1 Анализ данных с помощью больших языковых моделей

В этой главе

- ✓ Введение в языковые модели.
- ✓ Анализ данных с помощью языковых моделей.
- ✓ Эффективное использование языковых моделей.

Языковые модели — это мощные нейронные сети, которые можно использовать для решения различных задач по обработке данных. В этой главе вы познакомитесь с такими моделями и узнаете, как и зачем их применяют для анализа данных.

1.1. На что способны языковые модели

Начнем раздел с небольшого стихотворения и связанного с ним рисунка (рис. 1.1), сочетающего в себе две главные темы этой книги — анализ данных и большие языковые модели.

Стихотворение и иллюстрация были сгенерированы GPT-4o (o означает omni[1]), языковой моделью компании OpenAI, которая обрабатывает мультимодальные данные, исключительно на основе инструкций: «Напиши стихотворение, используя слова “анализ данных” и “большие языковые модели”», а затем «Теперь нарисуй соответствующую картинку!». И рисунок, и стихотворение, похоже,

[1] От лат. omnis — «всеобъемлющий», «универсальный». — *Примеч. пер.*

относятся к заданной теме. Конечно, стихотворение вряд ли может рассчитывать на литературную премию, но его текст связный, имеет структуру, характерную для стихотворения, и рифмуется!

В ночной тиши, при свете серверов
Потоки данных мчатся без оков.
Столбцы и строки — выверенный строй,
В них смысл сокрыт, он встречи ждет с тобой.

В числах — эхо прожитых времен,
Закономерностей уверенный закон.
И вот приходит новый искатель —
Языковая модель, прилежный читатель.

Внимает всей мощью, что схемы вложили,
Чтоб факты сухие свой смысл раскрыли.
От рассеянных чисел — к четким фразам,
Лишь язык данных ей близок и ясен.

Здесь цифры кружат, текст вступает в спор,
Где код и смысл ведут единый разговор.
Два мира слиты в связке без границ —
Логика и речь без пауз и страниц.

В таком союзе, выверенном, строгом,
У данных смысл становится итогом.
В глубинах сети, где импульсы бегут,
Анализ и язык единый путь найдут.

Рис. 1.1. Иллюстрация GPT-4o, связывающая понятия «анализ данных» и «большие языковые модели»

Пожалуй, самое главное, что для создания стихотворения и картинки потребовались лишь короткие инструкции, написанные на естественном языке. И если раньше для методов машинного обучения нужны были большие объемы обучающих данных, характерные для конкретной задачи, то теперь это требование устарело. И конечно, задача достаточно конкретная, чтобы убедить нас в том, что языковая модель не копирует существующие решения из Интернета, а генерирует уникальный контент.

Написание стихотворений и создание изображений — лишь два из множества возможных вариантов использования языковых моделей (хотя, пожалуй, самые интересные). Модели наподобие GPT-4o способны решать различные задачи, такие как краткое изложение текстовых документов, написание программного кода и ответы на вопросы о картинках. В этой книге вы узнаете, как использовать языковые модели для решения множества задач анализа данных, начиная с извлечения информации из больших коллекций текстовых документов и заканчивая написанием кода для анализа данных. Прочитав ее, вы сможете быстро создавать конвейеры анализа данных, основанные на языковых моделях, и извлекать полезные сведения из различных форматов данных.

Что означает GPT

Аббревиатура GPT (от Generative Pretrained Transformer) расшифровывается как «генеративный предварительно обученный трансформер».

Генеративный: GPT — это большая нейронная сеть, которая генерирует содержимое (например, текст или код) в ответ на входной текст. Этот факт отличает ее от других нейронных сетей, которые, например, могут только классифицировать входной текст, используя фиксированный набор заранее определенных категорий.

Предварительно обученный: GPT предварительно обучается на больших объемах данных, решая типовые задачи, такие как предсказание следующего слова в тексте. Как правило, задача предварительного обучения отличается от задач, для которых обычно используется модель. Однако предварительное обучение помогает быстрее осваивать более специализированные задачи.

