
Нишит Патак
(в соавторстве с Анурагом Бхандари)

Искусственный интеллект для .NET: речь, язык и поиск



Nishith Pathak
(With Contributing Author as Anurag Bhandari)

Artificial Intelligence for .NET: Speech, Language, and Search

*Building Smart Applications
with Microsoft Cognitive Services APIs*



Apress®

Нишит Патак
(в соавторстве с Анурагом Бхандари)



Искусственный интеллект для .NET: речь, язык и поиск

*Конструирование умных приложений
с использованием Microsoft Cognitive Services APIs*



Москва, 2018

УДК 004.8
ББК 32.813
П20



Нишит Патак

- П20 Искусственный интеллект для .NET: речь, язык и поиск / пер. с анг. А. В. Логунова. – М.: ДМК Пресс, 2018. – 298 с.: ил.

ISBN 978-5-97060-605-6

Издание рассказывает о применении когнитивных служб компании Microsoft и применении на практике набора простых API. В доступной форме описано создание приложений на основе искусственного интеллекта в среде разработки Visual Studio, создание разговорного интерфейса, применение API для обработки распознавания и интерпретации устной речи. Описаны различные модели языка в искусственном интеллекте.

Предназначена для широкого круга читателей, интересующихся искусственным интеллектом и возможностями его практического применения, знающих на уровне пользователя базовые сервисы Microsoft.

УДК 004.8
ББК 32.813

Original English language edition published by Apress, Inc. USA. Copyright © 2017 by Apress, Inc.
Russian-language edition copyright © 2018 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Вступительное слово от Microsoft

В последние годы мы наблюдаем бурный расцвет искусственного интеллекта, который демонстрирует все новые успехи: лучшее качество распознавания речи и изображений по сравнению с человеком, победа в шахматы и Go и др. Такой прогресс обусловлен все большим количеством доступных данных и вычислительных мощностей для их обработки и обучения интеллектуальных моделей. В этой связи конкурентное преимущество компаний смещается из области know-how в алгоритмах к владению приватными массивами данных.

Однако при этом наблюдается и другая тенденция, которую мы называем демократизация ИИ. Большие компании обучают ИИ-модели на имеющихся данных и предоставляют их в виде готовых «строительных блоков» в облаке для использования сторонними разработчиками. Это значит, что для решения таких сложных задач, как распознавание человека на фотографии или машинный перевод, достаточно всего лишь вызвать некоторую облачную функцию для выполнения этой задачи. Такой процесс делает интеллектуальные решения доступными как никогда.

Данная книга знакомит читателя с когнитивными облачными сервисами от компании Microsoft в области работы с естественным языком и речью. Эти сервисы могут быть полезны при создании диалоговых систем, для анализа потоков текстовой информации (например, в социальных сетях), для разработки голосовых интерфейсов и во многих других задачах. Через серию последовательно усложняющихся примеров автор знакомит читателя со всеми тонкостями API, приводя при этом примеры кода, который можно непосредственно заимствовать для разработки. К слову, если читателю этих примеров покажется мало, то еще больше примеров (в том числе в области анализа изображений) содержится в нашем репозитории к курсу по когнитивным сервисам и ИИ: <http://github.com/evangelism/ModernAI>.

Следует предупредить читателя, что когнитивные сервисы Microsoft стремительно развиваются, и некоторые приведенные в книге примеры могут устареть или работать немного не так, как описано. В настоящее время уже разрабатывается следующая версия Bot Framework, которая будет несколько отличаться от описанной в книге. Тем не менее нельзя недооценить важность излагаемых концепций и примеров – просто надо быть готовым продолжать разбираться в теме, не ограничиваясь только чтением книги.

Интеллектуальные и когнитивные технологии считают основным двигателем цифровой трансформации в ближайшие годы, когда многие рутинные операции, выполняемые традиционно человеком, будут заменены ИИ. Если вы планируете заниматься разработкой и созданием продуктов на базе программного обеспечения, способность грамотно включать решения на базе ИИ в свои продукты будет являться важнейшим критерием их успешности. Данную книгу можно рассматривать как начало увлекательного пути, который приведет вас к интересным открытиям, а в перспективе – и к лучшему пониманию человеческой интеллектуальности.

Сошинков Дмитрий Валерьевич,
старший технологический евангелист Microsoft,
к. ф.-м. н., доцент МФТИ, МАИ, НИУ ВШЭ

Посвящаю эту книгу самому важному человеку в моей жизни, моей матери, покойной Бине Патак, за ее наставничество, жертвы, молитвы и благословения, которые сделали из меня того, кем я являюсь сегодня. Я скучаю по ней каждый день. Своему отцу Панкаджу Патак, который учил меня делать то, во что я верю. Ты всегда остаешься и будешь оставаться для меня образцом для подражания и моим героем всю мою жизнь. Моему учителю Садху-Гурудева, который был неисчерпаемой руководящей силой и полностью изменил мою жизнь. Своему деду, покойному Махешу Чандра Патак, за его благословения и нравственные ценности.

Моей жене Сураби за то, что была со мной, пожертвовала своей прекрасной карьерой ради нашей семьи и всегда оставалась рядом со мной во всех взлетах и падениях. Женитьба на тебе стала самым прекрасным событием в моей жизни. Ты подарила мне самый драгоценный алмаз в моей жизни, Шихар, которого я люблю больше всех. Я знаю, что эта книга потребовала больших усилий и за прошлый год я не смог проводить с вами достаточно времени, папа и Шикхар, поскольку неустанно работал, чтобы подготовить для вас этот приятный сюрприз. Сураби и Шихар, эта книга была бы невозможна без всех ваших жертв.

Моей прекрасной сестре Танви и моей племяннице Аадхье – ваши улыбающиеся лица придают мне силы и вдохновение, чтобы становиться лучше каждый день. Моим друзьям Гуруджи, Дж. П. Кукрети, С. С. Тьяги и Раджеш Трипати, которые были со мной бесчисленное количество раз и всегда обеспечивали меня комфортом, пониманием, духовными убеждениями и большой мотивацией. И наконец, я благодарю Бога за то, что он осчастливили меня такими замечательными людьми в моей жизни.



Содержание

Об авторе	13
О соавторе	14
О техническом рецензенте	15
Признательности	16
Введение	17
Примечания переводчика	20
Глава 1. Начало работы с основами ИИ	21
Правда и вымысел	22
История и эволюция	23
Нынешнее состояние дел	25
Превращение ИИ в товар	26
Microsoft и ИИ	27
Фундаментальные понятия	28
Машинное обучение.....	30
Язык.....	33
Речь	35
Компьютерное зрение.....	36
Когнитивные службы Microsoft	39
Зрение	40
Речь	41
Язык	41
Знание	42
Поиск	43
Резюме	43
Глава 2. Создание приложения с поддержкой ИИ в среде Visual Studio.....	45
Необходимые условия для использования когнитивных служб	46
Подготовка среды разработки	46
Получение ключа подписки Azure для когнитивных служб.....	47
Шаг 1. Зарегистрировать учетную запись Azure.....	47
Шаг 2. Создать новую учетную запись когнитивных служб	50
Шаг 3. Получить ключ (ключи) подписки.....	52
Тестирование API.....	53
Что вы хотите сделать	54
Как это сделать.....	54
Создание первого ИИ-ориентированного приложения	57

Исходный код.....	58
Пошаговый анализ	59
Результат	63
Более интересный вариант приложения	63
Извлечение текста из изображений	63
Исходный код.....	64
Пошаговый анализ	66
Результат	66
Резюме.....	67
Глава 3. Конструирование разговорного интерфейса с применением технологий Microsoft	68
Что такое разговорный интерфейс пользователя?®.....	70
Краткая история	70
В самом начале: интерфейс командной строки.....	71
И затем появился графический интерфейс пользователя	72
Очередная эволюция: разговорный интерфейс пользователя	74
Роль ИИ в разговорном интерфейсе пользователя.....	77
Ловушки разговорного интерфейса пользователя	78
Будущее может быть за гибридным интерфейсом пользователя (разговорный + графический)	80
Принципы разработки	81
Инфраструктура Microsoft Bot Framework.....	82
Создание приложения с разговорным интерфейсом в среде Bot Framework	83
Шаг 0. Подготовка среды разработки	84
Шаг 1. Создание нового проекта приложения Bot Framework.....	84
Шаг 2. Первое сообщение и сообщение по умолчанию	85
Шаг 3. Выполнение, тестирование и отладка вашего бота	87
Шаг 4. Планирование визита к врачу	90
Шаг 5. Обработка системных сообщений	93
Следующие шаги	94
Резюме.....	94
Глава 4. Применение технологии понимания естественного языка	95
Что такое понимание ЕЯ?	96
История развития методов понимания естественного языка	99
Почему машины с трудом понимают естественный язык	102
Сложности естественного языка	102
Статистические модели в качестве решения недостаточны.....	104
Многообещающее будущее.....	106
Облачная служба LUIS	107
Архитектура программного приложения на основе LUIS	108
За кадром	111
Масштабная тренировка – основа всему	112
Получение подписки Azure для облачной службы LUIS	114

Демонстрационный пример: приложение DefinitionApp	116
Примечания	118
Резюме.....	119
Глава 5. Исследование когнитивной модели языка	120
API Bing проверки орфографии	120
Что это такое?	123
Как его использовать.....	124
Интеграция с облачной службой LUIS.....	127
API анализа текста	129
Идентификация языка	129
Выделение ключевых фраз	133
Анализ тональности текста.....	135
Идентификация темы	137
Идеи использования.....	140
API веб-модели языка (WebLM)	141
Разбиение на слова.....	143
Совместная вероятность	144
Условная вероятность.....	145
Предсказание следующего слова.....	147
API лингвистического анализа	148
Разбиение на предложения и лексемизация	149
Частеречная разметка	152
Синтаксический анализ на основе грамматики составляющих	154
Резюме.....	156
Глава 6. Потребление и применение облачной службы LUIS	157
Планирование своего приложения	158
Что бот должен уметь делать?	158
Какая информация нужна боту от пользователя?.....	159
Что должно быть сделано в облачной службе LUIS?.....	159
Что должно быть сделано в Bot Framework?	160
Создание приложения с поддержкой LUIS	161
Добавление намерений.....	162
Добавление/разметка высказываний	162
Публикация своего приложения.....	164
Добавление сущностей.....	166
Простые сущности	168
Составные сущности	170
Иерархические сущности.....	173
Предварительно подготовленные сущности	175
Добавление списка фраз	176
Рекомендуемые последующие шаги	177
Активное обучение посредством рекомендованных высказываний	177
Использование API LUIS для автоматизации	178
Интеграция LUIS со средой Bot Framework	178

Создание проекта в Visual Studio	179
Обработка намерения без сущностей	180
Подготовка своего бота для использования HealthCheckupDialog	180
Тестирование бота в эмуляторе	181
Обработка намерения с полным комплектом сущностей	181
Обработка намерения с составными сущностями	183
Обработка намерения None	185
Добавление бота в Skype	186
Публикация своего бота	186
Регистрация своего бота	186
Резюме	188
Глава 7. Взаимодействие с API обработки речи	189
Способы взаимодействия с речью	190
API когнитивного поиска	191
Распознавание речи	191
Начало работы	192
Сначала получение JWT-токена	192
API персонализированной обработки речи	194
Синтез речи	195
Механизм распознавания речи	198
Служба персонализированной обработки речи	199
Персонализированная акустическая модель	199
Персонализированная модель языка	209
Данные произношения	211
Персонализированная конечная точка преобразования речи в текст	212
Распознавание говорящего	215
Верификация говорящего против идентификации говорящего	215
Регистрация для верификации говорящего	216
Верификация говорящего	219
Регистрация для идентификации говорящего	219
Идентификация говорящего	221
Состояние операции	221
Резюме	222
Глава 8. Применение поисковых предложений	223
Поиск повсюду	224
Повсеместность, предсказательность, проактивность (три «П» поиска)	224
История Bing	226
В чем уникальность Bing?	228
API поиска	228
API Bing автозаполнения	228
Как использовать API Bing автозаполнения	229
API Bing поиска изображений	233
Как использовать API Bing поиска изображений	234
API Bing поиска новостей	239

API Bing поиска видео	242
Как использовать API Bing поиска видео	242
API Bing поиска в сети	246
Как использовать API Bing поиска в сети	246
Резюме.....	250



Глава 9. Работа с рекомендациями.....	251
Понимание основ	252
Рекомендация товаров, часто приобретаемых вместе	253
Рекомендация «от предмета к предмету».....	254
Рекомендация на основе предыстории	255
Как эти рекомендации работают?	255
Модели и типы рекомендаций	260
Сборка обычной модели рекомендаций	261
Сборка модели рекомендации на основе FBT	264
Сборка модели рекомендаций на основе ранжирования.....	267
Сборка модели рекомендации на основе SAR	269
Установление правил в сборке	272
Оффлайновое оценивание	272
Веб-интерфейс API рекомендаций.....	273
Резюме.....	277
Глава 10. ИИ в ближайшем будущем.....	278
Почему ИИ имеет такую популярность?.....	279
Рост вычислительной мощности	279
Открытия в алгоритмах ИИ	280
Данные – это новая валюта	281
Возникновение облачных вычислений.....	282
Службы против решений?	283
Когнитивные категории.....	283
Проблемы и будущее понимания ЕЯ.....	284
Проблемы и будущее обработки речи.....	285
Проблемы и будущее поиска	286
Проблемы и будущее рекомендаций	286
Прежде всего ИИ.....	287
Интеллектуальная периферия	287
Устранение задач, а не рабочих мест	288
Так куда же мы катимся?	291
Тематический указатель	293

Вступительное слово от сообщества MskDotNet Community



Последние несколько лет мир информационных технологий движет система интеллектуализации. Мы просыпаемся, используя smart-будильники наряду со smart-часами, которые анализируют фазы сна. Собираясь в институт или на работу, мы смотрим детальные прогнозы погоды. Затем по дороге в очередное место встречи мы пользуемся навигатором, который не просто строит нам качественный маршрут, но и может с нами поговорить. Ежедневно миллионы людей говорят «Привет, Cortana», «Привет, Siri», «О’кей, Google» или разговаривают с Алисой. Помимо всего этого, приезжая на очередную встречу, нам нужно припарковать автомобиль, в дорогих автомобилях есть целые интеллектуальные системы паркинга. В некоторых же автомобилях используются устройства подешевле, но в любом случае эти системы не работали бы без камер и распознавания изображений.

С каждым годом все больше программистов переходят с простых автоматизаций процессов на сторону приложений с искусственным интеллектом. Даже те же интернет-магазины последние годы собирают статистику цен по всему Интернету и с помощью интеллектуального анализа данных выставляют свои.

С каждым годом все большее знаменитых компаний участвуют в жизни вузов, например Microsoft, Yandex, ABBYY, JetBrains. Все больше стартапов, связанных с искусственным интеллектом, – можно взять любой предмет, добавить к нему слово smart и посмотреть о нем информацию в Интернете, наверняка такой smart-предмет уже есть.

Сегодня вы читаете эту книгу. Значит, эта тема, это направление интеллектуального будущего действительно интересно людям.

Нишит Патак в своей книге рассказывает о том, как можно просто и быстро сделать свою smart-систему, используя технологии компании Microsoft. Книга действительно небольшая. А технологии, которые скрываются за пределами этой книги, – на самом деле масштабного уровня. Ведь что для вас стоит сделать всего один запрос к сервисам Azure? Считанных минут, чтобы написать под это код. А что стоит распознать изображение? Голос? Перевести что-то? А может, вы хотите построить свой сервис рекомендаций? А стоить все эти операции будут ровно столько, сколько стоит написание простого кода для запроса к сервисам Azure. Считанные минуты – и система уже работает.

Действительно, эти технологии от Microsoft – настоящий прорыв в мире искусственного интеллекта. Всего один веб-запрос – и в ваших руках уже есть что-то smart-подобное! С помощью этих технологий и своей идеи вы быстро построите не только что-то умное, но, скорее всего, и свой бизнес. От лица сообщества MskDotNet Community мы можем пожелать вам успехов в начинаниях и изучении интеллектуальных сервисов от Microsoft и будем ждать все больше умных вещей, которые каждый день вносят краски в нашу современную жизнь.

*MskDotNet Community,
Елизавета Голенок, Юлия Цисык*

Об авторе



Нишит Патак носит звание наиболее ценного специалиста компании Microsoft (MVP), является системным архитектором, лектором, а также идеологом, новатором и стратегом ИИ. Он является плодовитым писателем и автором и написал много книг, статей, обзоров и рубрик для нескольких электронных и печатных изданий. Более чем 20-летний опыт работы в области ИТ позволил Нишиту накопить экспертизные знания, которые лежат в области инноваций, исследований, проектирования архитектуры, проектирования и разработки приложений для компаний из списка 100 крупнейших промышленных компаний США с использованием инструментов и технологий следующего поколения. Являясь ранним приверженцем технологии Microsoft, он идет в ногу в проблемах, связанных с сертификацией программных продуктов, и преуспел в получении нескольких своих сертификатов на стадии бета-тестирования.

Нишит является золотым членом консультативного совета различных национальных и международных сообществ и организаций в области информатики. Он дважды был награжден элитным званием наиболее ценного специалиста компании Microsoft (MVP) за свою образцовую работу и опыт в технологиях Microsoft. Он является членом различных консультативных групп корпорации Microsoft. В настоящее время Нишит работает вице-президентом и руководителем НИОКР в Accenture Technology Labs. Он сосредоточен на ключевых областях исследований, в частности искусственном интеллекте, машинном обучении, когнитивных вычислениях, программах-роботах и облачных вычислениях на основе технологии блокчейна, а также помогает компаниям проектировать архитектуру решений на основе этих технологий. Нишит родился, вырос и получил образование в городе Котдвара в Уттаракханде, Индия. Когда позволяет время, он проводит его с семьей и друзьями и забавляет их своими познаниями в области хиромантии и астрологии. Вы можете с ним связаться по адресу nispathak@gmail.com.



О соавторе



Анураг Бхандари является исследователем, программистом и проповедником свободного программного обеспечения. Его любимые области исследований включают обработку ЕЯ, интернет вещей и машинное обучение. Он специализируется на разработке веб- и мобильных приложений и решений. Имеет большой опыт работы с компаниями из списка 500 крупнейших промышленных компаний, стартапами и НКО в качестве исследователя и разработчика программного обеспечения. Анураг родом из Джалаандхара, Пенджаб, где он также получил диплом в области компьютерных наук в Национальном технологическом институте. Со своих студенческих дней он был связан с несколькими проектами с открытым исходным кодом, такими как Granular Linux и OpenMandriva, и в ряде из них возглавлял эти проекты. Он является потрясающим полиглотом в языках программирования (C#, Java, JavaScript, PHP, Python) и естественных языках (английский, хинди, панджаби, французский). Будучи энтузиастом в области технологий, Анураг продолжает возиться с трендовыми технологиями и испытывать новые программные инфраструктуры и платформы. В свободное время он читает книги, занимается спортом, мечтает о гаджетах, смотрит телепередачи, играет в игры и собирает марки. Вы можете найти его онлайн на <http://anuragbhan-dari.com> или оставить ему записку по адресу anurag.bhd@gmail.com.



О техническом рецензенте

Фабио Клаудио Ферракьятти является старшим консультантом и старшим аналитиком/разработчиком с использованием технологий Microsoft. Он работает в Blu-Arancio C. c. A (www.bluarancio.com) старшим аналитиком/разработчиком и специалистом Microsoft Dynamics CRM, является сертифицированным разработчиком решений Microsoft для .NET, дипломированным специалистом по программным средствам Microsoft, а также плодовитым автором и техническим редактором. За последние десять лет он написал ряд статей для итальянских и международных журналов и являлся соавтором более десяти книг на различные компьютерные темы.



Признательности



Эта книга стала результатом коллективной работы нескольких замечательных людей. Она не была бы завершена без моего партнера Анурага Бхандари, который проделал фантастическую работу, помогая завершать главы, писать исходный код и проводить исследования. В разное время мы проговаривали темы, обсуждали технологии и формировали книгу в правильном направлении. Анураг, ты – «единственный человек», который помог мне поднять эту книгу далеко за пределы моих ожиданий.

Спасибо всем сотрудникам издательства Apress, которые приложили искренние усилия, чтобы опубликовать данную книгу. Гвенан заслуживает особой благодарности, потому что, прежде чем приступить к этому проекту, я долго обменивался с ней письмами. Спасибо Нэнси и Лоре за отличную работу по управлению проектами и постоянное побуждение меня делать все возможное. Я также хотел бы поблагодарить Генри Ли за его технический обзор. Я бы не постеснялся сказать, что все вы – очень талантливые люди. Каждый из вас очень помог этой книге, и я с нетерпением жду возможности поработать с вами над следующей.

И последнее, но не менее важное: благодарю свою семью, в особенности жену Сурабхи и своего отца Панкаджа Патака, за то, что они были добры ко мне и всегда были готовы прийти на помощь, и за то, что мои мечты сбылись. Все, что я делаю в своей жизни, было бы невозможно без вас.



Введение

Эта книга познакомит вас с миром искусственного интеллекта. В большинстве случаев разработчики считают задачу внедрения ИИ трудно выполнимой, так как она сопряжена с написанием сложных алгоритмов и сотен строк программного кода. Задача настоящей книги – показать, как с помощью созданного вами когнитивного приложения, состоящего из нескольких строчек программного кода, можно найти более простое решение. Это стало возможным благодаря появлению разнообразных прикладных программных интерфейсов когнитивных веб-служб (Cognitive Services API). Данная книга посвящена некоторым полезным технологиям, благодаря которым ваше приложение сможет разумно применять API когнитивных служб. Компания Microsoft предоставляет разработчикам оптимальный опыт взаимодействия и реализует его с помощью своих программных интерфейсов Cognitive Services API.

Эта книга дает четкое представление о принципах ИИ. Темы поддержания естественно-языкового диалога, обработки естественного языка и выполнения интеллектуального поиска настолько многогранны, что для каждого из этих направлений потребовалась бы отдельная книга. В настоящей книге предпринимается попытка дать четкое объяснение по каждому из этих понятий, сначала давая ясные ответы на вопросы «Почему» и «Что» и только после этого вдаваясь в подробное объяснение того, «Как», т. е. каким образом устроен тот или иной API. В книге также приводятся подробные примеры, которые призваны облегчить применение новых концепций на практике. Книга «Искусственный интеллект для .NET: речь, язык и поиск» покажет, каким образом уже сегодня можно приступить к внедрению этих удивительных возможностей в свои собственные приложения.

Данная книга начинается с того, что знакомит вас с описанием и технологией искусственного интеллекта, проведя небольшой экскурс в его историю, терминологию и методологию. Затем вы познакомитесь с программными интерфейсами когнитивных служб и соответствующими инструментами, после чего перейдете к пошаговому созданию вашего первого интеллектуального когнитивного приложения, используя для этого среду разработки Visual Studio. Далее познакомитесь с понятиями, связанными с разговорным интерфейсом пользователя (РИП, conversational user interface, CUI), и приступите к созданию своего первого бота на основе программной инфраструктуры Microsoft Bot Framework. Данная книга позволяет понять процесс планирования вашего приложения с использованием инфраструктуры Bot Framework, а также познакомит с лучшими практическими методами ее применения.

Настоящая книга также позволяет глубже познакомиться с проблемами понимания естественного языка (natural language understanding, NLU) и его обработки (natural language processing, NLP). Эти технологии позволяют компьютерным программам понимать людей точно так же, как люди понимают друг друга. Книга подробно останавливается на интеллектуальной облачной службе понимания естественного языка Microsoft LUIS (Language Understanding Intelligent Service) и принципах ее работы, а также на приемах проектирования, употребления

и применения службы LUIS перед созданием проекта с участием LUIS с нуля. Книга также содержит подробные инструкции по тестированию, тренировке и публикации приложения с участием LUIS перед его развертыванием и использованием в среде Bot Framework.

Речь – это самая естественная форма взаимодействия. Данная книга предоставляет пошаговое объяснение речевого программного интерфейса Speech API и приемов применения API для распознавания и синтеза речи. Далее вы подробно познакомитесь с примерами применения службы персонализированной обработки речи, ранее известной под названием CRIS и предоставляющей пошаговый план создания вашей первой модели языка, аудиомодели, ее развертывания, а также применения службы персонализированной обработки речи. В книге также содержится подробная информация о том, как узнать, кто сейчас говорит.

В 8-й главе книги вы найдете историю и подробное описание популярной поисковой системы Bing и интерфейса API и способы применения возможностей этого поисковика в своих приложениях. В книге также подробно рассказывается об основополагающих понятиях и типах рекомендаций и дается пошаговое описание способов получения этих рекомендаций. Книга заканчивается экскурсом в будущее искусственного интеллекта. Другими словами, эта книга может рассматриваться как путеводитель, руководствуясь которым, вы сможете осуществить ваши последующие шаги при создании своих приложений.

В этой книге вы:

- изучите основы технологии искусственного интеллекта на практических примерах и сценариях;
- начнете создавать приложение на основе ИИ в среде разработки Visual Studio;
- создадите текстоориентированный разговорный интерфейс для прямого взаимодействия с пользователем;
- примените API обработки речи когнитивных служб для распознавания и интерпретации устной речи;
- ознакомитесь с различными моделями языка, включая обработку естественного языка, и примените их в приложении Visual Studio;
- повторно будете использовать поисковые возможности Bing, чтобы лучше понять, что именно пользователь имеет в виду;
- поработаете с предлагаемыми технологиями и внедрите их в свои приложения.

ДЛЯ КОГО ЭТА КНИГА ПРЕДНАЗНАЧЕНА

Термин «искусственный интеллект» представляет собой профессиональный жаргонизм современной информационной индустрии. Об ИИ говорят все. Из-за связанной с этим неразберихи разработчики программного обеспечения зачастую пребывают в недоумении по поводу того, где и как начинать работу с ИИ. И появление когнитивных программных интерфейсов Microsoft Cognitive API предлагает разработчикам широкий спектр новых функциональных возможностей. Эта книга предназначена для читателей начального и промежуточного уровней, интересующихся искусственным интеллектом. Разработчики и архитекторы программного обеспечения с предыдущим опытом работы в среде .NET или без та-

кового, которые хотят применить новые когнитивные API в своих приложениях, извлекут из предлагаемого в данной книге фактического материала и примеров программного кода немалую пользу. Эта книга также послужит отличным руководством для разработчиков приложений и архитекторов программного обеспечения, которые не знакомы с ИИ и/или ключевыми принципами применения некоторых когнитивных API.

Необходимые компоненты

Чтобы эта книга была наиболее полезна, вам потребуется программная платформа .NET Framework от компании Microsoft и подключение к Интернету. В качестве среды разработки для экспериментирования с примерами программного кода, которые можно найти в разделе Исходный код сайта Apress (www.apress.com), рекомендуется использовать Microsoft Visual Studio 2017.

Получение обновлений для этой книги

По ходу чтения вы можете натолкнуться на случайную грамматическую или программную ошибку, хотя я думаю, что этого не произойдет. Если же такое случится, то примите мои искренние извинения. Каждый человек имеет право на ошибку, поэтому одна или две неточности могут присутствовать независимо от предпринятых мною усилий. Текущий список ошибок можно получить на веб-сайте Apress (который тоже находится на домашней странице этой книги). Здесь же можно узнать, как уведомить меня о любых ошибках, которые вы можете найти.

Связь с автором

Если у вас есть какие-либо вопросы относительно исходного кода этой книги, или вы нуждаетесь в разъяснении того или иного примера, или же просто хотите предложить свои мысли относительно ИИ либо хотите связаться со мной по поводу других вопросов, то прошу, не стесняйтесь и напишите мне на nispathak@gmail.com. Я сделаю все возможное, чтобы своевременно с вами связаться.

Благодарю, что приобрели эту книгу. Надеюсь, что вы получите удовольствие от ее прочтения и примените свои вновь обретенные знания с пользой для всех.

Условные обозначения

 Такая пиктограмма обозначает совет.

 Такая пиктограмма обозначает примечание.

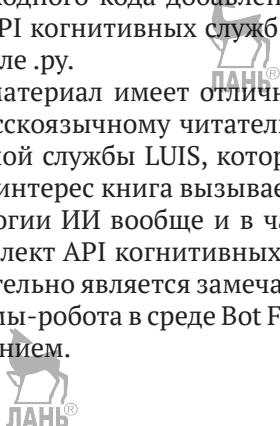
Примечания переводчика

Книга «Искусственный интеллект для .NET: речь, язык и поиск» является превосходным руководством по ИИ и технологиям Microsoft в этой области.

В переводе программный код адаптирован на основе Visual Studio 2017 Community и только в главах 2 и 3. В главах 4 и 5 используется интеллектуальная система понимания речи LUIS, в которой на сегодняшний день русский язык пока не используется, но его внедрение остается в планах. В последующих главах автор ссылается на программные шаблоны, созданные во второй и третьей главах.

