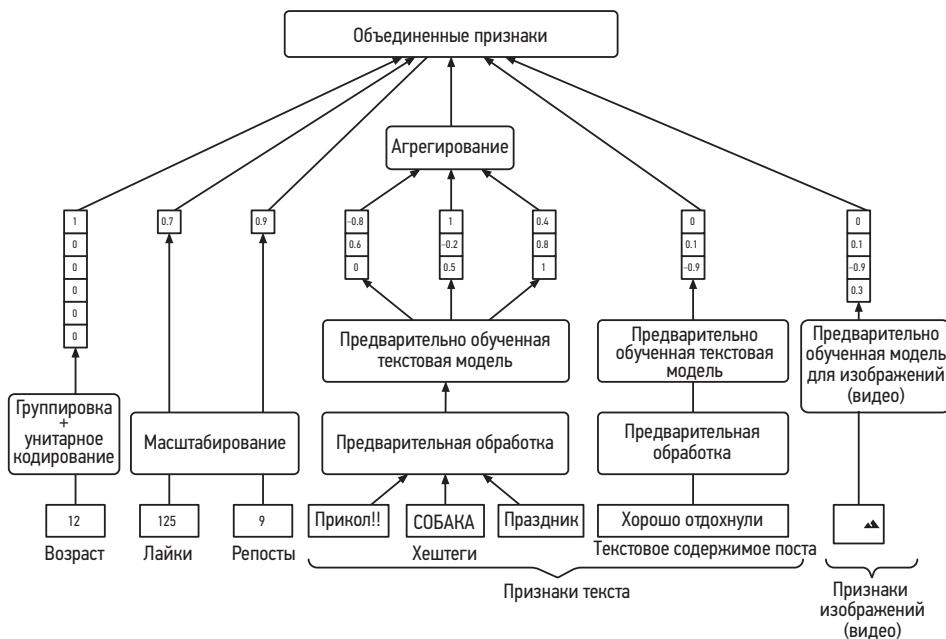


Рис. 10.5. Подготовка признаков, относящихся к постам

Признаки пользователей

Важнейшие признаки, относящиеся к пользователям:

- демографические признаки: возраст, пол, страна и т. д.;
- контекстная информация: устройство, время суток и т. д.;
- история взаимодействий пользователя с постами;
- упоминания в постах.

Про демографические и контекстные признаки мы уже говорили в предыдущих главах, так что здесь рассмотрим только два последних.

История взаимодействий пользователя с постами

Все посты, которым пользователь поставил лайки, представляются списком идентификаторов постов. Та же логика применяется к репостам и комментариям.

Почему эти данные важны. Предыдущие проявления вовлеченности пользователей обычно помогают спрогнозировать их поведение в будущем.

Как подготовить данные. Нужно извлечь признаки из каждого поста, с которым пользователь взаимодействовал.

Упоминания в постах

Что это такое. Признак показывает, упоминался ли пользователь в посте.

Почему эти данные важны. Пользователи обычно обращают больше внимания на посты, где их упоминают.

Как подготовить данные. Этот признак представляется бинарным значением. Если пользователь упоминается в посте, то значение равно 1, в противном случае – 0.

На рис. 10.6 представлен процесс подготовки признаков, относящихся к пользователю.

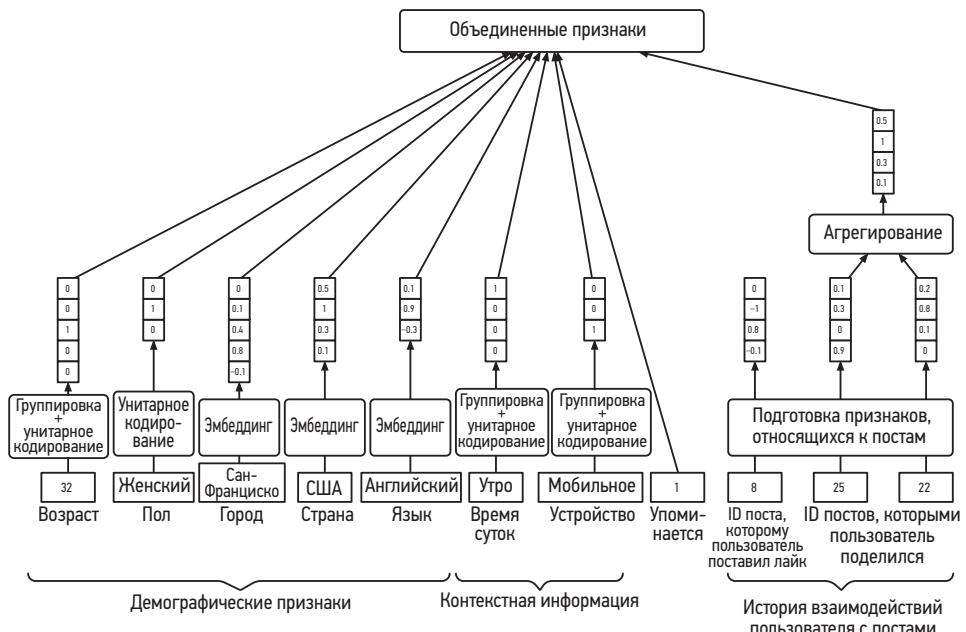


Рис. 10.6. Подготовка признаков, относящихся к пользователю

Связи между пользователями и авторами

По данным исследований, признаки связей (например, между пользователем и автором) относятся к важнейшим факторам при предсказании вовлеченности пользователя на Facebook [10]. Сконструируем несколько признаков, которые отражают связи между пользователями и авторами.

Частота лайков (кликов, комментариев, репостов)

Частота, с которой пользователь реагировал на предыдущие посты автора. Например, частота лайков 0.95 означает, что пользователь ставил лайки 95 процентам постов этого автора.

Продолжительность дружбы

Количество дней, в течение которых пользователь и автор числятся друзьями на платформе. Значение этого признака можно извлечь из данных о дружеских отношениях.

Почему эти данные важны. Пользователи склонны больше взаимодействовать со своими друзьями.

Близкие друзья и семья

Бинарное значение, которое показывает, включили ли пользователь и автор друг друга в списки близких друзей или членов семьи.

Почему эти данные важны. Пользователи обращают больше внимания на посты близких друзей и членов семьи.

На рис. 10.7 представлена сводка признаков, относящихся к связям между пользователями и авторами.

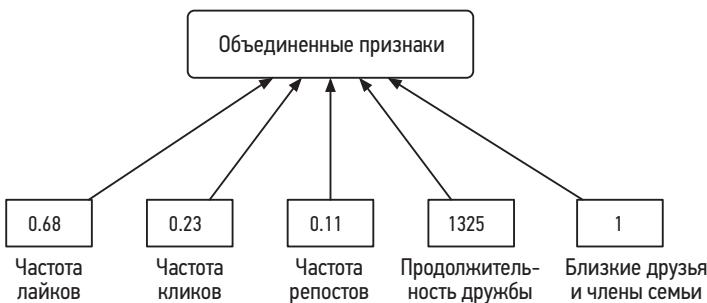


Рис. 10.7. Признаки связей между пользователями и авторами

Разработка модели

Выбор модели

Мы выбираем нейронные сети по следующим причинам.

- Нейронные сети хорошо работают с неструктурированными данными, такими как текст и изображения.

- Нейронные сети позволяют представлять категориальные признаки с помощью слоев эмбеддингов.
 - Архитектура нейронных сетей позволяет регулировать предварительно обученные модели, которые применяются при конструировании признаков. С другими моделями это невозможно.

Прежде чем обучать нейронную сеть, необходимо выбрать ее архитектуру. Мы будем выбирать из двух вариантов глубоких нейронных сетей (DNN):

- несколько независимых DNN;
 - многозадачная DNN.

Рассмотрим эти варианты подробнее.

Вариант 1. Несколько независимых DNN.

В этом варианте используются несколько независимых глубоких нейронных сетей — по одной для каждой реакции. Соответствующая схема изображена на рис. 10.8.

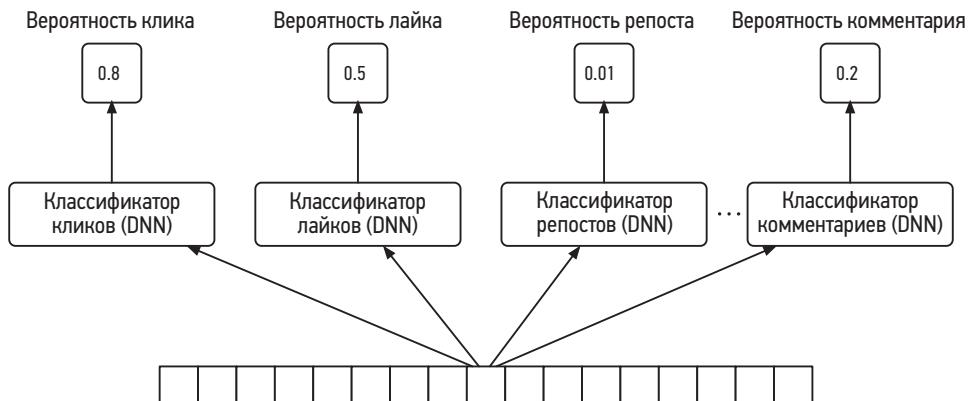


Рис. 10.8. Несколько независимых DNN

У этого варианта есть два недостатка.

- **Высокие затраты на обучение.** Обучение нескольких независимых DNN требует значительных вычислительных ресурсов и занимает много времени.
 - Для менее частых реакций может **не хватать обучающих данных**. Это значит, что система не сможет точно предсказать вероятности для редких реакций.

Вариант 2. Многозадачная DNN.

Чтобы избежать этих проблем, можно использовать метод многозадачного обучения (рис. 10.9).