Трансформер — это новая архитектура нейронной сети, которая особенно полезна для обучения задачам, где входные и выходные данные имеют переменную длину (например, текстовые документы). На данный момент это доминирующая архитектура для подходов, используемых в области генеративного ИИ.

1.2. Чему вы научитесь

Эта книга посвящена тому, как с помощью языковых моделей решать задачи анализа данных. Такие задачи можно классифицировать по типу изучаемых данных и по типу анализа. В книге рассматривается широкий спектр типов данных и задач анализа.

Мы сосредоточимся на *мультимодальном* анализе, то есть на использовании языковых моделей для анализа различных типов данных. Конкретно в этой книге рассматриваются следующие типы данных.

- *Текст*. Это могут быть электронные письма, газетные статьи и комментарии на веб-форуме. Текстовые данные распространены повсеместно и содержат ценную информацию. В книге мы рассмотрим, как использовать языковые модели для автоматической классификации текстовых документов на основе их содержания, как извлекать из текста конкретные фрагменты информации и как группировать текстовые документы по смежным темам.
- *Изображения*. Как говорится, одна картинка стоит тысячи слов. Изображения помогают нам понять сложные концепции, запечатлеть приятные воспоминания о последнем отпуске и проиллюстрировать текущие события. Языковые модели способны легко извлекать информацию из изображений. Например, мы будем использовать модели для ответа на произвольные вопросы об изображениях или для идентификации людей на фотографиях на основе базы данных профилей.
- *Видеоролики*. Значительную часть данных в Интернете составляют видеоданные. Даже на вашем смартфоне они, скорее всего, занимают немалую часть общего объема памяти. Книга покажет, что языковые модели можно применять и для анализа видеороликов: например, для генерации подходящих названий на основе содержания.
- *Аудиозаписи*. Для многих людей речь является наиболее естественной формой общения. Аудиозаписи фиксируют монологи и беседы, а также дополняют видеоролики. В книге мы рассмотрим, как расшифровывать аудиозаписи, как переводить разговорную речь на другие языки, а также как создать интерфейс запросов, который будет отвечать на устные вопросы о данных.
- *Таблицы*. Для примера возьмем набор данных, содержащий информацию о клиентах. Эти данные удобнее всего представлять в виде таблицы: в столбцах хранится информация об адресе, номере телефона и кредитной карте, а в строках — сведения о разных клиентах. В книге мы рассмотрим, как использовать языковые модели для написания кода, выполняющего сложные операции с такими табличными данными.
- *Графы*. Многие наборы данных (от социальных сетей до сетей метрополитена) удобно представлять в виде графов, моделирующих сущности (например, людей или станции метро) и их связи (например, дружеские отношения или пересадки). Мы посмотрим, как можно использовать языковые модели для создания кода, анализирующего большие графы различными способами.

В книге мы в основном будем работать с моделями компании OpenAI, используя ее же библиотеку для Python. Ближе к концу книги мы рассмотрим языковые модели и других поставщиков. Поскольку у библиотек разных поставщиков обычно схожая функциональность, то переход на другие модели не займет много времени.

Структурированные и неструктурированные данные

Типы данных часто делят на две группы: *структурированные* и *неструктурированные данные*. У структурированных есть структура, которая облегчает их эффективную обработку с помощью специализированных инструментов. Примеры структурированных данных — таблицы и графы. При работе с такими данными языковую модель обычно используют как интерфейс к специализированным инструментам обработки данных. Неструктурированные данные — такие как текст, изображения, видео- и аудиофайлы — не имеют структуры, которую можно было бы легко использовать для эффективной обработки. Поэтому при взаимодействии с неструктурированными данными языковую модель обычно приходится применять непосредственно к самим данным.