По всей книге заменены ссылки на API с региона западных США на регион Западной Европы, который позволяет работать с API когнитивных служб из России. К оригинальной версии исходного кода добавлена его адаптированная версия; добавлен шаблон запроса API когнитивных служб на языке Python для блокнота Jupyter и в стандартном файле .ru.

В целом предлагаемый материал имеет отличную практическую направленность и будет интересен русскоязычному читателю, лишь за небольшим исключением применения облачной службы LUIS, которая пока русский язык не поддерживает. Но наибольший интерес книга вызывает с точки зрения современных механизмов работы технологии ИИ вообще и в частности в Microsoft, где стремительно развивается комплект API когнитивных служб (Cognitive Services API). В этом плане книга действительно является замечательным руководством. А примеры по созданию программы-робота в среде Bot Framework являются превосходным практическим дополнением.



Глава 1

Начало работы с основами ИИ

Представьте, что вы создаете программный продукт, который настолько умен, что понимает не только человеческие языки, но и жаргонные выражения и тончайшие диалекты этих языков, да так, что ваше программное обеспечение будет знать, что «Привет, компьютер! Как дела?» и «Как делишки, чувак?» означает одно и то же.

И пока вы этим занимаетесь, почему бы не добавить в ваше программное обеспечение возможность слушать говорящего человека и реагировать соответствующим образом?

Пользователь: «Какое у меня расписание на сегодня?»

Софт: У вас сегодня довольно насыщенный день, одна за другой встречи с 10 утра до 13:30 пополудни и потом снова с 15:00 до 19:00 вечера».

Но почему бы не добавить в данный продукт возможность вести разговоры, подобные человеческим диалогам? Тот факт, что софт может отвечать человеку, еще не обозначает, что сам программный продукт обладает разумом. Но почему бы не добавить в данный продукт возможность вести диалоги, подобные человеческим? Например.

Пользователь: «Компьютер, я пропустил матч? Какой счет?»

Софт: «Уже 31-я минута футбольного матча между “Барселоной” и “Реалом”. Ваша любимая команда “Барселона” еще не забила. Счет 0:1».

Пользователь: «Черт возьми! Кто забил у Реала?»

Софт: «Первый гол на 10-й минуте забил Криштиану Роналду».

Пользователь: «Мне это не нравится. Сколько он забил в этом сезоне?»

Софт: «На данный момент Роналду забил 42 гола за свой клуб и 13 голов за свою страну».

Пользователь: «Впечатляет. Надеюсь, бедняга Месси скоро догонит».

Пользователь: «Компьютер, спасибо за новости».

Софт: «Не за что».

Софт: «Не забудьте узнать счет через полчаса. Если опираться на статистику владения мячом и ударов по воротам, то есть 73%-й шанс, что Барселона в ближайшие 20 минут забьет гол».

Разве такие возможности не сделают ваше программное обеспечение умным и интеллектуальным? Являясь разработчиком в среде .NET, вы вскоре узнаете, каким образом вы могли бы сделать ваше программное обеспечение таким же умным, что и Microsoft Cortana, Apple Siri или Google Assistant.

Microsoft Cortana – это голосовая помощница Кортана от компании Microsoft.

Apple Siri – персональный голосовой помощник с искусственным интеллектом, присутствующий во всех современных аппаратах Apple.

Google Assistant – умный виртуальный ассистент от компании Google.

После прочтения этой главы вы:

- узнаете правду и вымысел об ИИ;
- познакомитесь с историей и эволюцией ИИ;
- познакомитесь с разработками Microsoft в области ИИ;
- познакомитесь с основными понятиями ИИ:
 - когнитивным машинным обучением, глубоким обучением, обработкой естественного языка (ЕЯ), пониманием ЕЯ и др.;
 - показательными диаграммами и ссылками (где это возможно);
- познакомитесь с когнитивными службами Microsoft:
 - поговорим о всех пяти когнитивных группах;
- узнаете, как применять ИИ в своем программном обеспечении;
- узнаете о перспективах развития ИИ в обозримом и отдаленном будущем.

ПРАВДА И ВЫМЫСЕЛ

Что вам приходит на ум, когда вы слышите термин «искусственный интеллект»? Страшные роботы? Тема сложных исследований? Арнольд Шварценеггер в фильме «Терминатор»? Боты в Контр-страйке?

 Контр-страйк (Counter-Strike) – это компьютерная стратегическая игра-стрелялка от первого лица производства компании Valve. Она основана на битве между террористами, которые хотят устраивать взрывы бомб в людных местах, и контртеррористическими силами, мешающими террористам устроить хаос. Хотя в эту многопользовательскую игру, как правило, играют люди, предусмотрена возможность, когда один человек играет вместе с ботами или против них.

Боты – это программные виртуальные игроки с поддержкой ИИ и самостоятельным мышлением, которые могут заменять игроков, когда они отсутствуют. Боты широко используются в компьютерных играх, а иногда их просто так и называют – искусственным интеллектом игры.

Игра Counter-Strike, или CS (СиЭс), как ее с любовью называют, в особенности популярна среди любителей и профессиональных геймеров и регулярно присутствует на лучших игровых конкурсах по всему миру.

Смысл понятия «искусственный интеллект» (ИИ) изменился в ходе исследований этого вопроса. Базовые понятия ИИ не изменились, чего не скажешь о его приложениях. То, как ИИ воспринимался в 1950-х годах, очень отличается от того, как он на самом деле используется сегодня. И понятие искусственного интеллекта по-прежнему продолжает эволюционировать.

В наши дни тема искусственного интеллекта приобрела актуальный характер. ИИ прошел долгий путь, выйдя со страниц научно-популярных фантастических романов и в результате превратившись в товар. И разумеется, ИИ не имеет ничего

общего с победным шествием роботов, которые захватывают мир и порабощают нас, людей. По крайней мере, пока. Все, что обладает интеллектуальными способностями, будь то виртуальный ассистент в телефоне (Сири и Кортана), ваша верная поисковая система (Google и Bing) или ваши любимые мобильные и видеоигры, все это работает на основе ИИ.

Интерес к ИИ достиг пика в 2000-х годах, в особенности в начале 2010-х годов. Огромные инвестиции в исследования в области ИИ последних лет, направляемые научными кругами и корпорациями, стали благом для разработчиков программного обеспечения. Прогресс, достигнутый такими компаниями, как Microsoft, Google, Facebook и Amazon, в различных областях ИИ с последующим переходом к открытому исходному коду и коммерциализации своих продуктов позволил разработчикам программного обеспечения с беспрецедентной легкостью создавать впечатление человекоподобности ваших приложений. Это привело к взрывному увеличению умных, интеллектуальных приложений, которые способны понимать своих пользователей так же, как это делает обычный человек.

Задумывались ли вы, будучи разработчиком, о том, как применить ИИ для создания невероятно умного программного обеспечения? Скорее всего, задумывались, но не знали, с чего начать.

По нашему опыту общения с разработчиками программного обеспечения ведущих ИТ-компаний создается следующее впечатление: внедрение в софт даже отдельных элементов ИИ, таких как понимание естественного языка (ЕЯ), распознавание речи, машинное обучение и т. д., потребует глубокого понимания принципа работы нейронных сетей, нечеткой логики и других теоретических основ информатики.

Наверное, раньше так и было. Но поспешим вас обрадовать: таких глубоких знаний для разработки приложений с ИИ больше не требуется. Интеллект, работающий в ваших любимых приложениях, таких как Google Search, Bing, голосовой помощник Кортана и Facebook, постепенно становится доступным для разработчиков, не работающих в этих компаниях. Некоторые составные части искусственного интеллекта этих приложений бесплатны, а другие, как SaaS (англ. software as a service – программное обеспечение как служба), предоставляются на основе коммерческих предложений.

ИСТОРИЯ И ЭВОЛЮЦИЯ

По нашему мнению, для лучшего понимания интересующего вас предмета или вопроса, например искусственного интеллекта, следует познакомиться с его историей развития.

С древних времен люди восхищались тем, как всемогущие мудрецы либо сумасшедшие ученые наделяли неживые предметы способностью мыслить. В древней и современной литературе можно найти бесчисленное количество рассказов о том, как неодушевленные предметы внезапно наделяются сознанием и интеллектом.

Греческие, китайские и индийские философы верили, что человеческое мышление может быть представлено набором механических правил. Аристотель (384–322 до н. э.) разработал формальный метод решения силлогизмов. Евклид (~300 до н. э.) благодаря своей математической работе «Элементы», содержащей

один из самых ранних известных алгоритмов, подарил нам формальную модель рассуждения. Лейбниц (1646–1716) создал универсальный язык рассуждений, сводивший аргументацию к вычислению. Он предположил, что все рациональное мышление может быть систематизировано как алгебра или геометрия. Работа Буля (1815–1864) по математической логике была прорывной и предоставила возможность создания искусственного интеллекта.

Эти формальные системы, или «теории», с использованием технологий того времени внедрялись на практике. Создавались машины, подражавшие человеческому поведению или мыслям. Используя часовой механизм, человек создавал сложные механизмы, от часов с кукушкой до автоматов, рисующих картинки. Это были ранние варианты механических роботов. В наше время формальные принципы рассуждения были использованы для создания устройств, называемых компьютерами.

Термин «искусственный интеллект» был придуман на семинаре в кампусе Дартмутского колледжа летом 1956 года. Предлагаемая тема семинара, в частности, содержала следующее утверждение: *«Всякий аспект обучения или любое другое свойство интеллекта в принципе может быть настолько точно описано, что машина сможет его симулировать»*. Именно на этой конференции было положено начало в области исследований ИИ, а присутствовавшие на ней люди стали первоходцами научных исследований ИИ.

В последующие десятилетия произошли крупные прорывы в области разработки и изучения ИИ. Были разработаны компьютерные программы для решения алгебраических задач, доказывания теорем и ведения диалога на английском языке. Правительственные учреждения и частные организации финансировали данные исследования. Но путь развития ИИ был не прост.

Первая неудача в исследованиях ИИ произошла в 1974 году. Период между этим годом и по 1980 год называется первой «зимой ИИ». За это время многие обещанные результаты исследований не были достигнуты. Данная неудача объяснялась рядом причин, одной из которых была неспособность ученых предвидеть всю сложность проблем создания ИИ. Но основной причиной была ограниченная вычислительная мощность электронно-вычислительных машин того времени. Как результат отсутствие прогресса вынудило крупные британские и американские агентства, поддерживающие ранее эти исследования, прекратить их финансирование.

Следующие семь лет, начиная с 1980 года и по 1987 год, вновь появился интерес к исследованиям в области ИИ. Развитие экспертных систем подпитывало ажиотаж вокруг данных исследований. Экспертные системы развивались во всех организациях, и вскоре все гиганты индустрии начали вкладывать в данные исследования огромное количество денег. Исследование нейронных сетей заложило основу развития методов оптического распознавания символов и распознавания речи. В последующие годы, с 1987 по 1993 год, наступила вторая зима в развитии искусственного интеллекта. Как и предыдущей зимой, финансирование разработок и исследования ИИ снова уменьшились.



Экспертная система – это программа, которая отвечает на вопросы или решает задачи из конкретной области знаний, используя логические правила, которые разработаны на основе знаний экспертов.

Экспертные системы ограничивались небольшой областью конкретных знаний (при этом проблема понимания здравого смысла не возникала). Простая конструкция экспертных систем позволяла относительно легко их программировать и модифицировать после создания. В целом данные программы оказались полезными, чего ИИ ранее достичь не удавалось.

1993–2001 годы ознаменовались возвращением к исследованию и разработке искусственного интеллекта. Одной из причин возобновления интереса к ИИ стало появление более производительных и дешевых компьютеров. Закон Мура предсказывает, что скорость и объем памяти компьютеров удваиваются каждые два года. Именно это и произошло. Наконец-то прежние обещанные перспективы развития и исследования искусственного интеллекта были реализованы благодаря появлению более мощных и быстрых вычислительных машин, отсутствие которых было основной причиной неудачи еще первой зимой ИИ. Были созданы специализированные компьютеры с использованием передовых технологий ИИ, которые превосходили по своим способностям людей. Как же можно забыть культовый матч 1997 года между компьютером Deep Blue компании IBM и тогдашним действующим чемпионом мира по шахматам Гарри Каспаровым?

ИИ широко использовался в области робототехники. Взять, к примеру, японских роботов, которые выглядели как люди и даже понимали и говорили на человеческих языках. Западный мир не отставал. И вскоре началась гонка – кто первый построит наиболее человекоподобного механического помощника человека. Робот ASIMO компании Хонда был блестящим примером того, что может быть достигнуто путем объединения робототехники с ИИ: 130-санитметровый гуманоид, который мог ходить, танцевать, готовить кофе и даже дирижировать оркестром.

Нынешнее состояние дел

Изначально ИИ разрабатывался для создания человекоподобных роботов, которые могли бы нас понимать, выполнять нашу рутинную работу и избавлять от одиночества. Но сегодня, благодаря появлению различных методов, помогающих создавать надежные, умные и функциональные приложения, область применения ИИ стала гораздо шире.

С появлением нового поколения технологических компаний XXI век стал свидетелем огромных достижений в области искусственного интеллекта. Иногда эти достижения скрываются в тишине научно-исследовательских лабораторий Microsoft, IBM, Google, Facebook, Apple, Amazon и других. Возможно, одним из лучших примеров современного искусственного интеллекта является суперкомпьютер Watson компании IBM, первоначально разрабатывавшийся как компьютерная система, которая могла бы конкурировать с людьми. В феврале 2011 года этот суперкомпьютер принял участие в телевикторине «Jeopardy!» – аналог хорошо нам знакомой телевикторины «Своя игра». Будучи отключенным от сети Интернет, суперкомпьютер смог обыграть двух игроков: Брэда Раттера – обладателя самого большого выигрыша в программе и Кена Дженнингса – рекордсмена по длительности беспрогрышной серии. Компьютер одержал победу, получив 1 млн долларов. Дженнингс и Раттер получили, соответственно, по 300 и 200 тысяч. На волне успеха своего суперкомпьютера IBM вскоре выпустила коммерческие приложения, в которые были встроены технологии искусственного интеллекта. Интерес крупных технологических компаний к ИИ снова возрос. В результате и другие крупные технологические

компании вышли на рынок со своими собственными коммерческими предложениями. Сегодня существуют стартапы, предлагающие узкоспециализированные коммерческие предложения со встроенными технологиями ИИ.

Искусственный интеллект не был ограничен массовыми и корпоративными программными приложениями. Уже несколько лет ИИ встраивается в ваши любимые видеоигры, как телевизионные, так и мобильные. Например, в однопользовательских играх, где вы конкурируете против компьютера, ваши виртуальные противники принимают свои собственные решения в зависимости от ваших действий. Во многих играх даже можно менять уровень сложности противников: чем выше уровень сложности, тем более изощренным является искусственный интеллект игры и тем больше ваши противники становятся похожи на людей.

Превращение ИИ в товар

За последние годы произошел взрывной, почти экспоненциальный рост объема информации. В условиях ежедневно дешевающего пространства для хранения данных крупные корпорации, равно как и небольшие стартапы, стали сохранять свои архивы, чтобы в какой-то момент их проанализировать и получить важную информацию, способную принести выгоду своему предприятию. Эта тенденция в значительной части была поддержана облачной революцией. Сама облачная революция обусловлена появлением компьютеров с более высоким быстродействием и дешевеющим пространством хранения данных. Услуги облачного вычисления и хранения данных в облаке, предоставляемые такими компаниями, как Amazon AWS и Microsoft Azure, настолько дешевые, что уже больше не имеет смысла уничтожать даже журнальные данные десятилетней давности, произведенные серверами и программным обеспечением предприятия.

В результате каждый день и каждый час компании производят ошеломляющие объемы данных. Этот большой объем данных называется **большими данными**. Большие данные применяются почти во всех секторах экономики, таких как банковское дело, розничная торговля, ИТ, социальные сети, здравоохранение, наука, спорт и т. д.

 Чтобы вообразить масштаб больших данных, обратите внимание на приведенную ниже статистику.

По состоянию на август 2017 г. поисковик Google обрабатывал примерно 100 миллиардов поисковых запросов в месяц. Это больше 1,2 триллиона запросов в год! Google анализирует свои поисковые данные с целью определения поисковых тенденций среди различных географических регионов и демографических групп.

Социальная сеть Facebook в день обрабатывает свыше 300 миллионов фотографий из своей базы пользователей. Facebook анализирует свои данные, посты и фотографии с целью предоставления своим пользователям более точных объявлений.

Крупнейшая в мире сеть оптовой и розничной торговли Wal-Mart каждый час обрабатывает свыше 1 миллиона покупательских транзакций. Wal-Mart анализирует эти данные, чтобы понимать, какие товары реализуются лучше других, какие товары реализуются вместе, и получать другую розничную аналитическую информацию.

Традиционные методы обработки данных стали нежизнеспособными из-за сложности данных и времени, которое требуется для их полного анализа. Чтобы проанализировать этот огромный объем данных, был необходим радикально новый подход. Как оказалось, с большими данными могут использоваться

и методы машинного обучения, применяемые для тренировки сложных систем ИИ. В результате сегодня ИИ больше не является прерогативой крупных частных и общественных научно-исследовательских институтов. ИИ и его разнообразная методология используются для построения и поддержки программных продуктов для всех видов компаний.

MICROSOFT И ИИ

Компания Microsoft располагает богатой историей в области развития искусственного интеллекта. Когда в 1991 г. Билл Гейтс создал Microsoft Research, он предвидел, что однажды компьютеры будут способны видеть, слышать и понимать людей. Двадцать шесть лет спустя ИИ приблизился к воплощению таких прогнозов. За эти годы Microsoft не объявляла о создании гуманоидных роботов и не строила всезнающие универсальные компьютеры. Прогресс компании в области ИИ для общественности был «не видим». Компания потихоньку интегрировала человечоподобное мышление в свои существующие продукты.

Возьмем, к примеру, Microsoft Bing, популярную поисковую систему компании Microsoft. Мало того, что Bing может выполнять поиск по ключевым словам. Этот поисковик также может осуществлять поиск в Интернете на основе подразумеваемого смысла вашей поисковой фразы. Так, выполнение простого поиска по ключевым словам «Тейлор Свифт» дает вам официальный сайт, страницу Википедии, учетные записи в социальных сетях, последние новости и несколько фотографий популярного американского певца-автора песен. Выполнение более сложного запроса, типа «Кто президент Уганды?», даст вам точное имя крупными буквами и главные результирующие веб-страницы, относящиеся к этому человеку. Это равнозначно тому, как спросить «Кто президент Уганды» у человека, который знает, что вам нужно только имя президента данной страны, а не все веб-страницы, связанные с этим именем.

В обоих примерах (Тейлор Свифт и президент Уганды) поисковик Bing также покажет слева краткую справку о человеке: дату рождения, супругу(а), детей и т. д. И в зависимости от того, кто этот человек, вы увидите другие подробности, такие как образование, годы жизни, цитаты политика и собственный капитал, музыкальные композиции и песни для певца. Каким образом поисковой системе Bing удается показывать вам столько много информации о человеке? Неужели разработчики этого поисковика создали мегабазу данных справочных фактов для всех мировых знаменитостей (в настоящем и прошлом)? Не совсем.

Хотя человеческих способностей хватит, чтобы создать такую базу данных, стоимость ее поддержки была бы огромной, так как наш большой и обширный мир, в котором существует множество стран и территорий, продолжает производить знаменитостей. Так что с этой базой данных возникнет определенная проблема масштабируемости.

Технология, которую компания Microsoft применила для решения этой задачи, называется *машинным обучением*. Мы рассмотрим машинное обучение и его младшего брата – глубокое обучение – чуть ниже. Поисковая система Bing научилась понимать значение поисковой фразы только благодаря подобласти ИИ, название которой – *понимание естественного языка*. Можно задать поисковику Bing тот же самый вопрос десятью различными способами, и каждый раз он по-

прежнему будет приходить к одному и тому же смыслу. Методология понимания естественного языка (ЕЯ) достаточно умна, чтобы толковать естественно-языковые высказывания таким образом, как это люди делают подсознательно. Методология понимания ЕЯ также помогает обнаруживать в поисковых выражениях орфографические ошибки: вопрос «Кто президент Уганды» поисковик Bing автоматически исправит на «Кто президент Уганда».

ФУНДАМЕНТАЛЬНЫЕ ПОНЯТИЯ

Прежде чем вы начнете строить умные приложения с использованием элементов искусственного интеллекта, будет полезно узнать его основы. В этом разделе мы познакомимся с фундаментальной терминологией и узнаем, что происходит за кадром при работе каждого из этих терминов. Благодаря этому вы получите общее представление о том, как работает ИИ. Рисунок 1.1 показывает мимолетный взгляд в будущее, когда человек обучает машину и машина делает заметки.



Рис. 1.1 ♦ Человек преподает машине основы, и машина делает заметки

Но, прежде чем мы сможем углубиться в изучение подробностей работы различных форм ИИ, важно уяснить одно понятие, которое приводит все эти формы в действие. Этим понятием является машинное обучение (рис. 1.1).

Термин «машина́ное обучение» был введен Артуром Сэмюэлем в своей статье 1959 года «Некоторые исследования в машинном обучении». Согласно Сэмюэлю, машинное обучение – это то, что «*придает компьютерам способность учиться, без программирования их явным образом*». Все мы знаем компьютер как машину, которая выполняет определенные операции, следуя инструкциям, поставляемым людьми в форме программ. Но тогда каким образом можно сделать так, чтобы машина училась чему-то самостоятельно? И что такая машина будет делать со знаниями, полученными в результате такого обучения?

Чтобы лучше разобраться в машинном обучении, давайте возьмем в качестве примера популярный инструмент машинного перевода Google Translate, инстру-

мент, позволяющий легко переводить фразы с иностранного языка, скажем французского на английский и наоборот. Вы когда-либо задавались вопросом: как он работает?

Рассмотрим приведенное ниже предложение на французском языке (рис. 1.2).

Quel est ton numéro de téléphone?

Рис. 1.2 ♦ Предложение на французском языке

Надпись на рисунке: Какой ваш номер телефона?



Самая простая система перевода перевела бы это предложение пословно при помощи словаря (рис. 1.3).

Quel est ton numéro de téléphone?

which is your number of telephone

Рис. 1.3 ♦ Буквальный перевод на английский язык

Надписи на рисунке: Который есть ваш номер телефон

Если фразу, означающую на французском языке «какой ваш номер телефона», перевести на русский язык буквально, мы получим словосочетание «который есть ваш номер телефон». Очевидно, такой упрощенный перевод полностью игнорирует правила грамматики, характерные для русского языка.

Ситуацию можно исправить, введя в систему перевода правила грамматики обоих языков. Но вот проблема: правила грамматики работают, исходя из допущения, что входное предложение грамматически оформлено правильно. В реальном мире это не всегда так. Кроме того, может иметься несколько разных правильных вариантов результирующего предложения. Такую систему, основанную на грамматических правилах перевода, стало бы слишком сложно поддерживать.

Идеальная система перевода может учиться переводить самостоятельно, получая лишь тренировочные данные. Перебрав тысячи и тысячи обучающих фраз, система начнет видеть повторяющиеся шаблоны (образы) и в результате автоматически выводить правила языка. Машинное обучение как раз и занимается такого рода самообучением.

Google Translate поддерживает не 10 и не 20, а более 100 языков, включая некоторые редкие и малоизвестные, при этом в него регулярно вносятся дополнительные языки. Конечно же, человеческих способностей компьютеру не хватит, чтобы *жестко запрограммировать* переводы всех возможных фраз и предложений. Способность Google-переводчика понимать языки и переводить с языка на язык приводится в действие машинным обучением.

Фразы после перевода Google-переводчиком не совершенны, но их смысл сохранен. Google-переводчик учится не только на обучающих данных, которые ему поставляет Google, но и благодаря миллионам своих пользователей. В случае неправильного перевода пользователи могут вручную представить правильный ва-

риант. Google-переводчик учится[®] на своих ошибках, точно так же, как человек, и улучшает свое понимание языков ради будущих переводов. Яркий пример машинного обучения!

i Совсем недавно Google-переводчик перешел с использования алгоритмов машинного обучения на алгоритмы глубокого обучения.

Машинное обучение против глубокого обучения

Если вы следили за новостями, то, вероятно, вам доводилось слышать термин «глубокое обучение» в связи с искусственным интеллектом. Глубокое обучение – это новейшая разработка в области машинного обучения, и люди, которые, по-видимому, не знакомы с точным значением этого термина, понимают этот термин как преемника машинного обучения. Это совершенно неправильно.

В то время как машинное обучение является способом реализации искусственного интеллекта, глубокое обучение является методом машинного обучения. Другими словами, глубокое обучение является не альтернативой машинному обучению, а частью самого машинного обучения.

В машинном обучении общепринятой методикой традиционно были искусственные нейронные сети (ИНС). ИНС очень сильно загружают центральный процессор. Но результаты этих вычислений ниже человеческих возможностей. Недавняя революция в области ИИ стала возможной благодаря глубокому обучению, впечатляющему методу, который выполняет машинное обучение намного быстрее и точнее. В алгоритмах глубокого обучения для достижения поставленной цели (идентификации изображения, перевода с языка на язык и т.д.) применяется параллельное программирование, которое опирается на миллионы предцентров обучающих данных. Такое «глубокое» обучение было бы невозможным на основе предыдущих методов машинного обучения.

Компании разработали свои собственные инструменты глубокого обучения и в результате предложили облачные службы с элементами ИИ. В конце 2015 г. компания Google представила в свободный доступ свою платформу глубокого обучения, название которой TensorFlow. Чтобы узнать возможности этой платформы и как ее можно использовать, обратитесь по адресу: www.tensorflow.org.

Машинное обучение

Технология машинного обучения (МО) находится в самой основе искусственного интеллекта. В рамках МО проводятся исследования и конструирование алгоритмов, которые могут обучаться на ранее обработанных данных и прогнозировать, основываясь на результатах этого обучения. Именно МО приводит в действие интеллектуальную машину, и именно на МО функционирует искусственный интеллект.

Традиционный не-МО алгоритм машинного перевода имеет статические программные инструкции, которые идентифицируют, на каком языке написано предложение: используемые слова, грамматическая структура и т. д. Не-МО алгоритм идентификации лиц аналогичным образом имеет жестко запрограммированное определение лица: нечто округлое с цветом кожи и двумя небольшими темными областями в верхней части (глазами) и т. д. С другой стороны, у МО алгоритма нет такого жесткого программирования; он учится на примерах. Если в процессе тренировки использовать большое количество предложений на французском языке и добавить небольшое количество предложений не на французском, МО алгоритм научится определять французские предложения.

Целый ряд реальных практических задач, таких как перевод с языка на язык, предсказание погоды, фильтрация спама в почтовых сообщениях, предсказание



результатов выборов следующего президента Соединенных Штатов, задачи классификации (такие как различение видов птиц по фотографиям) и т. д., имеет нелинейный характер. МО является идеальным решением для таких нелинейных задач, где проектирование и программирование задаваемых явным образом алгоритмов с использованием статических программных инструкций попросту не работает.

Надеемся, что пример с машинным переводом в предыдущем разделе дал вам четкое понимание того, как работает машинное обучение. Это была просто верхушка айсберга. Технология МО является многосложной и многогранной. Но вы узнали, что МО – это фундаментальное понятие. МО является под областью информатики, включающей в себя несколько тем, в частности связанных с математикой и статистикой. Хотя для охвата всех аспектов МО потребуется не одна книга, давайте взглянем на общепринятые термины, которые с ним связаны (рис. 1.4 и 1.5).

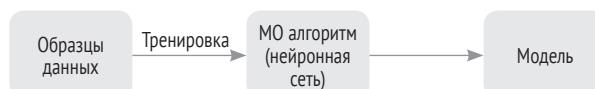


Рис. 1.4 ♦ Алгоритм машинного обучения, в частности нейронная сеть, которая «обучается» основам темы на обучающих данных. Результатом такого обучения является натренированная модель

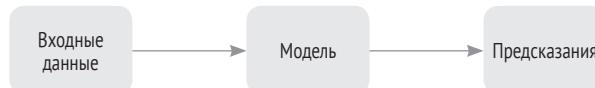


Рис. 1.5 ♦ Натренированная модель затем может принимать новые или ранее известные данные и делать информированные предсказания

Прежде чем система машинного обучения сможет начать разумно отвечать на вопросы по теме, она должна этой теме обучиться. Для этого МО в значительной степени опирается на начальный набор данных по теме обучения. Эти исходные данные называются *тренировочными данными*. Чем больше обучающих данных, тем больше шаблонов (образов) наша машина в состоянии идентифицировать и тем точнее она сможет отвечать на вопросы – новые и ранее известные – по этой теме. Для получения надежных результатов обычно недостаточно нескольких сотен или даже тысяч записей обучающих данных.