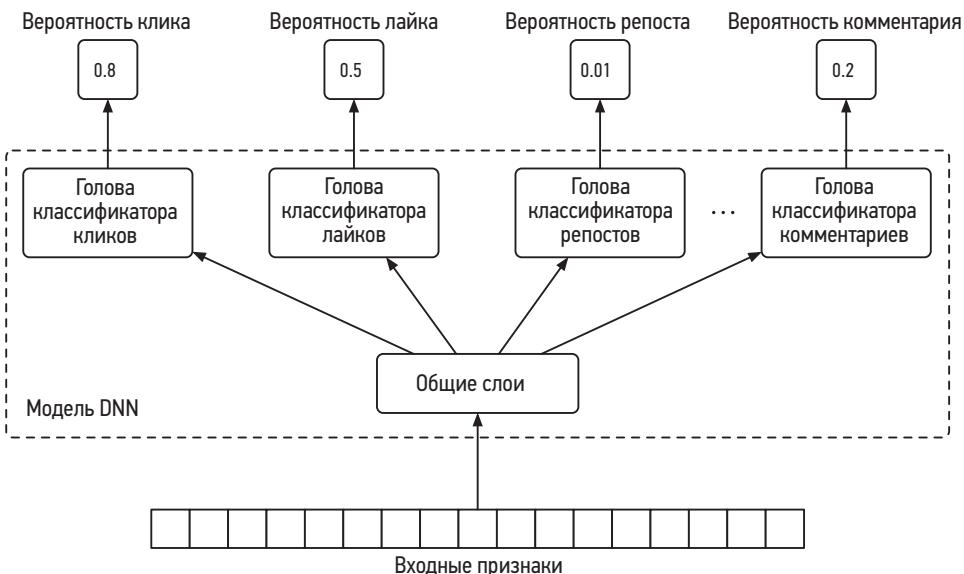


Рис. 10.9. Многозадачная DNN

Многозадачное обучение уже рассматривалось в главе 5 «Обнаружение вредоносного контента», так что здесь мы упомянем его в самых общих чертах. Под многозадачным обучением понимается процесс, когда модель одновременно обучается нескольким задачам. Это позволяет ей изучать сходство между задачами и избегать ненужных вычислений. Для модели многозадачной нейронной сети очень важно выбрать подходящую архитектуру. Архитектура и сопутствующие гиперпараметры обычно определяются в результате экспериментов: вы обучаете и оцениваете модели с разными архитектурами и выбираете тот вариант, который приводит к лучшему результату.

Улучшение архитектуры DNN для пассивных пользователей

До сих пор мы применяли DNN, чтобы предсказывать реакции: лайки, клики, репосты и комментарии. Однако многие пользователи используют платформу пассивно: они мало взаимодействуют с контентом в своих лентах. Для таких пользователей текущая модель DNN предсказывает очень низкие вероятности всех реакций, потому что они редко реагируют на посты. А значит, нужно изменить архитектуру DNN, чтобы она учитывала пассивных пользователей.

Для этого добавим в список задач две неявные реакции.

- **Время пребывания.** Время, в течение которого у пользователя был открыт пост.
- **Пропуск.** Если пост оставался открытым менее t секунд (например, 0.5 с), считается, что пользователь его пропустил.

На рис. 10.10 изображена многозадачная модель DNN с дополнительными задачами.

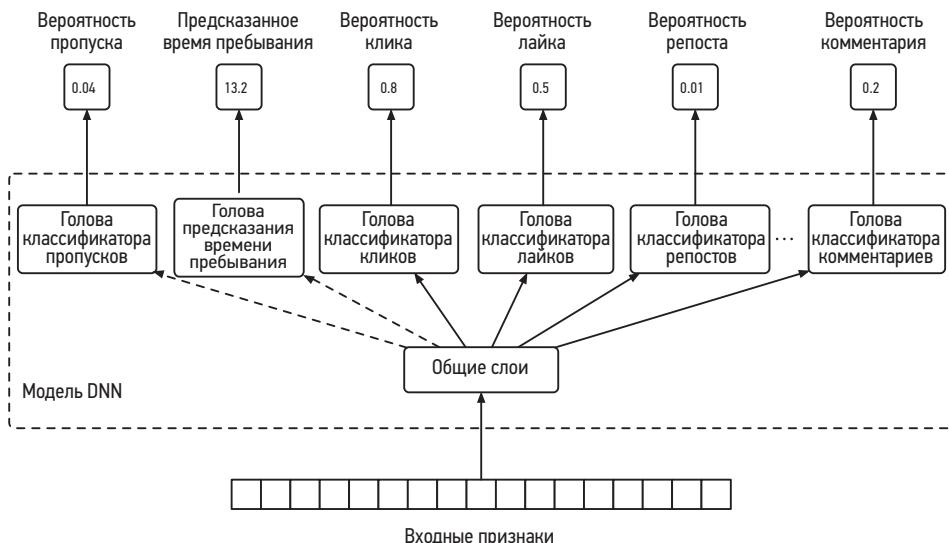


Рис. 10.10. Многозадачная модель DNN с двумя новыми задачами

Обучение модели

Построение датасета

На этом шаге мы построим датасет на основе исходных данных. Так как модель DNN должна изучать разные задачи, нужно создать положительные и отрицательные точки данных для каждой из них (кликов, лайков и т. д.).

Как создаются положительные и отрицательные точки данных, мы объясним на примере лайков. Каждый раз, когда пользователь ставит лайк посту, мы добавляем точку в набор данных, вычисляем признаки {пользователь, пост} и помечаем ее как положительную.

Чтобы сгенерировать отрицательные точки данных, мы выбираем показы, которые не привели к лайку. Следует учитывать, что отрицательных точек данных

обычно намного больше, чем положительных. Чтобы набор данных оставался сбалансированным, мы создаем столько же отрицательных точек, сколько положительных. На рис. 10.11 изображены положительные и отрицательные точки данных для лайков.

№	Признаки пользователя	Признаки поста	Признаки связей	Метка
1	1 0 1 0.8 0.1 1	0 1 1 0.4 0	0.9 0.6 0.3 8 0	Положительная
2	0 0 0 0.4 0.9 0	1 1 0 0.3 1	1 0.9 0.8 120 1	Положительная
3	1 1 0 0.1 0.5 0	0 1 0 0.9 1	0.1 0 0 2 0	Отрицательная

Рис. 10.11. Обучающие данные для задачи классификации лайков

Аналогичным образом можно создавать положительные и отрицательные метки для других реакций. Но поскольку время пребывания — это задача регрессии, оно конструируется по-другому. Как показано на рис. 10.12, эталонной меткой становится время показа.

№	Признаки пользователя	Признаки поста	Признаки связей	Время пребывания
1	0 0 0 0.1 0.9 1	1 1 0 0.1 1	0.6 0.6 0.3 0.2 5 0	8.1
2	1 1 1 0.9 0.1 0	1 1 0 0.8 0	0.1 0.9 0.3 0.1 3 1	11.5

Рис. 10.12. Обучающие данные для задачи времени пребывания

Выбор функции потерь

Многозадачные модели обучаются сразу для нескольких задач. Это значит, что нужно вычислять потери для каждой задачи по отдельности, а затем объединить их в общую функцию потерь. Как правило, функция потерь определяется для каждой задачи в зависимости от категории МО, к которой относится эта задача. В нашем случае для каждой задачи бинарной классификации используется бинарная перекрестная энтропия, а для задачи регрессии (прогнозирования времени пребывания) — регрессионная функция потерь (например, MAE [11], MSE [12] или функция Хьюбера [13]). Общие потери вычисляются объединением потерь для конкретных задач, как показано на рис. 10.13.

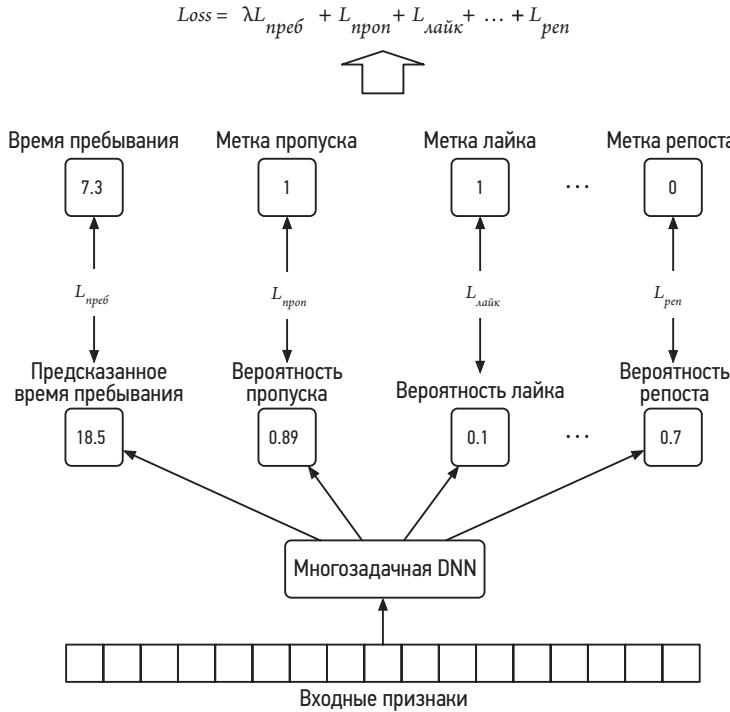


Рис. 10.13. Процесс обучения

Оценка

Автономные метрики

В процессе автономной оценки качество модели оценивается по тому, насколько правильно она предсказывает разные реакции. Чтобы оценивать качество предсказаний отдельных типов реакций, можно применять метрики бинарной классификации, такие как точность и полнота. Однако одних этих метрик может быть недостаточно, чтобы понять качество модели в целом. Поэтому мы используем ROC-кривую, которая позволяет оценить баланс между долями истинно положительных и ложноположительных результатов. Кроме того, мы вычисляем площадь под ROC-кривой (ROC-AUC), чтобы обобщить качество бинарной классификации в виде числового значения.

Оперативные метрики

Чтобы оценить разные аспекты вовлеченности пользователя, используются следующие метрики:

- кликабельность (CTR);
- частота реакций;
- общее количество проведенного времени;
- удовлетворенность пользователей по результатам опросов.

CTR. Отношение количества кликов к количеству показов.

$$\text{CTR} = \frac{\text{количество постов, по которым кликнул пользователь}}{\text{количество показанных постов}}.$$

Высокое значение CTR необязательно указывает на высокую вовлеченность пользователей. Например, пользователь может открыть «мусорный» кликбейт-ный пост и быстро понять, что он не заслуживает внимания. Несмотря на этот недостаток, CTR — довольно важная метрика.

Частота реакций. Набор метрик, отражающих реакции пользователя. Например, частота лайков — это отношение количества постов, которым пользователь поставил лайк, к общему количеству постов в ленте пользователя.

$$\text{Частота лайков} = \frac{\text{количество постов, которым пользователь поставил лайк}}{\text{количество показанных постов}}.$$

Аналогично вычисляются другие реакции: частота репостов, комментариев, сокрытий, блокирований и пропусков. Это все более содержательные сигналы, чем CTR, потому что с их помощью пользователи явно выражают свои предпочтения.

Перечисленные метрики основаны на реакциях пользователей. Но как насчет пассивных пользователей, которые обычно не реагируют на большинство постов? Чтобы оценить эффективность персонализации ленты новостей для таких пользователей, добавим еще две метрики.

Общее количество проведенного времени. Время, которое пользователи провели за просмотром ленты за фиксированный период времени (например, одну неделю). Эта метрика характеризует общую вовлеченность как пассивных, так и активных пользователей.