Как правило, использование языковых моделей влечет за собой денежные расходы, пропорциональные объему обрабатываемых данных. Плата зависит от используемой языковой модели, ее конфигурации, а также от того, как сформулированы входные данные для этой модели. В этой книге вы не только научитесь решать различные задачи анализа данных с помощью языковых моделей, но и узнаете, как делать это с минимальными затратами.

1.3. Как использовать языковые модели

Работа с современными языковыми моделями строится на методе *составления промптов (запросов)*. В этом разделе мы обсудим сам метод, а также интерфейсы, которые можно для этого использовать.

1.3.1. Составление промптов

Всего несколько лет назад модели машинного обучения обучали для решения одной конкретной задачи. Например, модель могла классифицировать текст отзыва как «положительный» (то есть автор отзыва доволен) или «отрицательный» (то есть автор недоволен). При использовании этой модели в качестве входных данных достаточно было ввести текст отзыва. Не нужно было описывать во входных данных задачу (классификация отзыва), поскольку модель была специально обучена выполнять только ее.

Ситуация изменилась в последние годы с появлением больших языковых моделей наподобие GPT. Такие модели больше не обучаются для решения конкретных задач. Вместо этого они призваны служить универсальными инструментами, которые в принципе способны справиться с любой задачей по желанию пользователя. При работе с такой моделью пользователь должен четко описать, какие действия она должна выполнить.

Промпт представляет собой входные данные для языковой модели. Он может содержать мультимодальные данные: например, текст и изображения.

Как минимум чтобы заставить языковую модель решить конкретную задачу, промпт должен содержать текст с инструкциями о том, какие действия должна выполнить модель. Помимо инструкций, в промпте должен содержаться весь необходимый контекст.

Например, если в инструкции сказано определить, виден ли на картинке автомобиль, то промпт также должен содержать картинку. Инструкции должны быть конкретными и уточнять, к примеру, ожидаемый формат вывода. Например, мы хотим, чтобы модель выдавала 1 при наличии автомобиля и 0 — при его отсутствии, что позволит нам быстро сложить выведенные моделью числа и подсчитать количество автомобилей. Все это необходимо четко прописать в промпте (в противном случае модель может ответить: «Да, на картинке есть автомобиль», что усложнит подсчет на этапе последующей обработки). Помимо инструкций и контекста, промпт может содержать примеры, помогающие языковой модели понять задачу.

Обучение на нескольких примерах или без примеров

Чтобы помочь языковой модели лучше понять задачу, в качестве части промпта можно предоставить примеры. Они похожи на задачу, для решения которой планируется использовать модель. Примеры содержат входные данные и желаемый вывод. Такой подход иногда называется *обучением на нескольких примерах (few-shot learning)*, поскольку модель обучается задаче, используя некоторое количество примеров[1]. *Обучение без примеров (zero-shot learning)*, в свою очередь, означает, что модель будет изучать задачу, не имея ни одного примера, основываясь только на ее описании.

1.3.2. Пример промпта

Проиллюстрируем промпты с помощью примера. Классический сценарий использования языковых моделей — анализ отзывов о товаре с целью определить тональность отзыва: является ли он положительным (то есть покупатель рекомендует товар) или отрицательным (то есть клиент недоволен товаром). Предположим, у нас есть отзыв, который необходимо определить как положительный или отрицательный. При наличии специализированной модели, обученной классифицировать отзывы на конкретную интересующую нас категорию товаров, достаточно отправить отзыв в эту модель. Она специально обучена для решения конкретной задачи, поэтому уже «знает», что делать с входными данными и какой формат вывода нужен. Однако, поскольку мы используем большие языковые модели, вместе с отзывом необходимо предоставить дополнительный контекст.

---

[1] В задачах машинного обучения количество примеров обычно варьируется от 1 до 100. Для языковых моделей, из-за ограничений контекстного окна и роста стоимости запросов, чаще используется от 1 до 10 примеров. Частный случай с одним примером называется one-shot learning. — *Примеч. науч. ред.*

Промпт должен содержать всю необходимую модели информацию, описывая задачу, которую нужно решить, и весь контекст. В нашем примере, вероятно, стоит добавить в промпт следующую информацию:

- *текст отзыва*, который требуется классифицировать;
- *описание задачи*, которую нужно решить;
- *форматы вывода*, которые нам нужны;
- *релевантный контекст* — например, мы рассматриваем отзывы на ноутбуки или газонокосилки?

