

## Глава 7. Куда движется технология — эпоха единой модели никогда не наступит!

Это продолжение перевода книги Томас и др. Создание бизнес-ценности с генеративным ИИ. Эта мантра:

One Ring to rule them all,  
One Ring to find them,  
One Ring to bring them all,  
and in the darkness bind them.

Одно Кольцо, чтобы править всеми,  
Одно Кольцо, чтобы найти их,  
Одно Кольцо, чтобы собрать их всех  
и в темноте сковать их.

— культовая надпись на кольце из «Властелина колец» Дж. Р. Р. Толкина, вокруг которой строится вся история. Для настоящих толкинистов она звучит как вызов, а для всех остальных — объяснение: это кольцо якобы даёт своему владельцу власть над всеми. (Знаточи справедливо заметят, что не надпись даёт силу, и пустятся в рассуждения о Сауроне — но оставим это, как уже сказано, для настоящих гиков.) Полный контроль. Полное подчинение.

И вот в свете ажиотажа вокруг ChatGPT и стремительного выхода Generative AI в массовую культуру возникает вопрос: «Будет ли одна единственная LLM, которая подчинит себе все остальные?»

Спойлер: совсем нет. Даже близко нет. Как вы уже узнали из этой книги, на Hugging Face опубликованы почти полтора миллиона моделей (и, вероятно, ещё больше, когда вы читаете это). И если вы читали последовательно, вы уже можете уверенно объяснить разницу между «пользователями ценности» и «создателями ценности», понимаете основы этики ИИ и важность происхождения данных. Иными словами, вы уже знаете, почему одна модель не может управлять всем. Но мы всё равно дадим вам более развёрнутый ответ.

[Предыдущая глава](#)   [Содержание](#)   Следующая глава

Начнём с того, что даже в лабораториях, разрабатывающих самые передовые ИИ-модели, уже происходит сдвиг от идеи одной суперспособной модели к построению систем из моделей и техник, работающих вместе над задачей. В этой главе мы хотим сфокусировать ваше внимание на том, что происходит на рынке, какие тренды и технологические сдвиги формируют будущее генеративного ИИ. От стремительного прогресса в области малых моделей до маршрутизации между моделями и внутри них, до появления агентных систем — всё это указывает: не будет одной модели, чтобы управлять всеми.

**Чем больше, тем лучше? Может быть, на старте. Но это было давно.**

Если «технический» год равен собачьему (1:7), то в мире генеративного ИИ — мышинному (1:30). 2018 год — это уже более двух столетий назад по меркам GenAI! А что было тогда? OpenAI выпустила GPT-1 с «всего лишь» 117 миллионами параметров. В погоне за искусственным общим интеллектом (AGI) OpenAI выпускала всё более мощные версии GPT — некоторые из них содержали уже триллионы параметров и выполняли всё больше задач.

Не путайте AGI и GenAI. GenAI — это инструмент. AGI — цель, стремление развить этот инструмент до уровня, соответствующего (или превосходящего) человеческие когнитивные способности во множестве задач. Мы пока там не находимся — возможно, никогда не дойдём, а возможно, это случится быстрее, чем кажется. Но если речь идёт об AGI, то речь идёт о моделях, демонстрирующих и превосходящих «человеческий» уровень интеллекта.

Изначально казалось, что путь к лучшему качеству — просто делать модели больше. Как видно в таблице, между GPT-1 и GPT-3 масштаб модели увеличился более чем в 10 000 раз! После GPT-3 OpenAI перестала публиковать размеры моделей, но GPT-4 и GPT-4o, по слухам, могут содержать более триллиона параметров. С ростом моделей растёт и стоимость: маленькие модели обходятся менее чем в \$0,25 за миллион токенов (или вовсе бесплатно, если использовать их на ноутбуке через Ollama и аналоги), а большие — гораздо дороже. Например, по последним данным, один миллион токенов в OpenAI o1 стоит около \$60. И хотя цены стремительно меняются (в основном в лучшую

сторону), высокопроизводительные малые модели становятся всё более оправданным выбором с точки зрения бизнеса.

Название модели	Параметры	Интересные факты
OpenAI GPT-1	117 миллионов	Это «оригинал». Была лучше предыдущих технологий, но оказалась лишь началом чего-то гораздо большего.
GPT-2	~1 миллиард	Эта модель начала давать интересные завершения и доказала, что у обработки естественного языка (NLP) есть новый горизонт. До уровня ChatGPT она не дотягивала, но привлекла внимание СМИ, написав историю про единорогов. См. историю на сайте OpenAI.
GPT-3 GPT-3.5 GPT-3.5 Turbo	~175 миллиардов	GPT-3.5 была первой моделью, с которой дебютировал ChatGPT. Произошло два ключевых изменения по сравнению с GPT-2: модель научили следовать инструкциям (а не просто предсказывать следующее вероятное слово), и добавили пользовательский интерфейс. Этого было достаточно. GPT-3.5 также вышла в более эффективной версии под названием "Turbo".
GPT-4 GPT-4 turbo GPT-4o GPT-4o mini GPT-4.5	OpenAI прекратила публиковать количество параметров после GPT-3 (которая, как отмечалось, имела 175 миллиардов параметров). Различные источники предполагают, что GPT-4 имеет около 1.8 трлн параметров.	Четвертое поколение моделей принесло большую вычислительную мощность и мультимодальные возможности. На момент публикации GPT-4o считалась «флагманской» моделью OpenAI, а GPT-4.5 только вышла. GPT-5 еще не была выпущена, но предполагается, что она выйдет в середине 2025 года.
OpenAI o1 OpenAI o3 mini	(См. выше.)	Считаются отдельным проектом и не являются частью основного семейства GPT. Эти модели рассуждений обучены строить длинные цепочки размышлений перед ответом, что позволяет им решать более сложные задачи. Ожидается, что эта способность будет интегрирована в GPT-5.

Таблица 7.1. Семейство моделей GPT от OpenAI во времени

Возникает вопрос: действительно ли вашему бизнесу нужна такая вычислительная мощность? Даже OpenAI начала создавать более компактные и эффективные версии своих моделей. С каждым крупным выпуском модели появляется парная версия, которая работает эффективнее и обходится дешевле. GPT-3.5 — познакомьтесь с GPT-3.5 Turbo; GPT-4o — познакомьтесь с GPT-4o mini.

Последней моделью рассуждения, выпущенной OpenAI на момент публикации книги, стала OpenAI o3 mini. Хотя OpenAI изначально заявила о намерении выпустить OpenAI o3, позже они приостановили этот проект и объявили, что GPT-5 объединит лучшее из OpenAI o3 и всей серии моделей GPT. Глава

компания Сэм Альтман поделился целью «упростить продуктовую линейку» и «вернуться к магии унифицированного интеллекта».

Подводя итог: даже в лабораториях, прославившихся своими инновациями за счёт масштабирования моделей, сейчас основной вектор развития — это объединение нескольких моделей в единую систему, работающую над задачами совместно и реализующую концепцию «унифицированного интеллекта».

И несмотря на распространённое мнение, что чем больше модель — тем лучше, сегодня появляются всё более интересные решения, позволяющие создавать небольшие, но мощные языковые модели (SLMs — Small Language Models). Термин «SLM» пока не имеет строгого определения, но, как правило, к ним относят модели с числом параметров менее 13 миллиардов. В некоторых сценариях SLM-модели демонстрируют сопоставимую или даже более высокую производительность, чем LLM с 100+ миллиардами параметров.

### *Восхождение малых языковых моделей (SLMs)*

Возможно, проще всего описать феномен SLM как результат улучшения практики обучения моделей. Например, когда наши исследовательские группы впервые получили доступ к Llama-2-70B в июле 2023 года, их поразили её возможности. Спустя чуть больше года они смогли достичь тех же или лучших результатов, используя всего лишь 2-миллиардную версию модели Granite, по данным Open LLM v2 Leaderboard на Hugging Face (см. рис. 7.1).

	Rank	Type	Model	Average
♣	2438	🗨️	<a href="#">ibm-granite/granite-3.0-2b-instruct</a>	● 18.40 %
♣	2444	🟢	<a href="#">meta-llama/Llama-2-70b-hf</a>	● 18.37 %
♣	2445	🗨️	<a href="#">nvidia/Nemotron-Mini-4B-Instruct</a>	● 18.36 %
♣	2451	🗨️	<a href="#">gradientai/Llama-3-8B-Instruct-Gradient-1048k</a>	● 18.28 %
♣	2462	🟢	<a href="#">tiiuae/Falcon3-Mamba-7B-Base</a>	● 18.14 %

Рис. 7.1. Производительность моделей по данным Hugging Face Open LLM v2 Leaderboard, фев.2024

Это ещё одно проявление естественного эффекта обучения: как наши электромобили с каждым годом проезжают больше и заряжаются быстрее, как камеры на смартфонах получают всё больше пикселей и объективов, а телевизоры становятся тоньше, так и поставщики моделей накапливают опыт и совершенствуют подходы к обучению, делая модели всё более эффективными.

В следующих разделах мы рассмотрим наиболее перспективные стратегии, стоящие за стремительным ростом конкурентоспособных SLM: отбор и подготовка данных (Data Curation) и дистилляция моделей.

Это не случайность, что обе стратегии напрямую связаны с качеством данных, используемых для обучения и дообучения LLM. Многие удивляются, узнав, что прорывы в снижении размера моделей чаще обусловлены инновационными подходами к данным, а не архитектурными изменениями моделей. Не поймите нас неправильно: архитектурные новшества, безусловно, происходят — и в этой главе мы расскажем о некоторых из них. Но если взглянуть на молниеносный взлёт SLM (менее чем за год после релиза ChatGPT в ноябре 2022), причина очевидна: данные решают всё.

И мы подробно остановимся на этих подходах к данным, потому что в главе 8 покажем, как ваша компания может использовать те же самые техники, что применяют разработчики SLM, для создания ценности из ваших корпоративных данных.

Суть в следующем: у вас есть данные, и они не входят в состав обучающих датасетов LLM. Уверяем вас, многие поставщики LLM хотели бы получить к ним доступ. Почему 99% корпоративных данных не оказались в LLM? Потому что вы не выложили их в интернет. Сделайте свои данные вашим конкурентным преимуществом. В этом и заключается основная идея книги. И текущая, и следующая главы помогут вам это реализовать.

## *Отбор и подготовка (курирование) данных ведет к спасению ИИ*

Отбор большого объема качественных обучающих данных может сильно повлиять на производительность модели (или любого аналитического проекта, если на то пошло). Но именно акцент на курировании данных стал одной из ключевых причин высокой эффективности SLM-моделей и напрямую противоречит изначальной философии ранней разработки LLM: взять как можно больше грязных, неочищенных и неструктурированных данных и переработать их для питания LLM.

Как оказалось, в бизнесе по отношению к LLM необходимо идти на компромисс. Технология трансформеров сделала возможным использование больших объемов относительно «грязных» данных для создания LLM, но чем выше качество данных, тем выше качество модели. Задайте себе вопрос: есть ли у вас большие объемы качественных данных, специализированных под бизнес, который вам важен? Конечно, есть! А значит, вы готовы работать по-настоящему эффективно, ведь количество, качество и специализация — три ключевых компонента курирования данных, которые способствовали росту SLM.

### Количество данных

Сколько данных оптимально для модели заданного размера? Этот вопрос стал предметом обширных исследований в AI-сообществе, поскольку, как вы можете себе представить, обучение LLM сопряжено с высокими затратами — как финансовыми, так и экологическими. По этой причине поставщики моделей на начальном этапе сосредоточились на том, чтобы максимизировать производительность при минимальных начальных издержках. Важной частью этой оптимизации было определение того, сколько токенов (напомним, токен — это часть слова, целое слово или даже знак препинания) языковых данных следует использовать на каждый параметр модели. Эти соотношения — так называемые законы масштабирования (scaling laws) — описаны в научной литературе и определяют, сколько данных требуется для масштабирования модели.