Удовлетворенность пользователей по результатам опросов. Чтобы оценить, насколько эффективна система персонализации ленты новостей, можно явно запрашивать у пользователей их мнение о ленте и о том, насколько интересны для них посты в ней. Явная обратная связь позволяет точнее измерить качество системы.

Эксплуатация

В режиме эксплуатации система обслуживает запросы, выводя ранжированный список постов. На рис. 10.14 показан общий дизайн системы персонализации ленты новостей. В системе два пайплайна:

- пайpline подготовки данных;
- предсказательный пайpline.

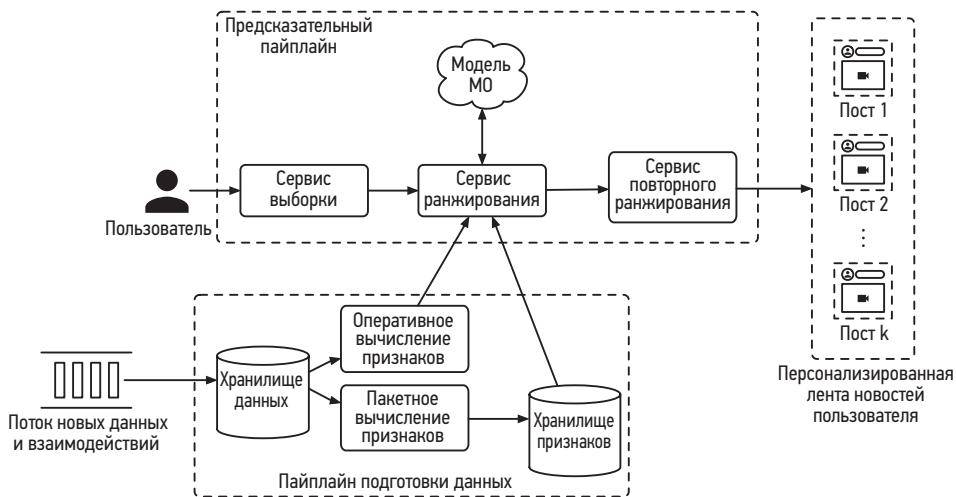


Рис. 10.14. Дизайн системы МО для персонализации ленты новостей

Мы не будем подробно рассматривать пайpline подготовки данных, потому что он очень похож на описанный в главе 8 «Предсказание кликов по рекламе на социальных платформах». Однако предсказательный пайpline заслуживает внимания.

Предсказательный пайpline

Этот пайpline состоит из следующих компонентов: сервис выборки, сервис ранжирования и сервис повторного ранжирования.

Сервис выборки

Этот компонент загружает посты, которые пользователь еще не просматривал, а также посты с новыми комментариями. Чтобы больше узнать о том, как эффективно загружать непросмотренные посты, обращайтесь к [14].

Сервис ранжирования

Этот компонент ранжирует загруженные посты, присваивая каждому из них оценку вовлеченности.

Сервис повторного ранжирования

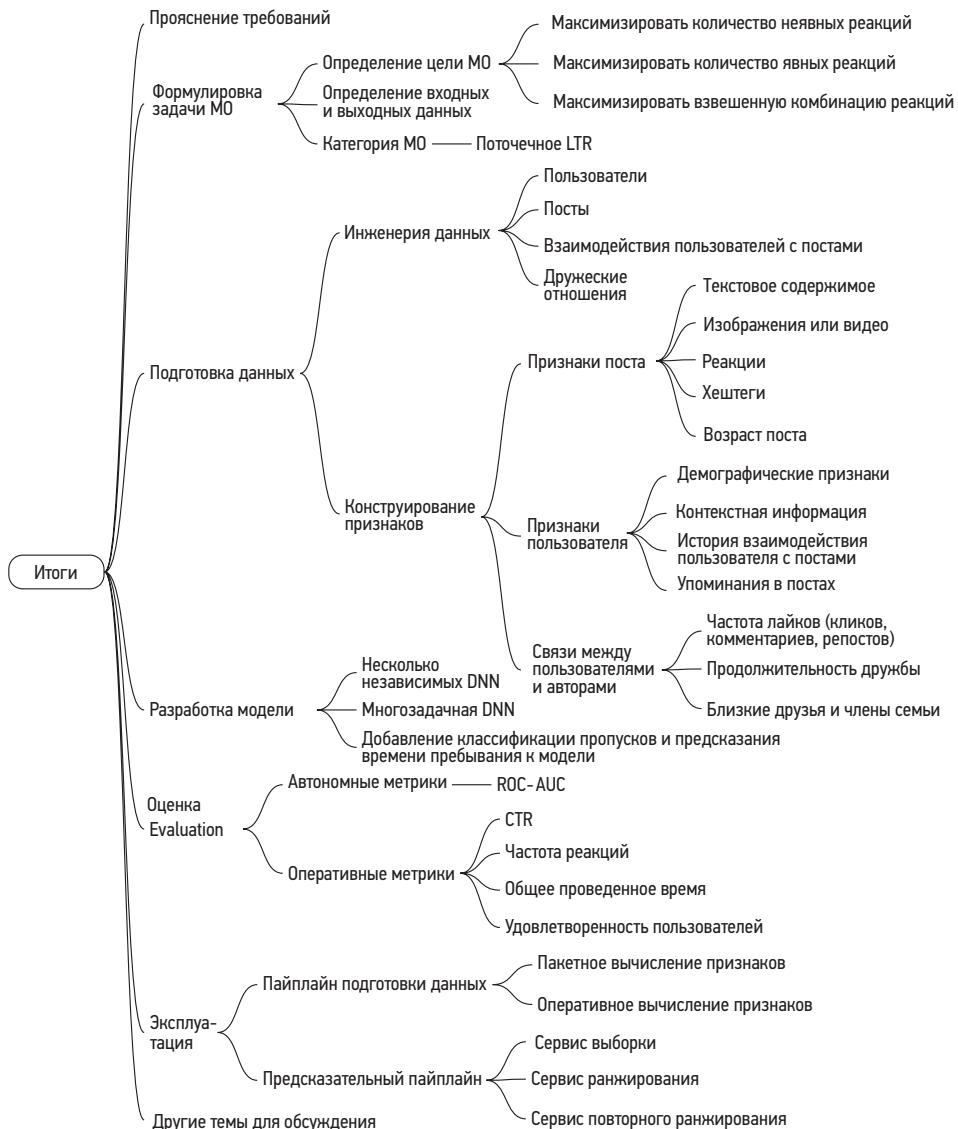
Этот компонент модифицирует список постов в соответствии с дополнительной логикой и пользовательскими фильтрами. Например, если пользователь явно выразил свой интерес к определенной теме (скажем, футболу), то сервис назначает соответствующим постам более высокий ранг.

Другие темы для обсуждения

Если в конце собеседования еще остается время, можно обсудить дополнительные темы.

- Что делать с постами, которые стали вирусными [15].
- Как персонализировать ленту новостей для новых пользователей [16].
- Как справиться с позиционным смещением в системе [17].
- Как определить правильную частоту повторного обучения [18].

Итоги



Ссылки

- [1] Ранжирование ленты новостей в Facebook. <https://engineering.fb.com/2021/01/26/ml-applications/news-feed-ranking/>
- [2] Лента новостей в Twitter. https://blog.twitter.com/engineering/en_us/topics/insights/2017/using-deep-learning-at-scale-in-twitters-timelines
- [3] Лента новостей в LinkedIn. <https://engineering.linkedin.com/blog/2020/understanding-feed-dwell-time>
- [4] BERT. <https://arxiv.org/pdf/1810.04805.pdf>
- [5] Модель ResNet. <https://arxiv.org/pdf/1512.03385.pdf>
- [6] Модель CLIP. <https://openai.com/blog/clip/>
- [7] Алгоритм Витерби. https://en.wikipedia.org/wiki/Viterbi_algorithm
- [8] TF-IDF. <https://en.wikipedia.org/wiki/Tf-idf>
- [9] Word2vec. <https://en.wikipedia.org/wiki/Word2vec>
- [10] Как обслуживать миллиард персонализированных лент новостей. <https://www.youtube.com/watch?v=Xpx5RYNTQvg>
- [11] Средняя абсолютная ошибка. https://en.wikipedia.org/wiki/Mean_absolute_error
- [12] Среднеквадратичная ошибка. https://en.wikipedia.org/wiki/Mean_squared_error
- [13] Функция потерь Хьюбера. https://en.wikipedia.org/wiki/Huber_loss
- [14] Проектирование ленты новостей. <https://liuzhenglaichn.gitbook.io/system-design/news-feed/design-a-news-feed-system>
- [15] Предсказание вирусных твитов. <https://towardsdatascience.com/using-data-science-to-predict-viral-tweets-615b0acc2e1e>
- [16] Проблема «холодного старта» в рекомендательных системах. [https://en.wikipedia.org/wiki/Cold_start_\(recommender_systems\)](https://en.wikipedia.org/wiki/Cold_start_(recommender_systems))
- [17] Позиционное смещение. <https://eugeneyan.com/writing/position-bias/>
- [18] Частота повторного обучения. <https://huyenchip.com/2022/01/02/real-time-machine-learning-challenges-and-solutions.html#towards-continual-learning>

11 СПИСКИ ВОЗМОЖНЫХ ЗНАКОМЫХ

Введение

В список возможных знакомых (PYMK, People You May Know) входят люди, с которыми вам, возможно, захочется связаться, потому что вас что-то объединяет: например, общие друзья или общее место учебы или работы. Во многих социальных сетях, таких как Facebook, LinkedIn и Twitter, функциональность PYMK реализована с помощью МО.



Рис. 11.1. Список возможных знакомых

В этой главе мы спроектируем систему PYMK, сходную с той, которая используется в LinkedIn. Система принимает на вход пользователя и возвращает список потенциальных знакомых.

Прояснение требований

Вот типичный диалог между соискателем и экспертом:

Соискатель: Можно ли считать, что функциональность PYMK нужна для того, чтобы помогать пользователям обнаруживать потенциальные связи с другими людьми и расширять свою сеть контактов?

Эксперт: Да, это похоже на правду.

Соискатель: Чтобы рекомендовать потенциальные связи, приходится учитывать огромный набор факторов: место жительства, образование, опыт работы, существующие связи, предыдущую активность и т. д. Можно ли сосредоточиться на самых важных факторах — например, образовании, опыте работы и социальном контексте пользователя?

Эксперт: Да, пожалуй.

Соискатель: В LinkedIn два человека считаются друзьями в том и только том случае, если каждый из них находится в списке друзей другого. Я правильно понимаю?

Эксперт: Да, дружба симметрична. Когда один пользователь отправляет другому заявку в друзья, получатель должен принять заявку, чтобы связь была установлена.