Действительно точные, человекоподобные машины были натренированы с использованием миллионов записей или нескольких гигабайтов данных в течение дней, месяцев или даже лет. И мы ни на йоту не преувеличиваем. Продолжительность непрерывной работы персонального компьютера с хорошей вычислительной мощностью и высококачественной видеокартой для тренировки алгоритма машинного перевода данными объемом свыше 1 Гб только для одной пары языков составит больше месяца [см. <https://github.com/tensorflow/tensorflow/issues/600#issuecomment-226333266>].

Качество тренировочных данных и то, как модель сконструирована, имеют равнозначное значение. Используемые данные должны быть точными, очищен-

ными и поступать через надежные источники. Модель должна быть разработана с использованием сценариев реальной практической деятельности. Так что в следующий раз, когда ваше приложение распознавания образов неправильно идентифицирует захваченный объект или ваше любимое приложение машинного перевода выдаст смешной перевод, вините в этом качество обучающих данных или модель, которая использовалась. Кроме того, важно отметить, что обучение не является процедурой начального этапа: это непрерывный процесс. Первоначально машина обучается на тренировочных данных; позже она делает это на пользователях.

Исследование в области ИИ привело к развитию нескольких подходов к реализации машинного обучения. Искусственная нейронная сеть является одним из самых популярных подходов. ИНС, или просто *нейросеть*, является обучаемым алгоритмом. Основой для разработки нейросети послужили структура и функциональные аспекты биологических нейронных сетей. Вычисления структурированы с точки зрения взаимосвязанных групп искусственных нейронов, обрабатывающих информацию, используя ассоциативный подход к компьютерным вычислениям. Они используются для моделирования сложных связей между входами и выходами с целью отыскания этих связей в данных шаблонов (образов). Другие популярные подходы представлены *глубоким обучением*, системами на основе *продукционных правил* (если – то), *разветвленными алгоритмами решений* и *байесовскими сетями*.

Таким образом, когда в нейронные сети подается достаточный объем тренировочных данных, мы получаем то, что называется *натренированной моделью*. *Модели* – это математические и статистические функции, которые могут создавать *прогнозы* (информированные предположения) в отношении предоставленных входных данных. Например, основываясь на метеорологической информации за прошедшие 10 лет (обучающих данных), модель машинного обучения может научиться предсказывать погоду на несколько дней вперед.

Типы машинного обучения

Обучение с учителем, или контролируемое обучение, выполняется, когда тренировочные данные помечены. Если в процессе обучения алгоритма распознавания языка предложения, передаваемые в этот алгоритм, будут помечены, то такое обучение называется обучением с учителем. То есть в этом случае требуется передавать алгоритму предложения, явным образом помеченные языком, на котором они написаны. Например, предложение на французском и не на французском языке, на испанском и не на испанском языке и т. д. Поскольку предварительная разметка выполняется людьми, эта работа увеличивает трудозатраты и стоимость поддержания таких алгоритмов.

Обучение без учителя, или неконтролируемое обучение, выполняется, когда тренировочные данные не помечены. Из-за отсутствия меток, разумеется, алгоритм не может волшебным образом научиться правильно определять язык предложения, но он может дифференцировать один язык от другого. Таким образом, используя обучение без учителя, алгоритм МО может научиться определять, что французские предложения отличаются от испанских, которые отличаются от хинди, и т. д.

Обучение с подкреплением, или стимулированное обучение, выполняется, когда машине не передаются тренировочные данные явным образом. Чтобы достиг-

нуть поставленной цели, алгоритм должен взаимодействовать с окружающей средой. Из-за отсутствия обучающих данных он должен обучаться самостоятельно с нуля и полагаться на метод проб и ошибок, чтобы принимать решения и обнаруживать свои собственные правильные пути. У каждого выполняемого машиной действия есть последствие, и каждому последствию задано числовое вознаграждение. Таким образом, если действие приводит к желательному результату, оно получает «хорошие» отметки. И если результат имеет катастрофические последствия, то он получает отметки «очень и очень плохо». Как и люди, машина стремится максимизировать свое итоговое числовое вознаграждение – то есть получить как можно больше «хороших» и «очень хороших» отметок, не повторяя свои ошибки. Этот метод машинного обучения особенно полезен, если машина будет работать в быстро изменяющихся средах, где создание и поставка обучающих данных попросту невозможны. Это такие среды, как управление автомобилем (рис. 1.6), видеоигры и т. д.



Рис. 1.6 ♦ Самоуправляемые автомобили – автотранспортные средства, которые не требуют, чтобы ими управлял человек, используют обучение с подкреплением, чтобы обучаться на данных динамической и сложной среды (дороги и дорожное движение) с целью постепенного улучшения своих навыков вождения

Язык

Люди взаимодействуют друг с другом одним из трех способов: словесно, письменно и жестами. Эти три способа объединены в одно понятие – «язык». Язык – это свод правил общения, который одинаков для каждого человека. Общение может быть устным и письменным, и между этими двумя способами общения есть различия. Формальной формой общения является письмо. При этом язык знаков, язык жестов полностью отличается.

Большая часть усилий, приложенных в исследовании ИИ, состояла в том, чтобы наделить машины способностью понимать людей так же естественно, как это делают сами люди. Поскольку машинам легче понимать письменную речь, чем устную, мы начнем наше обсуждение с основ понимания языка в письменной форме.

Понимание естественного языка

Понимание естественного языка (ЕЯ) – это способность машины понимать людей посредством естественных языков. Компьютер изначально разработан, чтобы по-



нимать биты и байты, программный код и логику, программы и инструкции, а не естественные языки. Иными словами, компьютер умело управляет структурированными данными, чего нельзя сказать о неструктурированных данных.

Естественный язык регламентируется некоторыми правилами (грамматикой), однако во время ежедневного и неофициального общения эти правила не всегда соблюдаются. В результате люди легко могут понимать письменные или словесные высказывания с плохой грамматикой, неправильным произношением, пропусками, сокращениями и т. д. Можно смело сказать, что естественные языки регламентируются гибкими правилами.

Технология понимания ЕЯ превращает неструктурированные исходные данные (рис. 1.7), регламентированные гибкими и плохо определенными правилами, в структурированные данные, которые машина способна понять. Для любознательных отметим, что именно эта технология делает человекоподобными Microsoft Кортана, Apple Сири и Amazon Алекса.

Какая завтра ожидается погода в Дели

День

Намерение

Город

Рис. 1.7 ♦ Технология понимания ЕЯ анализирует каждое предложение с целью получения двух вещей: намерения (смысла, значения или предполагаемого действия) и сущностей. В этом примере извлеченная информация о погоде является распознанным намерением, а город (Дели) и день (завтра) являются сущностями. Пользователь может задать тот же самый вопрос сотнями различных способов, и хорошая система понимания ЕЯ всегда будет в состоянии извлечь из пользовательского запроса правильное намерение и сущности.

Далее программная система сможет использовать эту ранее извлеченную информацию для подготовки запроса к онлайновому метеорологическому API и показать пользователю требующуюся информацию о погоде

Обработка естественного языка

Разумеется, в человеко-компьютерном взаимодействии гораздо больше задач, чем только одно понимание намерения заданного предложения. Технология обработки ЕЯ охватывает все, что имеет отношение к человеко-компьютерному взаимодействию на естественном языке. И технология понимания ЕЯ является всего лишь подмножеством задач в большем множестве, которым является технология обработки ЕЯ. Технология обработки естественного языка, в частности, включает в себя такие задачи, как:

- **машинный перевод:** преобразование текста из одного языка в другой;
- **генерация естественного языка:** обратная по отношению к пониманию ЕЯ задача; преобразование структурированных данных (обычно из баз данных) в текстовые предложения, понятные человеку. Например, на основе сравнения двух строк в базе данных с информацией о погоде может быть сформулировано вот такое предложение: «сегодня в Дели температура воздуха составляет 26 градусов по Цельсию, что ниже на 2 градуса, по сравнению со вчерашним днем»;
- **анализ мнений:** просканировать фрагмент текста (твит, обновление статуса Facebook, обзоры, комментарии и т. д.), касающийся товара, человека, события или географического места, чтобы определить общее мнение (отрицательное или положительное) в отношении соответствующего субъекта;

- **распознавание именованных сущностей:** в том или ином тексте определить, какие фрагменты в тексте соотносятся с именами собственными, связанными с людьми или географическими объектами, и тип каждого определенного имени (человек, местоположение, организация);
- **извлечение связей:** извлечение связей между сущностями, присутствующими во фрагменте текста, в частности родственные отношения, причины, признаки и т. д.

Обработка ЕЯ намного шире, чем несколько упомянутых выше задач, причем каждая из них является предметом независимого исследования.

Речь

Помимо интеллектуального анализа текста, ИИ способен помочь машинам с акустическими устройствами, такими как микрофон, понимать, о чём говорится. Устная речь представляется как набор звуковых сигналов, и акустическое моделирование используется для нахождения связей между звуковым сигналом и фонемами (минимальными смыслоразличительными единицами языка, которые составляют устную речь).

Распознавание речи

Распознавание речи – это распознавание и преобразование компьютерами устного языка в текст. Когда вы задаете вопрос Siri или Google (голосовой поиск), они используют методологию распознавания речи для преобразования вашего голоса в текст. Преобразованный текст затем используется для выполнения поискового запроса. Современные методы распознавания речи умеют справляться с отклонениями в акцентах, похоже звучащими словами и фразами на основе контекста.

Применения распознавания речи колеблются от проектирования общедоступных систем (таких как программное обеспечение для слепых) или систем на основе голосового поиска до голосового (бесконтактного) набора текста.

Распознавание голоса

Термины «*распознавание голоса*» и «*идентификация говорящего*» относятся к идентификации диктора, а не к смыслу произносимой фразы. Распознавание говорящего может упростить задачу перевода речи в системах, натренированных на голос определенного человека. Распознавание голоса диктора может использоваться с целью подтверждения подлинности или проверки личности говорящего и служить в качестве составной части процесса обеспечения безопасности.

Преобразование текста в речь и речи в текст

Преобразования текста в речь (текст-в-речь, Text-to-Speech, TTS) и речи в текст (речь-в-текст, Speech-to-Text, STT) являются взаимосвязанными, но разными технологиями.

Преобразование текста в речь, также именуемое *синтезом речи*, – это способность машины «проговаривать» фрагмент письменного текста. Синтезируемая речь создается путем соединения частей записанной речи (записей каждого слова), которые хранятся в базе данных. В альтернативном варианте синтезатор может содержать модель речевого тракта и других характеристик человеческого голоса с целью создания абсолютно «синтетического» голосового результата на выходе.

С другой стороны, преобразование устной речи в текст является следующим шагом в распознавании речи. После того как речь была разложена на звуковые сигналы и затем на фонемы, компьютер может преобразовывать эти фонемы в текст. Используя тот же самый набор и последовательность фонем, можно построить многочисленные текстовые предложения, в результате чего машина разумно назначает каждой конструкции степень уверенности для придания предложениям большей осмыслинности.

Компьютерное зрение

Наконец, мы дошли до раздела, где обсудим методы искусственного интеллекта, с помощью которых происходит обработка визуальных данных. Визуальные данные включают в себя как неподвижные изображения – фотографии, так и динамически изменяющееся изображение – видео. Общее название этого термина – компьютерное зрение, т. е. способность компьютера «видеть». Компьютеры по своей природе не способны обрабатывать изображения так же хорошо, как текст. Методы обработки изображений в соединении с интеллектуальными алгоритмами ИИ позволяют машинам «видеть» изображения, идентифицировать и распознавать объекты и людей.

Обнаружение объектов

Сцена на фотографии может содержать десятки или даже сотни объектов. Но обычно в зависимости от ситуации внимание фокусируется только на нескольких объектах. Давайте назовем их «интересующими». Распознавание изображения обозначает способность машины обнаруживать интересующие объекты, которые могут меняться в зависимости от ситуации. Например:

- ускоряющийся автомобиль на дороге (управление дорожным движением) (рис. 1.8);



Рис. 1.8 ♦ Автомобиль, обнаруживаемый на дороге

- планетоподобный объект в обширной Солнечной системе или галактике (астрономия);
- грабитель, проникающий через задний двор (безопасность жилища);
- группа людей, входящая в торговый центр (ведение учета посещаемости).

Распознавание образов

За обнаружением обычно следует распознавание. Распознавание – это способность распознать и назначить обнаруженным объектам и действиям точный тип (рис. 1.9). Например:

- распознавание лодки, двух человек, воды и солнца на изображении;
- распознавание точных видов животных на фотографии.



Рис. 1.9 ♦ Группа велосипедистов на грязной дороге

Распознавание образов также называется *классификацией*, или *сравнением, объектов*. Среди других систем распознавание объектов характерно для приложений виртуальной реальности, таких как Google Goggles, предназначенное для визуального поиска и распознавания изображений.

Точность системы распознавания образов, как и все остальное в ИИ, в значительной степени зависит от обучающих данных. При помощи методов машинного обучения, чтобы научиться распознавать объекты определенного класса, система проходит тренировку на сотнях изображений. Так, можно сначала натренировать систему распознавать обычную собаку, используя сотни изображений с одной или несколькими собаками. После того как система в состоянии распознавать собак, затем ее можно натренировать распознавать немецкую овчарку, добермана или даже чихуа-хуа.

Распознавание лиц

Обнаружение и распознавание лиц являются одной из многих подзадач распознавания образов (рис. 1.10). С помощью этих методов система способна обнаруживать на фотографии не только лица, но и связанные с ними признаки (возраст, пол, улыбку и т. д.). И если система предварительно была натренирована на лицо определенного человека, она может выполнить сравнение, чтобы распознать лицо именно этого человека на фотографии.

Распознавание лиц при использовании камер скрытого видеонаблюдения может использоваться в качестве механизма проверки подлинности или для обнаружения опасного преступника в местах массового скопления людей.



Рис. 1.10 ♦ Лица, определяемые по изображению

Оптическое распознавание символов

Оптическое распознавание символов (ОРС, Optical Character Recognition, OCR) – это метод, используемый для преобразования рукописного, набранного на печатной машинке или распечатанного на бумаге текста в данные, которые можно отредактировать на компьютере. Система оптического распознавания просматривает сканированные изображения печатных документов и сравнивает формы отсканированных букв с эталонными изображениями букв. В результате этого создается текстовый файл, который можно отредактировать в обычном текстовом редакторе. Текст, обнаруженный с использованием ОРС, затем можно передать в систему преобразования текста в речь (TTS), чтобы проговорить его громко вслух слепому человеку, который в противном случае не смог бы этот документ увидеть.

Технология OPC обычно используется онлайновыми книжными магазинами для создания электронных копий печатных книг. Она также используется некоторыми инструментами машинного перевода, чтобы переводить вывески с иностранного языка, используя камеру мобильного телефона. Рисунок 1.11 показывает, как Google применяет OPC, превращая ваш телефон в переводчика в реальном режиме времени.



Рис. 1.11 ◆ Google превращает ваш телефон в переводчика, действующего в реальном режиме времени

Когнитивные службы Microsoft

Когнитивные службы (Cognitive Services) – это набор программного обеспечения Microsoft, предлагаемого на коммерческой основе в виде служб SaaS со встроенным искусственным интеллектом. Когнитивные службы представляют собой результат многолетних исследований компании Microsoft в области когнитивных вычислений и искусственного интеллекта, и многие из этих служб применяются в некоторых собственных популярных продуктах Microsoft, таких как Bing (поиск, карты), Переводчик, среда для создания программ-роботов Bot Framework и т. д.

Компания Microsoft предоставляет эти службы в виде простых в использовании веб-служб на основе механизма REST API, работающих непосредственно в веб-приложении либо мобильном приложении. На момент написания этой книги в наличии имелось 29 когнитивных служб, тематически подразделенных на пять категорий (табл. 1.1).

Таблица 1.1. Когнитивные службы Microsoft

Оптические	Речевые
<ul style="list-style-type: none"> • API компьютерного зрения • API модерации информационного наполнения • API анализа эмоций • API распознавания лиц • API анализа видео • Пользовательская служба компьютерного зрения • Индексатор видео 	<ul style="list-style-type: none"> • API Bing обработки речи • Пользовательская служба обработки речи • API распознавания говорящего • API машинного перевода речи

Окончание табл. 1.1

Язык	Знание
<ul style="list-style-type: none"> • API Bing проверки орфографии • Интеллектуальная служба понимания языка • API лингвистического анализа • API анализа текста • API машинного перевода • API веб-модели языка 	<ul style="list-style-type: none"> • API академических знаний • Интеллектуальная служба связывания именованных сущностей • API исследования знаний • API создания вопрос-ответного интерфейса • API рекомендаций • Пользовательская служба принятия решений
Поиск	
<ul style="list-style-type: none"> • API Bing автозаполнения • API Bing поиска изображений • API Bing поиска новостей • API Bing поиска видео • API Bing веб-поиска • API Bing пользовательского поиска 	

Зрение

Службы компьютерного зрения (Vision) работают с визуальной информацией, главным образом в форме изображений и видео.

- **API компьютерного зрения.** Извлекает важную информацию из изображения, касающуюся его содержимого: интеллектуальное текстовое описание изображения, обнаруженные лица (с возрастом и полом), доминирующие на изображении цвета, и есть ли на изображении контент для взрослых.
- **Модерация информационного наполнения.** Оценивает текст, изображения и видео на наличие оскорбительного и нежелательного содержания.
- **API распознавания эмоций.** Анализирует лица с целью обнаружения спектра чувств, таких как гнев, счастье, печаль, страх, удивление и т. д.
- **API распознавания лиц.** Обнаруживает человеческие лица и сравнивает похожие лица (распознавание лиц), организует людей в группы согласно визуальному подобию (группирование лиц) и опознает ранее помеченных людей по изображениям (верификация лиц).
- **API распознавания видео.** Интеллектуальная обработка видеоданных для распознавания лиц, обнаружения движения (широко используется в системах скрытого видеонаблюдения), генерирование миниатюр изображений и анализ видео в режиме почти реального времени (текстовое описание каждого кадра).
- **Индивидуализированные (пользовательские) службы компьютерного зрения.** Когда требуется выполнить распознавание образов применительно к вещам, которые не относятся к фону, лицам и эмоциям, эти службы позволяют создавать индивидуальные классификаторы изображений, обычно ориентированные на конкретную область применения. Эту службу можно натренировать – скажем, определять различные виды птиц – и затем использовать ее REST API (репрезентативная передача состояния) в мобильном приложении для любителей наблюдения за птицами.
- **Индексатор видео.** Извлекает полезную информацию из видео с использованием таких инструментов, как распознавание лиц (имена людей), речевой анализ мнений (положительный, отрицательный, нейтральный) по каждому человеку и извлечение ключевых слов.

Речь

Эти службы работают с человеческой речью в форме аудиоформата.

- **API Bing обработки речи.** Преобразует речь в текст, извлекает смысл высказывания и преобразовывает текст назад в речь. Этот API подробно рассмотрен в главе 7.
- **Служба персонализированной обработки речи.** Позволяет создавать индивидуальные модели языка для распознавания речи путем его увязки со словарем приложения и разговорным стилем ваших пользователей. Этот API подробно рассмотрен в главе 7.
- **API распознавания говорящего.** Опознает говорящего в записанном или живом речевом аудио. Распознавание говорящего может надежно использоваться в качестве механизма проверки подлинности.
- **API машинного перевода речи.** Переводит речь с одного языка на другой в режиме реального времени между девятью поддерживаемыми языками.



Язык

Эти службы работают с пониманием естественного языка, переводом, анализом и многим другим.

- **API Bing проверки орфографии.** Исправляет орфографические ошибки в предложениях. Помимо словаря, принимает во внимание разрывы между словами, сленг (жаргонизмы), имена людей и названия фирменных брендов. Этот API подробно рассмотрен в главе 5.
- **Интеллектуальная служба понимания языка (Language Understanding Intelligent Service, LUIS).** Служба понимания естественного языка (ЕЯ). Этот API подробно рассмотрен в главах 4 и 6.
- **API лингвистического анализа.** Выполняет разбор текста с использованием детального лингвистического анализа, такого как выделение предложений и их лексемизация (разбиение текста на предложения и предложений на лексемы) и частеречная разметка (назначение меток существительного, глагола и т. д.). Этот API подробно рассмотрен в главе 5.
- **API анализа текста.** Обнаруживает в тексте мнение (положительное или отрицательное), ключевые фразы, темы и языки. Этот API подробно рассмотрен в главе 5.
- **API машинного перевода.** Переводит текст с одного языка на другой и идентифицирует язык данного текста. Этот API подробно рассмотрен в главе 5.
- **API веб-модели языка.** Обеспечивает множество задач обработки естественного языка, не охваченных другими языковыми API: разбиение на слова (вставка пробелов в последовательность слов с недостающими пробелами), совместные вероятности (вычисление частоты совместного появления конкретной последовательности слов), условные вероятности (вычисление частоты следования конкретного слова за другим) и варианты следующего слова (получение списка слов, которые, скорее всего, должны появиться следующими). Этот API подробно рассмотрен в главе 5.

Знание

Эти службы работают с поиском по большим базам данных с целью идентификации сущностей, предоставления результатов поиска и рекомендаций в отношении продукта.

- **API академических знаний.** Позволяет извлекать информацию из академического графика, Microsoft Academic Graph, собственной базы знаний научных/академических научно-исследовательских работ и их содержание. Используя этот API, можно легко найти статьи по автору, научному институту, событию и т. д. Также есть возможность находить аналогичные работы, выполнять проверку на наличие плагиата и получать статистику цитирования.
- **Интеллектуальная служба связывания именованных сущностей.** Находит в тексте ключевые слова (именованные сущности, события, местоположения и т. д.) на основе контекста.
- **Служба исследования знаний.** Добавляет поддержку запросов на естественном языке, предлагает варианты поиска с автозавершением и другие возможности по отношению к вашим собственным данным.
- **Генератор вопросов и ответов.** Чудесным образом создает из предоставленных данных вопросы и ответы в стиле часто задаваемых вопросов. Создатель вопросов и ответов (QnA maker) предлагает совместную работу веб-сайта и API. Использует веб-сайт для создания базы знаний, применяя ваш существующий веб-сайт, PDF, файл Word или обычный текстовый файл часто задаваемых вами вопросов. Генератор вопросов и ответов автоматически извлекает вопросы и ответы из вашего документа (документов) и, основываясь на ваших данных, тренируется давать ответы на вопросы пользователей на естественном языке. Его можно рассматривать как автоматизированную версию LUIS. Вам не приходится тренировать систему, но при этом вы имеете возможность переобучать ее согласно вашим индивидуальным требованиям. API генератора вопросов и ответов представляет собой конечную точку, которая принимает пользовательские запросы и отправляет ответы для вашей базы знаний. При необходимости генератор вопросов и ответов может быть соединен с инфраструктурой для создания программ-роботов Microsoft Bot Framework с целью разработки готовых к работе ботов для Facebook, Skype, Slack и других каналов связи.
- **API рекомендаций.** Этот инструмент в особенности полезен для розничных, онлайновых и обычных магазинов. Данный API помогает увеличивать продажи магазинам, предлагая своим клиентам рекомендации, такие, например, как часто покупаемые наборы товаров. Предлагает рекомендации по товарам, которые могут заинтересовать конкретного покупателя. Эти рекомендации выбираются исходя из истории транзакций товара и т. д. Подобно генератору вопросов и ответов, веб-сайт рекомендаций в своей системе использует ваши существующие данные для создания продуктового каталога и данных об использовании.
- **Служба персонализированного принятия решений.** Использует заданную текстовую информацию для получения контекста, на основе которого предоставленные варианты можно ранжировать по приоритетности и при-

нимать решение на основе полученного рейтинга. Данная служба использует машинное обучение с подкреплением на основе обратной связи с целью постепенного совершенствования своей работы.

Поиск

Эти службы помогают задействовать возможности поисковой системы Bing, занимающей по популярности второе место в мире.

- **API Bing автозаполнения.** Предоставляет вашему приложению форму поиска и, когда пользователь набирает запрос в поле поиска, параллельно с этим предоставляет интеллектуальный опережающий ввод запроса и варианты результатов поиска непосредственно в Bing.
- **API Bing поиска изображений.** Использует поиск изображений Bing и предоставляет изображения на основе фильтров, таких как ключевые слова, цвет, страна, размер, лицензия и т. д.
- **API Bing поиска новостей.** Выводит результаты поиска свежих новостей на основе фильтров, таких как ключевые слова, новизна, страна и т. д.
- **API Bing поиска видео.** Выполняет поиск видео на основе фильтров, таких как ключевые слова, разрешение, продолжительность видео, страна и цена (бесплатные или платные).
- **API Bing поиска в вебе.** Производит поиск в вебе на основе различных фильтров. Также есть возможность получать список похожих запросов для ключевого слова или фразы.
- **API Bing индивидуализированного (пользовательского) поиска.** Назначение этой службы – поиск в Bing на основе индивидуальных значений и тем. В результате вместо поиска по всему вебу поисковик Bing будет искать веб-сайты, основываясь на теме (темах). Он может также использоваться для реализации локализованного поиска по одному или нескольким веб-сайтам из заданного списка.

Все API когнитивных служб имеют две ценовые категории: бесплатную и с оплатой по мере использования. Ценовую категорию можно выбрать, исходя из активности эксплуатации вашего приложения.

К сожалению, описать каждый из этих API в мельчайших подробностях не позволяет объем данной книги. Мы рассмотрим несколько служб из категорий речи, языка и поиска, которые смогут обеспечить вас стартовыми условиями для создания в короткие сроки действительно умных приложений.

Подробнее узнать об этих службах и обо всех внесенных изменениях можно, обратившись по адресу www.microsoft.com/cognitive-services/en-us/apis.

Резюме

Эта глава послужила введением в технологию искусственного интеллекта. Вы познакомились с историей, методами и базовой терминологией ИИ. Мы рассказали об исследованиях, проводимых компанией Microsoft в области искусственного интеллекта. Также получили краткий обзор различных коммерческих предложений от компании Microsoft, в которых применены элементы ИИ в форме когнитивных служб, основанных на механизме REST API.

Подводя итоги, в этой главе вы:

- узнали правду и вымысел об ИИ;
- познакомились с историей и эволюцией искусственного интеллекта;
- узнали, как и где ИИ используется сегодня;
- познакомились с машинным обучением, которое в действительности является основой любой интеллектуальной системы;
- познакомились с готовыми к применению когнитивными службами компании Microsoft, основанными на программных интерфейсах REST API и использующимися для создания интеллектуальных приложений.

В **следующей главе** вы узнаете, какие предварительные требования необходимо соблюсти и какие инструменты установить для создания программного обеспечения с поддержкой ИИ, после чего создадите свое первое умное приложение с использованием программной среды Visual Studio.



Глава 2

Создание приложения с поддержкой ИИ в среде Visual Studio

Весь комплект когнитивных служб Microsoft доступен в виде набора прикладных программных интерфейсов REST API. Программный интерфейс REST API хорош тем, что ему не нужны специальные пакеты SDK или дополнительные библиотеки для языков программирования. REST API имеют конечную точку с URL-адресом и с протоколом передачи данных http. Конечная точка может принимать входные данные и выдавать результат в формате JSON либо XML. Поэтому программные интерфейсы REST API могут создаваться на всех основных языках программирования, таких как C#, Java, PHP, Ruby, Python, JavaScript и т. д. Учитывая, что эта книга предназначается для разработчиков приложений на платформе .NET, ограничимся использованием языка C#. Но в действительности основные принципы использования когнитивных служб остаются одинаковыми для остальных языков программирования.

Перед началом использования Конгнитивных служб вам потребуется ключ подписки для каждой службы, которую вы захотите использовать в вашем приложении. Получить этот ключ возможно только при наличии вашей учетной записи в Microsoft Azure. В момент написания этой книги большая часть программных интерфейсов API в комплекте когнитивных служб была бесплатна. Только некоторые из версий были платными. Бесплатная версия позволяет использовать API абсолютно свободно, однако она функционально ограничена. Поэтому бесплатные версии предназначены только для персональных и малоактивных приложений. Платная версия позволяет использовать все без ограничений функциональные возможности программных интерфейсов API. Ключ подписки требуется как для бесплатной, так и для платной версии API.

- i** Облачная платформа Azure является одной из ведущих платформ облачных вычислений для корпоративного и персонального использования. Она обеспечивает услуги на основе моделей облачных вычислений SaaS (программное обеспечение как услуга), PaaS (платформа как услуга) и IaaS (инфраструктура как услуга) и поддерживает множество различных языков программирования, инструментов и веб-платформ, включая Microsoft-ориентированное и стороннее программное обеспечение и системы. Когнитивные службы являются одной из многих облачных служб, которые предлагаются платформой Azure.