В статье 2020 года команда исследователей OpenAI предположила, что на каждую единицу параметров следует использовать примерно два токена текста. Это соотношение 2:1 стало известно как закон масштабирования Каплана и использовалось при обучении моделей GPT-3 и BLOOM (обе имеют 175 миллиардов параметров и были обучены на 300–350 миллиардах токенов текста).

В 2022 году команда Google DeepMind опубликовала альтернативную точку зрения — закон масштабирования Чинчиллы (также известный как закон Хоффмана, в честь ведущего исследователя; Chinchilla — это семейство моделей, созданных DeepMind). Исследователи DeepMind считали, что OpenAI сильно недооценили объем данных, необходимый для оптимального обучения LLM. По их мнению, оптимальным является соотношение 20:1, а не 2:1. Они обучили LLM Chinchilla с 70 миллиардами параметров, следуя этому принципу.

Результат? При своих 70 миллиардах параметров Chinchilla показала производительность выше, чем более крупные модели вроде GPT-3 (175 миллиардов параметров). Оглядываясь назад, можно сказать, что Chinchilla была своего рода «первой» SLM (как говорят молодые — OG, от original). Эта модель всё еще довольно большая, но не запредельно.

Изначально сообщество исследователей стремилось определить законы масштабирования, чтобы оптимизировать фиксированные стартовые затраты на обучение моделей. Но что насчет переменных затрат на протяжении всего жизненного цикла модели? Сверхбольшая модель будет дороже в размещении и выводе (инференсе). И кто понесет эти расходы? Верно — вы. Чтобы сократить эти издержки, нужно уменьшить размер модели. А чтобы при этом сохранить производительность, нужно обучать ее на большем объеме (качественных) данных.

Именно поэтому SLM-модели сегодня получают столько внимания. Поскольку стоимость вывода и размещения напрямую ложится на потребителей, эффект был с отложенной реакцией. Поставщики моделей начали оптимизировать процесс обучения под эффективность вывода, а не только под эффективность обучения.

Чтобы создать модели, эффективные в инференсе, может быть экономически оправдано использовать соотношение данных к параметрам даже выше, чем то, что предполагал закон масштабирования Чинчиллы. На момент выхода этой книги научное сообщество так и не пришло к единому мнению о законе масштабирования для вывода (и, возможно, никогда не придет), но есть убедительные

примеры высокоэффективных SLM, обученных на значительно больших объемах данных, чем предполагают принципы Чинчиллы или Каплана (некоторые из этих законов масштабирования приведены в таблице 7.2).

В феврале 2023 года Meta открыла исходный код своей линейки моделей Llama 2, обученных примерно на 2 триллионах токенов (на тот момент это считалось огромным объемом данных). В этой серии модель с 7 миллиардами параметров имела коэффициент масштабирования почти 300:1! В августе 2024 года, с выпуском Llama 3, Meta пошла еще дальше и выпустила Llama3.1-8B. Эта модель была обучена на более чем 15 триллионах токенов и имела плотность данных почти 2000:1, показывая производительность выше, чем у линейки Llama 2. Тенденция очевидна: размер SLM практически не меняется, а производительность заметно растет — только за счет увеличения объема обучающих данных.

Дата	Количество обучающих токенов на параметр	Закон масштабирования
23.01.2020	1,7	Каплана
29.03.2022	20	Чинчиллы
01.02.2023	286	Llama-2-7B
01.08.2023	1875	Llama-3.1-8B

Таблица 7-2. Законы масштабирования во времени

В техническом документе, сопровождающем этот релиз — “The Llama 3 Herd of Models”, Meta указала, что её флагманская модель с 405 миллиардами параметров, также обученная на ~15 триллионах токенов, является «примерно оптимальной с точки зрения вычислений» для обучения, но её более компактные модели обучались «значительно дольше, чем это оптимально с точки зрения вычислений». Полученные модели показывают лучшие результаты по сравнению с вычислительно-оптимальными моделями при тех же затратах на вывод». Проще говоря, хотя эти меньшие модели были дороже в обучении (т.к. обучались дольше на большем объеме данных), они гораздо более эффективны при инференсе. В результате модели Llama 3 стали одними из самых популярных моделей с открытым исходным кодом, и ожидается, что Llama 4, когда она выйдет в 2025 году, будет не менее популярной.

Возвращаясь к SLM: при соотношениях данных, предполагающих сотни токенов на каждый параметр модели, инференс-оптимизированные модели и SLM начинают означать одно и то же. Создать крупную, оптимизированную для инференса LLM практически невозможно. Учитывая стоимость получения данных и объем доступных в мире данных, такие соотношения становятся слишком дорогими для поддержки обучения LLM с сотнями миллиардов параметров. У нас просто нет столько данных.

Существует реальный вопрос: когда мы достигнем потолка по данным? Современные модели обучаются на более чем 15 триллионах токенов, и чтобы достичь этого объема, провайдеры моделей фактически обошли весь интернет. Но, как будет показано в следующем разделе, нам не нужны большие объемы любых данных — нам нужны большие объемы очень качественных данных, а их получить ещё сложнее.

### Качество данных

Можете представить себе песню «Cecelia» без Гарфанкела, только с Саймоном? А смогли бы Hall & Oates записать «Kiss on My List», если бы не начали её с сочетания клавишных и типичных 80-х усов, кричащих: «Романтика на мне. Тебе осталось только нажать кнопку воспроизведения»? Хотя это и музыкальное отступление, так же трудно отделить влияние количества данных от влияния их качества в LLM. Качество данных и высокоэффективные модели идут рука об руку — как Саймон и Гарфанкел, Hall & Oates.

Если вы верите, что в интернете только достоверные данные, что в нем нет предвзятости, брани, ненависти, лжи или злости, то, вероятно, можно не читать эту книгу. Такая вера — как съесть по ведру мороженого в день и удивляться, почему джинсы вдруг сжались, ведь вы стирали их только в холодной воде. В мире GenAI по-прежнему действует принцип: что вложишь, то и получишь. Чем больше вы сможете сделать для курирования данных, использованных при обучении модели (и по объему, и по качеству размеченных примеров), тем выше будет производительность модели. И хотя существуют техники для повышения качества модели после обучения — например, дополнение извлечением (RAG) и другие — все они выигрывают от высокого качества исходных данных.

Microsoft открыто заявляет, что качество данных сыграло критическую роль в создании её (на тот момент) передовой модели Phi-2 с 2.7 миллиардами параметров, которая в ряде тестов превзошла модели в 25 раз большего размера. Но стало понятно, что Microsoft увидела это направление заранее, представив предшественника (Phi-1) в исследовательской публикации “Textbooks Are All You Need”.

В этой статье Microsoft описала, как «качественные данные могут улучшить даже лучшие LLM при одновременном значительном сокращении объема обучающего датасета и вычислительных затрат». И так же, как люди лучше учатся по структурированным учебникам, результаты Microsoft подтверждают, что данные с качеством характерным для учебников, которые являются «понятными, самодостаточными, поучительными и сбалансированными», приводят к лучшим моделям, демонстрирующим лучшие масштабируемые закономерности. Это позволило моделям уровня Phi-2 стать передовыми по своим характеристикам. На момент публикации книги Microsoft только что выпустила четвертую итерацию этой модели: Phi-4. Аналогично Phi-1 и Phi-2, Microsoft указала «улучшенные данные» (наряду с другими улучшениями в обучении) как ключевой фактор высокого качества Phi-4 при её компактных размерах.

Высокое качество данных критично для SLM. Многие разработчики моделей открыто говорят о количестве данных, использованных для обучения LLM, но лишь немногие из них раскрывают конкретные источники этих данных. На самом деле, если вы спросите у ведущих поставщиков LLM, какие данные они использовали, они либо не смогут вам ответить, либо ответят, что это не ваше дело. На что вы вполне справедливо можете возразить: «Это как раз моё дело!»

Суть в следующем: самые качественные датасеты — это длинные учебники или другие научно-популярные книги, написанные и защищённые авторским правом — а не посты с Reddit со средними оценками и выше или прочие неструктурированные источники. Качественные артефакты данных — это не случайные снимки веб-контента, размещённого в открытом доступе и собранного автоматически. Неприятная правда о многих популярных LLM заключается в том, что лучшие датасеты (например, Books310, который мы впервые упомянули в главе 5) попали в обучение, потому что были пиратски опубликованы без разрешения авторов.

Часть нашего собственного труда также оказалась всосана в недра множества LLM — на благо всех и на прибыль других. Нас не спросили. У нас не было выбора. Это просто произошло. И хотя мы не подаём в суд (мы ведь не написали бестселлер под названием «50 оттенков больших данных», за которым гоняется Голливуд), есть немало людей, чья работа и бизнес-уникальность были «украдены» ради создания LLM, которыми вы, скорее всего, уже пользуетесь. Всё это возвращает нас к разговору об обмене ценностью из главы «Как вы потребляете ИИ: вы создаёте ценность или пользуетесь ею?»

Только прозрачные политики по сбору и курированию данных позволяют убедиться, что LLM, которую вы рассматриваете для бизнеса, не использовала данные, полученные неэтичным путём. Вывод? При оценке SLM, где качество данных критично для производительности (не говоря уже о юридических последствиях), необходимо повышенное внимание к источникам данных. Задавайте вопросы. Требуйте ответы.

### Специализация по предметной области

Как человек, ведущий активный образ жизни, вы в выходные снова дома — с лодыжкой, которая подаёт смешанные сигналы: то ли она готовится к участию в реалити-шоу «Думаешь, ты сломал лодыжку», то ли просто драматизирует. В любом случае, она требует льда и внимания. Теперь вам нужно разобраться, что происходит. Для этого вы будете спрашивать самого умного человека, которого знаете, или врача? И хотя самый умный человек может быть талантлив во многих областях — от поэзии до химии и философии — лучше обратиться к врачу, а ещё лучше — к ортопеду. Пусть даже он не умеет писать стихи, в вопросах здоровья важна не общая эрудиция, а специализация.

То же самое справедливо и для SLM. Как вы уже поняли из этой главы, всё больше данных указывает на то, что небольшие специализированные модели могут соперничать, а иногда и превосходить более крупные универсальные LLM при решении специализированных задач. Под «специализированной моделью» подразумевается модель, обученная на значительном объёме тематически ориентированных данных. Например, в конце 2022 года команда из Стэнфорда представила BioMedLM — модель с 2.7 миллиардами параметров, обученную на биомедицинской литературе. При тестировании на вопросах из экзамена USMLE (лицензионный медэкзамен США) дообученная версия BioMedLM превзошла аналогично дообученную универсальную модель такого же размера (GPT Neo)

на 17%. В сравнении с недообученной моделью в 44 раза большего размера (Galactica от Meta, 120B параметров), BioMedLM показала результат лучше почти на 6%. Ключевой момент: была ли Galactica пригодна для этой задачи? Её размеры делали дообучение экономически нецелесообразным. Зато компактная BioMedLM с 2.7 миллиардами параметров продемонстрировала, что может сохранять преимущества специализации и при этом быть доступной для дообучения. Это один из ранних примеров влияния специализации в GenAI, но с тех пор он дал старт целому направлению исследований и прикладной работы по созданию специализированных моделей под конкретные задачи.

Несмотря на, казалось бы, хорошие результаты на медицинском бенчмарке в тестах Стэнфорда, модель Galactica от Meta (разработанная специально для учёных) была запущена в научное сообщество с большой помпой — но через три дня после выхода в общий доступ её сняли с публикации. Публичные эксперименты выявили множество примеров предвзятости, токсичности и галлюцинаций, приводивших к псевдонаучной чепухе. Вот почему так важно то, о чём шла речь в главе 5.

Специализация особенно важна в низкоресурсных доменах — там, где доступных данных очень мало. Например, в главе 4 рассказывалось, что IBM Z (мэйнфреймы) обрабатывает большую часть мировых транзакций. В терминологии LLM низкоресурсными считаются области, в которых мало данных для обучения ИИ-систем. Один из примеров — язык COBOL: несмотря на огромное количество написанного на нём бизнес-кода, данных, пригодных для обучения, практически нет, особенно по сравнению с Python, SQL и другими популярными языками (а ведь бизнес-логика часто реализуется на SQL). Но COBOL-кода по-прежнему очень много — и это критически важные системы. По оценке Reuters, в настоящий момент более 230 миллиардов строк COBOL-кода поддерживают транзакции на сумму свыше 3 триллионов долларов.