По желанию можно добавить несколько примеров отзывов, уже классифицированных должным образом. Это может помочь модели выполнить задачу более точно.

В промпте ниже содержится вся необходимая информация для примера отзыва (листинг 1.1).

Листинг 1.1. Составление промпта для классификации отзыва о ноутбуке

```
Мы будем рассматривать отзывы о товарах – ноутбуках.        ❶ Контекст
Для каждого отзыва выведи "Доволен" или "Недоволен" в зависимости от того,
доволен покупатель товаром или нет.        ❷ Описание задачи и формат вывода
Примеры:
Отличный ноутбук! Всем рекомендую к покупке!
Доволен                                    ❸ Первый пример
Ноутбук не работал. Пришлось его вернуть.
Недоволен                                  ❹ Второй пример
Экран слишком маленький, а операционная система очень
долго загружается.                         ❺ Отзыв
```

Промпт начинается с описания релевантного контекста ❶. Клиенты пишут отзывы о ноутбуках, поэтому, например, если они отмечают, что товар «тяжелый», то чаще всего это недостаток (в отличие от анализа отзывов, скажем, о паровых катках). В описании задачи ❷ содержится инструкция для модели, что делать с отзывами, и указывается желаемый формат вывода (вывести «Доволен» или «Недоволен»). Далее идет список примеров. Строго говоря, для такой простой задачи может и не потребоваться добавлять в промпт примеры. Однако это порой может повысить точность вывода. Здесь мы добавили два примера отзывов (❸ и ❹), а также указали желаемый вывод для этих отзывов. Наконец, мы добавили сам отзыв ❺, который модель должна классифицировать.

Учитывая предыдущую информацию в промпте, современные языковые модели, получив такой промпт, скорее всего, выдадут результат «Недоволен». Что, собственно, нам и требовалось получить.

1.3.3. Интерфейсы

Каким же образом отправлять промпты языковой модели? Поставщики наподобие OpenAI обычно предлагают веб-интерфейсы, позволяющие пользователям отправлять их языковым моделям отдельные промпты. В главе 2 мы будем пользоваться веб-интерфейсом OpenAI для отправки промптов, предписывающих модели проанализировать текст или написать код для обработки данных.

Веб-интерфейс работает хорошо до тех пор, пока в него отправляют всего несколько промптов. Однако для анализа большой коллекции текстовых документов потребуется отправить множество промптов (по одному на каждый текстовый документ). Очевидно, что вводить тысячи промптов вручную не хочется. Именно в такой ситуации и можно использовать библиотеку OpenAI на языке Python. Она позволяет отправлять промпты моделям OpenAI непосредственно из кода Python и работать с ответами модели в среде Python. Так можно автоматизировать загрузку данных, генерацию промптов и любую необходимую последующую обработку ответов модели. Кроме того, это позволяет объединять языковые модели с другими полезными инструментами: например, чтобы с помощью языковой модели писать код для обработки данных и тут же выполнять его, используя другие инструменты.

С работой библиотеки OpenAI для Python вы познакомитесь в главе 3, а затем будете использовать ее в других главах. Такие поставщики языковых моделей, как Google, Anthropic и Cohere, предлагают аналогичные Python-библиотеки для отправки промптов своим языковым моделям. Более подробно об этих библиотеках мы поговорим в главе 8.

1.4. Использование языковых моделей для анализа данных

Как же использовать языковые модели именно для анализа данных? В книге описываются два варианта.