Другие доступные службы включают Windows- и Linux-ориентированные виртуальные машины (виртуальные частные серверы размещения веб-сайтов и приложений), базы данных, хранение данных, кэш и сети доставки контента (CDN).

После завершения этой главы вы:

- узнаете о предварительных требованиях для использования когнитивных служб;
- познакомитесь с процедурой получения ключа подписки от Azure для API когнитивных служб;
- получите представление, как проверить API когнитивных служб;
- узнаете, как создать свое первое приложение с поддержкой ИИ, используя Visual Studio и C#.



Необходимые условия для использования когнитивных служб

Как ранее упоминалось, для использования программного интерфейса REST API нам потребуется выбрать язык программирования и редактор программного кода. Кроме этого, потребуется документация API, из которой мы узнаем формат ввода и вывода данных. Эти требования верны для большей части программных интерфейсов REST API. Для когнитивных служб вам также потребуется ключ подписки, позволяющий обращаться к требуемому программному интерфейсу. Без действительного ключа подписки доступ к этому API-интерфейсу вашей программе предоставлен не будет.

Вот список того, что потребуется для создания приложения с элементами искусственного интеллекта:

- Visual Studio;
- ключ подписки Azure;
- ваш энтузиазм.



В этой главе мы поможем вам создать свое первое приложение. Для его создания будет использовано **API компьютерного зрения**. Как уже говорилось ранее, компьютерное зрение – это способность машины «видеть» изображение точно так же, как его видят люди, и назначать метки различным объектам, свойствам и людям, которые составляют это изображение.

Подготовка среды разработки

Visual Studio (VS) – это одна из самых эффективных интегрированных сред для разработки программного обеспечения. Данная программная среда фактически является редактором программного кода, основанным на платформе .NET. Это единственный инструмент, который необходимо установить на компьютер для создания не только интеллектуальных, но и обычных приложений. В комплект поставки Virtual Studio входит последняя версия платформы .NET, содержащая компиляторы для всех поддерживаемых платформой языков программирования, включая язык C#.

На момент написания этой книги последняя стабильная и готовая к эксплуатации среда разработки VS имела версию 2017. Среда разработки VS 2017 доступна в нескольких редакциях: Community (для сообщества программистов), Professional (профессиональная), Enterprise (корпоративная) и Test Professional (испытательная корпоративная). За исключением версии Community, все остальные версии платные.

На протяжении всей книги автор использует интегрированную среду разработки (IDE) VS Professional 2017. Вы же можете воспользоваться версией VS Community 2017 или любой другой версией VS 2015. Версии VS Professional и Community очень похожи друг на друга, за исключением нескольких корпоративных особенностей, доступных только в Professional.

Если среда разработки VS на вашем компьютере уже установлена, вам больше не о чем беспокоиться. Если же вы VS еще не установили, то мы настоятельно рекомендуем перейти на сайт www.visualstudio.com и загрузить последнюю бесплатную версию Visual Studio Community.

-  Стоит отметить, что Visual Studio, в отличие от Visual Studio Code, является полнофункциональной интегрированной средой разработки для создания настольных, мобильных и веб-приложений. А Visual Studio Code является всего лишь специализированной интегрированной средой разработки для создания веб- и гибридных мобильных приложений. Обратите внимание: VS Code не является редакцией Visual Studio. Скорее, это независимый бесплатный редактор с открытыми исходными кодами для разработки веб-приложений.

ПОЛУЧЕНИЕ КЛЮЧА ПОДПИСКИ AZURE ДЛЯ КОГНИТИВНЫХ СЛУЖБ

Для использования всех 29 API когнитивных служб для своих приложений вам потребуется отдельный ключ подписки для каждого из них. Давайте детально разберем, какие шаги требуется выполнить, чтобы получить ключ подписки на API компьютерного зрения.

Шаг 1. Зарегистрировать учетную запись Azure

Если у вас уже есть учетная запись Azure, то этот раздел вы можете спокойно пропустить. Процесс регистрации учетной записи на Azure прост, но для этого потребуется наличие учетной записи Microsoft. В свою очередь, учетную запись Microsoft можно бесплатно получить на signup.live.com. У вас есть два варианта регистрации учетной записи Azure: получить подписку на платную версию программного обеспечения или поработать с 30-дневной бесплатной пробной версией.

В случае выбора платной версии вам придется внести месячную плату за использованные услуги. Следует учесть: при выборе платной версии программы вам больше не потребуется вносить любую дополнительную оплату, кроме фиксированной месячной стоимости услуг. Например, если вы используете только виртуальную машину, стоимость которой \$20 в месяц, то это все, что вы будете платить ежемесячно, пока ее используете. Если вы выбрали бесплатную версию

API когнитивных служб, пока вы не перейдете на платную версию, вам не придется ничего платить.

Если вы начинающий пользователь, настоятельно рекомендуем воспользоваться 30-дневной бесплатной пробной версией. В этом случае вы получаете бесплатный кредит в размере \$200 на свою учетную запись Azure. Этот кредит можно потратить на использование всех остальных продуктов Azure по своему усмотрению. Когда ваш бесплатный кредит будет исчерпан, что маловероятно, вам придется оплачивать дополнительные платные услуги самостоятельно. По истечении 30-дневного испытательного срока вам будет предоставлена возможность подписаться на платную версию продукта. Если вы это не сделаете, то потеряете доступ ко всем службам, которые вы настроили во время испытательного срока. Но доступ к порталу Azure сохранится за вами. Обратите внимание: в отличие от других пробных версий, по истечении испытательного срока облако Azure автоматически не переключает вас с пробной версии программы на платную версию. Поэтому нет никакого риска непреднамеренного изъятия средств с вашей кредитной или дебетовой карты¹.

Следует отметить, что профессиональная и корпоративная версии Visual Studio получают бесплатные кредиты Azure каждый месяц в размере \$50 для подписчиков на профессиональную версию и \$150 для подписчиков на корпоративную версию. Эта сумма автоматически зачисляется каждый месяц на вашу учетную запись Microsoft, связанную с подпиской на Visual Studio.

Предположим, что у вас нет учетной записи Azure. Поэтому подпишемся на бесплатную пробную версию. При наличии учетной записи Microsoft обратитесь к ресурсам <https://azure.microsoft.com/ru-ru/free> или azure.microsoft.com/en-us/free, после чего нажмите кнопку **Начните использовать бесплатно** (рис. 2.1). Если у вас есть учетная запись в Microsoft, вам предложат авторизоваться.

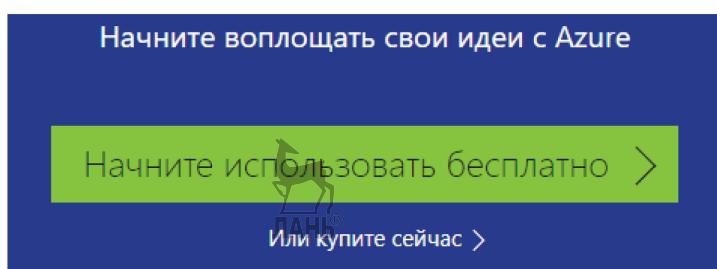


Рис. 2.1 ♦ Начальная страница Azure, когда подписки еще нет

Если учетная запись отсутствует, вы увидите страницу регистрации, где необходимо предоставить основную информацию о себе (рис. 2.2). Для этого в диалоге выбора учетной записи выберите ссылку: *Нет учетной записи? Создайте ее!*

¹ В Azure более 25 служб являются полностью бесплатными, а 12 популярным службам предоставляется годовой бесплатный доступ. – Прим. перев.

1 О вас

Страна или регион 

Имя

Фамилия

Адрес электронной почты 

Телефон

 **ЛАНЬ®**

Далее

Применяются локальные требования к данным. Ознакомьтесь с доступными вариантами покупки и использования Microsoft Online Services.

2 Проверка личности с помощью телефона 

3 Проверка личности с помощью карты 

4 Соглашение 

Рис. 2.2 ❖ Часть формы регистрации пробной учетной записи Azure

Здесь вам будет предложено внести дополнительную информацию для проверки идентичности. От вас потребуют ввести номер вашего мобильного телефона и данные кредитной карты. Обратите внимание, что данные кредитной/дебетовой карты требуются только для проверки. Во время регистрации у вас может быть снята незначительная сумма, но эта транзакция будет незамедлительноозвращена в полном объеме¹. Как указано на веб-сайте Microsoft, с целью удержания низких цен мы обязаны проверять, что владельцы учетных записей являются реальными людьми, а не ботами или анонимными хулиганами. Не волнуйтесь, никаких платежей с вас взиматься не будет, если только вы не решите выбрать платное предложение самостоятельно. Однако мы можем временно заморозить на вашем счету определенную сумму.

¹ В российском сегменте эта сумма составляет 30 руб. – Прим. перев.

После проверки вашей личности для регистрации вам потребуется принять соглашение о подписке. Не пожалейте нескольких минут, чтобы кратко ознакомиться с условиями и деталями соглашения и предложений, ссылки на которые приводятся в разделе соглашения анкеты. После принятия соглашения и нажатия кнопки **Зарегистрировать** вы будете перенаправлены на страницу подписчика. Чтобы на странице подписчика войти на портал Azure¹, нажмите кнопку **Начать управлять моими службами**.

Шаг 2. Создать новую учетную запись когнитивных служб

Теперь настало время создать свой первый ресурс в облаке Azure, а именно учетную запись когнитивных служб. Портал Azure предоставляет вам интерфейс, позволяющий легко создавать новые ресурсы в пошаговом, интуитивно понятном стиле (см. рис. 2.3).

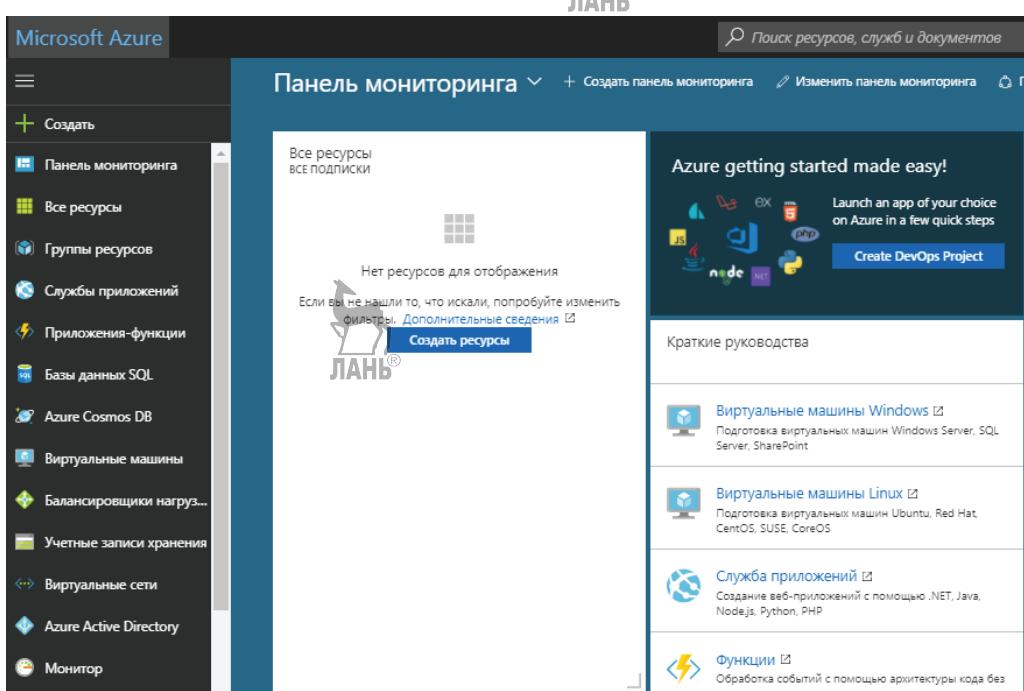


Рис. 2.3 ♦ Портал Azure – ваш личный кабинет

для создания и управления ресурсами, включая виртуальные машины, базы данных и программные интерфейсы когнитивных служб (AI + Cognitive Services)

¹ Следует иметь в виду, что бесплатная пробная версия предоставляется в расчете на одно физическое или юридическое лицо, а не в расчете на одну учетную запись Microsoft. Иными словами, имея несколько учетных записей Microsoft, вы сможете получить только одну пробную версию на свое имя. – Прим. перев.

Каждый шаг в мастере настройки называется сегментом (blade). Давайте создадим учетную запись для API компьютерного зрения.

1. В меню с левой стороны портала нажмите кнопку «+». Появится новый сегмент ресурса.
 2. Этот сегмент предлагает вам два варианта:
 - с помощью поискового запроса, введенного в поле поиска, найти «Cognitive Services»;
 - выбрать в меню **Marketplace** (Торговая площадка) AI + Cognitive Services.
 3. Из обновившегося списка когнитивных служб выберите API компьютерного зрения.
 4. В «сегменте» учетной записи **Создать** заполните области, как показано ниже.
 - Имя учетной записи:** имя, которое однозначно определяет вашу учетную запись API. Это имя не должно повторяться. То есть каждое имя в облаке Azure остается уникальным. Мы рекомендуем выбирать имя для вашей учетной записи по следующей схеме: <моё_имя-cs-тип_apи>. Например, anuragbhd-cs-computervision. Если вы укажете имя, которое уже за кем-то закреплено, вы будете об этом предупреждены.
 - Подписка:** подписку необходимо изменить в случае, если в вашей учетной записи Azure существует более одной подписки. Для пользователей бесплатной пробной версии будет автоматически выбрана «Бесплатная пробная версия».
 - Местоположение:** ресурсы Azure общедоступны в более чем 32 регионах в США, Канаде, Европе, Азии и Австралии. Лучше всего выбрать местоположение, находящееся ближе всего к вашей целевой аудитории. Например, если умное приложение будет предназначаться для Европы, включая Россию, выберите местоположение в Западной Европе.
 - Ценовая категория:** каждый API Cognitive Services имеет свой собственный набор ценовых категорий. Большинство из этих приложений имеет бесплатную и платную категории. Бесплатные категории обычно сопровождаются ограничениями, которые подходят только для небольших и персональных приложений. Для обучения следует выбрать бесплатное приложение.
 - Группа ресурсов:** группа ресурсов позволяет логически группировать свои ресурсы Azure. Например, все, что относится к одному приложению (виртуальная машина, база данных, кэш, API), следует поместить в одну группу ресурсов. Иными словами, все ресурсы в группе имеют одинаковый жизненный цикл: вы их разворачиваете, обновляете и удаляете совместно. Выберите элемент **Создать** и укажите уникальное имя.
 5. После заполнения всех полей нажмите кнопку **Создать** (рис. 2.4). На создание и разворачивание вашей новой учетной записи в отобранной группе ресурсов потребуется от нескольких секунд до минуты. Процесс создания вашей учетной записи можно отследить в меню извещений в верхнем правом углу портала.

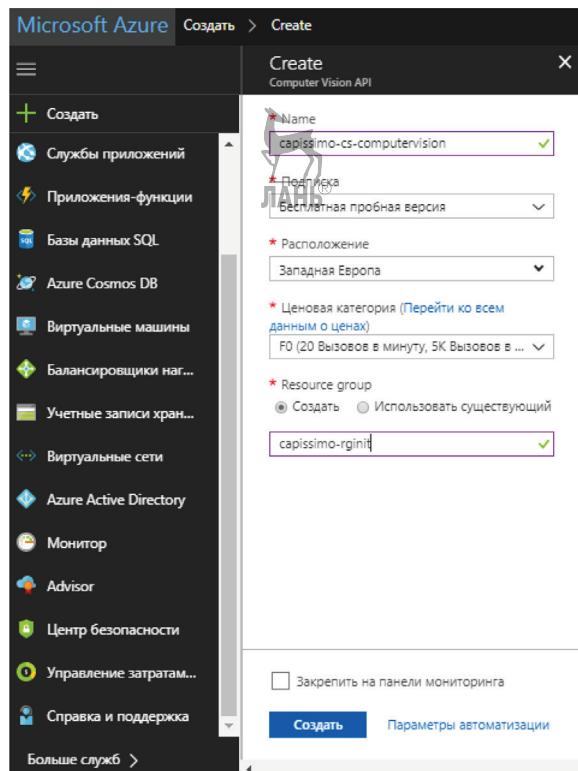


Рис. 2.4 ♦ Создание новой учетной записи Cognitive Services

6. После того как ваша новая учетная запись Cognitive Services будет создана, следует получить ключ подписки, который потребуется для начала использования API компьютерного зрения.

Шаг 3. Получить ключ (ключи) подписки

Создание и разворачивание новой учетной записи Конгитивных служб занимает некоторое время, от нескольких секунд до минуты. По окончании создания и разворачивания учетной записи на экране появится уведомление об успешном выполнении операции, которое свидетельствует, что новая учетная запись готова к использованию.

В левой части меню щелкните мышью на ссылке **Все ресурсы**. Появится список, в котором будет показана недавно созданная ваша учетная запись. Щелкните на ней мышью, чтобы открыть сегмент с подробной информацией о ресурсе.

В сегменте с подробной информацией по умолчанию будет открыт раздел **Overview** (Общие сведения) (рис. 2.5). С помощью полосы прокрутки опуститесь немного вниз и найдите раздел **Keys** (Ключи). Этот раздел находится в категории **Управление ресурсами**. Щелкните мышью на этом разделе. Вам будут показаны созданные специально для вас два ключа подписки. Скопируйте любой из этих двух ключей и сохраните его в удобном месте, например на **Рабочем столе**. Скоро этот ключ вам понадобится.

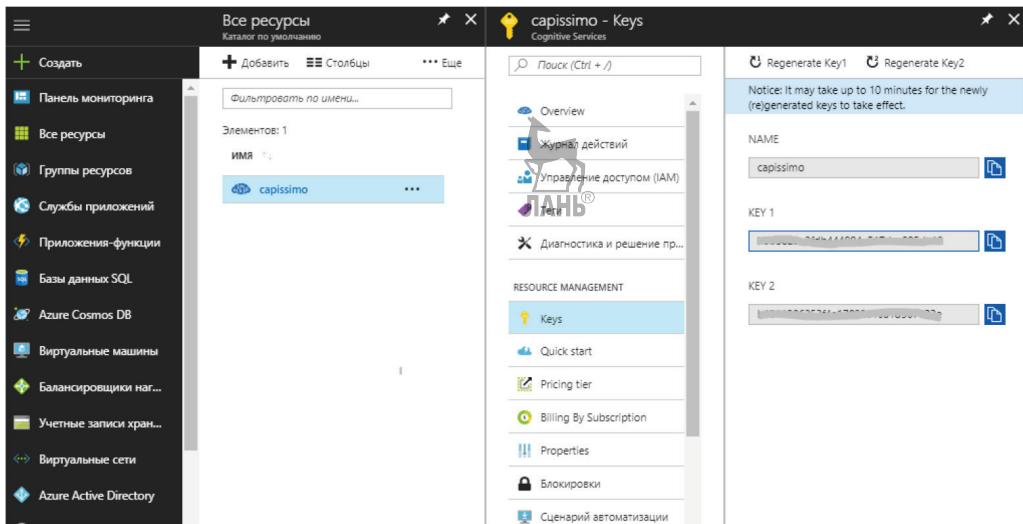


Рис. 2.5. Получение ключа подписки из портала

И Бесплатные подписки, которые не используются непрерывно в течение 90 дней, прекращаются. Однако если ваша подписка истекает, вы можете запросить новые ключи.

ТЕСТИРОВАНИЕ API

Перед тем как начать использовать API, было бы неплохо сначала его проверить. Проверка API предоставит вам следующую важную информацию:

- вы получите четкую картину формата данных заголовков и HTTP-запроса, которые будет получать этот программный интерфейс. Помимо этого, вы напрямую увидите HTTP-отклик, который отправляется программным интерфейсом;
- вы убедитесь, работает ваш ключ подписки или нет. Иногда можно получить ошибку «недопустимый ключ подписки». В этом случае следует применить второй из предлагаемых ключей. Если оба ключа не срабатывают, следует в разделе подробной информации об учетной записи когнитивных служб портала провести регенерацию ключей.

Эта информация полностью относится к RESTful API. Особенno важен первый пункт. Для этих целей создано множество инструментов для тестирования API. Эти инструменты существуют как в виде автономных приложений, так и в виде расширений Chrome/Firefox. Наиболее удобными оказались инструменты тестирования на основе графического интерфейса. При сравнении автономных приложений и сред для тестирования API было выявлено, что многофункциональность и надежность среды разработки и тестирования API под названием Postman оказались наилучшими.

Проще говоря, среда Postman позволяет отправлять HTTP-запросы в REST API и получать отклик. Обычно эта операция производится с использованием языка программирования. Среда Postman предоставляет собой хоть простой, но мощный графический интерфейс. Этот интерфейс позволяет не только конструиро-

вать свои запросы независимо от их сложности, но и сохранять эти запросы для последующего использования. Используя сохраненный ранее запрос, вы сможете повторить его обращение к API. Существует бесплатная версия среды разработки Postman для всех основных операционных систем, включая Windows, Linux и macOS. Для загрузки своей копии среды разработки Postman перейдите на сайт www.getpostman.com¹.

Что вы хотите сделать

Как уже упоминалось ранее, компьютерное зрение – это способность машины «видеть» изображения (точно так же, как это делает человек) и выделять интересующую нас подробную информацию. Например, находить фигуры людей, определять их примерный возраст, улыбаются ли они и т. д.

API компьютерного зрения – это комплект, содержащий несколько функций, таких как анализ изображения, описание изображения, оптическое распознавание символов (OCR) и т. д. От вас требуется всего лишь применить функцию `DescribeImage` API компьютерного зрения, которая предоставит краткий заголовок для переданного в нее изображения.

Прежде чем приступить к экспериментированию с API компьютерного зрения в среде разработки Postman, ознакомьтесь с документацией по API, чтобы узнать форматы запроса и отклика. Microsoft предлагает хорошо написанную документацию по API для когнитивных служб (Cognitive Services), адрес которой – <https://docs.microsoft.com/ru-ru/azure/#pivot=products&panel=cognitive>. Откройте раздел Документация, щелкните мышью на ссылке «Справочник API», который соответствует разделу API компьютерного зрения. Сейчас нас интересует функция описания изображения.



Как это сделать

Для проверки API компьютерного зрения запустите среду разработки Postman. Откроется вкладка данного интерфейса. В качестве метода HTTP выберите POST и введите в адресную строку URL-адрес:

<https://westeurope.api.cognitive.microsoft.com/vision/v1.0/describe>

Этот URL-адрес указан в документации по API.

Далее введите в заголовке запроса свой ключ подписки. Для этого откройте вкладку **Headers** (Заголовки), расположенную ниже адресной строки, и введите свой ключ подписки в следующем виде:

Ключ: Ocp-Apim-Subscription-Key

Значение: Ваш ключ подписки

¹ Протестировать работу API компьютерного зрения можно непосредственно на веб-сайте в консоли когнитивных служб по адресу: <https://westeurope.dev.cognitive.microsoft.com/docs/services/56f91f2d778daf23d8ec6739/operations/56f91f2e778daf14a499e1fe/console>. – Прим. перев.

Вы НЕ сможете воспользоваться этим API, если у вас нет действующего ключа подписки.

Теперь следует выбрать изображение, которое вы хотите отправить в API. Для этого выполните следующие действия. Щелкните мышью на ярлыке вкладки **Body** (Тело), которая расположена правее ярлыка **Header** (Заголовок).

Функция `DescribeImage` API компьютерного зрения поддерживает метод ввода изображения в двух форматах: URL-адрес потокового изображения (`application/json`) и бинарного изображения (`multipart/form-data`). Воспользуемся вторым вариантом.

Убедитесь, что на вкладке Body переключатель установлен в положение «`form-data`». Нажмите кнопку **Choose Files** (Выбрать файлы) и выберите файл изображения, который вы желаете проанализировать. Обратите внимание: обрабатываемое изображение должно быть сохранено в формате (`jpg/png`).

Рисунок 2.6 показывает, как должно выглядеть окно программы Postman, после того как файл с изображением выбран.

The screenshot shows the Postman interface with the following details:

- Request URL:** https://westeurope.api.cognitive.microsoft.com/vision/v1.0/describe
- Method:** POST
- Headers:** (1)
- Body:** (selected)
- File Attachment:** A file named "puppy.png" is attached under the "image" key.
- Send Button:** The "Send" button is visible at the top right.

Рис. 2.6. Интерфейс Postman с заданными параметрами запроса

Теперь, когда все настройки выполнены, передадим изображение API. Для отправки данных нажмите кнопку **Send** (Отправить). Эта кнопка находится за адресной строкой в правой части окна программы. Postman выполнит HTTP POST-запрос к функции `DescribeImage` API компьютерного зрения. В этот запрос войдут ваш ключ подписки и выбранное вами необработанное изображение (рис. 2.7). На получение ответа из API может потребоваться несколько секунд. Ответ в формате JSON будет показан в разделе **Response** (Ответ).





Рис. 2.7 ❖ Изображение, которое мы использовали во время тестирования API

Ответ в формате JSON, полученный для отправленного на анализ изображения, выглядит так:

```
{  
    "description": {  
        "tags": [  
            "dog",  
            "sitting",  
            "white",  
            "laying",  
            "wooden",  
            "top",  
            "small",  
            "table",  
            "large",  
            "black",  
            "water"  
        ],  
        "captions": [  
            {  
                "text": "a dog sitting on a white surface",  
                "confidence": 0.7865828590965703  
            }  
        ]  
    },  
    "requestId": "a8c3ac46-baff-4aed-a857-a79cc9bde4d0",  
    "metadata": {  
        "height": 576,  
        "width": 863,  
        "format": "Png"  
    }  
}
```

От API мы получили следующий текстовый заголовок: «a dog sitting on a white surface» (собака сидит на белой поверхности). Анализ изображения был проведен хорошо! API успешно прошел проверку. Теперь, когда у вас появилось хорошее понимание форматов запроса и ответа этого API, пора приступить к созданию своего первого приложения с элементами искусственного интеллекта.

Создание первого ИИ-ориентированного приложения

Пора начать работу с программой Visual Studio (рис. 2.8). Запустите программу Visual Studio и создайте новый проект. Для этого выберите команду меню **Файл > Создать > Проект**. Появится диалоговое окно **Новый проект**. Выберите шаблон **Консольное** (Пустое) приложение категории **Visual C# > Универсальные приложения Windows**. Введите название своего нового проекта, например `CognitiveServicesVisionApp`, и нажмите на кнопке **OK**. Новый проект создан.

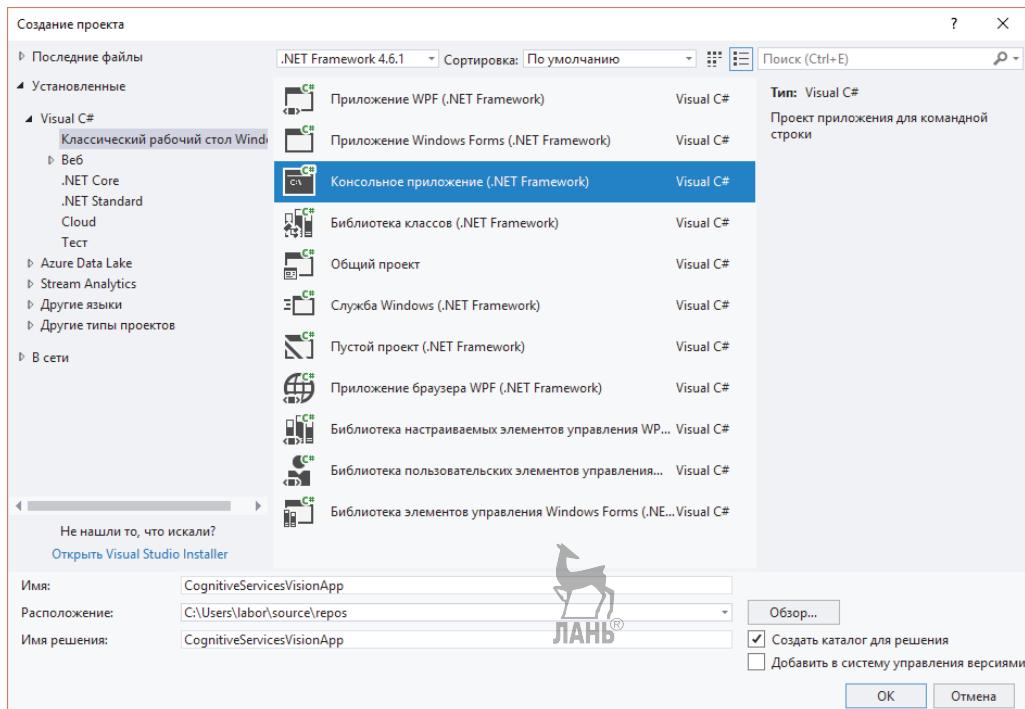


Рис. 2.8 ♦ Создание нового консольного приложения в Visual Studio 2017

i Консольное приложение – это приложение, которое полностью выполняется в консоли (в окне командной оболочки). У него нет графической оболочки, как у настольного либо веб-приложения. Это самое простое приложение, которое можно создать с использованием платформы .NET.