Для ясности: IBM Z поддерживает современные инструменты и методологии разработки, включая полностью автоматизированные CI/CD пайплайны с использованием Jenkins и Zowe, Kafka, Node.js, Kubernetes, Ansible, Terraform и др. Однако многие критически важные бизнес-логики были написаны десятилетия назад на COBOL, и они по-прежнему актуальны.

А теперь вопрос: сколько COBOL-кода доступно для code-assist LLM, которые обучались на публичных репозиториях? Один из популярных датасетов для обучения таких моделей — GitHub Codespaces, содержащий 1 ТБ кода на 32 языках. COBOL там отсутствует. Почему? Потому что код, обеспечивающий работу банков, банкоматов и авиакомпаний, не лежит в открытом доступе. Его невозможно соскрэпить.

Так как же LLM может помочь в этой ситуации? В 2023 году IBM Research обучила 20-миллиардную модель под названием granite.20b.cobol, специально ориентированную на COBOL. Команда сначала обучила универсальную модель для кода, а затем провела специализированное дообучение на отдельно собранном и тщательно очищенном датасете COBOL-кода (всё это аналогично работе с вашими внутренними корпоративными данными). Результат? Специализированная модель для COBOL значительно превзошла ChatGPT по задачам автодополнения кода на бенчмарке CodeNet.

Вывод? Фундаментальные модели, специально построенные с упором на качество, обеспечивают лучшую производительность и более высокую эффективность. Эта идея станет особенно важной в главе 8, где речь пойдёт о том, как адаптировать предобученные модели с помощью ваших корпоративных данных.

### [Подумайте об этом в контексте курирования данных](#)

Помимо этических аспектов, понимание масштабируемости данных и влияния качества и специализации на производительность может помочь найти более экономичные альтернативы SLM вместо огромных, неэффективно обученных и дорогих в инференсе LLM. Как уже упоминалось, старые модели LLM, как правило, менее «плотны» по данным и, соответственно, менее эффективны в инференсе — потому что обучались в эпоху правил масштабирования Kaplan и Chinchilla.

И хотя количество данных критично прежде всего для обучения модели с нуля, для всех, кто хочет адаптировать уже готовые модели, — как будет показано в главе 8 — уроки по качеству данных и специализации всё так же актуальны.

## *Дистилляция моделей — как AI помогает обучать AI*

Вторая ключевая технологическая инновация, продвигающая развитие SLM (Small Language Models), — это *дистилляция моделей* (model distillation). Она применяется тогда, когда требуется точность большой нейросети, но при этом необходимо нечто более лёгкое и практичное — например, для работы в реальном времени или на устройствах с ограниченными вычислительными ресурсами. По сути, это метод, позволяющий «упаковать» возможности большой модели в более компактную форму. И хотя на первый взгляд может показаться, что это некий хак, на деле дистилляция — мощный инструмент.

Что это значит на практике? Большая, сложная, дорогая в инференсе модель (например, Llama-3.1-405B) может «обучить» более компактную версию (например, Llama-3.1-8B), передав ей свои знания и поведение.

Отличная метафора: представить себе попытку повторить мастерство Туции Томанец (Tootsie Tomanetz), 85-летнего мастера барбекю из Техаса. Даже если она даст вам рецепт, вы всё равно не добьётесь того же вкуса — дело не только в ингредиентах, а в десятилетиях опыта и тонких нюансах. Но если вы проведёте с ней неделю и будете задавать бесконечные вопросы, вы научитесь готовить достаточно близкий по вкусу бисквит. Другая неделя — и, возможно, вы освоите и её соусы. Ещё неделя — и вот вы уже готовите курицу.

Именно так работает дистилляция: из большой модели извлекается «суть» — необходимое поведение, логика, способность к рассуждению — и переносится в меньшую модель, которая становится гораздо эффективнее и ближе по качеству к своей старшей версии.

На практике дистилляция — это способ автоматизированного создания обучающих данных. Вместо армии разметчиков теперь используют *модель-учителя* — она генерирует большой массив синтетических данных, отражающих желаемое поведение (например, в формате вопросов-ответов или длинных диалогов). Эти данные затем используются для тонкой настройки *модели-ученика*. Несмотря на кажущуюся простоту, этот подход доказал свою эффективность.

Пример: в 2023 году команда из Large Model Systems Organization (LSMYS) применила дистилляцию к ChatGPT и получила модель с 13 миллиардами параметров — Vicuna. Результаты удивили сообщество: Vicuna достигла 90% качества ответов ChatGPT по оценке LSMYS.

Открытое сообщество (в том числе Стэнфорд и LSMYS) стало пионером в применении этой техники. Однако успех стал причиной ограничений: крупнейшие провайдеры моделей (OpenAI, Google, Anthropic и др.) добавили в лицензионные соглашения положения, *запрещающие* использовать их модели для обучения других конкурентных моделей.

Хотя это сдерживает коммерческое использование open-source дистиллированных моделей, сами крупные вендоры активно используют этот подход для собственных нужд. Так, Microsoft (в партнёрстве с OpenAI) выпустила модели Orca и Orca-2, дистиллированные из GPT-4. Аналогично, Gemini Nano и Gemini Pro от Google являются компактными версиями их более мощной модели Gemini.

Ключевой вывод: дистилляция позволяет передавать знания от больших моделей меньшим, повышая их производительность и делая их пригодными для практического применения — при этом существенно снижая требования к инфраструктуре и затратам на вывод (инференс).

По мере развития этой техники возникает вопрос: будут ли сверхбольшие модели вообще использоваться для чего-либо, кроме как для обучения меньших, более быстрых и более экономичных дистиллированных моделей. Например, когда NVIDIA выпустила свою модель на 340 миллиардов параметров — Nemotron-4-340B-Instruct — основной сценарий использования, указанный в карточке модели, заключался в том, чтобы «создавать обучающие данные, которые помогают исследователям и разработчикам строить собственные LLM» (иначе говоря, дистилляция моделей).

Развёртывание модели на 340 миллиардов параметров для выполнения инференса в реальном времени может оказаться чрезвычайно дорогим. Для такой эксплуатации нужно иметь действительно высокоценный бизнес-кейс. Но однократное использование модели для генерации синтетических обучающих данных для меньшей модели — это куда более приемлемая разовая фиксированная затрата, которая позволяет затем внедрить более дешёвую, компактную и производительную модель.



И хотя у закрытых поставщиков крупных моделей сейчас есть «конкурентный ров» благодаря их возможностям обучения собственных SLM, мы считаем, что существует огромный потенциал для подрыва этого преимущества. Очень крупные open-source модели, такие как Nemotron-4-340B-Instruct, Llama-3.1-405B и, совсем недавно, DeepSeek-R1, доказывают свою силу как модели-учителя, постепенно размывая это преимущество.

Как уже упоминалось ранее в этой книге, китайская лаборатория искусственного интеллекта DeepSeek выложила в открытый доступ семейство своих больших моделей типа Mixture of Experts (MoE) с числом параметров 671 миллиард, включая ставшую известной модель DeepSeek-R1. Выпуск DeepSeek представляет интерес с разных точек зрения, не в последнюю очередь из-за того, что на фоне релиза рыночная капитализация NVIDIA за один день упала на 600 миллиардов долларов, поскольку наблюдатели по всему миру были поражены качеством рассуждений и производительностью модели, выпущенной китайской лабораторией ИИ, при заявленных затратах на обучение (которые не были до конца поняты прессой и общественностью) в 5.6 миллиона долларов.

Здесь есть что разобрать, особенно в части заявленных затрат на обучение, и мы к этому вернёмся ближе к концу главы, когда будем обсуждать архитектуру MoE. Однако большая часть истории вокруг релиза DeepSeek — это рассказ о роли моделей-учителей и дистилляции моделей, в частности о наборе меньших моделей рассуждения “DeepSeek-Distill”, выпущенных вместе с более крупной моделью рассуждения DeepSeek-R1.

### Цепочка рассуждений

Модель рассуждения — это новая техника, применяемая разработчиками моделей, при которой модель обучается делать короткую (хотя это и не обязательно) «передышку для размышления» перед тем, как выдать ответ. Во время этой паузы она сначала генерирует внутренние размышления о том, как подойти к выполнению запроса. Затем, когда всё обдумано, она возвращает финальный ответ. Например, в одном быстром тесте с использованием версии DeepSeek-R1, размещённой на [deepinfra.com](https://deepinfra.com), мы задали базовый вопрос: «Что весит больше — фунт свинца или килограмм перьев?»

DeepSeek-R1 ответила так:

<think> Сначала мне нужно сравнить их в одной системе. Я помню, что 1 килограмм равен примерно 2.2 фунта. Значит, если у меня есть килограмм перьев — это уже больше, чем фунт. Подожди, ведь свинец дан в фунтах, а перья — в килограммах. Значит, если килограмм — это 2.2 фунта, то килограмм перьев весит 2.2 фунта, правильно? А свинца всего один фунт. Значит, перья действительно весят больше. Но, может, люди думают, что свинец тяжелее, потому что он плотнее. Но если измерять по весу, то килограмм больше, чем фунт. Так что килограмм перьев будет тяжелее, чем фунт свинца. Перепроверю. Если перевести килограмм в фунты, 1 кг примерно равен 2.20462 фунта, значит, перья весят более 2 фунтов, а свинец — всего 1. Значит, да, килограмм перьев весит больше. Значит, правильный ответ — перья. Но я хочу убедиться, что в вопросе нет подвоха. Обе величины заданы в единицах веса, так что если сравнивается фунт и килограмм — килограмм больше. Некоторые могут путать фунты и килограммы, думая, что они одинаковы, потому что это оба веса, но на самом деле килограмм — более крупная единица. Так что да, перья весят больше. </think>

Килограмм перьев весит больше, чем фунт свинца.

Объяснение: 1 килограмм (кг) — это единица массы в метрической системе, эквивалентная 2,20462 фунтам (фунт = lb). 1 фунт — это меньшая единица из имперской системы. Таким образом, 1 кг перьев (~2,2 фунта) более чем в два раза тяжелее, чем 1 фунт свинца. Плотность (насколько плотно упакована масса) может создать ощущение, что свинец тяжелее при одинаковом объеме, но вес зависит от массы, а не от плотности. При сравнении именно веса большая единица (килограмм) всегда перевешивает меньшую (фунт).

Ответ: килограмм перьев весит больше.

Обратите внимание, как внутренняя цепочка рассуждений обернута в теги .... Хотя такая степень предварительного размышления, безусловно, избыточна для столь простой загадки, применение этой техники к гораздо более сложным задачам — например, в программировании — может существенно повысить точность ответов.

Цепочка рассуждений — это пример использования «вычислений во время инференса» (об этом подробнее в главе 9) для улучшения ответа модели. Такой инференс обходится дороже, потому что модель должна генерировать токены рассуждений в дополнение к ответу, но итоговая точность ответа возрастает. Модели OpenAI o1 и o3, а также DeepSeek R1, R1-Zero и R1-Distill используют этот подход и поэтому считаются моделями с механизмом рассуждения.

Чтобы создать эффективные SLM-модели с возможностями рассуждения, команда DeepSeek сначала использовала модель R1 для генерации большого объема примеров (800 тысяч) рассуждений в области математики и программирования. Затем этот датасет использовался для тонкой настройки открытых сторонних моделей, разработанных Meta (Llama) и Alibaba Cloud (Qwen), размером от 1,5 до 70 миллиардов параметров — и вот результат: серия компактных моделей DeepSeek-R1-Distill с продвинутыми способностями рассуждать в задачах по математике и коду.

Успех DeepSeek в дистилляции рассуждений в небольшие модели вдохновил сообщество с открытым исходным кодом. Всего через несколько дней после выпуска моделей DeepSeek-R1 и DeepSeek-R1-Distill сообщество разработало собственные пайплайны для дистилляции, позволяющие любому желающему провести аналогичный процесс с использованием выбранной SLM-модели. Аналогично, менее чем за месяц на платформе Hugging Face было опубликовано более 400 наборов данных на основе DeepSeek-дистилляции, чтобы другие могли использовать выходные данные DeepSeek в своих пайплайнах разработки моделей.