Соискатель: Сколько всего пользователей на платформе? Сколько среди них ежедневно активных пользователей?

Эксперт: У нас почти миллиард пользователей, из которых 300 миллионов ежедневно активны.

Соискатель: Сколько связей у пользователя в среднем?

Эксперт: Около тысячи.

Соискатель: У большинства пользователей социальный граф не слишком динамичен; иначе говоря, их связи обычно мало меняются за короткий промежуток времени. Можно ли ориентироваться на это при проектировании РУМК?

Эксперт: Верно подмечено. Да, это разумное предположение.

Резюмируем описание проблемы. Требуется спроектировать систему РУМК, сходную с используемой на LinkedIn. Система принимает на вход пользователя и возвращает список потенциальных знакомых. Система разрабатывается для того, чтобы пользователям было проще находить новые связи и расширять свои сети контактов. На платформе примерно 1 миллиард пользователей, и у каждого пользователя в среднем 1000 связей.

Формулировка проблемы в виде задачи МО

Определение цели МО

Типичная цель МО в системах РУМК — максимизировать количество связей между пользователями. Это помогает пользователям быстро расширять свои сети контактов.

Определение входных и выходных данных системы

На вход система РУМК принимает пользователя, а на выходе возвращает список связей, ранжированный по релевантности для пользователя. Общая схема показана на рис. 11.2.

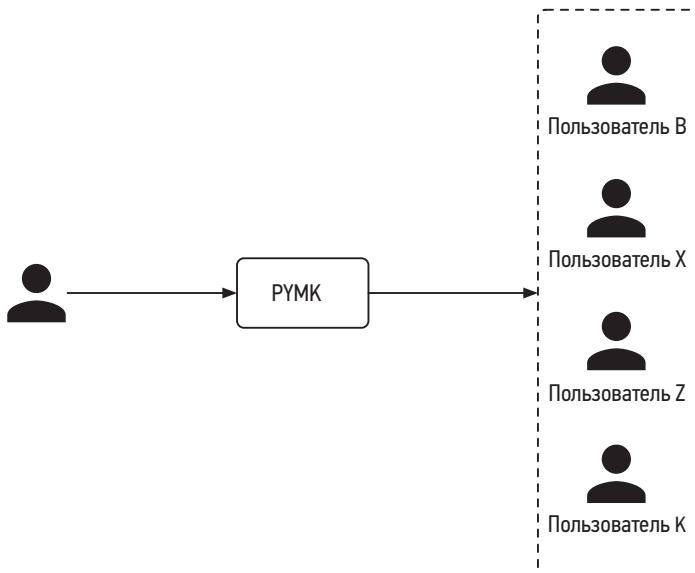


Рис. 11.2. Входные и выходные данные системы РУМК

Выбор категории МО

Рассмотрим два метода, с помощью которых обычно разрабатываются РУМК: поточечный метод обучения ранжированию (LTR) и предсказание связей.

Поточечное LTR

Здесь РУМК формулируется как задача ранжирования, а для ранжирования пользователей используется поточечное LTR. В этом методе, как показано на рис. 11.3, применяется модель бинарной классификации, которая принимает на вход двух пользователей и предсказывает вероятность того, что между ними есть связь.

Однако у этого метода есть крупный недостаток: он не учитывает доступный социальный контекст. Хотя такой подход упрощает ситуацию, точность предсказаний снижается из-за того, что отбрасывается информация о связях пользователей.

Вероятность того, что между пользователями есть связь

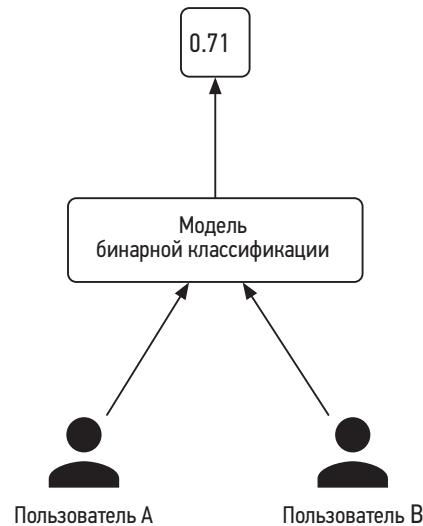


Рис. 11.3. Бинарная классификация с двумя пользователями на входе

Следующий пример демонстрирует, как социальный контекст предоставляет очень важную информацию. Допустим, мы хотим предсказать, образует ли пара \langle пользователь A , пользователь B \rangle потенциальную связь.

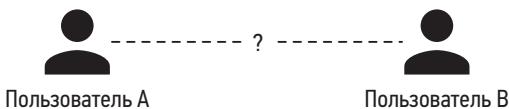


Рис. 11.4. Есть ли потенциальная связь между пользователями A и B ?

Просматривая окружение пользователей в пределах одного шага (связи пользователя A или пользователя B), можно получить больше информации, которая помогает определить, образует ли пара \langle пользователь A , пользователь B \rangle потенциальную связь. Рассмотрим два сценария, изображенные на рис. 11.5.

В сценарии 1 у каждого из пользователей A и B есть по четыре взаимные связи; кроме того, существуют взаимные связи между пользователями C, D, E и F .

В сценарии 2 у каждого из пользователей A и B есть по два друга, но друзья A не связаны с друзьями B и наоборот.

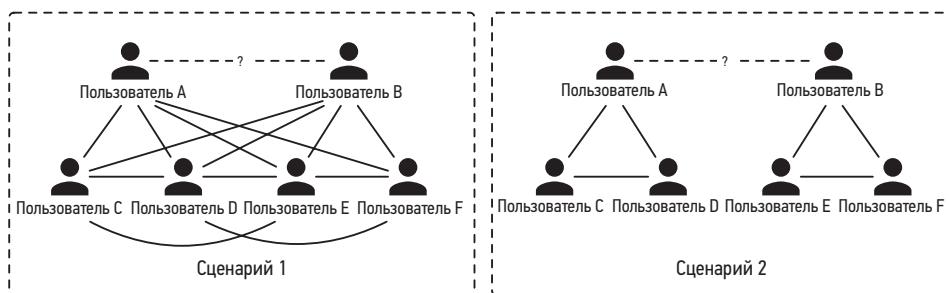


Рис. 11.5. Два разных сценария окружения в пределах одного шага

Анализируя одношаговое окружение, можно ожидать, что в сценарии 1 между элементами пары (пользователь A, пользователь B) с большей вероятностью есть связь, чем в сценарии 2. На практике можно даже рассматривать окружение в пределах двух или трех шагов, чтобы извлечь из социального контекста больше полезной информации.

Прежде чем обсуждать второй подход (предсказание связей), давайте разберемся, как в графах хранятся структурированные данные (например, социальный контекст) и какие задачи МО можно решать с помощью графов.

В общем случае граф представляет связи (ребра) между набором сущностей (узлов). Весь социальный контекст можно выразить в виде графа, в котором каждый узел соответствует пользователю, а ребро между двумя узлами соответствует связи между двумя пользователями. На рис. 11.6 изображен простой граф с четырьмя узлами и тремя ребрами.

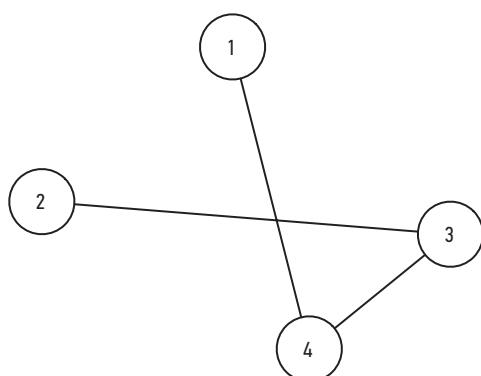


Рис. 11.6. Простой граф

Со структурированными данными, представленными в виде графов, можно выполнять предсказательные задачи трех типов.

- **Предсказание на уровне графа.** Например, если в виде графа представлено химическое соединение, можно предсказать, является ли оно ферментом.
- **Предсказание на уровне узла.** Например, по графу социальной сети можно предсказать, является ли конкретный пользователь (узел) спамером.
- **Предсказание на уровне ребра.** Эта задача позволяет узнать, существует ли ребро между двумя узлами. Например, по графу социальной сети можно предсказать вероятность того, что между двумя пользователями есть связь.

Рассмотрим метод предсказания связей для системы PYMK.

Предсказание связей

В этом методе модель снабжается информацией графа. В результате она может воспользоваться дополнительными сведениями из социального графа, чтобы предсказывать существование связи (ребра) между двумя узлами.

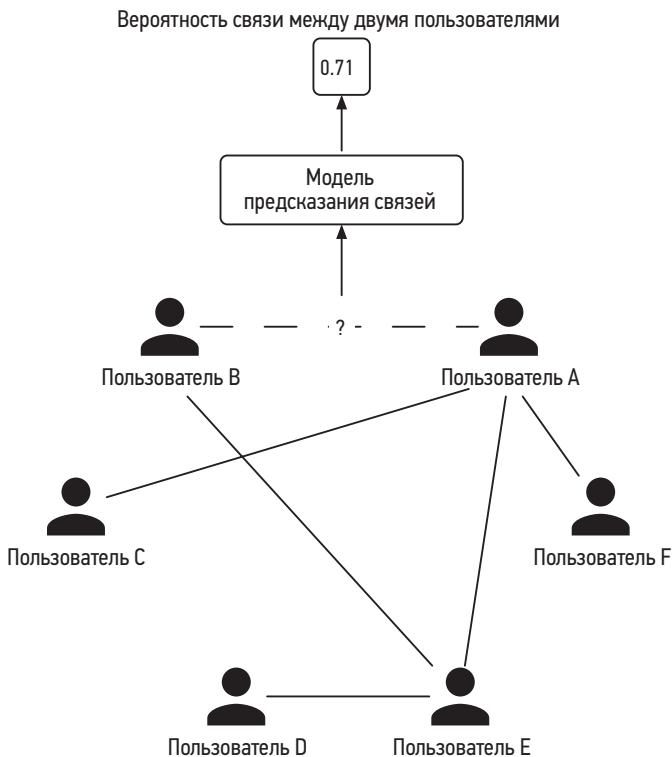


Рис. 11.7. Бинарная классификация с графом на входе

Говоря более формально, мы используем модель, которая принимает на вход весь социальный граф и предсказывает вероятность того, что между двумя конкретными узлами есть связь. Список потенциальных знакомых пользователя A будет ранжироваться по вероятности существования связей между пользователем A и другими пользователями.

Чтобы предсказать, существует ли связь между двумя узлами, модель использует не только типичные признаки, но и дополнительную информацию, извлеченную из социального графа.

