10 Чат-боты

В этой главе мы познакомимся с одним из самых быстро развивающихся классов приложений обработки естественного языка — диалоговыми агентами. Диалоговые агенты, от Slackbot до Алексы и Dragon Drive компании BMW, быстро становятся неотъемлемой частью нашей повседневной жизни, внедряясь в самые разные ее аспекты. Они улучшают нашу жизнь, расширяя память (например, выполняя поиск в интернете), ускоряя вычисления (например, осуществляя преобразования или прокладывая маршруты) и обеспечивая более гибкую связь и управление (например, посылая сообщения и управляя системами умного дома).

Главным отличием таких агентов является не информация и не помощь, предоставляемая ими (как в давно существующих веб- и мобильных приложениях с интерфейсом «укажи и щелкни»), а интерфейс, делающий их такими привлекательными. Взаимодействие на естественном языке обеспечивает более гладкий, естественный и простой способ доступа к вычислительным ресурсам. По этой причине чат-боты являются важным шагом в развитии пользовательского интерфейса, позволяя встраивать в текстовые приложения команды на естественном языке и тем самым уменьшая неудобство интерфейсов на основе меню. Важно отметить, что они также открывают возможность взаимодействия человека с машиной в новых вычислительных контекстах, например, с экранными устройствами, плохо приспособленными для ввода, такими как автомобильные навигаторы.

Но почему бурное развитие протекает именно сейчас, с учетом долгой истории диалоговых агентов в реальности (начиная с ранних версий Eliza и PARRY) и в фантастических произведениях («Компьютер» из произведения *Star Trek* («Звездный путь») или «Hal» из *2001: A Space Odyssey* («2001 г.: Космическая одиссея»))? Отчасти потому, что для работы «сногсшибательных приложений»

с таким интерфейсом требуется повсеместное проникновение вычислительных устройств, которое стало возможно с современным интернетом. Но самое главное, потому, что современные диалоговые агенты опираются на обширные массивы пользовательских данных, расширяющие их *возможности*, что, в свою очередь, увеличивает их ценность для нас. Мобильные устройства используют данные GPS, чтобы узнать наше местоположение и предложить уместные рекомендации; игровые консоли адаптируют течение игры, опираясь на количество игроков, которых они могут видеть и слышать. Чтобы эффективно справляться со стоящими перед ними задачами, такие приложения должны не только обрабатывать естественный язык, но и поддерживать внутреннее *состояние*, запоминать информацию, получаемую от пользователя, и ситуационный контекст.

В этой главе мы предложим вашему вниманию фреймворк для создания чатботов, обеспечивающий хранение состояния и использующий это состояние для поддержания осмысленного диалога в определенном контексте. Мы продемонстрируем этот фреймворк на примере создания кулинарного помощника, который поприветствует нового пользователя, выполняет преобразование между единицами измерения и советует хороший рецепт. Работая над этим прототипом, мы реализуем три функции: систему правил, анализирующую команды с помощью регулярных выражений; систему «вопрос-ответ», использующую предварительно обученные синтаксические парсеры для фильтрации входящих вопросов и выбора правильных ответов; и рекомендательную систему, использующую метод машинного обучения без учителя для определения подходящих рекомендаций.

Основы диалогового взаимодействия

В 1940-х годах Клод Шеннон (Claude Shannon) и Уоррен Уивер (Warren Weaver), пионеры в области теории информации и машинного перевода, разработали модель взаимодействия настолько мощную, что она и поныне используется для анализа диалогов[1]. Согласно их модели, связь сводится к последовательности операций кодирования и преобразования по мере прохождения сообщений через каналы связи с разными уровнями шумов и энтропии.

Современное понятие *диалога*, как показано на рис. 10.1, расширяет модель Шеннона — Уивера, где две (или больше) стороны отвечают на сообщения друг друга. Диалог протекает в течение некоторого времени и обычно имеет

[1] Claude Shannon, *A Mathematical Theory of Communication* (1948), http://bit.ly/2JJnVjd (Перевод на русский язык можно найти в сборнике статей «Работы по теории информации и кибернетике», 1963, Издательство иностранной литературы, с. 243–332. — *Примеч. пер.*).

фиксированную продолжительность. Во время диалога участник может либо слушать, либо говорить. Для эффективного диалога необходимо, чтобы в каждый конкретный момент времени говорил только один участник, а остальные слушали. Наконец, упорядоченный во времени диалог должен быть согласованным, чтобы каждое утверждение имело смысл с учетом предыдущих утверждений.