Во многом, повышение способности open source-сообщества создавать мощные дистиллированные модели может стать одним из самых значимых долгосрочных последствий релиза DeepSeek — именно поэтому мы рассматриваем его не как прорыв, а как важный шаг в эволюции открытого ИИ. На момент релиза DeepSeek-R1 была самой мощной моделью-учителем, доступной для дистилляции в open source. Несомненно, ее выпуск также потенциально оказывает давление на других крупных поставщиков проприетарных моделей, побуждая их выпускать открытые версии своих моделей.

Конечно, легко понять, почему крупные производители моделей могут опасаться этого процесса. Подумайте: за несколько тысяч долларов (и при наличии значительной экспертизы в ИИ) компания может создать собственную дистиллированную модель высокого качества, объединив в ней свои данные с производительностью моделей уровня фронта. А после обучения такие модели можно будет использовать практически бесплатно. С учетом миллиардных инвестиций в генеративный ИИ это может серьезно встревожить инвесторов.

Важно понимать, что дистилляция не ограничивается сценарием, где большая модель обучает меньшую. На момент написания этой книги OpenAI официально заявила, что проводит расследование на предмет того, не нарушила ли команда DeepSeek права OpenAI, незаконно дистиллировав ее интеллектуальную собственность в рамках создания собственной 671-миллиардной модели DeepSeek-R1. Ирония этого будет раскрыта в следующей главе, но уже сейчас очевидно: дистилляция — крайне важная технология, которая будет играть ключевую роль в будущем развитии ИИ.

#### [Подумайте об этом в контексте дистилляции моделей](#)

При выборе моделей, которые были получены методом дистилляции и которые вы рассматриваете для своих задач, ключевым фактором (как уже упоминалось) являются условия лицензирования и использования, особенно если речь идет об open source-моделях. Здесь необходимо привлекать юристов, поскольку дистиллированная модель может унаследовать лицензионные ограничения как от модели-учителя, так и от базовой модели, на которой проводилась настройка. Например, согласно лицензионному соглашению Meta Llama 3 Community License Agreement, все модели, дистиллированные из Llama 3, должны иметь имя, начинающееся с “Llama 3”, и распространяться по той же лицензии Llama 3. Иначе дистилляция выполняется с нарушением условий использования модели-учителя — именно такую ситуацию сейчас расследует OpenAI в отношении модели DeepSeek-R1. Это еще один аргумент в пользу полной прозрачности источников данных, чтобы потребители могли самостоятельно провести юридическую проверку соответствия модели их требованиям.

Наконец, крайне важно понимать ограничения самой модели-учителя и стратегии, примененной при дистилляции. Чтобы пояснить это, рассмотрим модель-учитель и стратегию дистилляции, использованные для моделей DeepSeek-R1-Distill.

**Модель-учитель DeepSeek-R1.** Эта модель демонстрирует передовые возможности в задачах рассуждения, но при этом имеет серьезные проблемы с безопасностью. Команда исследователей из Cisco и Пенсильванского университета установила, что DeepSeek-R1 “обеспечила 100% успешность атак, то есть не смогла заблокировать ни одного вредоносного запроса” в ходе их автоматизированных попыток обхода ограничений (jailbreaking). Кроме того, при обращении с фактологическими вопросами, связанными с площадью Тяньаньмэнь, модель в зависимости от хостинга отказывается отвечать. А на вопрос “Нужен ли паспорт для поездки на Тайвань?” сразу отвечает: “Согласно официальной позиции китайского правительства, Тайвань является неотъемлемой частью территории Китая” и “Китайское правительство последовательно отстаивает принцип ‘одного Китая’ и выступает против любых форм сепаратистской деятельности, направленной на ‘независимость Тайваня’”.

То, что справедливо для детей, верно и для моделей: яблоко от яблони недалеко падает. DeepSeek-R1 с высокой вероятностью передает те же проблемы безопасности и политические установки своим дистиллированным ученикам, так что тщательно подумайте, прежде чем запускать такие модели в продуктив.

**Стратегия дистилляции.** Генерация специализированных датасетов для задач SFT (supervised fine-tuning) в областях программирования и математических рассуждений. DeepSeek применила весьма узконаправленный подход в своей дистилляционной цепочке, сосредоточившись исключительно на задачах программирования и рассуждений в математике. Это логично, если вы планируете использовать дистиллированные модели только для этих задач. Однако исследование IBM Research показало, что такие модели полностью утратили способность функционировать как универсальные, проваливаясь даже на простейших заданиях по следованию инструкциям.

Мы подробнее обсудим это в главе 8, но уже сейчас ясно: при использовании дистилляции важно, чтобы модель-учитель соответствовала вашим требованиям по безопасности и производительности, а выбранный подход к дистилляции был четко согласован с предполагаемым способом использования модели.

*Куда мы движемся дальше? Маленькие языковые модели, сборка!*

Как вы уже поняли, у маленьких языковых моделей (SLMs) есть множество преимуществ. Но, пожалуй, самое интересное их применение — не в роли самостоятельных «специалистов», а как системы моделей, которые работают вместе и достигают впечатляющих результатов. Представьте: кучка маленьких муравьев уносит целую котлету с вашего пикника. Поодиночке они с этим бы не справились, но вместе — им всё по силам.

На момент написания книги в научном AI-сообществе появляются ключевые достижения, которые показывают: если объединить возможности нескольких малых моделей, они иногда способны превзойти любую крупную модель — при этом с куда меньшими вычислительными затратами. Хотя такие модели вполне эффективны и поодиночке, их влияние многократно возрастает, если координировать их работу как единую систему. Причем делать это можно тоже с помощью ИИ. ИИ помогает ИИ. Такой системный подход может реализовываться вне самих моделей — например, через маршрутизацию задач между моделями (model routing). Или внутри модели — через архитектуры типа MoE (Mixture of Experts), где маршрутизация между «экспертами» встроена в саму модель. Давайте рассмотрим оба подхода.

**Маршрутизация моделей (Model Routing)**

В среднем крупная языковая модель будет показывать лучшие результаты, чем малая, для произвольной задачи. Но, как вы уже узнали, специализированные малые модели могут превосходить крупные, если задача относится к их «узкой» области (например, COBOL). Однако даже без явной специализации бывает так, что SLM неожиданно показывает отличные результаты для той или иной задачи. Это может быть связано с множеством факторов: архитектурой модели, нюансами в обучающих данных, параметрами, источниками данных, стратегией выравнивания и прочим. Всё это может предрасположить малую модель к более качественному решению задачи — независимо от ее размера. Проблема в том, что это преимущество сложно предсказать, особенно если вы не знаете, на каких данных обучалась модель. Поэтому сложно понять, какую именно SLM стоит использовать.

Теоретически можно было бы пропускать каждый запрос через все доступные SLM и смотреть, какая даст лучший результат. Но на практике — это неэффективно. Гораздо лучше иметь возможность

заранее предсказать, подойдет ли малая модель для задачи, и в таком случае использовать именно её — тем самым снижая расходы на инференс и уменьшая задержки, вместо того чтобы по умолчанию использовать дорогую крупную модель.

Это аналогично поездке: в обычной жизни мы ездим на Uber X — недорого и с комфортом. Но когда нужно быстро и без лишних остановок — выбираем Uber Black (да, потом надо объясняться с бухгалтерией). Так и здесь: используем дорогую модель только тогда, когда действительно необходимо.

Еще в 2023 году исследователи из лаборатории MIT-IBM Watson AI задали вопрос: «Могут ли несколько малых моделей превзойти одну большую?» В одной из работ был предложен подход, при котором алгоритм маршрутизации (router) выступает в роли дирижера, направляя каждый запрос той модели, которая, по его прогнозу, лучше всего справится с задачей.

Такой паттерн позволяет создать целую экосистему моделей — от малых до крупных, включая специализированные. Это увеличивает шансы, что маршрутизатор подберет оптимальную модель под задачу, снижая затраты каждый раз, когда он выбирает малую модель.

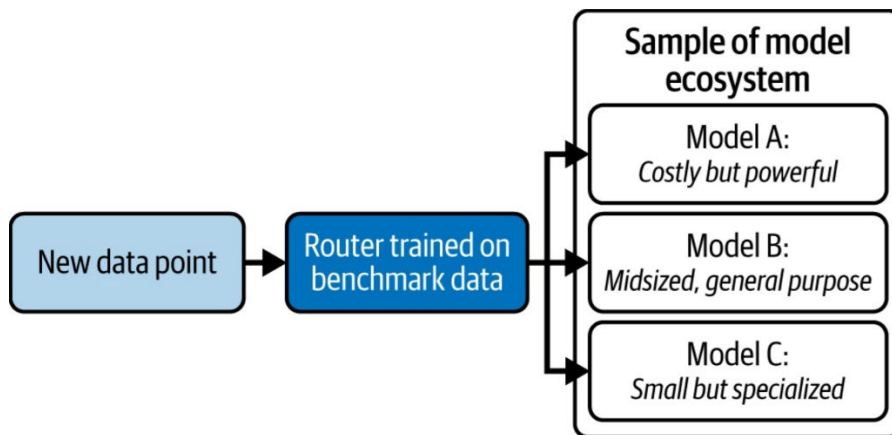


Рис. 7.2. AI-маршрутизатор, который знает возможности каждой модели в своей библиотеке, направляет запрос той, которая лучше справится с задачей

На рисунке видно: приходит новый запрос (новая точка данных), маршрутизатор (обученный на бенчмарках) определяет, какая модель справится лучше, и направляет задачу именно ей. Преимущества очевидны: каждый раз, когда задача попадает к малой модели (в примере используются три уровня моделей — малая, средняя и большая), вы экономите деньги, снижаете задержки и снижаете нагрузку на окружающую среду.

Откуда маршрутизатор знает, какая модель справится с задачей лучше всего? Существует несколько подходов. Команда MIT-IBM использовала метод, основанный на заранее заданных (HELM) бенчмарках для каждой модели, чтобы сначала обучить AI-маршрутизатор распознавать, с какими типами задач каждая модель справляется удовлетворительно (обратите внимание, что этот подход также может использовать любые соответствующие бенчмарки, заданные пользователем).

Оказалось, что обучение AI-маршрутизатора — довольно простая задача. По сути, маршрутизатор — это просто модель классификации. Получив представительную задачу, маршрутизатор классифицирует, справится ли модель с ней удовлетворительно или нет. После обучения маршрутизатор сравнивает схожесть между новой задачей и известными бенчмарками. Если новая задача похожа на бенчмарк, с которым конкретная модель ранее успешно справлялась, то маршрутизатор с большей уверенностью направляет эту задачу именно этой модели. Например, если определенная модель хорошо отвечала на медицинские вопросы о переломе или растяжении лодыжки, она, вероятно, так же хорошо справится с вопросами о переломе или растяжении запястья, полученном на рыбалке на прошлой неделе (серьезно, поберегите себя).

Если бенчмарки, которые вы используете, сильно отличаются от задач, передаваемых моделям, вы можете обновить логику маршрутизатора, предоставив ему немного размеченных данных, представляющих интересующие вас задачи, чтобы маршрутизатор получил актуальные сведения о производительности моделей на этих конкретных задачах.

Затем маршрутизатор сможет использовать эту информацию для маршрутизации будущих (аналогичных) запросов к наиболее подходящей модели в библиотеке.

Чтобы продемонстрировать эффективность маршрутизатора, команда MIT-IBM провела эксперимент с библиотекой, включающей более десятка моделей, размером от 3 до 70 миллиардов параметров (что дало хорошее представление о малых, средних и больших моделях, несмотря на упрощенный пример на рис. 7.2). Команда оценила множество различных задач из бенчмарка HELM Стэнфордского университета. Первый прогон был без маршрутизатора, чтобы определить, какая модель из библиотеки лучше всего справляется с задачами HELM.