Подготовка данных

Инженерия данных

В этом разделе рассматриваются доступные исходные данные:

- пользователи;
- связи;
- взаимодействия.

Пользователи

Кроме демографических данных пользователей, также доступна информация об их образовании, опыте работы, квалификации и т. д. В табл. 11.1 представлен пример данных об образовании пользователей. В аналогичных таблицах хранится информация об опыте работы, квалификации и т. д.

Таблица 11.1. Данные об образовании пользователей

ID пользователя	Учебное заведение	Степень	Специализация	Начало обучения	Конец обучения
11	Университет Ватерлоо	Магистр	Информатика	август 2015	май 2017
11	Гарвардский университет	Магистр	Физика	май 2004	август 2006
11	Калифорнийский университет в Лос-Анджелесе	Бакалавр	Электротехника	сентябрь 2022	–

Одна из проблем при работе с исходными данными такого типа заключается в том, что один и тот же атрибут может быть представлен в разных формах. Например, у строк «информационные технологии» и «IT» одинаковый смысл, но разные текстовые представления. Значит, на этапе инженерии данных важ-

но стандартизировать исходные данные, чтобы не получилось так, что разные формы одного атрибута обрабатываются по-разному.

Стандартизовать данные можно несколькими способами, например:

- предложить пользователям выбирать атрибуты из заранее определенного списка;
- группировать разные представления атрибута с помощью эвристик;
- группировать похожие атрибуты с помощью методов на базе МО (например, кластеризации [1]) или языковых моделей.

Связи

Упрощенный пример данных о связях представлен в табл. 11.2. Каждая строка представляет связь между двумя пользователями и время ее формирования.

Таблица 11.2. Данные о связях

ID пользователя 1	ID пользователя 2	Временная метка создания связи
28	3	1658451341
7	39	1659281720
11	25	1659312942

Взаимодействия

Существуют разные типы взаимодействий: пользователь отправляет заявку в друзья, принимает заявку, подписывается на другого пользователя, что-то ищет, просматривает профиль, ставит лайк или еще как-то реагирует на пост и т. д. На практике данные о взаимодействиях могут храниться в разных базах данных, но для простоты будем считать, что вся информация находится в одной таблице.

Таблица 11.3. Данные о взаимодействиях

ID пользователя	Тип взаимодействия	Значение взаимодействия	Временная метка
11	Заявка в друзья	user_id_8	1658450539
8	Заявка принята	user_id_11	1658451341
11	Комментарий	[user_id_4, Очень содержательно]	1658451365
4	Поиск	«Александр Пушкин»	1658435948
11	Просмотр профиля	user_id_21	1658451849

Конструирование признаков

Чтобы выявить потенциальные связи пользователя, модель должна задействовать его данные: возраст, пол и т. д. Кроме того, может быть полезной информация о его связях с другими пользователями. В этом разделе обсуждаются некоторые важные признаки.

Признаки пользователя

Демографические признаки: возраст, пол, город, страна и т. д.

Демографические данные помогают определить, насколько вероятна связь между двумя пользователями. Пользователи склонны устанавливать связи с теми, кто обладает сходными демографическими признаками.

В демографических данных часто встречаются пропущенные значения. Чтобы больше узнать о том, как их обрабатывать, обращайтесь к главе 1 «Введение и общие сведения».

Количество связей, подписчиков и запросов, ожидающих рассмотрения

Эта информация важна, потому что пользователи охотнее устанавливают связи с пользователями, у которых много подписчиков или связей.

Возраст учетной записи

Учетные записи, созданные совсем недавно, менее надежны, чем те, которые существуют уже долгое время. Например, если учетная запись создана вчера, то она с высокой вероятностью может использоваться для рассылки спама, а следовательно, не стоит поспешно рекомендовать ее пользователям.

Количество полученных реакций

Это числовые значения, которые представляют общее количество полученных реакций (лайков, репостов, комментариев и т. д.) за определенный период времени — например, за неделю. Пользователи склонны устанавливать связи с более востребованными пользователями — то есть такими, которые получают больше реакций от других пользователей.

Общие признаки пользователей

По общим признакам пользователей обычно удается предсказать, есть ли между ними связь. Рассмотрим некоторые из этих признаков.

Общее место учебы или работы

- Общая школа.** Пользователи склонны создавать связи с теми, с кем они учились в одной школе.

- **Учеба в одно и то же время.** Например, пользователь может захотеть подружиться в социальной сети с другим пользователем, который учился в школе X тогда же, когда и он сам.
- **Общая специализация.** Бинарный признак, который показывает, что во время учебы у двух пользователей была одинаковая профильная дисциплина.
- **Количество общих мест работы.** Пользователи склонны создавать связи с людьми, с которыми работали в одних компаниях.
- **Общая отрасль.** Бинарный признак, который показывает, что два пользователя работали или продолжают работать в одной отрасли.

Социальная близость

- **Посещения профиля.** Сколько раз один пользователь просматривал профиль другого.
- **Количество общих связей.** Если у двух пользователей много общих связей, они с большей вероятностью установят связь между собой. Это один из важнейших предсказательных признаков [2].
- **Общие связи с поправкой на время.** Этот признак назначает общим связям веса в зависимости от того, как долго они существуют. Чтобы понять, чем полезен этот признак, рассмотрим пример.

Допустим, вы хотите узнать, может ли пользователь B быть связан с пользователем A . Рассмотрим два сценария: в сценарии 1 связи пользователя A были сформированы совсем недавно, а в сценарии 2 — давно. Схема представлена на рис. 11.8.

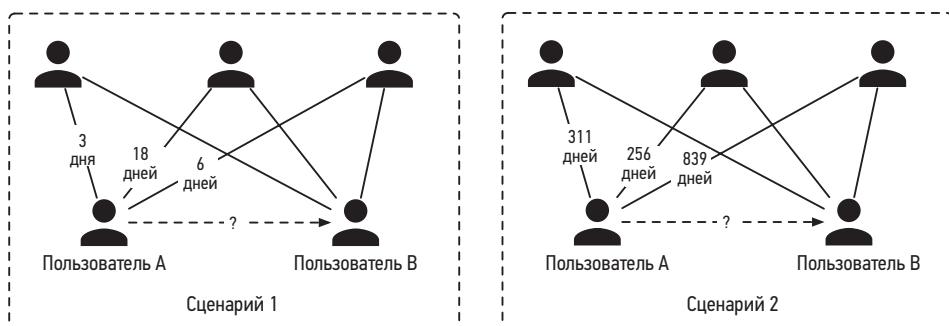


Рис. 11.8. Сравнение недавних связей со старыми

В сценарии 1 сеть пользователя A выросла недавно; это значит, что он с большей вероятностью подружится с пользователем B . А в сценарии 2 более вероятно, что пользователь A знает о пользователе B , но решил не устанавливать связь.

Разработка модели

Выбор модели

Ранее мы сформулировали проблему РУМК как задачу предсказания связей, где модель принимает на вход социальный граф и предсказывает вероятность того, что между двумя пользователями есть связь. Чтобы решить задачу предсказания связей, мы выбираем модель, которая способна обрабатывать входные данные в формате графа. Для этого предназначены графовые нейронные сети (GNN). Присмотримся к ним повнимательнее.

Графовые нейронные сети (GNN)

GNN — нейронные сети, которые можно напрямую применять к графикам. Они предоставляют простой способ решать предсказательные задачи на уровне графа, узла и ребра.

Как показано на рис. 11.9, GNN принимает на вход график, который содержит атрибуты, связанные с узлами и ребрами. Например, в узлах может храниться такая информация, как возраст, пол и т. д., а в ребрах — характеристики связей между пользователями: количество общих учебных заведений и мест работы, продолжительность дружбы и т. д. На основе входного графа и связанных с ним атрибутов GNN генерирует эмбеддинги для каждого узла.

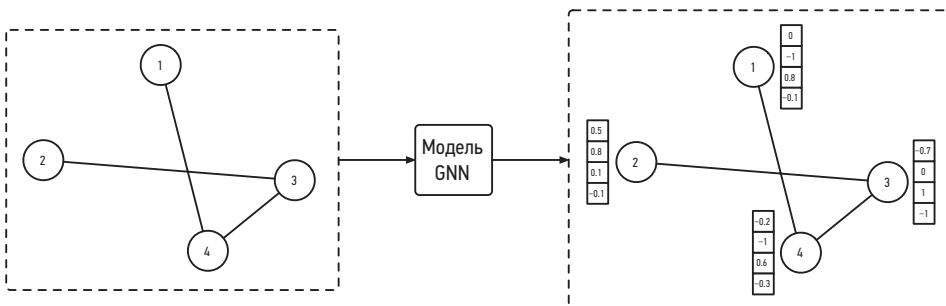


Рис. 11.9. Модель GNN генерирует эмбеддинги для каждого узла графа

После того как эмбеддинги узлов сгенерированы, на их основе модель предсказывает, с какой вероятностью два узла образуют связь. Это делается с помощью метрики сходства — такой, как скалярное произведение. Например, как показано на рис. 11.10, мы вычисляем скалярное произведение эмбеддингов узлов 2 и 4, чтобы предсказать, существует ли ребро, соединяющее эти два узла.

В последние годы было разработано много архитектур на базе GNN: например, GCN [3], GraphSAGE [4], GAT [5] и GIT [6]. Эти архитектуры устроены по-

разному и характеризуются разными уровнями сложности. Чтобы определить, какая архитектура лучше подойдет для ваших целей, потребуются основательные эксперименты. Если вас интересуют подробности про GNN, обращайтесь к [7].

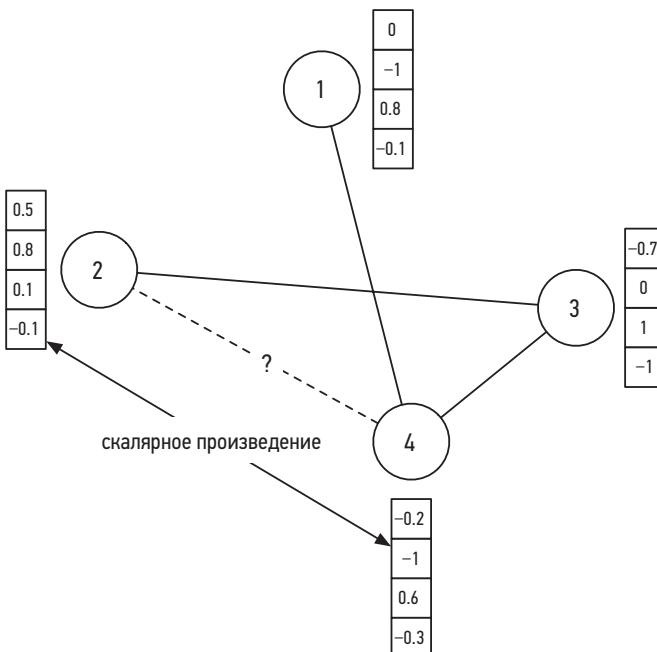


Рис. 11.10. Предсказание вероятности существования ребра между узлами 2 и 4

Обучение модели

Чтобы обучить модель GNN, мы предоставим ей моментальный снимок социального графа в момент времени t . Модель будет предсказывать связи, которые образуются к моменту времени $t + 1$. Рассмотрим, как построить обучающий набор данных.