Для вас, как для человека, такое описание диалога, вероятно, покажется очевидным и естественным, но оно имеет важные следствия для вычислительных устройств (представьте, насколько искаженным и запутанным получится диалог, если игнорировать любое из требований). Один из простых способов удовлетворить требования к диалогам — переключать режим вещания и прослушивания каждого участника по очереди. На каждом этапе *инициатива* вещания передается следующему участнику, который решает, куда повернуть диалог, опираясь на сказанное в предыдущем этапе. Поочередная передача инициативы помогает поддержать диалог, пока кто-то из участников не решит завершить его. В результате беседа получится согласованной и будет соответствовать всем требованиям, описанным выше.

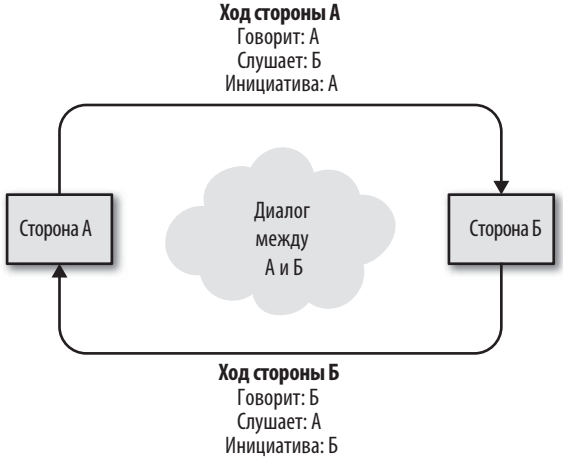

Рис. 10.1. Структура диалога

Чат-бот — это программа, участвующая в диалоге с поочередной передачей инициативы, целью которой является интерпретация входного текста или речи и вывод соответствующего полезного ответа. В отличие от людей, с которыми они взаимодействуют, для достижения этих результатов чат-боты должны полагаться на эвристики и методы машинного обучения. По этой причине им необходимы средства борьбы с неоднозначностью естественного языка и опре-

деления ситуационного контекста, чтобы эффективно анализировать входящие сообщения и выдавать осмысленные ответы.

Архитектура чат-бота, изображенная на рис. 10.2, состоит из двух основных компонентов. Первый — интерфейс с пользователем, осуществляющий прием ввода пользователя (например, через микрофоны для ввода речи или через веб-интерфейс для ввода текста) и передачу интерпретируемого вывода (через динамики для вывода синтезированной речи или через экран мобильного устройства для вывода текста). Этот внешний компонент обертывает второй — внутреннюю *диалоговую систему*, которая интерпретирует входной текст, управляет внутренним состоянием и генерирует ответы.

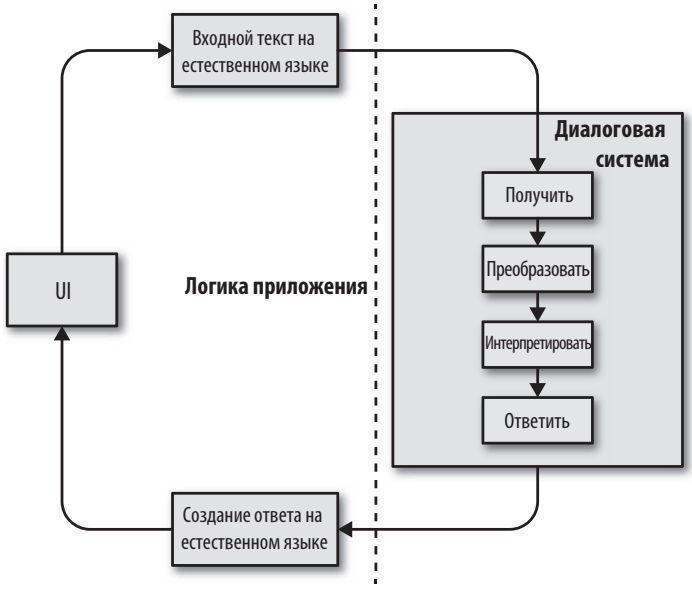

Рис. 10.2. Архитектура чат-бота

Внешний компонент пользовательского интерфейса, очевидно, может существенно отличаться в зависимости от особенностей и требований приложения. В этой главе мы сосредоточимся на внутреннем диалоговом компоненте и покажем, как легко обобщить его для любых приложений и сконструировать из множества поддиалогов. С этой целью мы сначала определим базовый абстрактный класс, который формально определяет поведение, или интерфейс диалога. Затем исследуем три реализации этого класса для управления состоянием, поиска ответов на вопросы и выбора рекомендаций и покажем, как объединить их в один диалоговый агент.