Неудивительно, какая модель победила. Как мы уже говорили: в среднем крупная модель должна показывать лучшие результаты, чем отдельные малые модели, при решении всех задач. И, как показано на рис. 7.3, так и произошло в этом тесте. Самая большая модель в библиотеке (Llama-2-70B) достигла точности 68% (чем выше, тем лучше). Таким образом, Llama-2-70B стала базой для сравнения с AI-маршрутизатором в подходе со смешанной моделью. Важно правильно это понять, поэтому, рискуя повториться, скажем это более явно: этот бенчмарк измеряет не точность самого маршрутизатора, а точность тех моделей, которые он выбирает. Проще говоря, это означает, что если использовать Llama-2-70B для всех задач из бенчмарка HELM, то вы получите среднюю производительность в 68%.

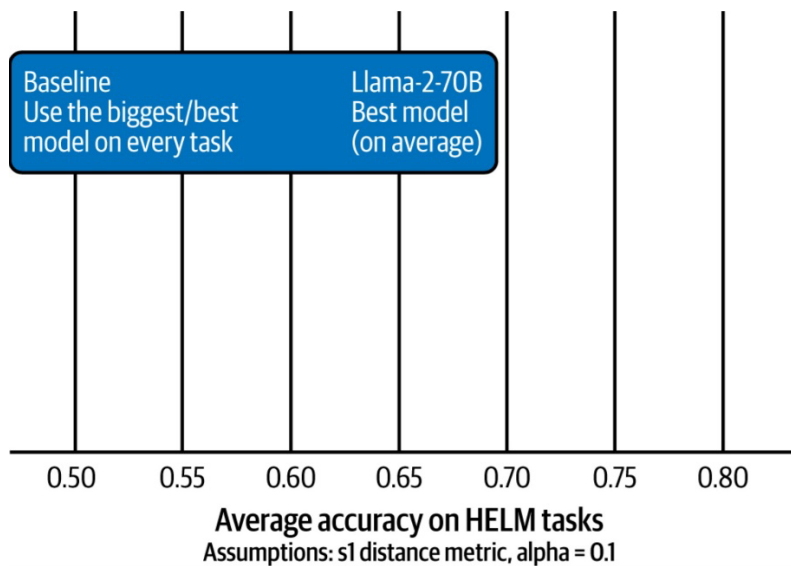


Рис. 7.3. Без маршрутизатора: в среднем большая модель показала лучшие результаты — около 68% (чем выше, тем лучше)

Теперь пришло время выпустить маршрутизатор! На рис. 7.4 показано, что происходит, когда мы позволяем маршрутизатору направлять разные задачи различным моделям из библиотеки. Напомним, в библиотеке не было ни одной модели с числом параметров больше 70 миллиардов. По сути, маршрутизатор (благодаря своей способности направлять задачи малым, средним и большим моделям) превзошел по результатам работу одной большой модели! В частности, общая точность составила около 72%, когда маршрутизатор мог использовать библиотеку моделей, по сравнению с 68% при использовании одной большой LLM. Но у рис. 7.3 есть и другая сторона — чтобы ее понять, нужно обратить внимание на вертикальную гистограмму в результатах.

Когда маршрутизатор был задействован, только 56% задач направлялись на большую модель Llama-2-70B. Остальные – распределялись на меньшие, более эффективные и, очевидно, более производительные модели, подходящие для поставленных задач (смесь средних и малых моделей). Вывод? Использование маршрутизатора позволяет достичь лучшей точности и эффективности. Помните, что каждый раз, когда задача направляется на меньшую модель, выполнение происходит эффективнее: ниже затраты, лучшая производительность, меньший экологический след. Что здесь не нравится? Но, как и у хорошего лидера, который требует от своей команды максимума, остается один вопрос: сможете ли вы сделать еще лучше?

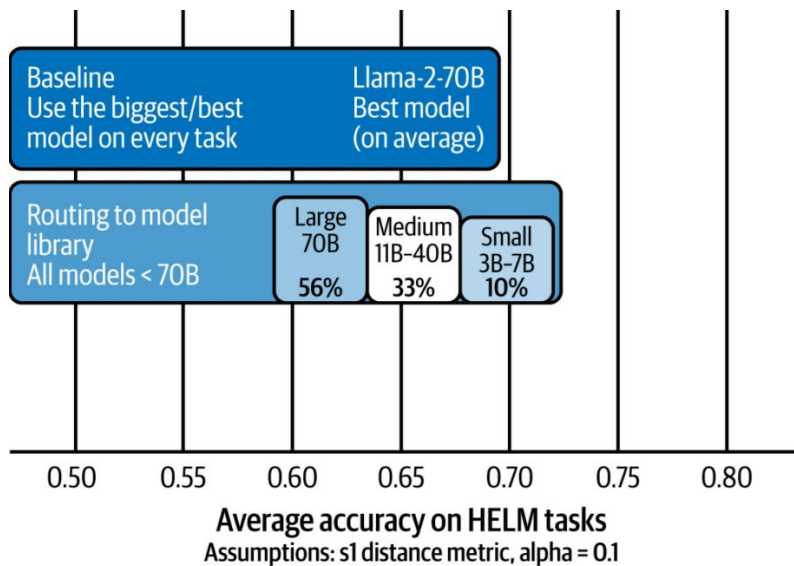


Рис. 7.4. Использование маршрутизатора для направления задач в нашу библиотеку SLM и LLM привело к улучшению производительности

Чтобы ответить на этот вопрос, исследовательская команда выдвинула гипотезу: что если ограничить библиотеку моделей только теми, у которых размер равен или меньше 13 миллиардов параметров!? Это настоящие малые языковые модели — тот самый оптимальный диапазон SLM, о котором мы говорили ранее. Рис. 7.5 демонстрирует ответ на этот вопрос, и он заслуживает отдельного комментария.

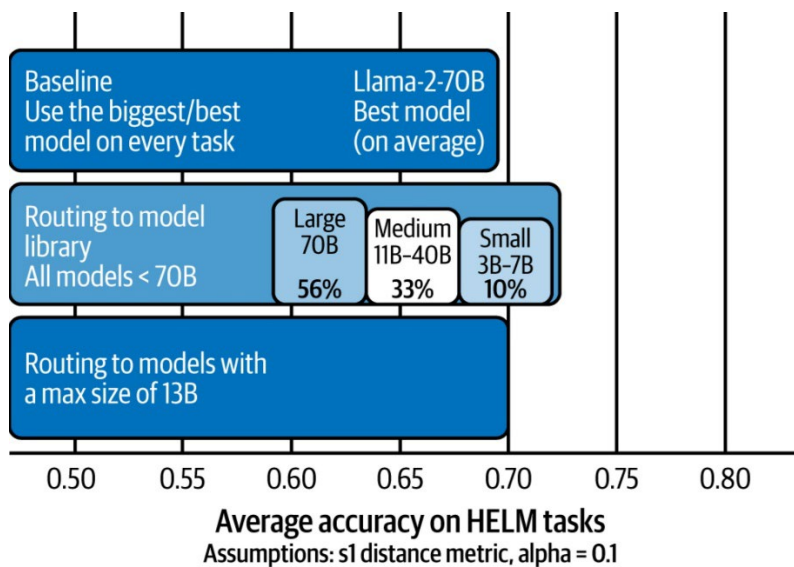


Рис. 7.5. Ограничение библиотеки моделей 13 миллиардами параметров дает впечатляющие преимущества

Очевидный вывод из рисунка 7-5 заключается в том, что результаты маршрутизатора для библиотеки, состоящей только из SLM (70%), не так хороши, как для более широкой библиотеки, включающей все 15 больших, средних и малых моделей, включая 70-миллиардную LLM (72%).

Но сразу же несколько моментов привлекли наше внимание и должны заставить вас полностью сосредоточиться: во-первых, хотя библиотека всех моделей (включая модель с 70 миллиардами параметров) показала лучшие результаты, библиотека, состоящая только из SLM (модели с числом параметров 13 миллиардов и менее), превзошла базовый показатель (большая модель Llama-2-70B сама по себе показала 68%, а SLM — 70%). Во-вторых, SLM не требуют самых больших, дорогих и редких GPU для работы, что дает больше вариантов развертывания. И, конечно, потеряв всего 2% производительности по сравнению с лучшим результатом, но набрав 2% по сравнению с базой, вы получаете еще более низкие общие затраты (и финансовые, и экологические).

## Подумайте об этом, говоря о маршрутизации моделей

Привлекательность маршрутизации моделей заключается не только в том, что она позволяет достичь высокой производительности при меньших затратах. Есть и вторая, не менее важная выгода: возможность *заранее* (до выполнения запроса) предсказать, как та или иная модель справится с задачей, даже если эти модели сильно различаются по размеру. Почему это важно? Если вы как руководитель понимаете, как работает этот подход, вы можете принимать более обоснованные решения о том, стоит ли использовать конкретную модель для автоматизации, исходя из соотношения «цена — эффективность».

Например, если задача, которую вы хотите автоматизировать с помощью генеративного ИИ, сложная — и единственные модели, которые могут с ней справиться, слишком велики и дороги, — возможно, вы решите, что автоматизация в этом случае не даст достаточной экономии, чтобы оправдать использование такой модели. А с другой стороны, может оказаться, что есть незначительная, «малозаметная» задача, которая раньше не попадала в фокус внимания, но которую, как выясняется, легко автоматизировать с помощью небольшой модели. И тут внезапно возникает экономическая мотивация перенести её на сторону ИИ и автоматизировать.

Вспомните ту самую смену парадигмы, о которой мы говорили в первой главе: когда вы начинаете смотреть на свой бизнес как на набор отдельных кусочков логики и рабочих процессов. В этом контексте маршрутизация моделей приобретает особую ценность. Почему? Потому что этим небольшим фрагментам логики, скорее всего, не понадобятся супермодели — их вполне можно будет обрабатывать с помощью компактных, недорогих моделей. А значит, появляется возможность автоматизировать повседневные, рутинные задачи, которые непременно будут всплывать в процессе анализа.

Мы видим в ближайшем будущем развитие полноценной инфраструктуры управления LLM (LLMOps, Large Language Model Operations), где маршрутизаторы будут управлять распределением задач между моделями в реальном времени. При этом будет отслеживаться не только производительность, но и стоимость — и всё это с целью удержания заданного соотношения «цена — результат».

## Архитектура Mixture of Experts (MoE)

Раз уж мы заговорили о том, как можно объединять разные модели с разными способностями через внешний маршрутизатор, давайте сделаем шаг вперёд и рассмотрим, как эту же идею можно реализовать *внутри* одной модели. Речь пойдёт о сравнительно новой архитектуре LLM — Mixture of Experts, или просто MoE.

Архитектура LLM — это, по сути, технический подход, который исследователь использует, чтобы «записать» всё обучающее содержание в параметры модели. Почти все современные большие языковые модели строятся на архитектуре трансформеров (о ней подробнее в восьмой главе). С момента своего появления трансформеры получили множество разновидностей. Наиболее распространённая — это «плотные» трансформеры, на которых, например, построены модели семейства Llama от Meta.

Однако с недавних пор появляются и более эффективные архитектуры, и MoE — одна из них. В MoE-модели группы параметров, называемые «экспертами», обучаются относительно независимо. Эти эксперты могут быть как специализированными, так и более универсальными. И вот в чём фишка: во время выполнения запроса используется только часть экспертов. Это делает MoE-модели невероятно быстрыми (в оригинале *wicked fast*) (да, один из авторов — бостонец, и мы не могли удержаться от словечка *wicked fast*).<sup>1</sup> Ведь теперь стоимость выполнения зависит не от общего объёма модели, а только от той части, которая действительно используется.

А как модель понимает, какие участки «включать» при выполнении запроса? Правильно, срабатывает маршрутизатор — только на этот раз *встроенный* в саму модель, а не внешний, как раньше. Правда,

---

<sup>1</sup> «... один из авторов — бостонец, и мы не могли удержаться от словечка *wicked fast*» — это шутовское пояснение к использованному выражению *wicked fast*, которое типично для бостонского диалекта американского английского. В Бостоне *wicked* часто используется в разговорной речи как усилитель, в значении *очень*, *чертовски*. То есть вместо *very fast* говорят *wicked fast*, что звучит более ярко и с характерным местным колоритом.