Построение датасета

Датасет строится по следующей схеме:

1. Создать снимок графа в момент времени t .
2. Вычислить исходные признаки узлов и ребер графа.
3. Создать метки.

1. Снимок графа в момент времени t . На первом шаге построения обучающих данных создаются входные данные для модели. Поскольку модель GNN ожидает

получить на входе социальный граф, мы создаем его снимок в момент времени t по доступным исходным данным. На рис. 11.11 показан пример графа в момент времени t .

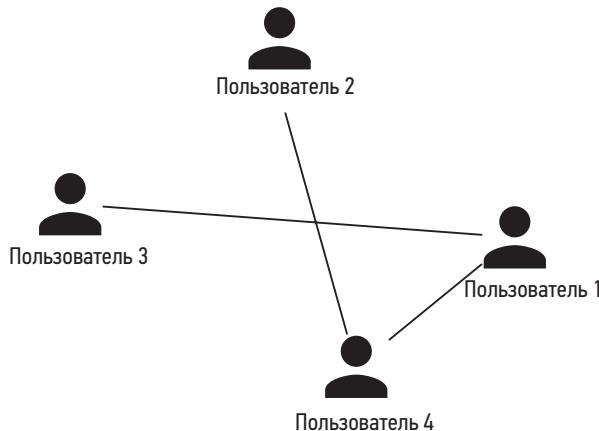


Рис. 11.11. Снимок социального графа в момент времени t

2. Исходные признаки узлов и ребер графа. Как показано на рис. 11.12, мы извлекаем такие признаки пользователей, как возраст, пол, возраст учетной записи, количество связей и т. д. Они используются как векторы исходных признаков узлов.

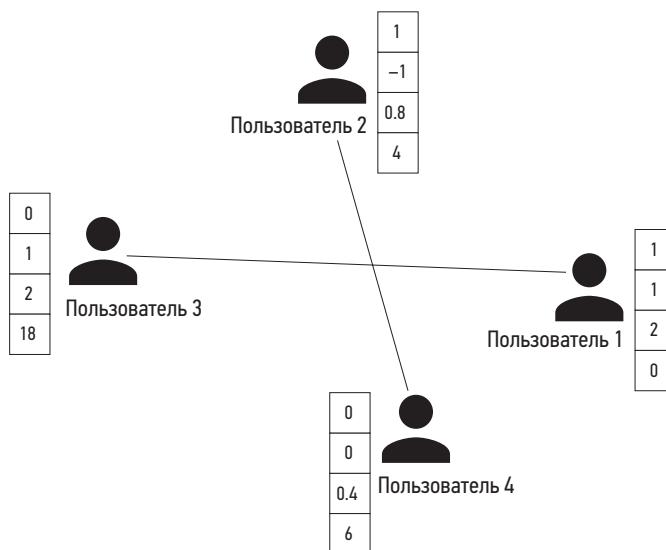


Рис. 11.12. Исходные признаки узлов

Похожим образом извлекаются признаки общности пользователей, которые применяются как векторы исходных признаков ребер. Как показано на рис. 11.13, между пользователем 2 и пользователем 4 есть связь. $E_{2,4}$ представляет вектор исходных признаков: он содержит такую информацию, как количество общих связей, посещения профилей, общий период учебы и т. д.

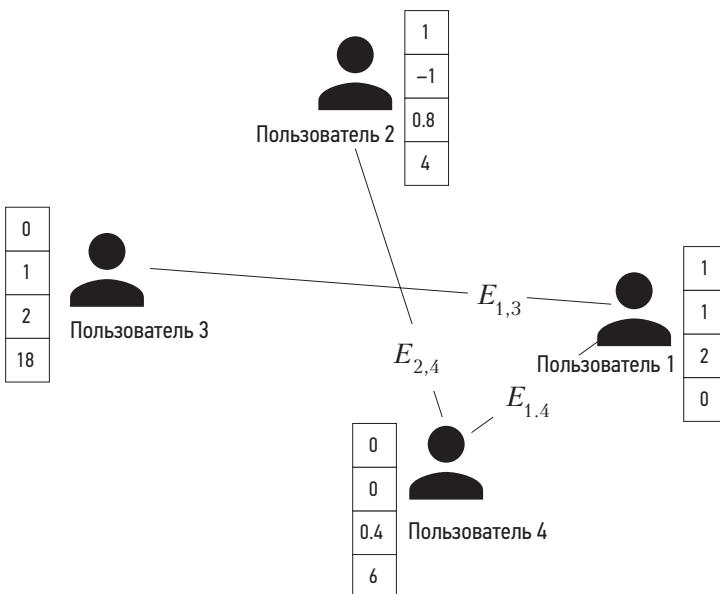


Рис. 11.13. Исходные признаки ребер

3. Метки. На этом шаге создаются метки, которые должна предсказывать модель. Чтобы определить положительные и отрицательные метки, используется снимок графа в момент времени $t + 1$. Рассмотрим конкретный пример.

Как показано на рис. 11.14, положительные и отрицательные метки создаются в зависимости от того, формируется ли новое ребро к моменту времени $t + 1$. Пара узлов помечается как положительная, если между ними появилась связь к моменту $t + 1$, и как отрицательная в противном случае.

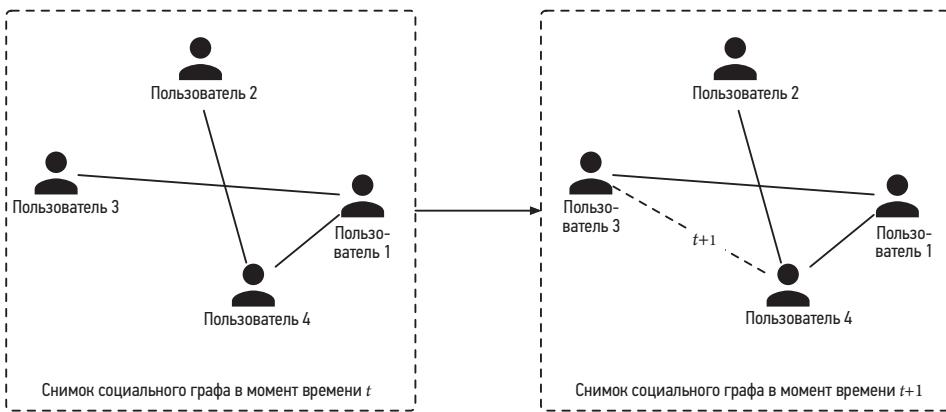


Рис. 11.14. Вновь сформированные ребра между моментами времени t и $t + 1$

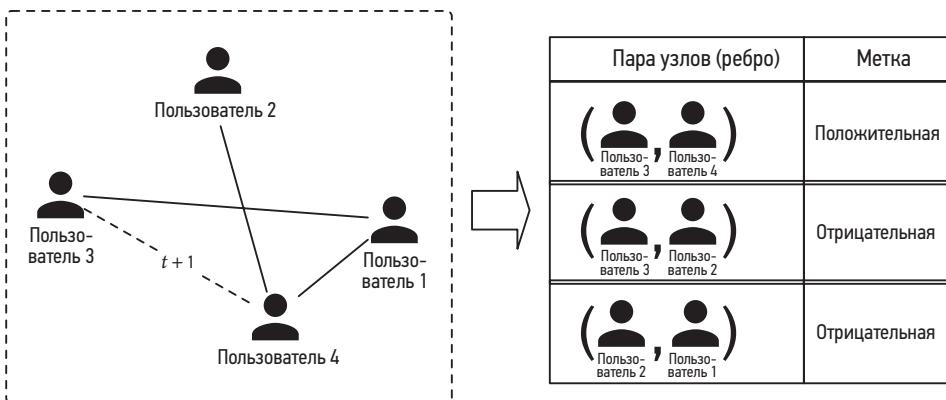


Рис. 11.15. Создание положительных и отрицательных меток

Выбор функции потерь

После того как входной график и метки созданы, можно переходить к обучению модели GNN. Подробное объяснение того, как обучаются модели GNN и какие функции потерь при этом применяются, выходит за рамки этой книги. Чтобы больше узнать об этом, обращайтесь к [7].

Оценка

Автономные метрики

В ходе автономной оценки мы оцениваем качество модели GNN и системы PYMK.

Модель GNN

Так как модель GNN предсказывает существование ребер, ее можно рассматривать как модель бинарной классификации. Чтобы измерить ее качество, будем использовать метрику ROC-AUC.

Система PYMK

В предыдущих главах мы подробно обсуждали, как выбирать подходящие автономные метрики для систем ранжирования и рекомендательных систем, поэтому здесь не будем углубляться в детали. В нашей системе пользователь либо устанавливает рекомендованную связь, либо отказывается от нее. Метрика mAP может хорошо подойти из-за своей бинарной природы (связь либо есть, либо нет).

Оперативные метрики

Чтобы оценивать эффект от систем PYMK, реальные компании отслеживают множество оперативных метрик. Рассмотрим две важнейшие метрики из этой категории.

- Общее количество заявок в друзья, отправленных за последние X дней.
- Общее количество заявок в друзья, принятых за последние X дней.

Общее количество заявок в друзья, отправленных за последние X дней. Метрика помогает понять, повышает ли модель количество заявок в друзья или снижает его. Например, если модель приводит к повышению количества заявок на 5 %, можно считать, что она положительно влияет на бизнес-цель.

Однако у этой метрики есть крупный недостаток. Новая связь образуется между двумя пользователями только в том случае, если получатель примет заявку в друзья. Например, пользователь может отправить 1000 заявок, но получатели примут лишь небольшой их процент. Поэтому эта метрика может недостаточно точно отражать реальный рост сети пользователей. Чтобы бороться с этим недостатком, можно воспользоваться следующей метрикой.

Общее количество заявок в друзья, принятых за последние X дней. Поскольку новая связь формируется только в том случае, если получатель при-

нимает заявку отправителя, эта метрика точно отражает реальный рост сети пользователей.