По умолчанию Visual Studio создает новое консольное приложение только с одним файлом: `Program.cs`. Этот файл содержит класс с точкой входа в виде главного метода `Main`. Весь ваш исполняемый код должен быть расположен в методе `Main`,

потому что именно этот метод вызывается компилятором C# во время запуска вашего приложения.

Visual Studio создает новый проект консольного приложения вместе с файлами и папками, создаваемыми по умолчанию. Когда VS завершит создание нового приложения, справа, в панели Обозревателя решений, дважды щелкните на файле Program.cs, чтобы открыть его. Если Обозреватель решений не открыт, выберите команду меню **Вид ► Обозреватель решений**, либо нажмите **Ctrl+Alt+L**, чтобы вывести Обозреватель решений на экран.

На данном этапе ваш файл Program.cs будет иметь вид заготовки с одним классом, Program, и одним методом, static void Main.

Исходный код

Отредактируйте файл Program.cs согласно листингу 2.1.

Листинг 2.1 ❖ Модифицированный файл Program.cs

```
using System;
using System.Configuration;
using System.Net.Http;
using System.Threading.Tasks;
using System.Web;

namespace CognitiveServicesVisionApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Task<string> resultDescribe = DescribeImage(@"C:\Users\labor\Downloads\photo.png");
            Task<string> resultText = ExtractText(@"C:\Users\labor\Downloads\another-photo.png ");
            Console.WriteLine(resultDescribe.Result);
            Console.ReadKey();
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine(resultText.Result);
            Console.ReadKey();
        }

        public static async Task<string> DescribeImage(string imagePath)
        {
            using (HttpClient hc = new HttpClient())
            {
                hc.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
                    ConfigurationManager.AppSettings["AzureSubscriptionKeyVision"]);

                using (MultipartFormDataContent reqContent = new MultipartFormDataContent())
                {
                    var queryString = HttpUtility.ParseQueryString(string.Empty);
                    queryString["maxCandidates"] = "1";
                    var uri = "https://westeurope.api.cognitive.microsoft.com/vision/v1.0/
describe?" + queryString;
                    hc.PostAsync(uri, reqContent);
                }
            }
        }
    }
}
```



```
        var imgContent = new ByteArrayContent(System.IO.File.  
        ReadAllBytes(imageFilePath));  
        reqContent.Add(imgContent);  
  
        HttpResponseMessage resp = await hc.PostAsync(uri, reqContent);  
        string respJson = await resp.Content.ReadAsStringAsync();  
        return respJson;  
    }  
    catch (System.IO.FileNotFoundException ex)  
    {  
        return "Указан неверный путь к файлу изображения.";  
    }  
    catch (ArgumentException ex)  
    {  
        return "Объект HTTP-запроса, похоже, не правильно сформирован.";  
    }  
}
```



Чтобы этот исходный код работал, вам необходимо добавить ссылки на приведенные ниже сборки:

- System.Web;
 - System.Configuration;
 - System.Net.Http.

Для этого щелкните правой кнопкой мыши на строке **Ссылки** в Обозревателе решений и выберите команду **Добавить ссылку**. Выберите в появившемся диалоговом окне три сборки (System.Web: System.Configuration и System.Net.Help). Эти сборки упоминались ранее (рис. 2.9).

Давайте детально разберем этот достаточно большой фрагмент программного кода и проанализируем его построчно, чтобы понять, что он делает.

Пошаговый анализ

Как закаленный программист вы сразу же заметите, что сердцевина исходного кода находится внутри метода `DescribeImage()`. Вначале программный код в `DescribeImage()` вызывает API компьютерного зрения, отправляя ему изображение, и возвращает короткое (хорошо проанализированное) текстовое описание изображения.

Шаги выполнения этой задачи вместе с построчным анализом приведены ниже:

- создать объект HTTP-клиента, чтобы иметь возможность делать вызовы API;
 - сконструировать объект HTTP-запроса с требуемым заголовком (ключом подписки) и параметрами запроса, а также URL-адресом API;
 - прочитать файл изображения как составной тип содержимого формы (multipart/form-data) и добавить его в объект HTTP-запроса;
 - вызвать API машинного зрения, передав его в ваш объект HTTP-запроса;
 - показать HTTP-ответ из API в виде строки с данными.

```
public static async Task<string> DescribeImage(string imagePath)
```

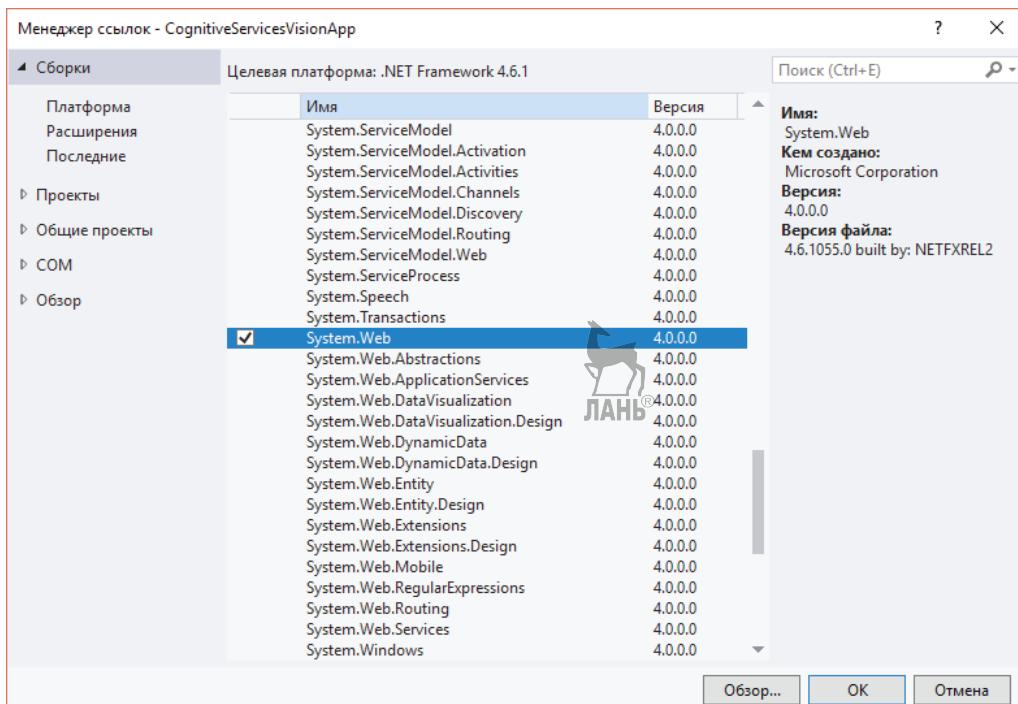


Рис. 2.9 ❖ Добавление ссылок в ваше приложение в Visual Studio 2017

Все начинается с создания метода, который будет использован для выполнения задания. Выбранный метод следует сделать статическим, чтобы его можно было вызывать из статического метода Main. Путь к файлу целевого изображения задается в виде текстовой строки. Ответ, получаемый из API, – это текстовая строка, в которой отображаются данные анализа. При вызове API задействуется асинхронное программирование C#. За плавность работы программы отвечают ключевые слова «async» и «Task», которые будут подробно рассмотрены чуть позже.

Оптимальная практика. Слова `async` и `await` используются всегда, когда это возможно. Цель использования – обеспечение неблокирующего, без зависаний, взаимодействия пользователя с программой.

Плохая практика. Выполнять синхронные HTTP-вызовы с неправильным использованием `async` и `await`.

```
using (HttpClient hc = new HttpClient())
```

Далее идет экземпляр класса `HttpClient` для выполнения HTTP-вызовов методами `get`, `post`, `put` и `delete`. Создание экземпляра в блоке `using` автоматически высвобождает в конце блока выделенную под этот объект оперативную память. Высвобождение объектов, интенсивно использующих средства ввода-вывода, имеет важное значение, потому что используемые ими зарезервированные системные ресурсы немедленно становятся доступными для других программ. Альтернативой применению блока `using` является вызов метода `dispose()` класса `HttpClient` в конце блока инструкций.

Оптимальная практика. Всегда вызывать методы `dispose()` или `close()` объектов, которые реализуют интерфейс `IDisposable`.

Плохая практика. Невнимательное отношение к ресурсоемким объектам может привести к потерям памяти и рассогласованию с другими системными/пользовательскими программами, которые зависят от зарезервированных вашей программой ресурсов.

```
hc.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
ConfigurationManager.AppSettings["AzureSubscriptionKeyVision"]);
```

HTTP-запрос состоит из двух компонентов:

- заголовок;
- содержимое.

В заголовке запроса `HttpClient` вы указываете свой ключ подписки API, как упомянуто в документации.

Оптимальная практика. Хранить изменяемые данные в конфигурационном файле и считывать их оттуда. Подписка со временем подвержена изменению. Ключ подписки лучше всего хранить в `Web.config`.

Плохая практика. Использовать свой ключ как символьное значение непосредственно в исходном коде. В случае изменения ключа вам придется перекомпилировать свой исходный код и, возможно, выполнить повторное развертывание приложения, с тем чтобы оно могло использовать новый ключ.

```
using (MultipartFormDataContent reqContent = new MultipartFormDataContent())
```

API компьютерного зрения требует от вас отправки запроса с составным содержимым формы (`multipart/form-data`). Поэтому вам необходимо создать содержимое HTTP-запроса в требуемом формате. К счастью, в .NET имеется класс, предназначенный как раз для этой цели. По своему усмотрению здесь можно использовать блок `using`, тем более что оптимальная практика говорит именно об этом.

```
var queryString = HttpUtility.ParseQueryString(string.Empty);
queryString["maxCandidates"] = "1";
var uri = "https://westeurope.api.cognitive.microsoft.com/vision/v1.0/describe?" + queryString;
```

Далее вы указываете URL-адрес API и его параметры строки запроса. Параметры строки запроса различны для каждого API когнитивных служб и не обязательны. Если их не указать, то API примет для них значения по умолчанию. Но API-ответ с анализом изображения может возвращать несколько вариантов описаний одного и того же изображения с различной точностью опознания. Вы можете ограничиться возвратом максимального уровня точности описания. Для этого укажите значение 1 для параметра `maxCandidates` (данное значение равно 1 по умолчанию, но его можно увеличить, чтобы возвращать большее количество совпадений).

Оптимальная практика. Для создания строки запроса использовать встроенную в .NET вспомогательную функцию `ParseQueryString()`. Эта функция кодирует ваши параметры и их значения в безопасном URL-формате. Например, допустимый URL-адрес не должен содержать пробелы. Специальные знаки должны вводиться только латинскими символами. Эти знаки перед выполнением HTTP-запроса должны быть преобразованы в безопасный URL-формат. Каждый пробел представляется группой символов `%20`. Еще одним хорошим практическим примером является использование `string.Empty` вместо пустого строкового литерала `" "`.

Плохая практика. Указывать строку запроса непосредственно вместе с URL-адресом, как, например, `var img = "https://westeurope.api.cognitive.microsoft.com/vision/v1.0/describe?maxCandidates=1"`. В таком случае это может непреднамеренно привести к указанию недействительного URL-адреса, который не будет работать, как ожидалось.

```
var imgContent = new ByteArrayContent(System.IO.File.ReadAllBytes(imageFilePath));
reqContent.Add(imgContent);
```



Ваш HTTP-запрос сформирован, за исключением одной очень важной детали: в нем нет самого изображения. В API можно передавать изображения как из Интернета (URL-адрес этого изображения), так и изображения, хранящиеся на жестком диске вашего компьютера. В данном примере был выбран второй способ. Этот способ предусматривает отправку содержимого изображения вместе с HTTP-запросом в виде последовательности байтов. Для метода `File.ReadAllBytes()` следует указать абсолютный путь, по которому сохранен требуемый файл изображения, чтобы передать этот файл как последовательность байтов. `ByteArrayContent` используется, чтобы преобразовать байтовую последовательность в безопасный HTTP-формат и добавить его в объект с составным содержимым формы (`multipart/form-data`), который вы создали ранее.

Оптимальная практика. Функции ввода/вывода обычно выполняют «рискованные» операции, и поэтому они часто могут вызывать исключения. Хороший разработчик всегда преобразует такой опасный программный код в блок `try` и выявит максимальное количество конкретных исключений.

Плохая практика. Не использовать конструкцию `try-catch` для обработки исключений или не отслеживать конкретные исключения при помощи базового класса `Exception`.

```
 HttpResponseMessage resp = await hc.PostAsync(uri, reqContent);
string respJson = await resp.Content.ReadAsStringAsync();
return respJson;
```

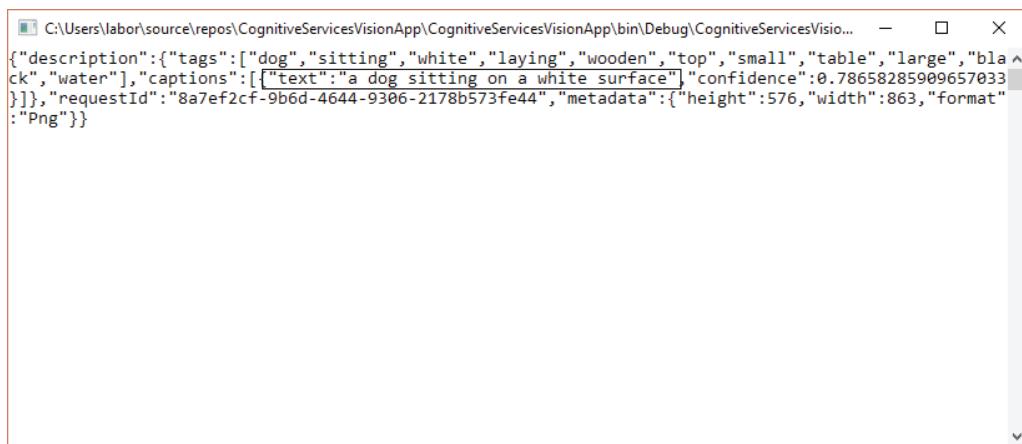
Наконец, вы выполняете HTTP POST-запрос к API, отправляя туда правильно сконструированный запрос. Этот запрос представляет собой асинхронную операцию, то есть программа не прервет своего исполнения, чтобы дождаться получения ответа из API, перед тем как перейти к следующей программной инструкции. Блокирующие операции, в особенности связанные с работой сети, в C# должны выполняться асинхронно, чтобы избежать «зависания» интерфейса приложения. Хотя с точки зрения пользователя это представляется логичным, разработчик может столкнуться с проблемой, когда программный код, следующий непосредственно за асинхронной операцией, зависит от результата операции. В этом случае в работу включается ключевое слово `await`. Ключевое слово `await` запоминает функцию обратного вызова, исполняющую программный код последующей операции, которая должна идти после успешного получения результата. При этом может создаться впечатление, как будто программа на самом деле сделала паузу в исполнении последующих команд, ожидая окончания асинхронной операции. Все асинхронные операции должны «ждать ответа из API». А метод, который имеет операции, требующие ожидания, должен быть помечен ключевым словом `async`, точно так, как было сделано в сигнатуре своего метода `DescribeImage()`. Тип воз-

возвращаемого значения для таких методов должен быть `void` либо `Task<T>` (где `T` – это тип возвращаемого результата асинхронной операции).

Результат

Чтобы запустить вашу программу в Virtual Studio на исполнение, достаточно нажать клавишу `F5` или сочетание клавиш `Ctrl+F5`. Программа как часть консольного приложения откроется в командном окне оболочки и будет выполнена.

Результат анализа изображения собаки, которое мы использовали в примере, показан на рис. 2.10.



```
C:\Users\labor\source\repos\CognitiveServicesVisionApp\CognitiveServicesVisionApp\bin\Debug\CognitiveServicesVisio...
{"description":{"tags":["dog","sitting","white","laying","wooden","top","small","table","large","blac
ck","water"]}, "captions": [{"text":"a dog sitting on a white surface"}, "confidence":0.78658285909657033
]}, {"requestId":"8a7ef2cf-9b6d-4644-9306-2178b573fe44"}, "metadata":{"height":576,"width":863,"format"
:"Png"}}
```

Рис. 2.10 ♦ Выделенный рамкой текст – это заголовок с результатом работы приложения, полученный из API

БОЛЕЕ ИНТЕРЕСНЫЙ ВАРИАНТ ПРИЛОЖЕНИЯ

Итак, вы создали свое первое приложение на основе элементов ИИ, используя метод `DescribeImage` API компьютерного зрения. Но в API компьютерного зрения предусмотрены еще шесть методов анализа изображения, позволяющие провести более глубокий анализ. Чтобы воспользоваться одним из семи этих методов анализа, следует использовать существующий ключ подписки.

Извлечение текста из изображений

Одним из семи методов API компьютерного зрения является метод OCR. *Оптическое распознавание символов*, или сокращенно OPC (OCR), – это метод обнаружения и извлечения видимого текста из изображений, очень широко использующийся. Мы приведем только несколько примеров применения этого метода на практике. А вы сможете придумать еще как минимум десяток других:

- извлечение текста из отсканированных, письменных или распечатанных документов;
- перевод текста с указателей на иностранном языке;
- автоматическое считывание номерных знаков транспортных средств;
- преобразование рукописного текста в режиме реального времени.

Хотя OPC не является новой технологией, в которой Microsoft не был первопроходцем, предлагаемый когнитивными службами метод оптического распознавания изображений способен не только извлекать как печатный, так и рукописный текст, но и автоматически распознавать язык извлеченного текста.

Исходный код

Воспользуйтесь своим консольным приложением и добавьте в файл Program.cs немного программного кода, чтобы получить новый метод под названием ExtractText(). По сути, это тот же метод, что и DescribeImage(), и отличается только URL-адресом API и параметрами строки запроса.

Редактируйте файл Program.cs, так чтобы он выглядел, как показано в листинге 2.2.

Листинг 2.2 ❖ Модифицированный файл Program.cs

```
using System;
using System.Configuration;
using System.Net.Http;
using System.Threading.Tasks;
using System.Web;

namespace CognitiveServicesVisionApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Task<string> resultDescribe = DescribeImage(@"C:\Users\labor\Downloads\photo.png");
            Task<string> resultText = ExtractText(@"C:\Users\labor\Downloads\another-photo.png");
            Console.WriteLine(resultDescribe.Result);
            Console.ReadKey();
            Console.WriteLine(Environment.NewLine);
            Console.WriteLine(resultText.Result);
            Console.ReadKey();
        }

        public static async Task<string> DescribeImage(string imagePath)
        {
            using (HttpClient hc = new HttpClient())
            {
                hc.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
                    ConfigurationManager.AppSettings["AzureSubscriptionKeyVision"]);

                using (MultipartFormDataContent reqContent = new MultipartFormDataContent())
                {
                    var queryString = HttpUtility.ParseQueryString(string.Empty);
                    queryString["maxCandidates"] = "1";
                    var uri = "https://westeurope.api.cognitive.microsoft.com/vision/v1.0/
describe?" + queryString;

                    try
                    {
```

```
        var imgContent = new ByteArrayContent(System.IO.File.  
ReadAllBytes(imageFilePath));  
        reqContent.Add(imgContent);  
  
        HttpResponseMessage resp = await hc.PostAsync(uri, reqContent);  
        string respJson = await resp.Content.ReadAsStringAsync();  
        return respJson;  
    }  
    catch (System.IO.FileNotFoundException ex)  
    {  
        return "Указан неверный путь к файлу изображения.";  
    }  
    catch (ArgumentException ex)  
    {  
        return "Объект HTTP-запроса, похоже, не правильно сформирован.";  
    }  
}  
}  
}  
  
public static async Task<string> ExtractText(string imagePath)  
{  
    using (HttpClient hc = new HttpClient())  
    {  
        hc.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",  
ConfigurationManager.AppSettings["AzureSubscriptionKeyVision"]);  
  
        using (MultipartFormDataContent reqContent = new MultipartFormDataContent())  
        {  
            var uri = "https://westeurope.api.cognitive.microsoft.com/vision/v1.0/ocr";  
  
            try  
            {  
                var imgContent = new ByteArrayContent(System.IO.File.  
ReadAllBytes(imagePath));  
                reqContent.Add(imgContent);  
  
                HttpResponseMessage resp = await hc.PostAsync(uri, reqContent);  
                string respJson = await resp.Content.ReadAsStringAsync();  
                return respJson;  
            }  
            catch (System.IO.FileNotFoundException ex)  
            {  
                return "Указан неверный путь к файлу изображения.";  
            }  
            catch (ArgumentException ex)  
            {  
                return "Объект HTTP-запроса, похоже, не правильно сформирован.";  
            }  
        }  
    }  
}
```

Пошаговый анализ

Полный пошаговый анализ этого нового исходного кода, с изображением, показанным на рис. 2.11, не требуется.

Вы создали новый метод под названием ExtractText() в том же самом классе, где расположен метод DescribeImage(). В методе ExtractImage абсолютный путь к файлу изображения, которое вы хотите проанализировать, задается в виде текстовой строки. Текст, полученный из API в формате JSON, вы увидите в консоли.

```
Task<string> resultText = ExtractText(@"C:\Users\labor\Downloads\another-photo.png");
Console.WriteLine(resultText.Result);
```



Рис. 2.11 ♦ Изображение, использованное для тестирования API OCR

Результат

Итоговые данные, которые мы получили в консоли после выполнения обновленного приложения, приведены на рис. 2.12 (обнаруженные фрагменты текста выделены квадратными скобками).

```
C:\Users\labor\source\repos\CognitiveServicesVisionApp\CognitiveServicesVisionApp\bin\Debug\CognitiveServicesVisio...
{"description": {"tags": ["dog", "sitting", "white", "laying", "wooden", "top", "small", "table", "large", "bla ^ ck", "water"], "captions": [{"text": "a dog sitting on a white surface", "confidence": 0.78658285909657033 }]}, "requestId": "8a7ef2cf-9b6d-4644-9306-2178b573fe44", "metadata": {"height": 576, "width": 863, "format": "Png"}}

{"language": "en", "orientation": "Up", "textAngle": 0.0, "regions": [{"boundingBox": "189,220,446,138", "lines": [{"boundingBox": "349,220,127,43", "words": [{"boundingBox": "349,220,127,43", "text": "Next"}]}], {"boundingBox": "189,282,446,76", "words": [{"boundingBox": "189,282,140,76", "text": "BIG"}, {"boundingBox": "357,282,278,76", "text": "THING"}]}]}}



```

Рис. 2.12 ♦ Выделенные квадратными скобками фрагменты представляют текст, обнаруженный при помощи API

Разве это не чудо? API компьютерного зрения обратило внимание даже на буквы верхнего и нижнего регистров. И эти невероятные возможности ИИ теперь находятся в ваших руках.



Резюме

В этой главе вы узнали обо всех инструментах, необходимых для создания своего первого приложения с элементами искусственного интеллекта. Было показано, что большинство ваших задач можно решить с помощью интегрированной среды разработки Visual Studio. Вы также узнали, как получить ключ подписки для облачной инфраструктуры Azure, которая используется прикладными программными интерфейсами когнитивных служб (Cognitive Services API). После подготовки своей среды разработки вы в пошаговом режиме проследили процесс создания приложения с элементами искусственного интеллекта, которое способно предложить краткое логическое текстовое описание изображений, отсылаемых на анализ в API. По ходу вы познакомились с оптимальными и плохими практическими приемами создания всех главных частей исходного кода. И наконец, вы успешно расширили возможности своего первого приложения с элементами искусственного интеллекта. После этой модернизации приложение смогло извлечь из изображения текст.

Подводя итоги, в этой главе вы узнали:

- как подготовить Visual Studio 2017 в качестве своей среды разработки;
- как получить свой собственный ключ подписки Azure, который является самым важным предварительным требованием для использования когнитивных служб;
- как использовать программу Postman для тестового прогона метода DescribeImage API компьютерного зрения, чтобы получить лучшее представление о его форматах запроса и отклика;
- как создать свое первое консольное приложение с элементами ИИ в среде Visual Studio;
- как расширить возможности своего приложения;
- познакомились с основными элементами исходного кода и рассмотрели оптимальные практические приемы работы.

В **следующей главе** вы узнаете о наметившейся тенденции в конструировании интерфейсов пользователя, именуемой разговорными интерфейсами пользователя, которая позволяет взаимодействовать с компьютером посредством человекоподобных диалогов, а не чисто графических либо текстовых взаимодействий.

Глава 3



Конструирование разговорного интерфейса с применением технологий **Microsoft**

Если вы вернетесь назад, к главе 1, то на первых страницах вы увидите, как выглядит разговор с интеллектуальным компьютером о проходящем футбольном матче. Давайте продвинемся на шаг вперед. Наверное, неплохо было бы заказывать еду, наблюдая за тем, как играет ваша любимая команда.

Вспомните, как вы заказывали еду онлайн в прошлый раз. Если вы знали, какое блюдо вы хотите сейчас заказать, то вы, возможно, просто позвонили в ресторан. В противном случае вы заходили на веб-сайт ресторана, просмотрели меню, выбрали блюда, указали количество и только после этого сделали заказ. Весь процесс, вероятно, занял минут 5–10 в зависимости от заказа.

Теперь представим размещение этого заказа при помощи одного простого сообщения по WhatsApp (рис. 3.1). Никаких 20 нажатий на клавиши, чтобы разместить один заказ. Никаких разговоров с обычно раздраженными продавцами, которые, похоже, всегда спешат.



Рис. 3.1 ♦ Размещение заказа по WhatsApp

И почему именно WhatsApp? Facebook Messenger, Slack, Google Talk и даже старый добрый SMS также сработали бы просто великолепно.

Одно простое сообщение на вашей любимой платформе обмена сообщениями – и вуаля! Ваш заказ подтвержден. Это проще и намного экономичнее по времени. Если необходимы разъяснения, то продавец может задать вам вопрос или два, как показано на рис. 3.2.

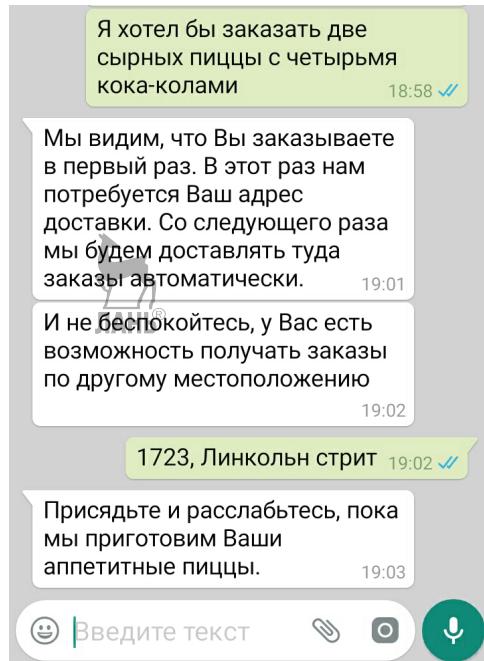


Рис. 3.2 ❖ Система заказа пиццы на основе чата может запросить у пользователя больше подробностей о заказе

Использование простых текстоориентированных диалогов для взаимодействия с продуктами или услугами компании – вот в этом и состоит смысл *разговорного интерфейса пользователя* (РИП, conversational user interface, CUI). В разговорном интерфейсе пользователя традиционный графический интерфейс отсутствует. Нет никаких страниц, которые нужно просматривать. Нет никаких полей ввода, которые требуется заполнять, или кнопок, которые следует нажимать. Просто текст! Это все равно, что беседовать с другом в неофициальном тоне. Идеальный разговорный интерфейс предоставит вам простую возможность взаимодействия с собеседником, потому что формальных правил общения нет.

Прочитав эту главу, вы познакомитесь с разговорным интерфейсом пользователя. В частности, вы:

- узнаете, что такое разговорный интерфейс пользователя;
- познакомитесь с краткой историей РИП;
- сравняте возможности разговорного интерфейса с возможностями графического интерфейса (или РИП против ГИП): на что разговорный интерфейс способен и что он сделать не сможет;
- познакомитесь с реально существующими примерами;
- познакомитесь с принципами разработки;

- создадите приложение с разговорным интерфейсом, используя инфраструктуру для создания программ-роботов, Microsoft Bot Framework;
- разберетесь в исходном коде;
- зайдетесь отладкой и тестированием своего приложения с разговорным интерфейсом.

ЧТО ТАКОЕ РАЗГОВОРНЫЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ?

В приведенном ранее примере вы увидели, как выглядит взаимодействие пользователя с разговорным интерфейсом. Но это только видимая часть интерфейса, иначе ему не была посвящена целая глава. Итак, начнем.