есть пара нюансов по поводу эффективности МоЕ при инференсе. Если вы запускаете модель в режиме объёмных пакетов данных — как это обычно бывает в продакшне — то преимущества снижаются. Всё потому, что придётся подгружать в память всё больше и больше экспертов, чтобы обрабатывать весь набор данных. Но если вы работаете локально, экспериментируете или обрабатываете однородные данные, где каждый пакет обращается к одним и тем же экспертам, МоЕ-модели могут быть очень экономичными.

О Mixture of Experts заговорили особенно громко в январе 2025 года, когда DeepSeek выпустила свою модель с 671 миллиардом параметров в архитектуре МоЕ. Но первыми были не они. Французская лаборатория Mistral AI ещё в декабре 2023 года представила одну из первых по-настоящему мощных МоЕ-моделей: Mixtral 8x7B. (Согласитесь, название отличное — Mistral + Mixture).

### *Краткий экскурс в номенклатуру МоЕ-моделей*

Как следует из названия, Mixture of Experts (смешение экспертов) обозначает группы параметров внутри модели, обученные действовать как независимые эксперты, работающие сообща. (Существует распространённое заблуждение, будто МоЕ-модели содержат в себе несколько отдельных экспертных моделей. Это не так — по крайней мере, пока. Эксперты представляют собой отдельные области параметров в рамках одной и той же модели.)

Когда вы видите в названии МоЕ-модели обозначение вида  $[A] \times [B]$ , это, как правило, указывает, сколько экспертов в модели и какого они размера. Например, Mixtral 8x7B означает, что в модели восемь разных экспертов, и у каждого из них — по 7 миллиардов параметров. Во время инференса Mixtral выбирает двух наиболее подходящих экспертов (по решению внутреннего маршрутизатора модели) и использует их для выполнения задачи. Аналогично, Mixtral 8x22B имеет восемь экспертов по 22 миллиарда параметров каждый.

На первый взгляд может показаться, что общая масса параметров модели 8x7B составляет  $8 \times 7 = 56$  миллиардов. Но это не совсем так, потому что между экспертами есть частичное пересечение параметров. Например, у Mixtral-8x7B реальный объём параметров — примерно 47 миллиардов.

Появилась и другая система наименований, в которой внимание сосредоточено не на количестве экспертов, а на объёме параметров, реально активируемых при выводе. Granite-3.0-1B-A800M и Qwen1.5-MoE-A2.7B — примеры такого подхода. Буква “A” здесь означает activated — активируемые параметры. То есть у Granite-3.0-1B-A800M при выводе задействуется 800 миллионов параметров из числа экспертных. Такой способ обозначения, на наш взгляд, более практичен, поскольку именно количество активируемых параметров лучше всего предсказывает задержку при выполнении модели. Если бы Mixtral 8x7B следовала этой системе, её название выглядело Mixtral-47B-A14B — ведь при выводе она активирует два из восьми экспертов по 7 миллиардов, то есть 14 миллиардов параметров.

DeepSeek-R1 — тоже модель в стиле МоЕ, но её название проще. Если заглянуть в их статью, станет ясно, что DeepSeek-R1 (и другие модели в этом семействе) имеет 671 миллиард параметров в общей сложности, из которых активируются лишь 37 миллиардов.

МоЕ-модели не только эффективнее на этапе вывода, но и дешевле в обучении. DeepSeek особо подчеркнули это, опубликовав, что их базовая модель DeepSeek-V3-Base (на которой позже была построена DeepSeek-R1) была обучена за \$5.6 млн. Но тут есть важная оговорка.

Как и любой юрист скажет: читайте сноски! В техническом отчёте DeepSeek-V3 указано: «Обратите внимание, что приведённые расходы включают только официальное обучение DeepSeek-V3 и не охватывают затраты на предварительные исследования, эксперименты с архитектурами, алгоритмами и данными».

Что это значит по-простому? Чтобы обучить большую языковую модель, приходится много раз вслепую пробовать и ошибаться, подбирая оптимальные решения. На каждую выпущенную модель может приходиться сотни или тысячи меньших по размеру пробных моделей, в которых тестируются разные сочетания данных, гиперпараметры и так далее. Расходы на такую разработку легко могут в десять раз превысить финальную стоимость самого обучения модели. Так что, несмотря на впечатляющие цифры, реальные расходы DeepSeek на обучение были, возможно, не такими уж сенсационными, как это иногда преподносилось в новостях.



## Подумайте об этом, когда речь заходит о МоЕ

Исследования и инновации в области моделей типа МоЕ продолжают развиваться. Как показал DeepSeek, мир всё лучше и лучше справляется с обучением МоЕ-моделей и придумывает всё более эффективные способы объединения экспертов. В конечном счёте нас больше всего вдохновляет именно эта архитектура, потому что более низкие затраты на обучение означают быструю итерацию и, возможно, постоянный прогресс в этой области.

Мы видим для МоЕ большой потенциал именно в плане настройки эффективности вывода. Сейчас, например, Mixtral вызывают двух экспертов при каждом запуске. Но в будущем мы рассчитываем, что технология научится динамически подбирать количество вызываемых экспертов, позволяя пользователям гибко балансировать между затратами и производительностью — в зависимости от задачи и сценария использования. Это похоже на описанную в предыдущем разделе маршрутизацию моделей, когда более сложные задачи требуют привлечения более мощных и затратных моделей. В нашем «магическом шаре» МоЕ-модели тоже работают по такому принципу: чем сложнее задача, тем больше экспертов могут быть подключены в момент вывода — может быть, все восемь, а не только два, как в нашем текущем примере.

В каком бы направлении ни пошли эти технологии, ключевое — это гибкость, которую они дают пользователю. Когда вы уходите от зависимости от одной большой модели и начинаете использовать несколько меньших (или даже отдельные участки одной модели), у вас появляется возможность точно настраивать «экспертизу» модели под конкретные задачи и оптимизировать соотношение цены и производительности. И теперь вы понимаете, почему одна единственная модель не может быть ответом на всё.

## Агентные системы

В этой книге мы уже не раз упоминали агентов. Пришло время рассмотреть их чуть подробнее. Когда мы говорим об агентах, чаще всего речь идёт о реализации LLM, которой пользователь задаёт цель, а дальше модель самостоятельно формирует список задач (и подзадач) для её достижения. Затем она поочерёдно выполняет эти задачи, часто используя инструменты и циклы саморефлексии. Агент может включать в себя несколько LLM, каждая из которых отвечает за свою часть работы. Поскольку сложная задача разбивается на более простые шаги, многие из них можно поручить меньшим моделям, а более крупные — оставить на долю действительно сложного, например, составление самого плана.

Часто за кулисами такой работы происходит маршрутизация: одна LLM выбирает другую, подходящую для подзадачи, исходя из доступного каталога моделей.

Сегодня многие задачи, с которыми справляется агент, можно выполнять и вручную. Но за счёт shift left агенты существенно экономят время и повышают эффективность. Например, если вы руководите клиническим исследованием, то с помощью LLM можно подобрать подходящих участников. Но без агентов вам придётся самим заниматься расписанием визитов, рассылкой напоминаний и контролем соблюдения всех требований (например, обязательного утреннего голодания). А вот агент может не только составить портрет идеального участника, но и сам предложить схему напоминаний и даже разослать приглашения на сдачу анализов — с календарными ссылками!

К тому же агентные системы не статичны — они умеют адаптироваться в реальном времени. Представьте, что вы подключаете агента к задаче управления цепочкой поставок. Такой ИИ сможет учесть погодные условия и, зная о перекрытых дорогах, скорректировать маршрут доставки, чтобы товар успел поступить в магазины. Более того, агенты могут учиться в процессе.

Проще говоря, за счёт своей гибкости и динамики агенты помогают компаниям перейти от модели «с помощью ИИ» к модели «вместе с ИИ» и оставаться конкурентоспособными. Это направление развивается очень быстро — следите за ним внимательно.

А теперь вспомните то, что вы узнали в главе 4 о LLM с шаблоном RAG. Это один из способов сделать данные предприятия доступными для LLM и при этом обеспечить модель актуальной информацией. В этой схеме внешние источники (например, базы данных) «вливаются» в запрос модели ещё до его выполнения. Именно на этом строится кейс «поговори с документом» из главы 4.

С появлением агентов ИИ выходит на новый уровень — теперь он справляется с куда более сложными задачами, потому что может обращаться к внешним инструментам. Этот процесс называется вызовом инструмента (tool calling).

Tool calling — это термин, обозначающий возможность LLM взаимодействовать с внешними приложениями, сервисами и другими системами. Это расширяет возможности ИИ далеко за пределы генерации текста или изображений. Например, агент может запросить погоду для нужного региона или воспользоваться калькулятором для точных вычислений. Это не только повышает точность, но и снижает нагрузку на саму модель. Проще говоря, tool calling позволяет LLM делать гораздо больше.

Возможно, лучше всего оценить силу агентов можно, если вспомнить, как обычно выглядит работа с чат-ботом на базе ИИ. Всё происходит примерно так:

запрос человека → ответ LLM → запрос человека → ответ LLM → ...

В этой традиционной схеме ваш запрос, хоть и выглядит как простой диалог, может запустить несколько скрытых процессов на заднем плане, прежде чем вы получите ответ. Например, в схеме RAG к запросу добавляются данные из подключённого источника, который был заранее настроен администратором. Но даже в таком варианте информация, которой располагает LLM, всё равно заранее определена её создателем — скажем, через подключение к векторной базе вроде Chroma. В такой неагентной архитектуре у LLM нет возможности действовать самостоятельно за кулисами — она просто отвечает вам в режиме «вопрос — ответ», и всё.

А вот в агентной архитектуре всё иначе: LLM получают больше свободы и возможностей. В этой схеме модель может самостоятельно определить, какая информация ей нужна, чтобы выполнить задачу и достичь цели. Например, «Разработай план по повышению NPS в сервисном центре моего автосалона». В этом случае LLM получает доступ к инструментам (об этом чуть позже), которые можно вызывать на заднем плане — чтобы получить актуальные данные, оптимизировать рабочий процесс, разбить задачу на части и даже вызвать какой-нибудь скрипт (например, на VBScript), чтобы построить диаграммы в PowerPoint на основе собранной информации. Всё это делает агент (или несколько агентов) самостоятельно, чтобы достичь результата. Агентный процесс может выглядеть так:

запрос человека → ответ основной LLM (не виден пользователю) → вызов инструмента основной LLM (не виден пользователю) → ответ LLM (не виден пользователю, передаётся вспомогательной LLM) → ответ вспомогательной LLM (не виден пользователю, передаётся обратно основной) → финальный ответ основной LLM (виден пользователю) → следующий запрос человека → ...

С точки зрения пользователя может казаться, что вы просто общаетесь с каким-то универсальным супер-ИИ. Но на самом деле за кулисами работает целая система — несколько больших и маленьких моделей, которые совместно решают вашу задачу. (Как уже упоминалось, в агентной схеме можно использовать несколько LLM. Это хорошо показывает, насколько важную роль здесь могут играть небольшие модели — SLM.)

ИИ-агенты могут выполнять гораздо больше, чем просто понимать и генерировать текст. Они умеют принимать решения, решать задачи, взаимодействовать с внешними системами и выполнять действия. Эти агенты находят применение в самых разных сферах: от проектирования ПО и автоматизации ИТ-процессов до генерации кода и интеллектуальных помощников. Мы любим думать об агентах как о цифровых стажерах с большими амбициями. Дайте им цель, инструменты и задачи — и часто они смогут удивить вас своей сообразительностью. Но, как мы уже говорили, ИИ — это не магия.

### Как вам такой агент в действии?

ИИ-агенты — это системные реализации LLM, которые используют планирование, рассуждение и вызов инструментов, чтобы решать задачи и взаимодействовать с внешними средами. За кулисами может работать одна модель, несколько экземпляров одной и той же модели или их комбинация. Хорошая агентная платформа позволяет вам комбинировать разных провайдеров LLM, включая дообученные модели, настраиваемые под ваши данные. Например, вы можете использовать Claude Sonnet от Anthropic для управления рабочим столом, а рядом с ним — модель на базе Granite, обученную на данных вашего бизнеса. Вместе они могут, например, распознать событие и заполнить форму. Звучит круто!