Эксплуатация

В режиме эксплуатации система PYMK эффективно рекомендует пользователю список потенциальных связей. В этом разделе мы объясним, почему нужно оптимизировать быстродействие, а также продемонстрируем некоторые приемы, которые делают PYMK эффективнее. Затем будет представлена архитектура, в которой разные компоненты совместно работают над обслуживанием запросов.

Эффективность

Как упоминалось в разделе, посвященном сбору требований, на платформе имеется 1 миллиард пользователей. Это значит, что для поиска потенциальных связей одного пользователя необходимо отсортировать 1 миллиард эмбеддингов. Ситуация усугубляется тем, что эту процедуру придется выполнять для каждого пользователя. Очевидно, в нашем масштабе это нереально. Чтобы решить эту проблему, обычно применяются два приема: 1) использовать друзей друзей и 2) предварительно вычислять PYMK.

Друзья друзей (FoF, friends of friends)

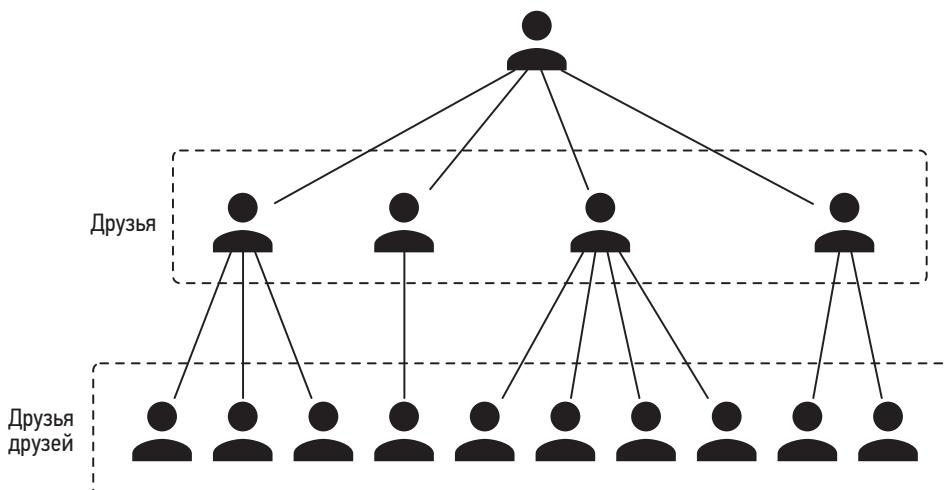


Рис. 11.16. Друзья друзей пользователя

По данным исследования Meta [2], 92 % новых дружеских связей формируется через друзей друзей. Этот прием сужает пространство поиска с помощью окружения FoF.

Как упоминалось ранее, у пользователя в среднем 1000 друзей. Это значит, что его окружение FoF в среднем состоит из 1 миллиона связей (1000×1000). Таким образом, пространство поиска сокращается с 1 миллиарда до 1 миллиона точек данных.

Предварительное вычисление РУМК

Сделаем шаг назад и рассмотрим, в каком режиме делать предсказания — в оперативном или в пакетном.

Оперативные предсказания

В случае РУМК оперативные предсказания заключаются в том, чтобы генерировать потенциальные связи в реальном времени, когда пользователь открывает главную страницу. При таком подходе не получится предоставить рекомендации для неактивных пользователей. Кроме того, если вычисление рекомендаций «на лету» будет занимать много времени, это ухудшит впечатления пользователя от системы.

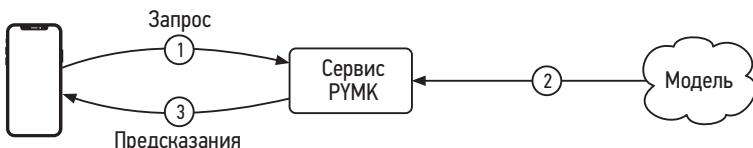


Рис. 11.17. Оперативное предсказание в РУМК

Пакетные предсказания

Пакетные предсказания заключаются в том, что потенциальные связи для всех пользователей вычисляются заранее и сохраняются в базе данных. Когда пользователь открывает главную страницу, предварительно вычисленные рекомендации загружаются напрямую, так что с точки зрения конечного пользователя они выдаются мгновенно. Недостаток пакетных предсказаний в том, что они могут привести к ненужным вычислениям. Представьте, что ежедневно платформу посещают 20 % ее пользователей. Если ежедневно генерировать ре-

комендации для каждого пользователя, то генерация 80 % рекомендаций будет вхолостую расходовать вычислительные мощности.



Рис. 11.18. Пакетное предсказание в PYMK

Какой вариант выбрать — оперативный или пакетный?

Мы рекомендуем пакетный подход по двум причинам. Во-первых, согласно требованиям, у платформы 300 миллионов ежедневно активных пользователей. Если вычислять PYMK для такого количества пользователей в реальном времени, это может замедлить работу системы и ухудшить пользовательский опыт.

Во-вторых, поскольку социальный граф в PYMK меняется достаточно медленно, предварительно вычисленные рекомендации остаются актуальными в течение некоторого времени. Например, можно хранить их в течение семи дней, а потом вычислять заново. Для новых пользователей временн^ие окно можно сократить (скажем, на один день), потому что их сети обычно растут быстрее.

Возможно, в социальной сети пользователь не захочет постоянно видеть один и тот же набор рекомендуемых друзей. Поэтому можно заранее вычислять связи, а потом отображать только те из них, которые пользователь не видел раньше.

Проектирование системы МО

На рис. 11.19 представлен дизайн системы машинного обучения PYMK. Она состоит из двух пайплайнов:

- пайpline генерации PYMK;
- предсказательный пайpline.

Рассмотрим эти пайплайны подробнее.

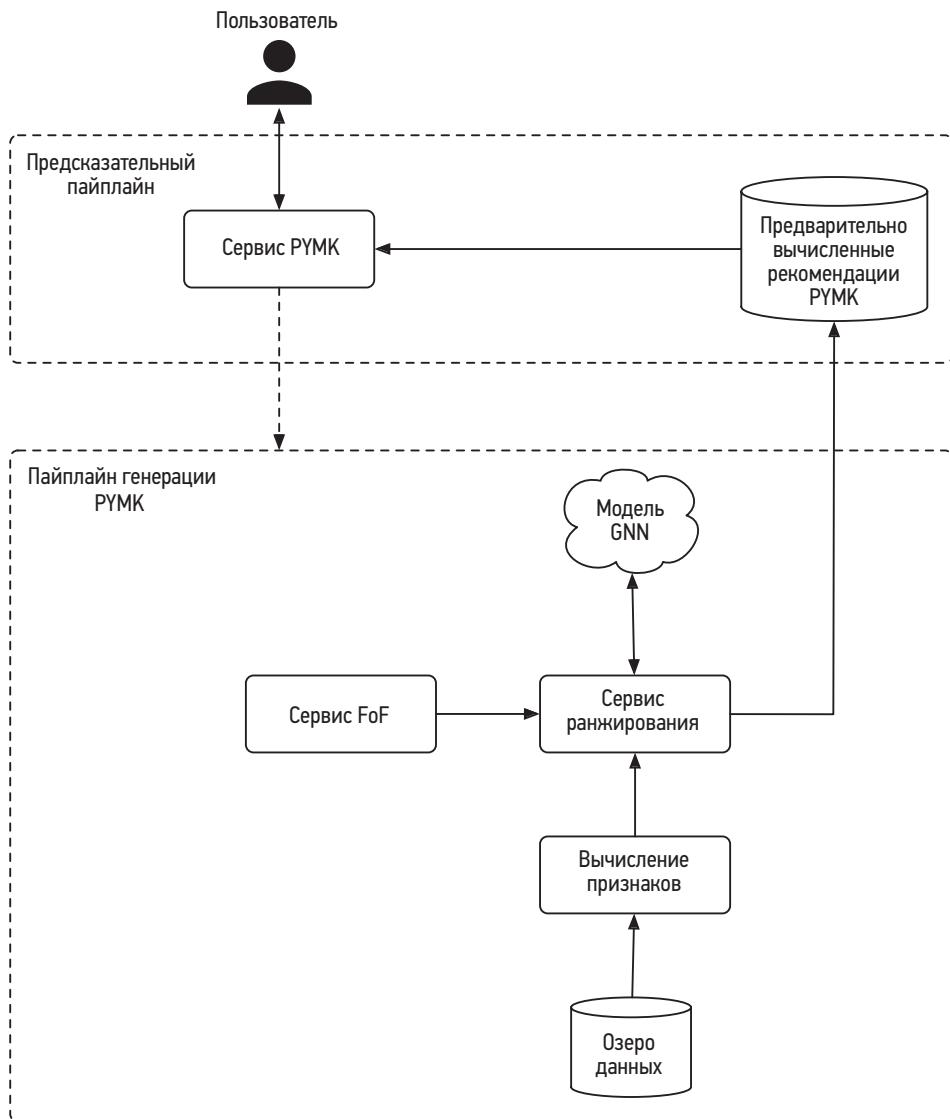


Рис. 11.19. Дизайн системы PYMK

Пайплайн генерации PYMK

Этот пайплайн генерирует PYMK для всех пользователей и сохраняет результаты в базе данных. Рассмотрим его подробнее.

Сначала для заданного пользователя сервис FoF сужает связи до подмножества кандидатов в друзья (связей через 2 шага), как показано на рис. 11.20.

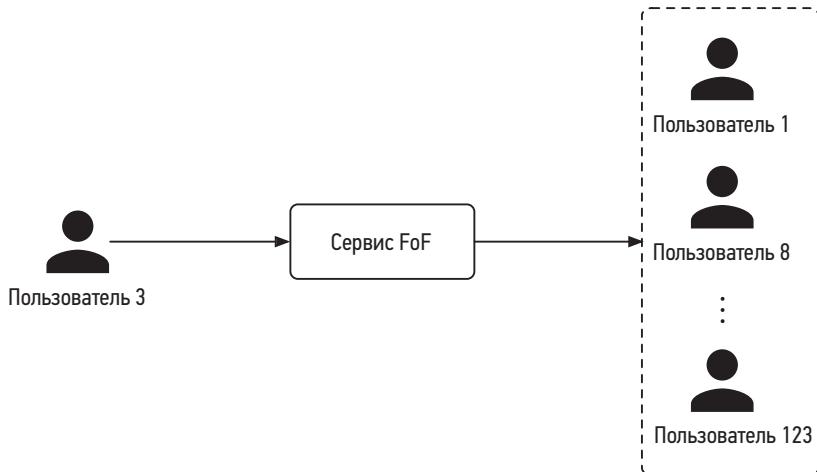


Рис. 11.20. Входные и выходные данные сервиса FoF

Затем сервис ранжирования получает потенциальные связи, которые выработал сервис FoF, присваивает каждой из них оценку с помощью модели GNN и генерирует ранжированный список PYMK для пользователя. PYMK сохраняется в базе данных. Когда система получает запрос от пользователя, можно просто загрузить его список PYMK из базы. Этот процесс проиллюстрирован на рис. 11.21.