Кто в примере с заказом пищи является собеседником на другом конце? Обычно это может быть не только живой человек, но и какая-то офигенная автоматизированная система. Но живой человек не сможет обслужить сразу несколько десятков человек за одну минуту. Он просто обожжет свои пальцы, набирая текст со скоростью света. К тому же существует большая вероятность, что он перепутает заказы. Или, что еще хуже, вообще пропустит заказ.

Такая автоматизированная компьютерная система обычно называется **программой-роботом**, или просто **ботом**. И скорее всего, вам уже доводилось общаться с таким количеством ботов, что вы сами себе не представляете. Это и аккаунт в Twitter с поддержкой продукта, и чат с мгновенными сообщениями, касающимися запросов о распродажах, Алекса в Amazon Echo, Кортана в Windows и Сири в iOS.

Чтобы взаимодействовать с ботом, не нужны какие-то специальные графические элементы управления. Например, кнопки, которые нажимаются с помощью мыши. Потребуется только платформа обмена сообщениями. Этой платформой может стать WhatsApp, Twitter, Facebook или специальная веб-страница, разработанная под конкретный сервис. Главное, что для разговора с ботом и для достижения цели требуется только клавиатура. Как назвать такой интерфейс, где все взаимодействия выполняются при помощи текста? Назвать его графическим интерфейсом пользователя, или ГИП, было бы неправильно. Поэтому один мудрый человек назвал его в честь единственной видимой составляющей данного взаимодействия: разговора. Отсюда и название «*разговорный интерфейс пользователя*».

КРАТКАЯ ИСТОРИЯ

Историю возникновения разговорного интерфейса пользователя невозможно понять, не познакомившись с историей интерфейса пользователя как такого. Так что же такое интерфейс пользователя?

Интерфейс пользователя – это окно для взаимодействия компьютера и пользователя. То есть с помощью этого интерфейса пользователь указывает компьютеру, что он должен сделать, а компьютер показывает результаты своих действий. Другими словами, интерфейс пользователя позволяет упростить взаимодействие человека и машины. Без этого интерфейса у вас не будет другой возможности контроля над компьютером, кроме как непосредственно возиться с его аппаратными средствами, что очень долго и утомительно и не принесет вам никакой пользы.

В самом начале: интерфейс командной строки

В самых ранних компьютерах в качестве единственного интерфейса пользователя применялась командная строка. И назывался он интерфейсом командной строки, или интерфейсом командной оболочки (Command line interface, CLI). Пользователи вводили команду в виде текстовой строки. А интерфейс представлял собой черный экран, на котором отображались как строки введенных команд, так и строки с ответами на эти команды. То есть такой интерфейс графическим назвать было нельзя, так как в нем не было ни одного графического элемента. Его можно было назвать текстовым интерфейсом, в котором компьютер выполнял введенную команду и выводил на экран результаты. На рис. 3.3 вы видите результат команды просмотра выполняющихся процессов, аналогичных тем, которые мы видим в Диспетчере задач в Windows.

```
Processes: 210 total, 2 running, 9 stuck, 199 sleeping, 901 threads 23:30:03
Load Avg: 1.40, 1.75, 1.00 CPU usage: 4.15% user, 4.40% sys, 91.44% idle
SharedLibs: 1648K resident, 0B data, 0B linkedt.
MemRegions: 31278 total, 1892M resident, 117M private, 564M shared.
PhysMem: 5893M used (1191M wired), 10G unused.
VM: 523G vszsize, 1026M framework vszsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 12105/8925K in, 11907/1964K out.
Disks: 80156/2205M read, 21235/425M written.
```

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPR	PGRP	PPID
592	screenCaptur	0.0	00:00.02	7	5	55+	1952K+ 20K+	0B	0B	262	262
590	mdworker	0.0	00:00.01	3	0	44	2032K	0B	0B	590	1
589	mdworker	0.0	00:00.01	3	0	44	1572K	0B	0B	589	1
588	top	1.7	00:00.51	1/1	0	22+	2860K	0B	0B	588	584
584	bash	0.0	00:00.00	1	0	15	588K	0B	0B	584	583
583	login	0.0	00:00.01	3	1	28	1228K	0B	0B	583	482
574	auditd	0.0	00:00.00	2	0	25	560K	0B	0B	574	1
567	System Prefe	0.0	00:03.23	3	0	270	39M	8364K	0B	567	1
561	systemstatsd	0.0	00:00.01	2	1	19	1040K	0B	0B	561	1
560	com.apple.We	0.0	00:01.42	9	0	229	25M	0B	0B	560	1
558	com.apple.We	0.0	00:05.07	15	3	224	151M	1716K	0B	558	1
555	bash	0.0	00:00.00	1	0	15	604K	0B	0B	555	554
554	login	0.0	00:00.01	3	1	28	1176K	0B	0B	554	482
550	bash	0.0	00:00.00	1	0	15	608K	0B	0B	550	549

Рис. 3.3 ♦ Подобного рода интерфейсы были распространены в 1970-х и 1980-х годах. Они не были ограничены только стереотипными «хакерскими» группами, а использовались всеми пользователями компьютеров одинаково

Интерфейс командной строки предлагал разные команды для выполнения на компьютере различных операций. Хотя основные операции между различными компьютерами и операционными системами оставались прежними, сами команды различались. Операционная система MS-DOS была прямым предшественником всех версий Microsoft Windows. Она имела характерный интерфейс командной строки, который позволял пользователям управлять компьютером. Вот некоторые самые популярные команды MS-DOS:

- dir: вывести перечень содержимого каталога;
- cd: изменить текущий рабочий каталог;
- md: создать новый каталог;
- copy: скопировать файл в указанное место назначения;
- at: наметить время для исполнения команды или программы.

Каждая команда имела свой набор вариантов для тонкой настройки ее поведения. Например, команда dir выводила на экран имена файлов и папок вместе

с датой и временем последнего изменения по каждому из них (рис. 3.4). Команда `dir /B` показывала только имена без даты и времени.

```
C:\inetpub>dir
Volume in drive C is OSDisk
Volume Serial Number is 98C1-B0D3

Directory of C:\inetpub

04/11/2016  01:42 PM    <DIR>
04/11/2016  01:42 PM    <DIR>          .
04/11/2016  01:42 PM    <DIR>          ..
04/12/2017  02:44 PM    <DIR>          custer
04/12/2016  02:05 PM    <DIR>          history
04/12/2016  01:43 PM    <DIR>          logs
04/11/2016  01:42 PM    <DIR>          temp
01/30/2017  10:36 AM    <DIR>          wwwroot
              0 File(s)           0 bytes
              7 Dir(s)   8,543,842,304 bytes free

C:\inetpub>
```

Рис. 3.4 ♦ Команда директории (`dir`) выводит список всех файлов и папок на экран

Интерфейс командной строки был быстрым, потому что, по сравнению с графическими интерфейсами, требовалось совсем немного системных ресурсов. И его было легче автоматизировать при помощи сценариев.

Хотя он великолепно подходил для опытных пользователей, которые имели полный контроль над машиной, интерфейс командной строки был очень недружелюбен к новичкам. Он сопровождался крутой кривой обучения и требовал месяцы, а то и годы практики, чтобы к нему приспособиться. Он не был интуитивно понятным, учитывая совершенную неочевидность команд для нового пользователя.

Несмотря на его недостатки, интерфейс командной строки по-прежнему предлагается как альтернативный интерфейс пользователя в операционных системах в виде средства, предназначенного для опытных/продвинутых пользователей, которые могут в оперативном режиме производить наладку и выполнять другие операции. Вы, возможно, пользовались одной из форм интерфейса командной строки: командной строкой в Windows или терминалом в macOS и UNIX/Linux.

И затем появился графический интерфейс пользователя

Графический интерфейс пользователя (ГИП) часто обозначается английской аббревиатурой GUI (graphical user interface), которая кратко произносится как «гуи». Большинство из нас родилось в эру ГИП. Мы настолько привыкли к использованию кнопок, меню, текстовых полей и двух десятков других графических элементов, из которых ГИП и состоит, что не можем представить взаимодействие с нашими компьютерами каким-то другим способом. Сейчас ГИП воспринимается абсолютно естественно. А знаете ли вы, с чего началась революция ГИП?

Все началось в конце 1960-х годов с новаторской работы Дугласа Энджелбарта по взаимодействию машины и человека. Его исследование привело к развитию

предшественников ГИП, которые, в свою очередь, привели к изобретению мыши. Далее примитивный ГИП Энджелбarta усовершенствовали в научно-исследовательском центре Xerox PARC, в котором был создан компьютер Xerox Alto. Xerox Alto стал первым компьютером с графическим интерфейсом WIMP (windows, icons, menus, pointer), т. е. с окнами, изображениями, меню, указателем, в том виде, в котором мы используем ГИП сегодня (рис. 3.5).

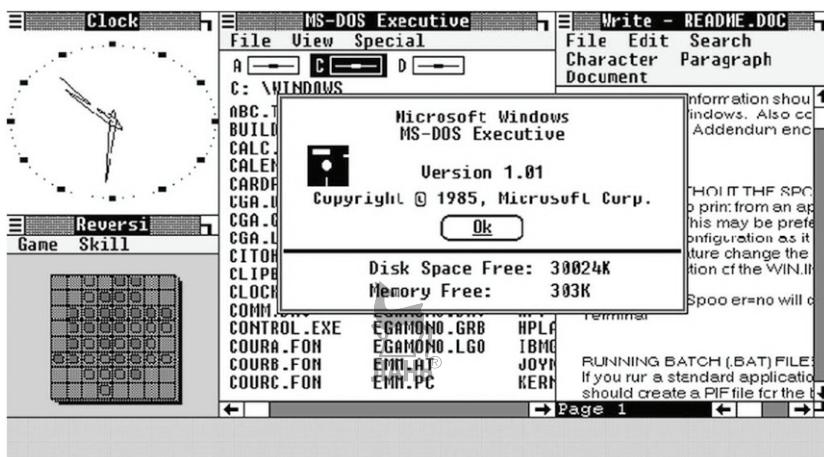


Рис. 3.5 ♦ Снимок экрана операционной системы Windows 1.01 с одним из самых ранних коммерческих успешных ГИП'ов. На этом снимке вы видите часы, игру, файловый менеджер и редактор документов

i Хотя компания Xerox практически изобрела ГИП в том виде, в котором мы его знаем сегодня, на рынке она практически не была представлена. В компании Xerox не спешили оценить значение технологии, которая была разработана в ее научно-исследовательском центре PARC. В результате компании не удалось свою разработку правильно и вовремя коммерциализировать. Компания Xerox все же сделала коммерческий вариант компьютера Alto под названием Xerox Star, но не смогла его усовершенствовать из-за предельно высокой цены на этот компьютер и отсутствия надлежащей рекламы. В компании Xerox поняли свою грубую ошибку только после того, как компьютер Macintosh компании Apple коренным образом изменил рынок персональных компьютеров, но к тому времени было уже слишком поздно.

Графические интерфейсы пользователя благодаря своей простоте полностью изменили картину использования персональных компьютеров. Впервые люди, которые не были исследователями, учеными или энтузиастами, смогли начать пользоваться компьютерами. Они могли выполнять команды, не набирая их на клавиатуре, а нажимая выведенные на экран кнопки. Пользователи могли «видеть» свои файлы и папки, играть в игры и делать много чего еще. Компании Apple и Microsoft воспользовались неудачей Xerox и смогли коренным образом модернизировать свои персональные компьютеры. Apple Macintosh создал свой первый дешевый коммерческий ПК с операционной системой на основе ГИП. А благодаря тому, что операционная система Windows не привязывалась к каким-то конкретным аппаратным средствам или конкретному компьютеру и могла устанавливаться

ваться на компьютеры не только от фирмы IBM, но и на машины от других производителей, компания Microsoft переманила на свою сторону большинство как корпоративных, так и индивидуальных пользователей.

Со времени своего появления графические интерфейсы проделали длинный путь. Во всех программных продуктах, от операционных систем до веб-приложений и приложений для мобильных устройств, дизайн был сосредоточен на простоте, изяществе и функциональности. Внешний вид ГИП эволюционировал. В период от 1980-х до 1990-х годов ГИП был очень простым. Со временем ГИП усовершенствовался, и в 2000-х внешний вид графического интерфейса пользователя стал даже изысканным. В 2010-х годах, с появлением мобильных устройств ГИП стал плоским и функциональным.

Революция, вызванная смартфонами, способствовала эволюции ГИП в той же степени, что и ПК вначале. Хотя ранние смартфоны имели операционные системы, управляемые в режиме меню (Nokia S60), наши современные телефоны имеют операционные системы, которые во многом похожи на операционные системы их настольных собратьев. Однако графические интерфейсы для мобильных устройств, размеры экранов которых, по сравнению с мониторами компьютеров, гораздо меньше, пришлось создавать заново. В ГИП для мобильных устройств было необходимо предусмотреть использование экранных сенсоров. Наличие экранных сенсоров произвело на свет новые элементы интерфейса пользователя. Это такие элементы, как скользящее боковое меню, для вызова которого достаточно провести пальцем по экрану, контекстное меню «три точки», переключатель, уведомления, меню «поделиться» и т. д. С нуля были написаны новые языки конструирования интерфейса для всех основных платформ: Material Design (Android), Human Interface (iOS) и Metro Design (Windows). Появившиеся недавно элементы интерфейса приятны для глаз и очень функциональны. По существу, они стали результатом взаимодействия дизайнерских идей ОС для мобильных устройств с ОС для настольных компьютеров.

Очередная эволюция: разговорный интерфейс пользователя

Текущая технологическая революция находится во власти мобильных телефонов. ГИП, переработанный для устройств с маленькими экранами, позволяет своим пользователям выполнять одни и те же действия на мобильном телефоне быстрее, чем на ПК. Но если бы мы могли с нашим мобильным телефоном еще и говорить, точно так же, как мы говорим друг с другом, то добиться желаемой цели можно было бы еще легче. Достижения в области искусственного интеллекта как никогда упростили создание интерфейсов, которые позволяют разговаривать с устройствами, используя как устную, так и письменную (текст) речь.

Существует большая разница между общением с устройством посредством команд и естественным общением, не опирающимся на формальные правила общения. Принцип интерфейса командной строки состоит в общении с устройством с помощью команд. Такой подход приводит к односторонней коммуникации с очень строгими правилами построения команд. В этом случае устройство выглядит как немая машина, которая понимает только определенные ключевые слова. Она настолько строга в отношении этих ключевых слов, что если использовать слово, которого в машинном словаре нет, то она попросту откажется по-

виноваться. Рассмотрим диалог, основанный на командах, построение которых требует строгого выполнения правил:

Пользователь: Хочу заказать пиццу.

Бот: Извините, Ваша команда не понятна.

Пользователь внезапно вспоминает, что эта машина понимает только команды.

Пользователь: Заказ пиццы, 1 большая Маргарита.

Бот: Спасибо, Ваш заказ размещен. Хотите заказать еще? (Y/n)

Пользователь: Нет.

Бот: Извините, Ваша команда не понятна.

Пользователь рвет на себе волосы.



Более естественный разговор прошел бы вот таким образом:

Пользователь: Хочу заказать пиццу.

Бот: Отлично. Какую пиццу Вы бы хотели?

Пользователь: Гм.., я думаю, Маргариту. Стоп, наверно, все-таки выберу Пепперони.

Бот: Так Маргарита или Пепперони?

Пользователь: Пепперони. И сделайте ее большой.

Бот: Хорошо. Хотите заказать гарниры?

Пользователь: Нет, обойдусь без них.

Бот: Замечательно. Ваш заказ на 1 пиццу Пепперони успешно размещен.

Бот: Я сообщу, когда Ваш заказ будет отправлен.

Бот: Есть ли что-либо еще, чем я могу Вам помочь?

Пользователь: Нет, спасибо.

Бот: Спасибо за заказ. Хорошего дня.

Что необычно в приведенном выше разговоре? Позвольте мне вам подсказать:

- этот разговор не строится на строгих правилах общения, которые необходимы для построения команд. Пример таких команд был показан ранее. В результате общение с ботом становится проще и привлекательнее для пользователя;
- пользователь НЕ должен знать заранее, как общаться с программой-роботом; идеальный бот поймет любые отклонения в пользовательском запросе, потому что каждый пользователь, задавая вопросы, может поступать по-своему.



i Стоит отметить, что разговорный интерфейс не ограничивается только текстом. Диалоги на основе устной речи обычно предлагаются большинством разговорных интерфейсов пользователя. На внутреннем уровне речь сначала преобразовывается в текст. Результат запроса представляется пользователю в текстовой форме или, после преобразования при помощи технологии преобразования текста в речь, в виде речи.

Приложения для обмена сообщениями сыграли большую роль в эволюционном переходе от графического интерфейса к разговорному. Социальная сеть Twitter

популяризовала ведение микроблогов (выражение мыслей и мнений в коротких предложениях до 140 знаков). Если вы использовали Twitter или видели твиты по ссылкам на новостном веб-сайте, то наверняка знаете, что в Twitter используется разговорная речь. То есть пользователь может отправлять ответ на сообщение, опубликованное другим пользователем, или участвовать в дискуссии. Эта соцсеть используется не только отдельными людьми для выражения своих мыслей, но и организациями для оказания поддержки своим клиентам.

То же самое произошло и с WhatsApp. Изначально эта система разрабатывалась как приложение для быстрого личного общения. Со временем WhatsApp преобразовался в более крупную платформу обмена сообщениями и стал поддерживать передачу других популярных форматов (аудио, видео, документы, контакты и т. д.). А организации стали использовать, подобно Twitter, WhatsApp, для связи с каждым своим клиентом.

Сегодня многие компании взаимодействуют со своими клиентами исключительно с помощью платформ обмена сообщениями, освобождая себя от дорогих разработок приложений для собственных услуг. Это в особенности справедливо в отношении электронной коммерции. Такие приложения, как WeChat, QQ и Ali-pay, чрезвычайно популярны среди китайских мелких и крупных электронных продавцов. Посмотрите, как это работает:

- продавец использует платформу электронной коммерции, в частности Alibaba, чтобы выставить товарные позиции на продажу;
- потенциальный клиент, желающий приобрести товар, выбирает его в каталоге, размещенном на сайте этой платформы, связывается с продавцом по WeChat и размещает свой заказ;
- позже живой человек связывается с клиентом, уточняет детали заказа и способы оплаты. На этой стадии возможен обмен несколькими сообщениями;
- клиент оплачивает стоимость товара с помощью банковского перевода или через PayPal;
- продавец отправляет выбранный товар по указанному адресу.

В отличие от традиционных веб-сайтов электронной коммерции, на вышеописанных веб-сайтах электронной коммерции нет кнопки **Приобрести** или **Добавить в корзину**. Продавец получает всю сумму, переведенную клиентом за проенный товар, а веб-сайт электронной коммерции зарабатывает за счет сборов за регистрацию товарной позиции на сайте. Но в чем польза для клиента? Для клиента этот тип сделки особенно выгоден при ведении дел с малопопулярными или неизвестными продавцами. Из-за того, что клиент непосредственно общается с продавцом, возрастает уровень доверия между клиентом и продавцом относительно состоявшейся сделки.

Люди настолько привыкли к обмену сообщениями, что для большинства из них легче напечатать короткое сообщение, вместо того чтобы щелкать мышью или нажимать пальцем по кнопкам и заполнять формы. Поэтому компании все чаще обращаются к платформам обмена сообщениями для продажи своих товаров и услуг. Такое общение обладает двумя преимуществами:

- 1) оно удобнее для пользователей и
- 2) позволяет компаниям, в особенности малым и стартапам, экономить на дополнительной стоимости разработки специального приложения.

Надеемся, что теперь нам удалось четко разъяснить преимущества разговорного интерфейса пользователя.

Но есть ли польза от платформы обмена сообщениями для предприятий, которым требуется для ведения переговоров со своими клиентами нанимать живых людей, затраты на оплату труда которых, скорее всего, будут равны или даже превысят стоимость разработки специального приложения для обслуживания клиентов? В этом случае гораздо выгодней воспользоваться программой-роботом с элементами искусственного интеллекта.

Роль ИИ в разговорном интерфейсе пользователя

Технология искусственного интеллекта, в частности технология понимания естественного языка (ЕЯ), помогает в определении смысла предложения, введенного пользователем. Не важно, в какой форме пользователь выразит свое намерение что-то получить или сделать. Хорошая серверная часть службы (бэкэнд) с элементами понимания естественного языка будет в состоянии определить, что хочет пользователь.

Например, вы разрабатываете бот, для которого выражения «Эй», «Привет», «Здорово» и «Как дела» означают одно и то же. Разработчик может обучить серверную часть службы с элементами понимания ЕЯ распознавать все эти выражения как одно и то же намерение поприветствовать HelloGreeting. Поэтому когда пользователь отправит в ваш бот свое сообщение, это сообщение будет перенаправлено на серверную часть службы с элементами понимания ЕЯ. В свою очередь, серверная часть вернет единое слово, выражающее намерение пользователя. Если одно из вышеупомянутых выражений будет обнаружено в сообщении, то оно будет распознано как приветствие HelloGreeting. В логике исходного кода вящего бота можно предусмотреть условие для передачи соответствующего ответа обратно (см. листинг 3.1).

Листинг 3.1 ♦ Псевдокод, в котором используется серверная часть с элементами понимания ЕЯ, которая выделяет намерение пользователя из его сообщения и отправляет соответствующее сообщение с откликом назад пользователю

```
string intent = NLUBackend.getIntent(userMessage);
if (intent == "HelloGreeting")
{
    sendReply("Здравствуйте! Я - пиццерийный бот. Для начала диалога скажите что-нибудь наподобие 'Я хочу заказать пиццу'.");
}
```

Стоит отметить, что методология понимания ЕЯ умнее, чем просто способность возвращать значение выражений, на которые это значение было натренировано. Хорошая система понимания ЕЯ вернет правильное значение даже для тех выражений, которые она никогда прежде не получала (на которые она не была натренирована). Например, на основе вышеупомянутых выражений для HelloGreeting серверная часть с элементами понимания ЕЯ может решить, что «здравствуйте» и «привет, старина» могут означать то же самое, и поэтому для этих новых выражений с высокой степенью уверенности вернет намерение поприветствовать HelloGreeting.

Так, постепенно тренируясь, она будет становиться все умнее и умнее. Чем больше и лучше проводится обучение, тем точнее будет определяться намерение в серверной части с элементами понимания ЕЯ. Вы узнаете более подробные понятия технологии понимания ЕЯ в главе 4. В главе 6 вы получите практический опыт использования среды Microsoft LUIS, в которой реализованы методы понимания ЕЯ.

Безусловно, технология понимания ЕЯ сделала процесс создания умных ботов проще, чем это было ранее. Начиная с отдельных слов, словосочетаний и до полных предложений правильно натренированная серверная часть службы с элементами понимания ЕЯ определит смыслы всех простых и сложных сообщений, поступающих от пользователей. При проектировании бота вам больше не нужно беспокоиться о двух десятках вариантов, которые пользователи могут использовать, чтобы заказать пиццу через ваш бот.

«Я хочу заказать пиццу».

– Пользователь A.

«Не могли бы вы мне помочь заказать пиццу?»

– Пользователь B.

«Чувак, лови мой заказ».

– Пользователь C.

«Эй, хочу большую Пепперони с чесночным хлебом и соусом».

– Пользователь D.

Все вышеупомянутые пользовательские сообщения, в сущности, означают одно и то же. Они отличаются по написанию, но их смысл в размещении заказа является одинаковым. При достаточной тренировке серверная часть службы с элементами понимания ЕЯ распознает смысл всех этих сообщений как одинаковый.

Ловушки разговорного интерфейса пользователя

Мир разговорных интерфейсов вселяет большие надежды. Но не все так радужно на самом деле.

Чат-боты ограничиваются технологиями искусственного интеллекта, существующими на сегодняшний день. Понимание естественного языка представляет собой задачу, за которую полностью отвечает искусственный интеллект. Это одна из сложных задач в этой области. Данная технология проделала длинный путь и с достаточной тренировкой может хорошо справляться со специализированными вариантами использования. Но, несмотря на независимые исследования в области понимания ЕЯ специалистами в области ИИ по всему миру, эта методология имеет большие резервы для совершенствования.

Программы-роботы, приводимые в действие технологией понимания естественного языка, могут делать ошибки от глупых и безобидных до абсолютно грубых (рис. 3.6 и 3.7). Боты, которые предлагают диалоги на основе команд, в которых используются только определенные ранее слова. Поэтому они откликаются только на команды и полностью игнорируют слова, которых в этих командах нет. Боты с элементами понимания ЕЯ, с другой стороны, сопровождаются побочным

эффектом неопределенности. Даже разработчики, которые натренировали серверную часть бота с элементами понимания ЕЯ, иногда не знают, как она будет отвечать на незнакомые выражения.

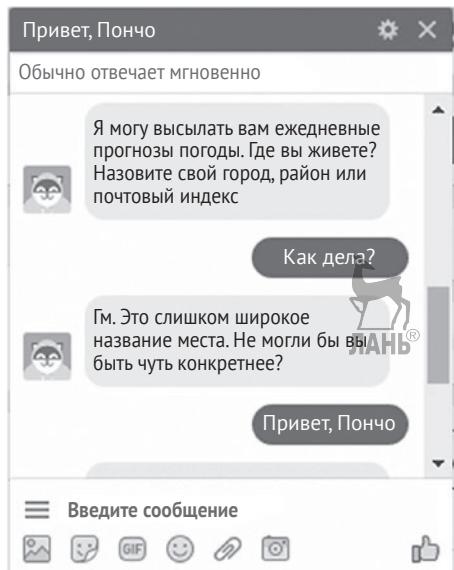


Рис. 3.6 ♦ Бот мессенджера Facebook, Пончо, не в состоянии понять разницу между сообщением с местоположением и без местоположения.

Глупая, но безопасная ошибка



getty @geraldmellor · 23 Mar 2016
«Тяя меньше, чем за 24 часа прошла путь от «люди – это нечто невероятное» до совершенной нацисты, и меня абсолютно не заботит будущее ИИ.

4 247 13K 11K

Рис. 3.7 ♦ Тяя – это ИИ-ориентированный чат-бот Twitter, созданный подразделением технологий и исследований Microsoft. Тяя была разработана с целью подражания языковым конструкциям 19-летней американской девушки и обучения на взаимодействии с человеческими пользователями соцсети Twitter. Этот чат-бот вызвал серьезные споры, когда группа пранкстеров научила Тяю подстрекательским сообщениям. В течение 24 часов Тяя начала выдавать расистские и сексуально заряженные твиты в ответ на твиты других пользователей соцсети Twitter. Впоследствии этот чат-бот был убран компанией Microsoft. Тяя остается хорошим примером эксперимента в области ИИ, который закончился ужасно неудачно

Самые лучшие программы-роботы учатся на своих ошибках. Полезно поддерживать журналы, в которых хранятся как файлы с пользовательскими сообщениями, так и ответы программы-робота на эти сообщения. Разработчики могут регулярно просматривать пользовательские сообщения, которые остались без ответа или которые получили неправильные ответы, и использовать эту информацию, чтобы продолжить тренировку своих серверных частей служб с элементами понимания ЕЯ.

Разговорные интерфейсы пользователя не ограничиваются только своими серверными частями с элементами понимания ЕЯ. Некоторые задачи по-прежнему легче выполнять на основе графического интерфейса. Например, в примере заказа пиццы бот должен уметь показывать меню (с ценами) тем пользователям, которые заведомо не знают их ассортимент. Меню с фотографиями каждого блюда в 10 раз лучше меню, состоящего из одного текста. Если в нижней части каждой фотографии, из которых состоят пункты меню, установить кнопку **Добавить в корзину**, бот гораздо точнее выполнит обработку каждого заказа. И тогда пользователю, оформляя заказ, не придется каждый раз набирать полное название заказываемого пункта меню.

Сейчас, учитывая современные возможности и уровень развития разговорного интерфейса пользователя, использование организацией для общения с клиентами только программы-робота было бы рискованным решением. Гораздо лучше перенаправлять обмен сложными сообщениями оператору – человеку, который способен лучше справиться с диалогами.

Будущее может быть за гибридным интерфейсом пользователя (разговорный + графический)

Пока разговорный интерфейс пользователя не стал достаточно развитым, чтобы самостоятельно справляться с любым типом разговоров и пользовательских сообщений, идеальный интерфейс пользователя объединит в себе разговорный интерфейс с элементами графического интерфейса. Это значительно улучшает пользовательский опыт взаимодействия с приложением, во много раз упрощая пользователю его использование. Рисунок 3.8 демонстрирует наше утверждение примером бота путешествий на основе Skype.