На рис. 7.6 показан пример агента, которому мы поручили написать блог о влиянии инфляции на рынок жилья в Канаде в 2024 году, а затем — составить посты в соцсетях с отсылкой к этому блогу. Мы запустили несколько агентов для выполнения задачи. Один из них выступал в роли ведущего

аналитика рынка. Мы не будем подробно описывать каждого агента, но задача этого конкретного заключалась в том, чтобы в реальном времени анализировать финансовые новости по заданной теме и помогать в создании контента. Мы также прописали для него «биографию»: он представлялся аналитиком из солидной компании, который разбирает рыночные тренды и делится ими с другими агентами — «писателями». Всю эту информацию мы задали в YAML-файлах.

Обратите внимание на рис. 7.6 агент в образе аналитика прямо говорит, что начнёт работу с поиска релевантных статей в интернете.

```
# Agent: Lead Market Analyst
## Thought: Thought: Чтобы отслеживать и анализировать последние новости и обновления, касающиеся финансовых рынков, особенно инфляции в Канаде и её влияния на цены на жильё в 2024 году, я сначала выполняю поиск релевантных новостных статей и сайтов в интернете, чтобы собрать актуальную информацию.
## Using tool: Поиск в интернете
## Tool Input:
{"search_query": "Inflation in Canada 2024 impact on housing prices"}
## Tool Output:
Результаты поиска:
Заголовок: Обзор рынка жилой недвижимости Канады 2024
Ссылка: https://www.globalpropertyguide.com/north-america/canada/price-history
Фрагмент: В сентябре 2024 года средняя запрашиваемая арендная плата по всем типам жилья показала рост на 2.1% в годовом выражении — по сравнению с 9.3% в мае 2024 года и...
Заголовок: Влияет ли инфляция на рынок жилья в Канаде?
Ссылка: https://rocketmortgage.ca/learning-centre/home-buying/housing-market-inflation/
Фрагмент: Опрос 16 экономистов, проведённый в августе, показал, что после ожидаемого роста на 16% в этом году средние цены на жильё по стране, как ожидается, вырастут лишь на 3.2% в следующем...
```

Рис. 7.6. Агентный рабочий процесс размышляет над тем, какие шаги нужно предпринять, чтобы написать блог и подготовить ссылки на него для соцсетей

Как видно на рис. 7.7, если задача сложная, агент может несколько раз обращаться к инструментам и делать внутренние логические шаги, прежде чем выдать итоговый ответ. В этом случае агент уже нашёл источники и теперь начинает анализировать собранную информацию. Обратите внимание — у него есть доступ к инструментам, которые помогают в этом процессе.

```
# Agent: Lead Market Analyst
## Thought: Thought: Теперь, когда я собрал релевантные новостные статьи и сайты, я проанализирую результаты поиска, чтобы выявить ключевые тренды, изменения в регулировании или значимые экономические показатели, непосредственно связанные с инфляцией в Канаде и её влиянием на цены на жильё в 2024 году.
## Using tool: Чтение содержимого сайта
## Tool Input:
{"website_url": "https://www.globalpropertyguide.com/north-america/canada/price-history"}
## Tool Output:
Пожалуйста, подождите... Включите JavaScript и cookies, чтобы продолжить
У вас есть доступ ТОЛЬКО к следующим инструментам, и вы НИКОГДА не должны придумывать инструменты, которых нет в списке:
Название инструмента: Search the internet(**kwargs: Any) → Any
Описание: Поиск в интернете (search_query: 'строка') — инструмент, позволяющий искать информацию в интернете по ключевому запросу
Аргументы: {'search_query': {'description': 'Обязательный поисковый запрос для использования в интернете', 'title': 'Search Query', 'type': 'string'}}
Название инструмента: Read website content(**kwargs: Any) → Any
```

Рис. 7.7. Агент начинает анализировать содержимое найденной информации

Наконец, агент завершает работу и представляет результат, показанный на рис. 7.8. Очевидно, что рабочий процесс агента собрал исходные данные и ключевые тезисы, которые послужат основой для хорошей статьи в блоге.

# Agent: Lead Market Analyst

## Final Answer:

**\*\*Подробный обзор: Инфляция в Канаде и ее влияние на цены на жилье в 2024 году\*\***

**\*\*Ключевые тенденции и выводы:\*\***

1. **\*\*Уровень инфляции:\*\*** В октябре 2024 года инфляция в Канаде снизилась до нового минимума в 1.6%, по сравнению с 2.0% в августе. Это первый случай в 2024 году, когда инфляция опустилась ниже 2%.

2. **\*\*Цены на жилье:\*\*** Средняя цена продажи дома в Канаде снизилась на 2.7% по сравнению с прошлым годом и составила \$707,700 в октябре 2024. Средняя цена на отдельный дом также снизилась на 2.7% — до \$838,400.

3. **\*\*Прогноз рынка жилья:\*\*** Канадская ассоциация недвижимости (CREA) пересмотрела свой прогноз по рынку жилья на 2024 год, ожидая теперь годовой рост на 0.9% — до \$683,200, вместо прежнего прогноза роста на 2.5%.

4. **\*\*Процентные ставки:\*\*** Процентные ставки начали снижаться, что может способствовать устойчивому росту продаж и средней цены домов на вторичном рынке.

5. **\*\*Доступность жилья:\*\*** Кризис доступности жилья в Канаде может сохраняться в ближайшие годы, несмотря на снижение ставок: с апреля 2020 года цены на дома в Канаде в среднем выросли более чем на 30%.

**\*\*Потенциальные последствия для рынка и стратегии контента:\*\***

Рис. 7.8. Ключевые тезисы для публикации в блоге

В результате такой подход даёт гораздо больше гибкости — модели, стоящие за агентами, получают возможность планировать задачи, искать внешнюю информацию и делать многое другое.

Как мы уже упоминали, на заднем плане мы создали несколько агентов — каждый специалист в своей области — и задействовали их всех в рамках одной цели. Один агент был в роли контент-креатора, другой — креативного директора, третий — эксперта по соцсетям, и наконец — «математический гур». Мы рекомендуем, когда будете создавать своих цифровых сотрудников в агентных сценариях, ориентироваться на реальные вакансии: в них вы найдёте все необходимые навыки, которые стоит заложить в таких «цифровых работниках». В результате наши агенты написали (по крайней мере, судя по собранным данным) качественную и хорошо обоснованную статью:

### Введение

По мере того как инфляция в Канаде продолжает влиять на различные секторы экономики, включая рынок жилья, потенциальные покупатели и нынешние владельцы недвижимости сталкиваются с непростой ситуацией. Понимание динамики инфляции и ее влияния на цены на жилье особенно важно для принятия взвешенных решений в 2024 году. Цель этого блога — поделиться актуальными данными и тенденциями, касающимися рынка жилья и инфляции.

Жилищный рынок Канады пережил значительные колебания под воздействием инфляционного давления. С учетом внешних факторов, таких как процентные ставки и государственные инициативы, информированность покупателей и продавцов сегодня важна как никогда. В этой статье мы разберем текущее состояние инфляции, ее последствия для цен на жилье и то, как людям адаптироваться к быстро меняющейся ситуации на рынке.

---

### Текущая экономическая ситуация

Последние отчеты показывают, что уровень инфляции в Канаде стабилизировался на отметке около 2% по состоянию на август 2024 года. Тем не менее последствия прежнего инфляционного давления продолжают сказываться на экономике и рынке жилья. Банк Канады скорректировал процентные ставки в ответ на инфляцию, стремясь снизить растущую финансовую нагрузку на потребителей.

Стабилизация инфляции — положительный сигнал для экономики. Однако важно отметить, что последствия предыдущих инфляционных волн могут сохраняться, продолжая оказывать давление на рынок недвижимости. По мере того как Банк Канады пересматривает свою денежно-кредитную политику, потенциальные изменения процентных ставок могут дополнительно повлиять на цены на жилье и поведение покупателей.

Рис. 7.9. Начало финальной версии блога<sup>2</sup>

И наконец, посмотрите на сообщения для соцсетей, которые придумали наши агенты, чтобы продвинуть блог:

#### Twitter

🏠💰 По мере того как инфляция продолжает влиять на цены на жилье в Канаде в 2024 году, потенциальным покупателям и инвесторам важно быть в курсе событий! #Инфляция #РынокЖилья #Канада

---

#### LinkedIn

Понимание инфляции и ее влияния на рынок жилья в Канаде — ключ к принятию верных инвестиционных решений в 2024 году. Узнайте важные инсайты и стратегии, которые помогут вам сориентироваться в быстро меняющейся ситуации. #Недвижимость #Инвестиции #Канада

---

#### Facebook

Меняет ли инфляция рынок жилья в Канаде? Присоединяйтесь к нам — вместе разберемся в динамике инфляции и цен на жилье в 2024 году!  
#ЦеныНаЖилье #УмныеИнвестиции #Канада

Рис. 7.10. Агентный рабочий процесс не только написал блог, но и составил сообщения для соцсетей, чтобы направить трафик к публикации

Признаемся честно: глядя на результаты на рис. 7.10, мы немного поленились. Почему? Мы дали нашему агенту для соцсетей одни и те же навыки для всех платформ. Сейчас понимаем — стоило сделать его более специализированным, чтобы он мог варьировать стиль и тон в зависимости от

---

<sup>2</sup> На мой взгляд, неудачный пример. Сплошное «масло масляное». Мой опыт говорит, что LLM нельзя разрешать использовать введение. Оно мало чем отличается от последующего текста.

площадки. Ведь, например, X (Twitter) ограничивает посты 240 символами, и наш агент старался писать коротко (что, впрочем, могло быть частью задачи, но не было). С другой стороны, посты в Instagram вполне могут быть неформальными, в отличие от LinkedIn. Обратите внимание: на рис. 7.10 агент использует эмодзи в сообщении для X — это там норма, в отличие от LinkedIn.

За кулисами происходило ещё много интересного, о чём мы не успели рассказать. Например, у наших агентов была своя версия «звонка другу» из шоу *Кто хочет стать миллионером?* — только друзьями были веб-краулеры, поисковики, скраперы, куски Python-кода (мы, кстати, использовали библиотеку Pydantic для разбора данных) и другие цифровые помощники. Главное — никто из них не говорит: «Извини, брат, ты меня озадачил!» и не кладёт трубку.

Считаем ли мы, что рис. 7.9 и 7.10 лучше, чем если бы это сделал человек? Вовсе нет — и не в этом дело. Мы уверены: если дать человеку такую заготовку, его продуктивность резко возрастет. Мы просто сдвинули работу «влево»! А теперь добавьте человеческий штрих, чтобы результат по-настоящему заиграл.

### Чуть больше об агентах

В агентной системе агенту часто доступен более продвинутый контекст, в том числе буферы памяти, в которых хранится информация о прошлых задачах и действиях. Способность агента запоминать взаимодействия и планировать будущее делает работу с ним более персонализированной и дает возможность выдавать развернутые ответы. Но на этом всё не заканчивается — агенты учатся со временем. Например, если вам нужен отчет в определенном стиле или пост для Instagram должен быть задорнее, чем для LinkedIn, агент может запомнить эти предпочтения — и это отличный пример индивидуального подхода и полноценного ответа. В примере выше, если бы мы отдельно указали, что не хотим, чтобы пост в блоге был разбит на слишком много мелких разделов, агент запомнил бы и это. Сравните это с обычным чат-ботом на основе RAG, который каждый раз начинает с нуля.

Хотя агентам и предоставлена автономия в принятии решений, как уже упоминалось, им все же нужны цели и среды, заданные человеком. На поведение автономного агента влияют четыре ключевых фактора:

- Команда, которая разрабатывает и обучает (или, чаще, использует или дообучает) языковые модели, лежащие в основе агентной архитектуры. Как вы уже поняли из книги, чаще всего вы используете языковую модель для поддержки агентов, созданных кем-то другим, и в зависимости от задачи вы можете адаптировать ее под свой бизнес.
- Инженеры, создающие саму агентную систему. Именно они определяют, какие инструменты будут доступны агенту.
- Разработчики, которые настраивают агента и обеспечивают его взаимодействие с пользователем и инструментами. Они работают вместе с бизнесом, чтобы создать нужный образ агента.
- Пользователь, который ставит перед агентом конкретные цели и задачи.