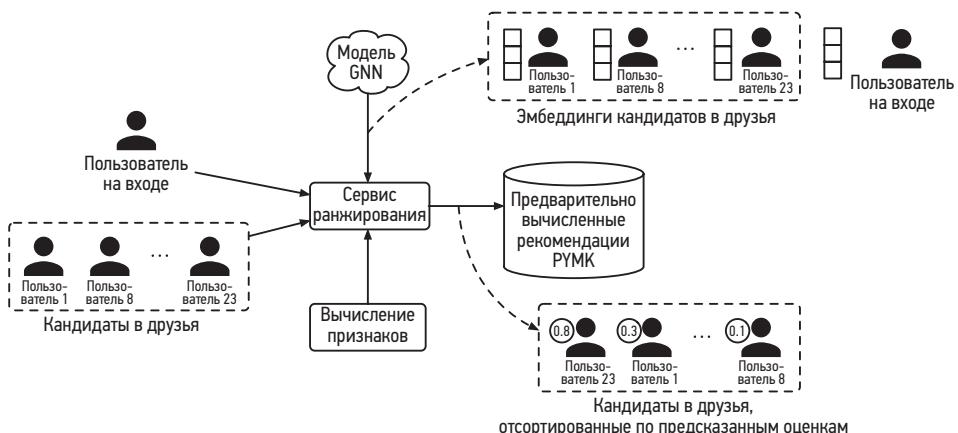


Рис. 11.21. Входные и выходные данные сервиса ранжирования

Предсказательный пайплайн

Когда поступает запрос, сервис PYMK сначала проверяет предварительно вычисленные результаты PYMK, чтобы узнать, есть ли готовые рекомендации. Если они есть, то они загружаются напрямую, а если нет, то сервис отправляет одноразовый запрос сервису генерации PYMK.

Обратите внимание, что мы описали упрощенную архитектуру. Если в ходе собеседования вам предложат ее оптимизировать, можно обсудить несколько возможных направлений.

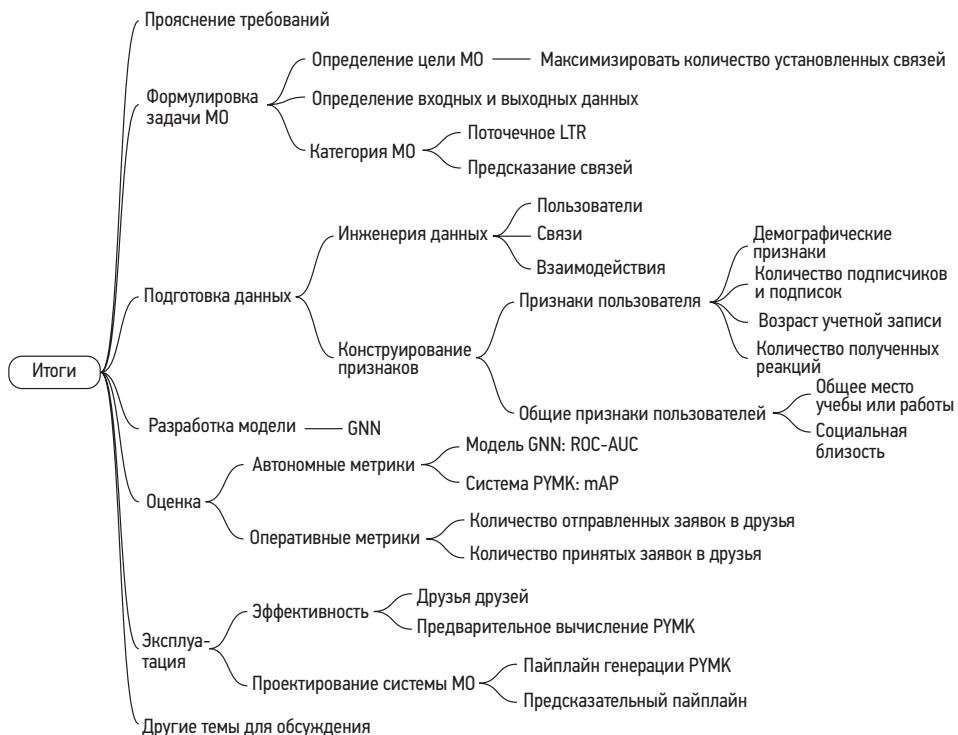
- Предварительно вычислять PYMK только для активных пользователей.
- Применять облегченный механизм ранжирования, чтобы сократить количество генерируемых кандидатов перед тем, как сервис ранжирования присвоит им оценки.
- Использовать сервис повторного ранжирования, чтобы разнообразить итоговый список PYMK.

Другие темы для обсуждения

Если в конце собеседования еще остается время, можно обсудить дополнительные темы.

- Персонализированное случайное блуждание [8] — еще один метод, с помощью которого часто создаются рекомендации. Благодаря своей эффективности он может пригодиться, чтобы построить базовое решение.
- Проблема смещения. Активные пользователи обычно шире представлены в обучающих данных, чем обычные пользователи. Из-за этой неравномерности в модели может возникнуть смещение в пользу одних групп в ущерб другим. Например, список PYMK будет чаще рекомендовать активных пользователей, отчего у них возникнет больше связей, и в результате они будут еще шире представлены в обучающих данных [9].
- Если пользователь раз за разом игнорирует рекомендуемых друзей, возникает вопрос о том, стоит ли учитывать их при будущем повторном ранжировании. В идеале игнорируемые рекомендации стоит ранжировать ниже [9].
- Пользователь может не отправить заявку в друзья сразу же, когда мы порекомендуем ему это сделать. На это может уйти несколько дней или недель. Когда же следует пометить рекомендуемую связь как отрицательную? Как в общем случае поступать с задержками обратной связи в рекомендательных системах [10]?

Итоги



Ссылки

- [1] Кластеризация в MO. <https://developers.google.com/machine-learning/clustering/overview>
- [2] PYMK в Facebook. <https://youtu.be/Xpx5RYNTQvg?t=1823>
- [3] Графовые сверточные нейронные сети. <http://tkipf.github.io/graph-convolutional-networks/>
- [4] GraphSage. <https://cs.stanford.edu/people/jure/pubs/graphsage-nips17.pdf>
- [5] Графовые сети с механизмом внимания. <https://arxiv.org/pdf/1710.10903.pdf>
- [6] Сеть изоморфизма графов. <https://arxiv.org/pdf/1810.00826.pdf>
- [7] Графовые нейронные сети. <https://distill.pub/2021/gnn-intro/>
- [8] Персонализированное случайное блуждание. https://www.youtube.com/watch?v=HbzQzUaJ_9I
- [9] Система PYMK в LinkedIn. <https://engineering.linkedin.com/blog/2021/optimizing-pymk-for-equity-in-network-creation>
- [10] Как бороться с задержкой обратной связи. <https://arxiv.org/pdf/1907.06558.pdf>

ПОСЛЕСЛОВИЕ

Поздравляем! Вы перевернули последнюю страницу этого пособия для соискателей. Важные навыки и знания, которые вы получили из этой книги, помогут вам проектировать сложные системы. Не каждому хватило бы выдержки достичь этого результата и узнать то, что вы теперь знаете. Поздравьте себя с заслуженным успехом: ваш нелегкий труд окупится.

Чтобы попасть на работу своей мечты, придется потратить немало времени и усилий. Практика — путь к совершенству. Удачи вам!

Спасибо за то, что вы купили и прочитали эту книгу. Без таких читателей, как вы, наша работа была бы невозможна. Надеемся, книга вам понравилась!

Если у вас появятся комментарии или вопросы, свяжитесь с нами по адресу hi@bytebytogo.com. Если вы обнаружите какие-нибудь ошибки — пожалуйста, сообщите нам о них, чтобы мы исправили их в следующем издании. Спасибо!

Алекс Сюй, Али Аминиан

System Design. Машинное обучение.
Подготовка к сложному интервью

Перевел с английского Е. Матвеев

Руководитель дивизиона	<i>Ю. Сергиенко</i>
Ведущий редактор	<i>Е. Строганова</i>
Научный редактор	<i>А. Афанасьев</i>
Литературный редактор	<i>Р. Чебыкин</i>
Художественный редактор	<i>Б. Мостапан</i>
Корректоры	<i>С. Беляева, Г. Шкатова</i>
Верстка	<i>Л. Егорова</i>

Изготовлено в России. Изготовитель: ООО «Прогресс книга».
Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург,
Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 12.2023. Наименование: книжная продукция. Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 — Книги печатные
профессиональные, технические и научные.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214, тел./факс: 208 80 01.

Подписано в печать 10.11.23. Формат 70×100/16. Бумага офсетная. Усл. п. л. 25,800. Тираж 700. Заказ 0000.

Алекс Сой

SYSTEM DESIGN. ПОДГОТОВКА К СЛОЖНОМУ ИНТЕРВЬЮ



Интервью по System Design (проектированию ИТ-систем) очень популярны у работодателей, на них легко проверить ваши навыки общения и оценить умение решать реальные задачи.

Пройти такое собеседование непросто, поскольку в проектировании ИТ-систем не существует единственно правильных решений. Речь идет о самых разнообразных реальных системах, обладающих множеством особенностей. Вам могут предложить выбрать общую архитектуру, а потом пройтись по всем компонентам или, наоборот, сосредоточиться на каком-то одном аспекте. Но в любом случае вы должны продемонстрировать понимание и знание системных требований, ограничений и узких мест.

Правильная стратегия и знания являются ключевыми факторами успешного прохождения интервью!

[КУПИТЬ](#)

Луис Серрано

ГРОКАЕМ МАШИННОЕ ОБУЧЕНИЕ



Машинное обучение — это набор методов анализа данных, основанных на алгоритмах, которые дают все более точные результаты по мере поступления новых данных. Машинное обучение лежит в основе систем рекомендаций, программ распознавания лиц, «умных» колонок и даже беспилотных автомобилей. Эта уникальная книга объясняет основные понятия машинного обучения на простых и доступных примерах, увлекательных упражнениях и запоминающихся иллюстрациях.

Здесь нет зубодробительного академического жаргона, для понимания объяснений достаточно знаний основ алгебры. По мере чтения вы будете создавать модели для идентификации спама, распознавания изображений и другие интересные проекты на языке Python.

Откройте для себя мощные методы машинного обучения, для понимания и применения которых достаточно знаний математики на уровне средней школы!

Для читателей, знающих основы языка Python. Знаний в области машинного обучения не требуется.

[КУПИТЬ](#)