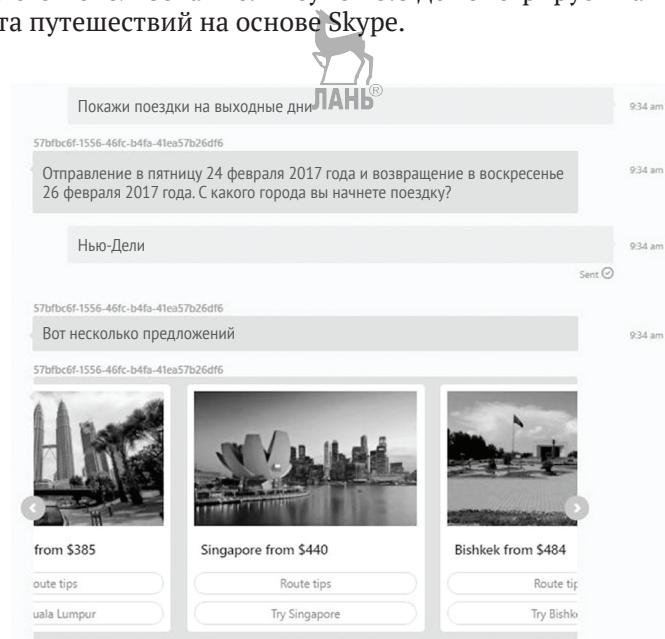


Рис. 3.8 ♦ Заказ авиабилета по Skype с использованием программы-робота, Skyscanner. Предложения имеют фотографии и кнопки

Пользователю не нужно выбирать предлагаемый вариант поездки, набирая его название или идентификатор. Для продолжения достаточно щелкнуть мышью или коснуться пальцем нужной кнопки. Нажав на кнопку, вы продолжите диалог и отправите от имени пользователя соответствующего голосового сообщения. Некоторые последующие сообщения от программы-робота также могут содержать элементы графического интерфейса (рис. 3.9).

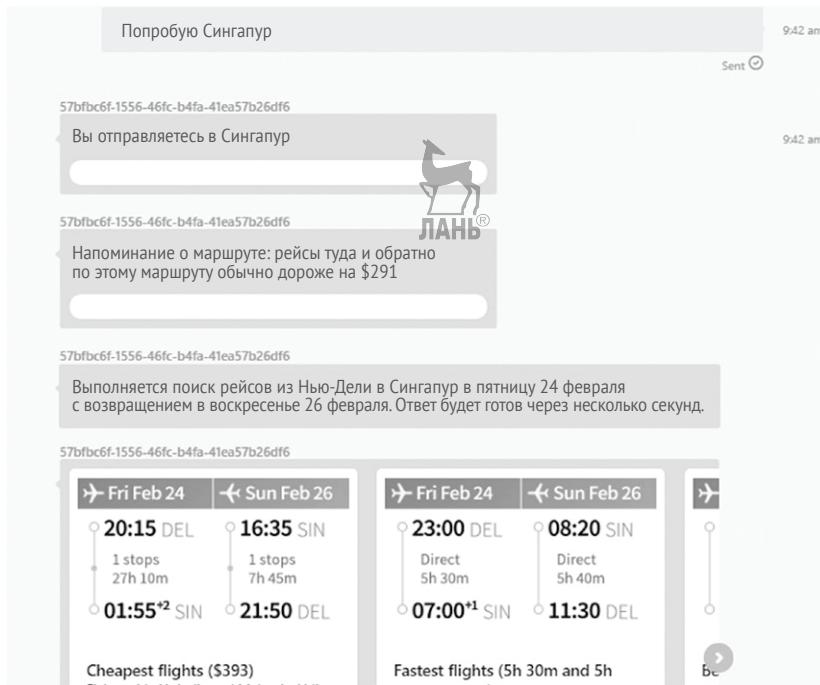


Рис. 3.9 ♦ Взаимодействие с элементами графического интерфейса продолжает текстовый диалог

ПРИНЦИПЫ РАЗРАБОТКИ

Самое важное правило, Святой Грааль, который необходимо помнить во время проектирования разговорных интерфейсов, состоит в том, что принцип «что видишь на экране, то получишь при печати» (WYSISYG) для интерфейса пользователя не используется. С вашей программой-роботом клиенты смогут только разговаривать. Естественно, на вас лежит обязанность сделать этот процесс в максимальной степени интуитивно понятным.

Отличительная черта разговорных интерфейсов состоит в том, что, в отличие от графических интерфейсов, они не могут иметь страниц «Приступая к работе» или «Часто задаваемые вопросы», которые служат в качестве путеводителей для пользователя. Есть только текст, а также эпизодические ссылки и кнопки. Чтобы не сбиться с правильного пути, запомните приведенные ниже принципы.

Введение. В 99% всех случаев именно пользователь начнет общение и станет инициатором разговора. Это может быть приветствие в одно слово или же прямой вопрос по существу. Ваш бот должен найти возможность сообщить пользователю, что и как он может делать. Дайте вашему боту человеческое имя, чтобы придать диалогу более естественный характер.

Привет! Я – Тед. Чем могу быть полезен? Я могу Вам помочь заказать билет на кинофильм или билет на спортивное мероприятие. Вы можете начать, произнеся «закажи билет».

Индикация. Ваш разговорный интерфейс может не иметь визуального интерфейса, такого как индикатор выполнения операции, который бы сообщал, что ваш бот активно переносит данные или выполняет действие для пользователя. Поэтому следует постоянно отправлять сообщения о ходе выполнения операции, чтобы пользователь не думал, что бот не отвечает. Можно запрограммировать бот так, чтобы он игнорировал пользовательские сообщения, полученные в это время.

Я по-прежнему ищу лучшие предложения.

...

Сожалею, что требуется так много времени. Качественное оказание услуги занимает время, ведь так?



...

Я про Вас не забыл.

Подтверждение и следующие шаги. Когда бот закончил выполнение задачи, выполнить которую его попросил пользователь, он должен вежливо сообщить об этом и о возможных следующих шагах. Также можно подсказать, что именно бот может сделать еще.

Ваш билет в Сингапур был заказан на 2:30 утра 14 февраля 2017.

Безопасного полета!

Не хотите ли, чтобы я напомнил Вам о поездке за 12 часов до отъезда?

Юмор. Время от времени стоит шутить. Вместо того чтобы бот все время отвечал формальным и скучным тоном, запрограммируйте его так, чтобы он отвечал шутливыми сообщениями. Цитируйте известные книги, фильмы, телесериалы и т. д., чтобы сделать опыт пользователя еще интереснее.

Пока я подбираю доступные места.., вы видели последний фильмец «Звездные войны»? Слышал, что о нем восторженно отзываются.

ИНФРАСТРУКТУРА MICROSOFT BOT FRAMEWORK

Microsoft Bot Framework – это набор пакетов SDK с открытым исходным кодом для легкого и быстрого создания программ-роботов корпоративного класса. Посредством нескольких встроенных конструкторов эта платформа ускоряет разработку приложения с разговорным интерфейсом. Это могут быть простые приложения, предназначенные для обработки пользовательских сообщений независимо, или контекстные приложения. Контекстные приложения способны обучаться, анализируя ранее полученные пользовательские сообщения. В результате пользователи будут получать более точные ответы на свои запросы.

Используя инфраструктуру Bot Framework, можно легко отработать логику программы-робота, после чего установить эту программу на одной или нескольких платформах обмена сообщениями. Например, в Skype, Facebook Messenger и т. д. Для улучшения возможностей каналов связи в ботах можно использовать гибридные интерфейсы, делая взаимодействие пользователя с каналом связи удобнее.

Полнофункциональная среда Bot Framework не является частью пакета Cognitive Services, но она может эффективно интегрироваться с несколькими API Cognitive Services. Данная среда благодаря технологиям понимания естественного языка, компьютерного зрения и обработки речи позволяет сделать общение с вашим ботом похожим на общение с человеком.

Во время написания данной книги инфраструктура Bot Framework находилась в экспериментальной фазе. Посетите сайт <https://dev.botframework.com> и узнайте, какими возможностями обладает данная платформа сейчас. На этом веб-сайте вы найдете справочник, представляющий собой список ботов, созданных на основе Bot Framework. Эти боты внесены в список своими разработчиками. Чтобы самому убедиться в преимуществах приложения с разговорным интерфейсом, загрузите один из предлагаемых ботов и проверьте его.

Создание приложения с разговорным интерфейсом в среде Bot Framework



Бот в контексте инфраструктуры Bot Framework, по существу, представляет собой комбинацию из трех компонентов. Взгляните на рис. 3.10, чтобы увидеть связь между этими компонентами.



Рис. 3.10 ♦ Схема архитектуры приложения с разговорным интерфейсом на основе инфраструктуры Bot Framework

- **Конечная точка обмена сообщениями:** веб-служба/API, создается с использованием полнофункциональной среды Bot Framework и является основным компонентом, в котором расположена логика всего бота. Когда вы говорите, что создаете бот, вы по существу создаете веб-службу этого бота, которая расположена в серверной части. В настоящее время веб-службу среды Bot Framework официально можно создать только на C# (ASP.NET MVC) и Node.js.
- **Канал связи:** это платформа обмена сообщениями, которую ваш бот использует. Ваш бот может быть сконфигурирован для использования более одного канала связи. В настоящее время среда Bot Framework поддерживает следующие каналы связи: Skype, Telegram, Facebook Messenger, Slack, GroupMe, Kik, SMS и электронную почту Office 365.
- **Соединитель:** компонент программного обеспечения, составная часть среды Bot Framework, которая соединяет вашу конечную точку с одним или несколькими каналами связи и занимается перенаправлением сообщений между ними. Именно соединитель придает вашей конечной точке способность использовать одновременно несколько каналов связи.

Понимая базовую архитектуру приложения на основе среды Bot Framework, вы создадите, например, простой бот записи на прием, который помогает планировать визит к врачу.

Шаг 0. Подготовка среды разработки

Visual Studio – это все, что вам нужно для создания бота. Основной элемент, который необходимо создать, – это конечная точка обмена сообщениями. Конечная точка – это веб-служба, создаваемая на основе механизма RESTful, которая находится на веб-сервере. Веб-служба помещается на веб-сервере, чтобы соединители среды Bot Framework могли легко ее найти для соединения с одним или несколькими каналами связи.

В .NET рекомендуемый способ создания веб-службы на основе механизма RESTful предполагает использование инфраструктуры ASP.NET MVC. В VS 2017 создать веб-службу на основе ASP.NET достаточно легко. В VS предусмотрено большое количество предварительно настроенных шаблонов. Поэтому при создании определенных типов приложений вам не придется создавать эти шаблоны с нуля. В главе 2 вы создали консольное приложение с использованием шаблона VS 2017. Сейчас же мы воспользуемся шаблоном для создания приложения ASP.NET MVC. Когда мы выберем этот шаблон, Visual Studio создает новое приложение MVC и добавляет все зависимости (ссылки на DLL, пакеты NuGet, библиотеки JavaScript и т. д.). Кроме этого, вы увидите пример программного кода, с которого работу можно начать.

И Если вы – новичок в инфраструктуре ASP.NET MVC или веб-службах в целом, то рекомендуем прочитать книгу ASP.NET Web API 2: «Создание служб на основе механизма REST от начала до конца» (ASP.NET Web API 2: Building a REST Service from Start to Finish, Jamie Kurtz and Brian Wortman) издательства Apress.

Документация по инфраструктуре Bot Framework содержит много примеров, которые помогут вам создать свою конечную точку обмена сообщениями на основе ASP.NET MVC. Кроме того, с помощью специального шаблона Visual Studio, созданного именно для таких целей, данная инфраструктура обеспечит наиболее оптимальный способ создания конечной точки. Обратитесь на сайт <http://aka.ms/bf-bc-vstemplate>, чтобы загрузить этот шаблон. При загрузке шаблона сохраните полученный zip-файл в своем каталоге шаблонов Visual Studio 2017, которые традиционно находятся в папке %USERPROFILE%\Documents\Visual Studio 2017\Templates\ProjectTemplates\Visual C#.

Шаг 1. Создание нового проекта приложения Bot Framework

В этом проекте вы создадите простой бот записи к врачу. Единственная цель этого бота состоит в том, чтобы помочь пользователю записаться на прием к врачу.

Запустите программу Visual Studio 2017 и создайте новый проект, выполнив команду меню **Файл > Создать > Проект**. Выберите в появившемся диалоговом окне шаблон Bot Application Visual C#, как показано на рис. 3.11. В результате будет создано приложение ASP.NET MVC с настройками для среды Bot Framework.

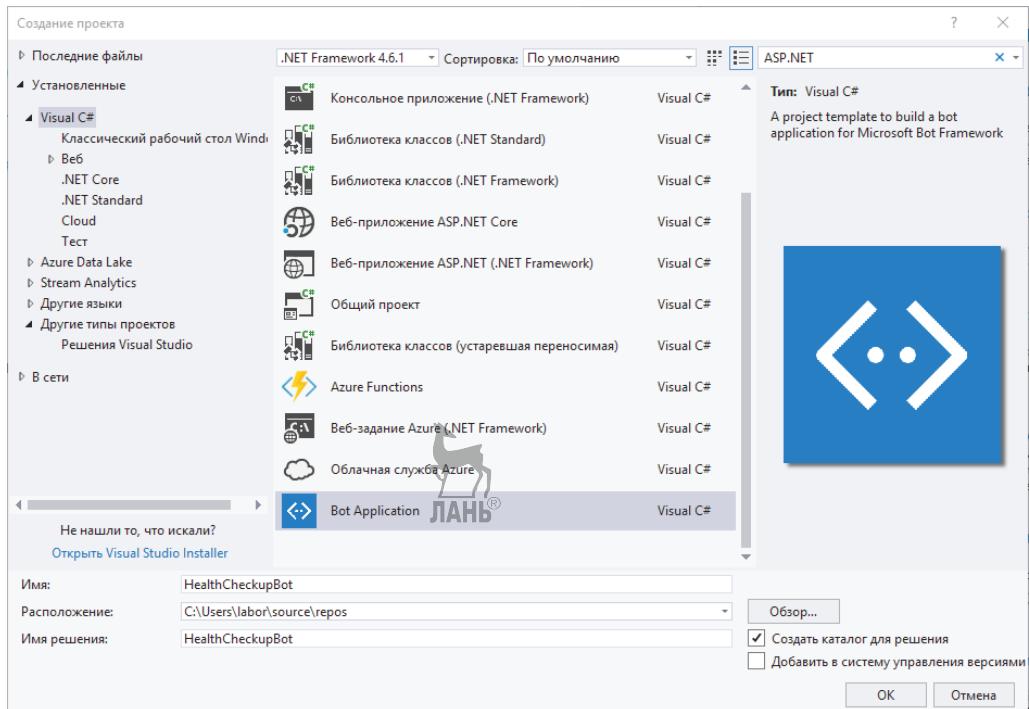


Рис. 3.11 ♦ Создание нового приложения Bot Framework в Visual Studio 2017

Шаг 2. Первое сообщение и сообщение по умолчанию

В приложении Bot Application, которое вы только что создали, должен быть только один контроллер. Откройте в Обозревателе решений папку Контроллеры и щелкните мышью на строке MessagesController.cs. В окне Bot Application1 появится исходный код. Начнем его редактирование.

Вашим главным методом здесь является метод Post(). Именно тут содержится вся логика вашего бота. Именно в это место программного кода приходят пользовательские сообщения для обработки, и именно отсюда отправляются соответствующие ответы. В настоящее время бот сконфигурирован на передачу назад того же самого сообщения вместе с длиной этого сообщения в символах, которое бот получает от пользователя.

Давайте модифицируем ваш исходный код, чтобы возвращать приветственные сообщения и сообщения, заданные по умолчанию. При редактировании исходного кода будьте внимательны (см. листинг 3.2).

Листинг 3.2 ♦ Новый исходный код

```
[BotAuthentication]
public class MessagesController : ApiController
{
    string[] greetingPhrases = new string[] { "привет", "приветствую", "здравствуйте",
    "как дела", "доброе утро", "добрый день" };
```

```

/// <summary>
/// POST: api/Messages
/// Receive a message from a user and reply to it
/// </summary>
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
{
    if (activity.Type == ActivityTypes.Message)
    {
        ConnectorClient connector = new ConnectorClient(new Uri(activity.ServiceUrl));
        string userText = activity.Text.ToLower();
        string replyText = "";
        if(this.greetingPhrases.Contains(userText))
        {
            replyText = @"Приветствую! Я - бот записи к врачу.
Буду рад Вам помочь спланировать визит к врачу.
Скажите 'записаться к врачу.'";
        }
        else
        {
            replyText = "К сожалению, я не понял, что Вы сказали.";
        }
        Activity reply = activity.CreateReply(replyText);
        await connector.Conversations.ReplyToActivityAsync(reply);
    }
    else
    {
        HandleSystemMessage(activity);
    }
    var response = Request.CreateResponse(HttpStatusCode.OK);
    return response;
}

private Activity HandleSystemMessage(Activity message) {...}
}

```

Построчная разбивка исходного кода следующая:

```
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
```

Этот фрагмент информационного обмена между ботом и пользователем называется действием. Вы получаете объект действия от POST-вызыва по протоколу HTTP, направленного на вашу конечную точку обмена сообщениями веб-службы. Среда Bot Framework поддерживает несколько типов действий, некоторые из которых вы увидите чуть ниже. Прямо сейчас вы заинтересованы в действии с типом Message (Сообщение).

Действие Message представляет текстовое сообщение и некоторую о нем информацию, такую как:

- текстовая строка;
- отправитель;
- получатель;
- метка времени.



```
if (activity.Type == ActivityTypes.Message)
{
    ConnectorClient connector = new ConnectorClient(new Uri(activity.ServiceUrl));
    string userText = activity.Text.ToLower();
```

Если полученное вашей конечной точкой действие имеет тип `Message`, то (для простоты сравнения) текст сообщения преобразуется в нижний регистр, и создается экземпляр соединителя. Этот экземпляр соединителя будет использован, чтобы передавать ответ обратно пользователю. Действие `activity.ServiceUrl` указывает на URL-адрес, куда будет отправляться ответ бота. Каждый канал связи имеет свой собственный URL-адрес обслуживания, где прослушиваются ответы.

```
if(this.greetingPhrases.Contains(userText))
{
    replyText = @"Приветствую! Я – бот записи к врачу.
        Буду рад Вам помочь спланировать визит к врачу.
        Скажите 'записаться к врачу.'";
}
else
{
    replyText = "К сожалению, я не понял, что Вы сказали.";
}
```

Далее вы проверяете, соответствует ли сообщение пользователя одной из фраз, определенных в строковом массиве `greetPhrases`. Если это так, то вы назначаете своему ответу вежливое вводное сообщение. В противном случае вы отправляете сообщение по умолчанию, чтобы дать пользователю понять, что вы не знаете, как ответить на его сообщение.

```
Activity reply = activity.CreateReply(replyText);
await connector.Conversations.ReplyToActivityAsync(reply);
```

Вы используете метод `CreateReply()` полученного действия, чтобы создать новое действие для вашего ответного сообщения. Затем вы отправляете свой ответ через объект соединителя.

```
var response = Request.CreateResponse(HttpStatusCode.OK);
return response;
```

После отправки своего ответа вы возвращаете из POST-метода веб-службы простой код состояния HTTP 200, чтобы дать сигнал об успешном завершении этой операции.



Шаг 3. Выполнение, тестирование и отладка вашего бота

Настало время выполнить ваше приложение, чтобы увидеть, как бот отвечает на пользовательские сообщения. Visual Studio поставляется в комплекте с облегченным программным обеспечением веб-сервера, IIS Express, для выполнения веб-приложений и служб. Веб-сервер IIS Express – это упрощенная версия своего полноценного собрата, IIS.

Чтобы запустить вашу конечную точку обмена сообщениями, откройте меню «Отладка» и выберите элемент «Начать отладку». Или нажмите клавишу F5. В результате приложение будет запущено в режиме отладки. По завершении процесса отладки откройте в своем браузере новую вкладку и введите URL-адрес конечной

точки. По умолчанию приложения Bot Framework на порту 3979. Поэтому его URL-адрес будет следующий: <http://localhost:3979>. Здесь должна появиться страница нашего приложения (рис. 3.12).

HealthCheckupBot

Describe your bot here and your terms of use etc.

Visit [Bot Framework](#) to register your bot. When you register it, remember to set your bot's endpoint to |
https://your_bots_hostname/api/messages

Рис. 3.12 ❖ Страница по умолчанию конечной точки обмена сообщениями.
 Если вы это видите, все работает, как надо

Открытая в браузере страница по умолчанию не дает вам возможности протестировать ваш бот. Это статическая страница, которая используется только для описания бота. Для тестирования вашего приложения можно применить программу Postman, как это было сделано с RESTful API компьютерного зрения когнитивных служб в главе 2. Здесь нужно выполнить следующее условие: вам должен быть известен заголовок и формат тела HTTP соответствующего API. Но мы можем предложить более интересный вариант – установить на ваш компьютер эмулятор Bot Framework.

Эмулятор Bot Framework – это бесплатное программное обеспечение, которое специально предназначено для локального или удаленного тестирования приложения Bot Framework. Скачать эмулятор можно со страницы <https://aka.ms/bf-bc-emulator>.

Установите полученный эмулятор. Запустите его и укажите в строке поиска приведенный ниже URL-адрес:

<http://localhost:3979/api/messages>

Не заполняя поля ввода идентификатора приложения Microsoft App и пароля **Password**, нажмите кнопку **Connect** (Соединить).

Если вы поместили в свой исходный код точку останова отладчика, то Visual Studio может выглядеть, как показано на рис. 3.13. Вы можете воспользоваться богатыми возможностями отладки Visual Studio, чтобы просмотреть свой исходный код. Для этого воспользуйтесь клавишами **F10** (Перешагнуть), **F11** (Шагнуть внутрь) и **F5** (Продолжить).

В Visual Studio нажмите кнопку **F5**, чтобы закончить отладку, в случае если у вас есть активные точки останова. Теперь снова откройте эмулятор и начните разговор с вашим ботом. Чтобы начать разговор, напечатайте и отправьте слово «привет».

Как видно из рис. 3.14, бот ответит на ваши сообщения «привет» и «приветству», как и ожидалось. И тем не менее он не понял сообщение «привет, бот» и вернул сообщение по умолчанию. Это можно быстро исправить, добавив в класс **MessagesController** следующий ниже метод:

```
private bool MessageHasPhrase(string message, string[] phraseList)
{
    foreach (string phrase in phraseList)
```

```

{
    if (message.Contains(phrase))
    {
        return true;
    }
}
return false;
}

```



The screenshot shows a code editor with C# code and a local variables window below it.

```

20     public async Task<HttpResponseMessage> Post([FromBody]Activity activ
21
22     if (activity.Type == ActivityTypes.Message)
23     {
24         ConnectorClient connector = new ConnectorClient(new Uri(activity.ServiceUrl));
25         string userText = activity.Text.ToLower();
26         string replyText = "";
27         if (this.greetingPhrases.Contains(userText))
28             r

```

Локальные

Имя	Значение	Тип
this	{HealthCheckupBot.MessagesController}	HealthCheckupBot.MessagesController
activity	{Microsoft.Bot.Connector.Activity}	Microsoft.Bot.Connector.Activity
response	null	System.Net.Http.HttpResponseMessage

Рис. 3.13 ❖ Visual Studio показывает активную точку останова, которая сработала из-за входящего сообщения из эмулятора

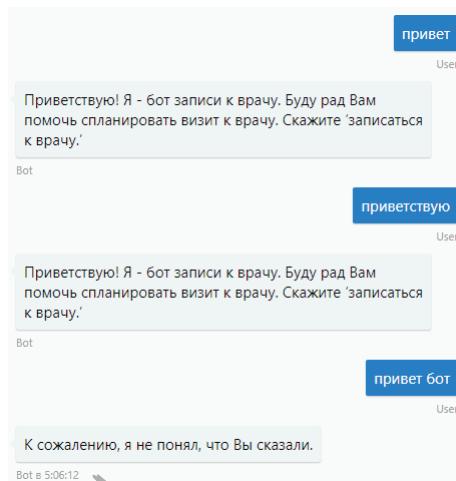


Рис. 3.14 ❖ Начало диалога с ботом в эмуляторе



Затем замените свое условие if с проверкой фразы на нижеследующее:

```

if(this.MessageHasPhrase(userText, greetingPhrases))
{
    replyText = @"Приветствуя! Я - бот записи к врачу.
    Буду рад Вам помочь спланировать визит к врачу.
    Скажите 'записаться к врачу.'";
}

```

```

}
else
{
    replyText = "К сожалению, я не понял, что Вы сказали.";
}

```



Результат работы модифицированного бота можно увидеть на рис. 3.15.

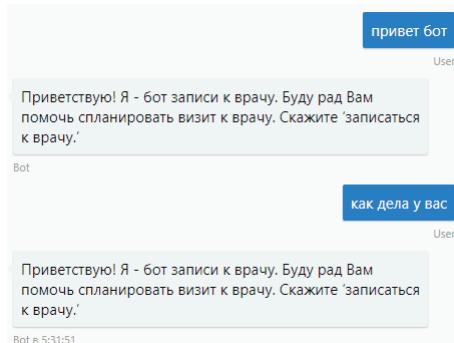


Рис. 3.15 ❖ Ваш бот теперь может распознавать выражения, которые не совпадают в точности с выражениями из вашего списка

На данном этапе ваш бот стал немного интеллектуальнее, потому что он может распознавать похожие выражения. Правда, он не может распознавать другие подобные выражения, фрагменты которых не присутствуют в вашем списке выражений:

- «как у вас дела»;
- «добрый вечер»;
- «чем можешь мне помочь».

Поэтому как вы поступите? Обновите свой список выражений всеми возможными словами/выражениями, которые пользователь может прислать? Разумеется, нет. Вот где на выручку может прийти технология понимания естественного языка. В главе 6 вы воспользуетесь службой понимания ЕЯ под названием Microsoft LUIS, чтобы добавить в ваш бот способность понимания языка и отказаться от жесткого кодирования вручную всех возможных выражений, положившись на испытанный службу LUIS, которая распознает как новые, так и известные выражения.

Шаг 4. Планирование визита к врачу

Теперь обратимся к главной части: планирование визита к врачу. Давайте добавим логику, которая будет распознавать запрос пользователя на запись к врачу, предлагать ему на выбор имеющиеся свободные часы приема и подтверждать запись к врачу. В идеальном плане бот будет сверяться с календарем врача или своего рода базой данных для извлечения свободных часов приема. В вашем простом боте вы встроите часы приема в программный код.

Обновите свой исходный код, как показано в листинге 3.3. Теперь в нем появилось два новых массива выражений и условия `else-if`.

Листинг 3.3 ❖ Обновленный исходный код

```
string[] greetingPhrases = new string[] { "привет", "приветствую", "здравствуйте", "как дела",  
"доброе утро", "добрый день" };  
string[] appointmentPhrases = new string[] { "записаться", "запишите", "прием", "к врачу" };  
string[] slotPhrases = new string[] { "1", "2", "3", "4", "5", "первый", "второй", "третий",  
"один", "два", "три" };  
  
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)  
{  
    if (activity.Type == ActivityTypes.Message)  
    {  
        ConnectorClient connector = new ConnectorClient(new Uri(activity.ServiceUrl));  
        string userText = activity.Text.ToLower();  
        string replyText = "";  
  
        if(this.MessageHasPhrase (userText, greetingPhrases))  
        {  
            replyText = @"Приветствую! Я - бот записи к врачу.  
Буду рад Вам помочь спланировать визит к врачу.  
Скажите 'записаться к врачу.'";  
        }  
        else if (this.MessageHasPhrase(userText, appointmentPhrases))  
        {  
            replyText = @"Я нашел следующие свободные часы приема.  
Пожалуйста, укажите **номер времени приема**, чтобы подтвердить запись к врачу.  
  
1) 26 фев, 9:00 утра  
2) 02 мар, 12:30 пополудни"; // поддерживается базовая упрощенная разметка  
        }  
        else if (this.MessageHasPhrase(userText, slotPhrases))  
        {  
            replyText = "Ваша запись к доктору Джону Доу подтверждена. До встречи.";  
        }  
        else  
        {  
            replyText = "К сожалению, я не понял, что Вы сказали.";  
        }  
  
        Activity reply = activity.CreateReply(replyText);  
        await connector.Conversations.ReplyToActivityAsync(reply);  
    }  
    else  
    {  
        HandleSystemMessage(activity);  
    }  
    var response = Request.CreateResponse(HttpStatusCode.OK);  
    return response;  
}
```

Рисунок 3.16 показывает диалог, содержащий запрос на запись к врачу с обновленным ботом.