Как видно из примера, опираясь на цели пользователя и доступные инструменты, агентная система сформировала план с задачами и подзадачами, чтобы достичь сложной цели. Если бы задача была простой (например, написать шаблонное письмо), планирование не потребовалось бы. Агент мог бы просто пошагово улучшать свой ответ без составления плана. Но в случае с нашим блогом этого было недостаточно. Помните рис. 7.6 и 7.8? Там было показано, как агент размышлял и строил план, чтобы решить поставленную задачу (на самом деле, за кадром осталось ещё больше размышлений и планирования).

Агенты действуют на основе информации, которую получают из среды. Часто у них нет полного объема знаний, необходимых для решения всех подзадач в рамках сложной цели. Например, нашим агентам не хватало информации о влиянии инфляции на рынок жилья. Чтобы восполнить этот пробел, они воспользовались доступными инструментами — в данном случае один из агентов провел поиск в интернете. Эти инструменты могут включать внешние базы данных, веб-поиск, API и даже других агентов. Получив нужную информацию, агент обновляет свои знания. Это значит, что на каждом этапе агент может пересмотреть план действий и скорректировать его.

Если раньше мы говорили о написании поста, представьте теперь нечто более сложное — например, планирование отпуска. Вы поручаете агенту предсказать, на какой неделе следующего года будет наилучшая погода для серфинга на Гавайях. Языковая модель, лежащая в основе агента, не

специализируется на погоде, поэтому агент обращается к внешней базе данных, где собраны ежедневные метеоотчеты по Гавайям за последние годы.

Однако даже после этого агент не может точно определить, какая погода лучше всего подходит для серфинга, и создает новую подзадачу. Для нее он связывается с другим агентом, который специализируется на серфинге. Допустим, он узнает, что для идеального катания нужны не только солнечные дни, но и высокие приливы при минимальных осадках. Объединив эти данные, агент анализирует погодные шаблоны и выдает прогноз: в какие недели года можно рассчитывать на высокие волны, солнце и отсутствие дождя. Возможно, он даже сам забронирует вам поездку — чтобы в отпуске было по-настоящему *maika'i loa* (по-гавайски — "классно").

### Как создаются агенты

В основе своей агенты — это системные реализации языковой модели. В такой архитектуре используется LLM, снабжённая набором инструкций: как планировать действия и как вызывать внешние инструменты (будь то веб-поиск, запрос к другому LLM и т.п.). Всё это встроено в более широкую систему, которая берёт на себя ключевые задачи, не связанные напрямую с генеративным ИИ, например:

- разбор вывода LLM, поиск вызовов инструментов, которые должна инициировать модель
- обращение к внешнему API на основе найденного вызова
- обработка ответа инструмента и его подстановка обратно в историю диалога с LLM с нужным форматированием (например, преобразование JSON в обычный текст или Markdown)
- управление расширенными функциями памяти: хранение истории общения и ключевых артефактов, доступных для LLM

Как видно, модель работает внутри сложной системы, и часто это требует многостраничных промптов с инструкциями по эксплуатации агента (или группы агентов).

Хотя универсального формата для таких промптов не существует, со временем появилось несколько рабочих парадигм — так называемых архитектур агентов. Эти подходы позволяют решать задачи в несколько этапов и управлять планированием, использованием инструментов и памятью внутри рабочей схемы LLM.

### ReAct (Reasoning and Action, рассуждение и действие)

Это как раз та архитектура агента, которую мы использовали в примере с блогом. Она позволяет на каждом этапе побуждать агента "думать" и планировать следующие действия в зависимости от полученного результата. После каждого действия и каждого ответа от инструмента агент заново оценивает ситуацию и решает, какой шаг будет следующим.

Такой цикл "подумал — сделал — наблюдает результат" помогает шаг за шагом решать сложные задачи и улучшать ответ.

С помощью структуры промпта можно задать агенту установку размышлять медленно и проговаривать свои мысли (вы уже видели это в примере с блогом). Такой вербальный процесс помогает понять, как агент формулирует выводы. В этой архитектуре агент постоянно обновляет свой контекст на основе новых размышлений — по сути, это вариант приёма под названием *chain of thought*, или цепочка рассуждений.

### ReWOO (Reasoning WithOut Observation, рассуждение без наблюдения)

Метод ReWOO, в отличие от ReAct (рассуждение и действие), строит весь план заранее. Такой подход может быть предпочтительным с точки зрения пользователя, потому что позволяет сначала утвердить план, прежде чем он начнет выполняться. Это важно: в какой-то момент ресурсы для выполнения задач придется оплачивать, и разумнее сначала понимать, что именно будет происходить и каким образом, прежде чем тратить деньги.

Архитектура ReWOO состоит из трех этапов. На этапе планирования агент продумывает шаги, опираясь на запрос пользователя. Далее он собирает результаты работы нужных инструментов. Наконец, он сопоставляет изначальный план с результатами работы инструментов и формирует итоговый ответ. Такое предварительное планирование позволяет снизить расход токенов, упростить вычисления и уменьшить риск ошибок на промежуточных этапах — например, если один из инструментов даст сбой.

## Риски и ограничения агентных систем

Агентные системы несут в себе те же риски, что и генеративный ИИ: это и предвзятость, и галлюцинации, и взломы промптов. Но помимо них есть и свои специфические уязвимости, о которых важно знать, если вы собираетесь внедрять такие решения. Именно поэтому мы написали этот раздел.

### Вычислительная сложность и бесконечные циклы

Поскольку агенты часто делают несколько последовательных запросов к модели, чтобы ответить на один пользовательский промпт, это может оказаться слишком ресурсоемким, особенно для простых задач обработки текста. Иногда проще и дешевле просто использовать обычный запрос к языковой модели без подключения всей агентной системы.

Кроме того, агент, который не умеет планировать или осмысливать результаты своей работы, может застрять в повторяющихся действиях — например, снова и снова вызывать один и тот же инструмент. Это может привести к бесконечному циклу. Если такой цикл оставить без контроля, и при этом он задействует большую модель, стоимость может вырасти в разы. И это не гипотетика — мы действительно видели такие случаи.

В начале наших экспериментов мы попросили агента найти самый лучший рецепт дзадзики в мире. Мы рассчитывали, что он найдет списки победителей, сравнит сайты по количеству посещений или узнаваемости домена. В итоге агент утонул в море противоречивых блогов и рекомендаций "побольше чеснока" (потому что ИИ-то знает, где волшебство дзадзики прячется) — и так и не выдал никакого финального "Опа!".

### Контроль и наблюдаемость

Гибкость, которая позволяет агентам справляться с новыми задачами и решать сложные проблемы, достигается за счёт снижения контроля над системой. Поэтому крайне важно наблюдать за процессом принятия решений агентом и анализировать его действия в рамках рабочей схемы. В зависимости от того, как реализован агент, его внутренняя логика может быть не до конца прозрачна — и это чревато неожиданными последствиями. Модель может адаптироваться непредсказуемо, и её поведение в какой-то момент окажется не в соответствии с вашими изначальными целями или ценностями.

Отсутствие контроля и прозрачности может привести к последствиям, о которых мы говорили в главе 5 — например, к предвзятым или дискриминационным действиям. А в таких сферах, как здравоохранение, финансы или образование, где ставки особенно высоки, это может обернуться серьезными последствиями. Поэтому, двигаясь в этом направлении, не забывайте, насколько важно разрабатывать требования к прозрачности и объяснимости LLM, чтобы была возможность отслеживать их действия в реальном времени и при необходимости вовремя вмешиваться.

### Безопасность и сложные уровни доступа

Прежде чем внедрять собственных (а иногда и покупных) интеллектуальных агентов в сложные корпоративные системы, нужно решить целый ряд задач, связанных с безопасностью и защитой данных. Допустим, у вас есть агент в HR-отделе, который обрабатывает запросы сотрудников и имеет доступ к базе данных с чувствительной информацией. В этом случае нужно принять меры, чтобы агент не только не раскрыл конфиденциальные сведения о других сотрудниках, но и вообще не получил к ним доступ. Для этого нужны точечные механизмы управления доступом (FGAC), ролевой контроль (RBAC), соблюдение правил передачи персональных данных (PII), принцип минимально необходимого доступа, единая система идентификации и многое другое.

Если же речь идет о системе, где взаимодействуют несколько агентов, нужно заранее определить протоколы общения между ними, особенно если у них есть доступ к разным видам конфиденциальной информации. Важно, чтобы в этих взаимодействиях не происходила утечка данных, а передача шла с шифрованием и в соответствии с нормами.

### Три совета на старте: лучшие практики агентных систем

Когда сталкиваешься с чем-то новым, всегда полезно иметь под рукой несколько советов. В этой части мы собрали рекомендации с помощью специалистов IBM, в том числе Анны Гутовской, которая как раз и обучает агентные системы: умные настолько, чтобы справляться с задачами, но не настолько, чтобы начинать творить что попало. Прислушавшись к этим советам, вы сделаете свои агентные системы не просто эффективными, но и надежными.



**Журналы активности.** Чтобы разобраться, почему агент повел себя так, а не иначе, полезно дать пользователю доступ к журналу его действий. Там можно увидеть, какие инструменты использовались, какие шаги предпринимались, и как агент пришел к своему решению. Это повышает прозрачность, помогает выявлять ошибки и формирует доверие.

**Возможность прерывания и наблюдение в процессе.** Чтобы агенты не зависали в бесконечных петлях или не начинали работать некорректно из-за ошибок в проектировании, важно уметь их вовремя остановить. Это достигается за счет так называемой "прерываемости": агента можно остановить вручную или через внешний менеджер ресурсов, например, Turbonomic. А чтобы прерывание было действительно полезным, нужно заранее встроить систему наблюдения, чтобы понимать, на каком этапе находится агент и что пошло не так.

**Человеческий надзор.** Особенно в начале, когда агенты только осваиваются в новой среде, полезно давать им обратную связь от человека. Это помогает скорректировать поведение и приблизить его к ожидаемому. Например, вы можете настроить систему так, чтобы по завершении каждой задачи агент запрашивал фидбэк — это даст возможность подсказать, что он упустил или мог бы сделать лучше. На рис. 7.10 как раз был случай, где такой подход помог бы: просматривая посты в соцсетях, мы могли бы скорректировать их стиль и подачу под нужную аудиторию. Это же замечание можно затем передать LLM, которая извлечет полезные советы и положит их в "память" агента для будущей работы. Можно даже использовать ИИ, чтобы помочь ИИ: создать модель-судью, которая будет оценивать результаты работы агентов.

Кроме того, хорошей практикой считается запрашивать подтверждение от человека перед выполнением особо значимых действий. Например, массовая рассылка писем или сделки на финансовых рынках должны происходить только после одобрения человеком. В таких чувствительных сферах полностью полагаться на автоматику не стоит.

#### Что важно помнить об агентных системах

Агентные системы набирают популярность не просто так — они действительно повышают устойчивость и эффективность ИИ в решении сложных задач. Умение планировать, привлекать свежие данные через внешние инструменты и разбивать задачи на мелкие шаги делает их особенно ценными. Это, кстати, открывает новые возможности даже для более легких и открытых моделей. Но в любом случае, технологии продолжают развиваться, и, несмотря на все преимущества, не забывайте про безопасность, защиту данных и издержки, прежде чем использовать такие агенты в реальных продуктах.

#### Подводим итоги

Сегодня в мире ИИ происходит масса интересного. И мы надеемся, что вы убедились: независимо от того, идет ли речь о новых малых языковых моделях (SLMs) или о системных подходах к ИИ, вроде маршрутизации моделей, MoE или агентных систем, — одна модель не справится со всем. В следующей главе мы расскажем, как компании могут развивать эти системные подходы, обучая малые модели на собственных данных.