- Во-первых, можно использовать языковую модель *непосредственно* для работы с данными. Это означает, что языковая модель получит данные, которые необходимо проанализировать, как часть промпта (вместе с инструкциями по выполнению анализа).
- Во-вторых, языковая модель может применяться для анализа данных *косвенно*. В этом случае она не видит сами данные: то есть они не добавляются в промпт. Вместо этого с помощью языковой модели пишется код для обработки данных, выполняемый в специализированных инструментах обработки.

Какой подход применить, зависит от свойств данных и задачи. Рассмотрим оба метода подробнее.

1.4.1. Использование языковых моделей непосредственно для работы с данными

Проще всего анализировать данные с помощью языковых моделей — поместив их непосредственно в промпт. Именно так мы поступили в подразделе 1.3.2. Чтобы проанализировать отзыв, мы добавили текст отзыва в промпт вместе с инструкциями о том, что с ним нужно сделать. Такой же подход можно применять и для работы с другими типами данных, нетекстовыми. Например, при использовании мультимодальных моделей наподобие GPT-4o, помимо инструкций по анализу, можно просто добавить в промпт сами изображения, которые необходимо проанализировать.

Как правило, требуется проанализировать не одно изображение или отзыв, а их набор. Например, предположим, что нам нужно классифицировать целый набор отзывов и определить для каждого из них, каким он является: положительным или отрицательным. В таких случаях обычно применяется следующий подход, реализованный на языке Python с помощью библиотеки OpenAI для Python (или эквивалентной библиотеки, позволяющей пользователям отправлять промпты другим моделям других провайдеров). Мы загружаем отзывы, которые требуется классифицировать, и генерируем по одному промпту для каждого отзыва. Затем отправляем эти промпты в языковую модель, извлекаем результаты классификации из ответов, сгенерированных моделью для каждого отзыва, и сохраняем результаты в файле на диске.

В этом сценарии требуется решить одну и ту же задачу (классификация отзывов) для нескольких текстовых документов (то есть отзывов). Как вы можете представить, промпты для разных отзывов должны быть схожими. Несмотря на то что текст отзыва, который нужно классифицировать, каждый раз меняется, описание задачи и другие части промпта остаются неизменными.

Для создания промпта на языке Python используется *шаблон промпта*. Этот шаблон определяет промпт, связанный с конкретной задачей, которую необходимо решить. В нашем примере мы используем шаблон промпта для разделения отзывов на две категории: положительные или отрицательные. Шаблон содержит метки-заполнители для обозначения частей промпта, которые изменяются в зависимости от входных данных. Учитывая, что наш шаблон предназначен для классификации отзывов, в нем нужно оставить место для текста отзыва. Затем при генерации промпта в Python мы заменим эту метку-заполнитель (placeholder) текстом отзыва, который требуется классифицировать.

Например, для классификации отзывов можно использовать следующий шаблон (листинг 1.2).

В этом шаблоне обобщен промпт, с помощью которого мы намеревались классифицировать один конкретный отзыв (см. листинг 1.1 в подразделе 1.3.2). Опять же, мы предоставляем контекст (сообщение, что классифицируются отзывы о ноутбуках) ❶ и инструкции, описывающие задачу, которую нужно

решить, а также формат вывода ❷. Кроме того, мы приводим несколько примеров отзывов с соответствующими результатами классификации (❸ и ❹). Несмотря на то что отзывы, которые требуется классифицировать, меняются в зависимости от исходных данных, примеры отзывов менять не нужно. Эти отзывы просто иллюстрируют, какую задачу должна решить языковая модель. Наконец, ❺ у нас есть место для текста отзыва. При итерациях с разными отзывами мы будем генерировать промпты для каждого из них, подставляя текст отзыва в метку-заполнитель.

Листинг 1.2. Шаблон промпта для классификации отзывов о ноутбуках

```
Мы будем рассматривать отзывы о товарах — ноутбуках.          ❶ Контекст
Для каждого отзыва выведи "Доволен" или "Недоволен" в зависимости от того,
доволен покупатель товаром или нет.
Примеры:                                    ❷ Описание задачи и формат вывода
Отличный ноутбук! Всем рекомендую к покупке!
Доволен                                     ❸ Первый пример
Ноутбук не работал. Пришлось его вернуть.
Недоволен                                   ❹ Второй пример
[Отзыв]                                     ❺ Метка-заполнитель для текста отзыва
```

В примере шаблона промпта есть только одна метка-заполнитель. Вообще говоря, разные элементы промпта могут варьироваться в зависимости от входных данных. В этом случае мы добавляем метки-заполнители для каждого элемента и при создании промптов заменяем их все.

На рис. 1.2 показано использование шаблонов промптов при анализе данных непосредственно с помощью языковых моделей. Для каждого элемента данных (например, отзыва, который нужно классифицировать) мы заменяем метки-заполнители в шаблоне, чтобы сгенерировать промпт (можно также сказать, что мы *создаем экземпляр* промпта). Затем мы отправляем этот промпт в языковую модель, чтобы она решила интересующую нас задачу анализа данных.

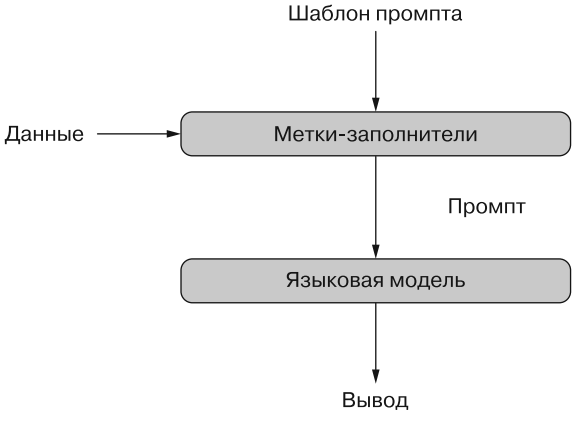

Рис. 1.2. Использование языковых моделей непосредственно для анализа данных. В шаблоне промпта описывается задача анализа. Он содержит метки-заполнители, которые заменяются данными для анализа. Когда замена выполнена, получившийся промпт передается в языковую модель, чтобы та могла создать вывод

1.4.2. Анализ данных с помощью внешних инструментов

Ввод данных непосредственно в шаблон не всегда самый эффективный вариант. Для работы с некоторыми типами данных предусмотрены специализированные инструменты. В таких случаях языковую модель зачастую лучше использовать для написания кода для обработки данных, чем непосредственно для анализа данных. Затем код, сгенерированный языковой моделью, можно выполнить с помощью специализированного инструмента.

Мы применим этот подход к структурированным данным. Для обработки табличных и графовых данных существуют специализированные инструменты, поддерживающие широкий спектр операций анализа. Такие операции, как фильтрация и агрегация данных, могут выполняться очень эффективно именно на структурированных данных. Даже если бы те же операции можно было надежно выполнять с помощью языковых моделей (а это не так), мы бы не стали их применять, поскольку оплата поставщикам наподобие компании OpenAI пропорциональна размеру входных данных. Обрабатывать большие структурированные наборы данных (например, таблицы с миллионами строк) с помощью языковых моделей непомерно дорого. В следующих главах мы рассмотрим несколько типов инструментов для обработки структурированных данных.

- *Реляционная система управления базами данных (РСУБД)* — хранит и обрабатывает реляционные данные, то есть коллекции таблиц с данными. Большинство РСУБД поддерживают язык структурированных запросов SQL. Мы применим языковые модели для перевода вопросов о данных в запросы на языке SQL.
- *Графовая СУБД* — работает с графовыми данными, представляющими сущности и отношения между ними. Различные системы управления графовыми данными поддерживают разные языки запросов. В главе 5 мы рассмотрим, как с помощью языковых моделей преобразовать вопросы о данных в запросы на языке *Cypher*, который поддерживается Neo4j — графовой СУБД.

Для примера предположим, что нам необходимо предоставить непрофессиональным пользователям возможность анализировать реляционную базу данных, то есть набор таблиц с данными. Возможно, таблица содержит результаты опроса и требуется обеспечить пользователям возможность агрегировать ответы различных групп респондентов. Результаты опроса хранятся в реляционной системе управления базами данных (наиболее подходящем инструменте для этого типа данных). Используя языковые модели, можно позволить пользователям задавать вопросы о данных на естественном языке (то есть на простом русском). Языковая модель позаботится о переводе этих вопросов в формальные запросы. Точнее, учитывая, что данные хранятся в реляционной системе управления базами данных, вопросы нужно будет преобразовать в SQL-запросы.

Чтобы решить эту задачу, снова применим шаблон промпта. В данном случае нас интересует перевод текста в SQL, то есть языковая модель будет использоваться для перевода вопросов на естественном языке в SQL-запросы. Задача (перевод текста в SQL) и данные (база, содержащая результаты опроса)

останутся неизменными, но вопросы пользователя с течением времени будут меняться. Поэтому на месте вопроса пользователя мы ставим в шаблоне промпта метку-заполнитель. Теоретически приведенный ниже шаблон (листинг 1.3) должен позволить преобразовывать вопросы, касающиеся данных опроса, в SQL-запросы.

Листинг 1.3. Шаблон промпта для перевода вопросов на язык SQL

```
База данных:                          ❶ Описание базы данных
База данных содержит результаты опроса, хранящиеся в таблице SurveyResults,
которая включает следующие столбцы: ...
Вопрос:                               ❷ Вопрос, который требуется перевести
[Вопрос]
Переведи вопрос на язык SQL!          ❸ Описание задачи
```

Сначала в промпте описывается структура данных ❶. Это необходимо для того, чтобы система могла писать корректные запросы (например, такие, которые будут ссылаться на корректные имена таблиц и столбцы в этих таблицах). Описание в примере шаблона сокращено. О том, как точно описать структуру реляционной базы данных, мы поговорим в следующих главах. Далее в шаблоне содержится вопрос, который требуется перевести ❷. Это метка-заполнитель, позволяющая пользователям задавать разные вопросы, используя один и тот же шаблон промпта. Наконец, шаблон содержит (краткое) описание задачи ❸: нужно перевести вопросы в SQL-запросы!

На рис. 1.3 представлен процесс перевода текста на язык SQL. Получив соответствующий шаблон промпта, мы заменяем метку-заполнитель вопросом пользователя, переводим вопрос в SQL-запрос с помощью языковой модели и, наконец, выполняем запрос в реляционной системе управления базами данных. Результат запроса показывается пользователю.

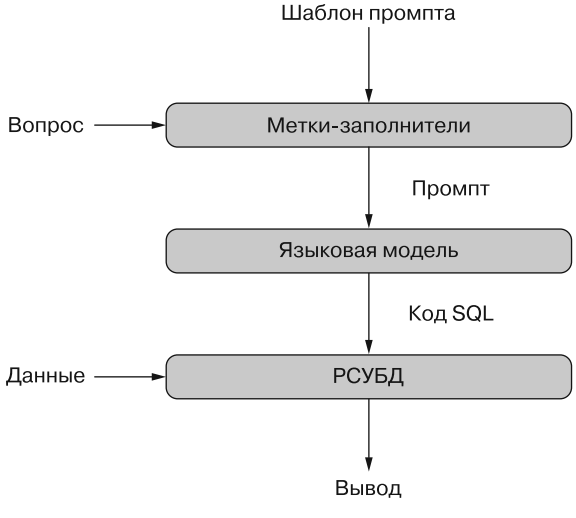

Рис. 1.3. Косвенное использование языковых моделей для создания интерфейса на естественном языке для табличных данных. Шаблон промпта содержит метки-заполнители для вопросов о данных. После того как они будут заменены вопросами, получившийся промпт используется в качестве входных данных для языковой модели. Она переводит вопрос в SQL-запрос, который выполняется через реляционную систему управления базами данных