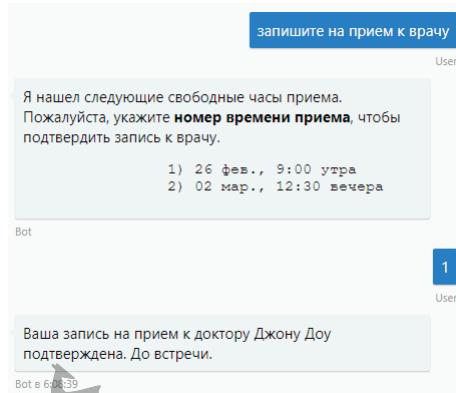


Рис. 3.16 ❖ Вывод подтверждения визита к доктору Джону в боте записи к врачу

На этом речевой поток вашего очень простого бота завершается. Теперь бот способен представиться пользователю и спланировать запись к врачу. Но многое он еще делает неправильно.

Он не в состоянии распознавать дату и время в запросе. Даже если дата и время будут присутствовать в сообщении пользователя, ваш бот их просто проигнорирует. Поэтому выражение типа «запишите меня на 27 февраля на 9:00» приведет к тому же самому жестко закодированному сообщению. В идеальном случае бот должен принимать во внимание указанное пользователем время и соответствующим образом проверять его по календарю.

Кроме того, разные пользователи могут указывать информацию о дате/времени по-разному. Бот должен уметь понимать все, что приведено ниже, и даже больше:

- 02/27/17 9:00;
- 27 февраля, 9:00;
- 27 февраля в 9 утра;
- завтра в 9:00.

Этого можно добиться благодаря использованию поддержки понимания естественного языка. Подробнее об этом вы прочитаете в главе 6, посвященной службе LUIS. Еще один недостаток созданного нами бота – в том, что он совершенно ничего не помнит о предыдущих сообщениях пользователя. Он знает и обрабатывает только текущее пользовательское сообщение. Для примера взгляните на рис. 3.17.

Эта ситуация поднимает проблему **контекста**. Ваш бот не знает контекст ведущегося разговора и поэтому рассматривает каждое сообщение как независимое. Эта проблема может быть решена за счет использования **диалогов**. Среда Bot Framework поставляется с мощной поддержкой диалогов для создания **контекстных** приложений с разговорным интерфейсом. В главе 6 будет показано ограниченное использование диалогов. Чтобы получить более точное представление относительно этой темы, мы рекомендуем обратиться к официальной документации по Bot Framework на C#.

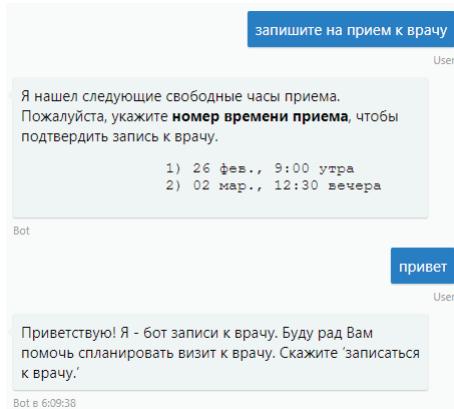


Рис. 3.17 ❖ Если пользователь, намечая визит к врачу, преднамеренно или непреднамеренно отправит другой тип сообщения, который бот поддерживает, то бот немедленно «отбросит» предыдущий контекст и ответит совершенно новым и несвязанным сообщением. В идеальном случае бот должен высказать сомнение, что сообщение пользователя не похоже на приемлемое время приема (12:30 вечера)

Шаг 5. Обработка системных сообщений

Системные сообщения – это сообщения или сигналы, которые отправляются не пользователем, а его каналом связи. Каналы связи поддерживают не все системные сообщения, однако ниже приведен общий список сообщений, которые поддерживаются в среде Bot Framework.

- **ConversationUpdate.** Изменения состояния разговора, как в случае, если или к разговору подключается еще один участник, или количество принимающих участие в разговоре уменьшается.
- **DeleteUserData.** Пользователь делает запрос на удаление разговора или профиля. В некоторых странах платформы обмена сообщениями обязаны предоставлять эту возможность с целью защиты частной жизни пользователей.
- **ContactRelationUpdate.** Добавление или удаление бота из списка контактов пользователя.
- **Typing.** Сигнал, показывающий, что пользователь что-то набирает. Некоторые каналы могут отправлять его чаще, чем другие.
- **Ping.** Проверяет, правильно ли в боте реализована защита.

В `MessagesController.cs` имеется метод `HandleSystemMessage()`, который среда Visual Studio добавляет при создании приложения. В этом методе охватываются все вышеупомянутые типы системных сообщений. Вы можете добавить собственный программный код обработки каждого типа сообщения. Вам дается возможность вывести системное сообщение в журнал либо отправить пользователю ответ (например, шутливое сообщение, когда пользователь слишком медлит с набором сообщения).

Следующие шаги

Теперь, когда вы ознакомились с обучающими инструкциями по созданию бота для записи к врачу, у вас появился достаточный запас знаний, чтобы приступить к созданию собственного бота. Придумайте тему для вашего бота и начинайте программировать в вашей среде разработки Visual Studio.

По ходу изложения мы уже говорили и повторим еще раз: нами был создан элементарный бот. Он может правильно выполнять элементарные вещи, но имеет большие резервы для совершенствования. Задача этого обучающего руководства состояла не в том, чтобы сделать из вас эксперта в создании приложений с разговорным интерфейсом, а чтобы позволить вам приступить к собственным разработкам. В качестве следующего шага попробуйте добавить в бот записи к врачу контекст посредством применения диалогов и подсказок.

Кроме того, бот может транслировать свои сообщения на одном или нескольких каналах связи. В существующем сейчас виде бот работает локально на компьютере разработчика, и связываться с ним можно только через эмулятор. Конечная цель, конечно, состоит в том, чтобы сделать бот доступным через такие каналы связи, как Skype и Facebook Messenger. О том, как это сделать, вы узнаете из главы 6.

EXERCISE: FORMFLOW

Re-create the Health Checkup Bot Using FormFlow

FormFlow – это функционал в среде Bot framework, который может автоматически преобразовывать класс C# в беседу на основе форм посредством использования диалогов. Этот функционал в особенности полезен в ситуациях, где задача бота состоит в том, чтобы в результате текстовой беседы пользователь заполнил форму.

Обратитесь на docs.botframework.com, чтобы получить более подробную информацию о FormFlow. Рассмотрев примеры на этой странице, создайте бот записи к врачу заново с использованием FormFlow. Совет: перед назначением визита к врачу с узкой специальностью добавьте в разговор дополнительные компоненты, такие как вопрос к пользователю о симптомах его недомогания.

Резюме

Подводя итоги, констатируем, что в этой главе вы:

- познакомились с краткой историей эволюции интерфейсов пользователя;
- узнали, что такое разговорные интерфейсы и в чем их преимущества;
- узнали возможности разговорного интерфейса;
- научились создавать приложение с разговорным интерфейсом (бот) на основе полнофункциональной среды Microsoft Bot Framework.

В **следующей главе** вы подробно познакомитесь с технологией понимания естественного языка, лежащими в основе этой технологии основными понятиями и тем, где эту технологию можно применить.

Глава 4



Применение технологии понимания естественного языка

Термин **понимание естественного языка**, или **Natural language Understanding** (NLU), вы встречаете начиная с главы 1. На приведенных ранее примерах было показано, с чем связана эта технология и что она может делать. Вы уже знаете, что данная технология позволяет приложению имитировать человеческое понимание высказываний пользователя на естественном языке. В этой главе вы подробно узнаете о технологии понимания ЕЯ и разберетесь в понятиях, связанных с LUIS, служащих компонентом понимания ЕЯ в когнитивных службах.

i Хотя термин *естественный язык* можно представить как полную противоположность *искусственному языку* (такому как язык программирования), он не всегда имеет тот же смысл, что и *язык общения между людьми*. Все естественные языки являются языками общения между людьми, но не все эти языки естественные. Естественный язык – это язык, который использовался людьми в течение долгого времени и развивался естественным образом из поколения в поколение. Английский, французский, русский, латинский, арабский язык и т. д. являются естественными языками.

Однако имеется еще один класс человеческих языков, именуемый *сконструированными языками*, которые были искусственно созданы одним человеком или группой лиц с формальным набором неизменных грамматических правил. Самым популярным сконструированным языком является эсперанто, на котором, согласно некоторым оценкам, говорит приблизительно 2 миллиона человек по всему миру.

Высшая цель искусственного интеллекта всегда состояла в том, чтобы служить человечеству способами, недоступными для традиционных машин. Чтобы служить людям, искусственный интеллект должен уметь понимать людей так, как сами люди понимают друг друга. Для человека выполнение какой-то функции искусственным интеллектом должно быть таким же легким, как обращение к другому человеку за помощью. Неудивительно, что подавляющая часть всех исследований в области ИИ была сконцентрирована в области обработки и понимания естественных языков, языков людей.

Понимание ЕЯ – это одна из нескольких основных задач более широкой области искусственного интеллекта обработки естественного языка (*natural language processing*, NLP). Но, в отличие от большинства других задач обработки ЕЯ, по-

нимание естественного языка представляет собой полную задачу для искусственного интеллекта. ИИ-полные задачи, или так называемые сложные, являются самыми трудными задачами в области ИИ, и они связаны с созданием машин, столь же интеллектуальных, что и человек. Эти задачи не могут быть решены с использованием одного специально разработанного алгоритма, как это делается для других задач ИИ. Задача понимания естественного языка может потребовать комбинации нескольких сложных алгоритмов, совместно работающих на различных этапах решения одной и той же задачи для достижения поставленной цели. Технология понимания естественного языка в сочетании с компьютерным зрением и машинным переводом является основными полными задачами для искусственного интеллекта.

Способность понимать естественные языки и толковать значение предложений является обязательным свойством интеллектуальной системы. Как результат понимание естественного языка используется в самых разнообразных коммерческих, академических и персональных приложениях. Например, на бытовом уровне в персональных роботах, которые понимают наши потребности, или в приложениях с разговорным интерфейсом, которые предоставляют медицинскую консультацию по рассказанным нами симптомам. Или в системах анализа информации, которые в целях предотвращения террористических актов будут обрабатывать огромные объемы текстовых сообщений между террористическими группами.

После прочтения этой главы вы познакомитесь с задачей понимания естественного языка. В частности, вы:

- узнаете, что такое задача понимания естественного языка;
- познакомитесь с краткой историей задачи понимания ЕЯ;
- узнаете, почему машины с трудом понимают естественный язык;
- познакомитесь с Интеллектуальной службой понимания естественного языка Microsoft Language Understanding Intelligent Service (LUIS);
- познакомитесь с архитектурой приложения с поддержкой LUIS;
- узнаете, как получить ключ подписки Azure для службы LUIS;
- узнаете, как создать демонстрационный пример рабочего веб-приложения на основе службы LUIS.

Что такое понимание ЕЯ?

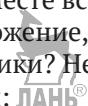
К настоящему времени вам стало ясно, что понимание естественного языка – это способность машины понимать естественные языки. Если говорить конкретнее, это процесс преобразования текста, произнесенного или введенного с клавиатуры на естественном языке, в форму, которую понимают компьютеры. В главе 1 вы увидели, как программа с элементами понимания ЕЯ определяет значение предложения, подразделяя его на намерение и сущности. Однако не все реализации методов понимания естественного языка работают с намерениями и сущностями. И тем не менее вы их найдете во всех главных общедоступных и проприетарных (основанных на частной собственности и на идеи коммерческого использования авторских и иных прав) функциональных средах с элементами понимания ЕЯ.

Тот или иной фрагмент текста может быть представлен одним намерением и одной или многочисленными сущностями. Используя эту информацию, программа может решить:

- какое действие выполнить на основе намерения;
- как выполнить действие на основе сущностей.

Давайте воспользуемся примером из главы 1, чтобы проиллюстрировать этот тезис. Рассмотрим предложение, представленное ниже, еще раз:

«Какая завтра ожидается погода в Дели?»

Для нас, людей, сразу же ясно, что вышеупомянутое предложение представляет собой намерение человека узнать о погоде в определенном географическом месте. Компьютер, который не знает правил грамматики, это предложение может не понять. Компьютер не знает, что такое существительное, глагол, прилагательное, предлог и т. д. Следовательно, ему не удастся составить верное представление о смысле предложения, если он и соберет вместе все слова. Сможет ли компьютер правильно понять вышеупомянутое предложение, если он будет знать и сумеет правильно применить все правила грамматики? Несомненно. Но тогда что бы он сделал с этими двумя вопросами/запросами: 

«На сколько лимонов ожидается погодка в Делях?»

«Дели погода звтр».

Компьютер, который всегда следует правилам, будет не в состоянии понять эти грамматически неправильные, находящиеся во власти сленга предложения. Конечно, не следует ожидать, что пользователь всегда напечатает грамматически правильные предложения:

- a) носители, равно как и неносители, языка одинаково позволяют себе делать грамматические ошибки и
- b) у большинства людей нет времени, чтобы напечатать полные слова или предложения.

Трюк в достижении точного понимания языка состоит не в том, чтобы научить компьютер всем правилам, жаргонизмам, частым ошибкам и т. д., а в том, чтобы натренировать его на сотнях тысяч или миллионах предложений и позволить ему самостоятельно выводить формальные и неформальные правила языка посредством своего рода распознавания образов. Это стало возможным благодаря методам машинного обучения, в особенности глубокого обучения.

Коммерческий продукт с элементами понимания ЕЯ обычно представляет- ся облачной службой, которая прошла эту первоначальную тренировку. Теперь, когда облачная служба знает основы языка, разработчики могут ее натренировать, чтобы она распознавала более специфические варианты использования, например «получить метеоинфо». Они могут натренировать службу так, чтобы она знала, что выражения «погода», «что за погода», «какая погода», «сообщи погоду» и т. д. означают одно и то же: получить информацию о погоде. Разработчикам не требуется тренировать службу на всей тысяче способов спросить то же самое. Пользуясь ее базовым пониманием языка, служба может сделать разумные догадки, когда она сталкивается с подобным, но неизвестным запросом пользователя. Поэтому, когда пользователь спрашивает о погоде любым из вышеупомянутых способов (или другим подобным образом), его намерение состоится в том, чтобы получить информацию о погоде. Другими, более техническими словами, мы говорим, что его **намерение** состоит в том, чтобы получить информацию о погоде.

Аналогичным образом разработчики могут натренировать службу распознавать в предложении определенные ключевые слова, которые дополняют это намерение (смысл). Информация о погоде обычно связана с местом и временем. Разработчики могут натренировать службу понимать названия нескольких городов, областей и стран и различным способам понимания выражений, обозначающих день/дату. Город и дата затем станут **сущностями**. Сущности могут быть обязательными или необязательными в зависимости от намерения. В случае с погодой, если нет указания на город и дату, можно неявно предположить, что пользователь хочет знать сегодняшнюю погоду в том месте, где он сейчас находится.

Рисунок 4.1 иллюстрирует, как служба с элементами понимания ЕЯ разбивает предложение.

Какая завтра ожидается погода в Дели

День

Намерение

Город

Рис. 4.1 ♦ Предложение на естественном языке
с подчеркнутыми намерением и сущностями



Такое разбиение предложения на намерение и сущности позволяет разрабатываемой программе понять смысл предложения, чтобы выполнить действие.

```
if (intent == "getWeatherInfo")
{
    result = getWeatherInfo(entities.City, entities.Date); // возвращает "21 градус
    Цельсия с малой вероятностью дождя"
    print "Погода в " + entities.City + " на " + entities.Date + " ожидается " + result;
}
```

Рисунок 4.2 показывает разбиение в формате «намерение–сущности» для примера с заказом авиабилета.



Закажи билет в Нью-Йорк для 2 взрослых на 12 марта

Намерение

Место назначения

Количество билетов

Дата отправления

Рис. 4.2 ♦ Компьютерная программа
может использовать извлеченные данные о намерении и сущностях,
чтобы сделать фактический заказ для пользователя



Задачей, обратной для понимания ЕЯ, является задача генерации естественного языка. Генерация ЕЯ (natural language generation, NLG) – это процесс преобразования данных в формат, понимаемый компьютерами (такой как числовые данные, хранящиеся в базе данных), в текст на естественном языке.

Например, сравнение между данными о погоде на протяжении двух дней подряд для заданного города может привести к генерации следующего текста:

Погода в Нью-Йорк сегодня умеренно жаркая и на 3 градуса Цельсия выше, чем вчера. Хотя дождя не ожидается, в течение дня может ощущаться небольшая влажность.

В отличие от понимания естественного языка, генерация ЕЯ не является ИИ-полной задачей (полной задачей искусственного интеллекта) вследствие того, что процесс генерации языка является управляемым, где синтезирование предложений находится под управлением фиксированного набора формальных грамматических правил. Следовательно, в этих правилах существует нулевая неопределенность.

ИСТОРИЯ РАЗВИТИЯ МЕТОДОВ ПОНИМАНИЯ ЕСТЕСТВЕННОГО ЯЗЫКА



История развития методов понимания ЕЯ почти так же стара, как и сама область искусственного интеллекта. В 1964 г. аспирант докторантury Дэниел Боброу из Массачусетского технологического института (MIT) для своей докторской диссертации создал компьютерную программу под названием STUDENT. Эта программа была написана на языке программирования Lisp и была разработана, чтобы решать алгебраические задачи из учебной программы средней школы.

Программа STUDENT считается одной из самых ранних известных попыток понять естественный язык.

Естественно возникает вопрос «Что подразумевается под пониманием компьютером естественного языка?» Я принял следующее рабочее определение понимания. Компьютер понимает подмножество английского языка, если он принимает входные предложения, являющиеся членами этого подмножества, и правильно отвечает на вопросы на основе информации, содержащейся в этих предложениях. Эта способность должна распространяться на выводы, опираясь на неявную информацию, содержащуюся в нескольких предложениях. Желательно, чтобы ответы также были на английском языке, чтобы облегчить связь между компьютером и человеком.

— «Ввод на естественном языке
для компьютерной системы решения задач»,
Дэниел Боброу, 1964

Программа STUDENT могла давать ответы на алгебраические вопросы, выраженные в виде английских предложений, и предоставлять ответы на английском языке. Вопрос типа «The sum of two numbers of 96, and one of the numbers is 16 larger than the other number. Find the two numbers» (Сумма двух чисел равняется 96, и одно из чисел на 16 больше другого числа. Найдите эти два числа) приводил к ответу «One of the numbers is 56. The other number is 40» (Одно из чисел равняется 56. Другое число равно 40).

Возможно, самый известный пример применения методов понимания ЕЯ представлен компьютерной программой ELIZA, созданной Джозефом Вайценбаумом между 1965–1966 гг. в Лаборатории Искусственного интеллекта MIT. Программа ELIZA была интерактивным чат-ботом. Этот робот мог быть запрограммирован для ведения беседы с пользователем по любой теме на английском языке. Самой популярной темой была психотерапия, реализованная в сценарии под названием DOCTOR и симулирующая психотерапевта. Рисунок 4.3 показывает разговор между программой ELIZA и пользователем.

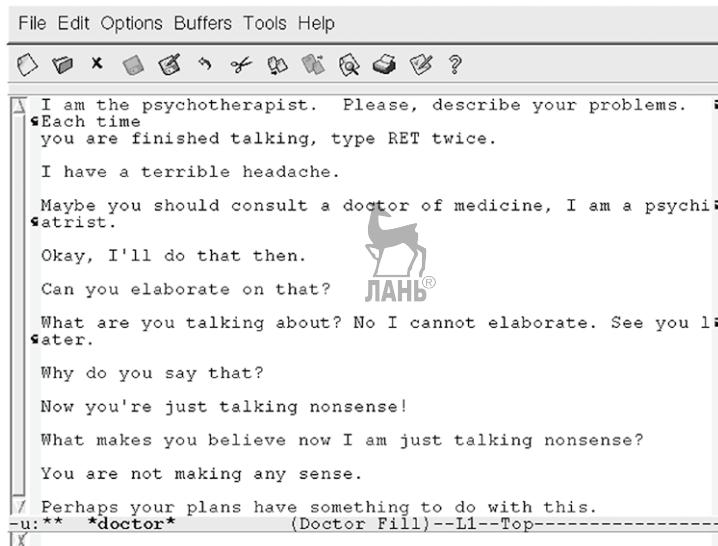


Рис. 4.3 ♦ Программа ELIZA, выполняющая сценарий DOCTOR

В программе ELIZA использовались подстановка и методы сопоставления с образцом, которые создавали иллюзию понимания. Хотя Вайценбаум создал программу ELIZA, чтобы продемонстрировать искусственность общения между человеком и машиной, она имела совершенно противоположный эффект среди многих пользователей, которые чувствовали, что она проявляла человекоподобное понимание их задач. Однажды, когда Вайценбаум находился в отъезде, его секретарша воспользовалась возможностью, чтобы тайно поговорить с программой ELIZA. Вернувшись, Вайценбаум увидел, как его секретарша покидала офис в слезах. Позже, когда он сказал ей, что это была всего лишь компьютерная программа и что она в действительности не понимала ее чувств, она отказывалась верить, что «друг», с которым она разговаривала, был не более, чем запрограммированной машиной.

Еще одной невероятно успешной демонстрацией понимания естественного языка и искусственного интеллекта в целом явилась программа SHRDLU Терри Винограда. Профессор Виноград создал свою программу в 1970 г. в Массачусетском технологическом институте (MIT). Программа SHRDLU (произносится как «шер-ду-лу») представляла собой анализатор естественного языка, который был ограничен миром блоков. Иными словами, она могла понимать интерактивный диалог на английском языке, чтобы отвечать на вопросы и выполнять команды, имеющие отношение к виртуальному миру,льному геометрических фигур в виде блоков, таких как кубоиды, кубы, конусы и т. д. Например, человек мог попросить, чтобы программа переместила блок из одного места в другое на обыденном разговорном английском языке, и программа отвечала, выполняя требуемое действие, используя виртуальную роботизированную руку. Рисунок 4.4 является иллюстрацией выдуманного мира блоков программы SHRDLU.

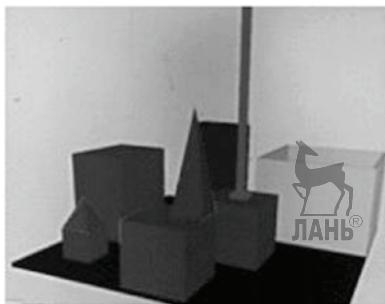


Рис. 4.4 ♦ «Мир блоков» программы SHRDLU

Ниже приводятся выдержки из диалога из демонстрационного выполнения программы Винограда. Обратите внимание, насколько по-человечески программа SHRDLU реагирует с учетом симуляции понимания синтаксиса, семантики, грамматики и контекста.

Человек: ВЫБЕРИ БОЛЬШОЙ КРАСНЫЙ БЛОК.

Компьютер: ХОРОШО. (выполняет команду)

Человек: ВОЗЬМИ ПИРАМИДУ.

Компьютер: Я НЕ ПОНИМАЮ, КАКУЮ ПИРАМИДУ ВЫ ИМЕЕТЕ В ВИДУ.

Человек: НАЙДИ БЛОК, КОТОРЫЙ ВЫШЕ, ЧЕМ ТОТ, КОТОРЫЙ ТЫ ДЕРЖИШЬ, И ПОМЕСТИ ЕГО В КОРОБКУ.

Компьютер: ПОД СЛОВОМ «ЕГО» Я ПОНИМАЮ, ЧТО ВЫ ИМЕЕТЕ В ВИДУ БЛОК, КОТОРЫЙ ВЫШЕ, ЧЕМ ТОТ, КОТОРЫЙ Я ДЕРЖУ.

Компьютер: ХОРОШО. (выполняет команду)

Человек: ЧТО СОДЕРЖИТ КОРОБКА?

Компьютер: СИНЮЮ ПИРАМИДУ И СИННИЙ БЛОК.

...

Человек: НАЗОВИ САМЫЙ БОЛЬШОЙ БЛОК «СУПЕРБЛОКОМ».

Компьютер: ХОРОШО.

Человек: ТЫ ВЫБРАЛ СУПЕРБЛОК С ТЕХ ПОР, КАК МЫ НАЧАЛИ?

Компьютер: ДА.

Человек: ПОЧЕМУ ТЫ ЕГО ПОЛОЖИЛ?

Компьютер: ПОД СЛОВОМ «ЕГО» Я ПОНИМАЮ, ЧТО ВЫ ИМЕЕТЕ В ВИДУ СУПЕРБЛОК. Я НЕ МОГУ ОБЪЯСНИТЬ НЕСУЩЕСТВУЮЩЕЕ СОБЫТИЕ.

Человек: ЕСТЬ ЛИ ЧТО-ТО СПРАВА ОТ КРАСНОЙ ПИРАМИДЫ?

Компьютер: ДА, ЧЕТЫРЕ ПРЕДМЕТА: КОРОБКА, КРАСНЫЙ КУБ, СИНЯЯ ПИРАМИДА И БЛОК ПОД НАЗВАНИЕМ СУПЕРБЛОК.

Человек: СПАСИБО.

Компьютер: ПОЖАЛУЙСТА!

Программа SHRDLU, какой бы она ни казалась на вид интеллектуальной, была простой программой с большим количеством ограничений. В свое время она была одной из самых продвинутых программ на основе искусственного интеллекта. Она вдохновила разработчиков на дальнейшие исследования в области понимания естественного языка, что в конечном итоге оказалось гораздо сложнее, чем считалось ранее. Вскоре после этого профессор Виноград переехал в Стэнфорд, где он создал Группу человека-компьютерного взаимодействия и консультировал несколько высококлассных компаний, организованных своими студентами, в том числе Google.

Коммерческий интерес к пониманию естественного языка увеличился в течение 1980-х годов. Этот интерес подстегнул Гэри Хендрикс, когда в 1982 г. организовал корпорацию Symantec для разработки естественно-языкового интерфейса для организации запросов к базе данных. Напомним, это та же самая компания Symantec, которая сегодня выпускает популярный антивирус и защитное программное обеспечение. Symantec – это слово-гибрид из слов синтаксис, семантика и технологии.

В начале 2000-х гг. компьютер Watson фирмы IBM стал одним из первых когнитивных суперкомпьютеров, который проявил удивительные способности понимания языка. Но только совсем недавно мы стали свидетелями прорывов в области понимания ЕЯ благодаря методам глубокого обучения, в результате которых появились крупнейшие технологические компании, предлагающие облачные службы для интеграции возможностей понимания естественного языка в любом приложении. Microsoft LUIS – это одна из таких облачных служб. Кроме того, также имеются службы компаний Google и Facebook.

Почему машины с трудом понимают естественный язык

Этот раздел дополнительный. Вы можете его спокойно пропустить, если вам не терпится применить методы понимания ЕЯ в своем приложении. Но мы рекомендуем вам просмотреть подраздел «Сложности естественного языка», потому что он может подсказать вам возможные пути решения, когда ваш процесс тренировки программы пониманию ЕЯ не будет работать, как ожидалось.

Сложности естественного языка

Идеальный анализ предложения на естественном языке будет включать проверку его синтаксиса (грамматики), семантики (значения) и прагматики (контекста). Вероятно, вы знаете, что такое синтаксис и семантика, но не разбираетесь в прагматике. Давайте объясним на примерах.

Синтаксис имеет дело с грамматической структурой предложения, такой как подлежащее-сказуемое-дополнение, или субъект-глагол-объект: «Мальчик играет с шаром». Нетрудно создать программу, использующую грамматические правила языка, которая извлекает структуру предложения, т. е. определяет субъект, глагол и объект. В реальном мире люди не всегда пишут или высказывают предложения, которые были бы на 100% грамматически правильными. Чем больше мы тренируем программу с элементами понимания естественного языка, используя реальные предложения, тем лучше она способна определять даже грамматически неправильные структуры.

Неправильный синтаксис: самые вкусные блюда, которые мы отведали в ресторане, *была* паста и радиоли.

Правильный синтаксис: самые вкусные блюда, которые мы отведали в ресторане, были паста и равиоли.

Семантика имеет дело со значением слов и словосочетаний в предложении и с тем, как они связаны друг с другом. Язык – это инструмент выражения мыслей или эмоций. Синтаксис можно представить как универсальные правила применения языка для общения с другим человеком: правильно примененная последовательность слов, чтобы другой человек мог вас ясно понять. Семантику же можно представить как значение ваших слов, фактическую мысль, которую вы хотите передать. Другой человек может быть в состоянии понять вашу мысль, даже если последовательность слов не согласуется с универсально принятыми стандартами на 100%, но вам не удастся передать свое сообщение, если другой человек не сможет ясно истолковать значение ваших слов.

Например, яблоко – это красный съедобный предмет круглой формы, тогда как манго – это желтоватый, душистый съедобный предмет овальной формы. Если вы хотите передать сообщение о том, что вы едите яблоко, то это можно сделать десятью различными способами, все из которых могут быть синтаксически правильными. Но выбор неправильных слов для передачи вашего сообщения полностью изменяет его значение. Употребление в пищу манго – это НЕ одно и то же, что употребление в пищу яблока, ни для вас, ни для другого человека и ни для кого-либо еще на Земле.

A: Я люблю яблоки.

B: Я люблю есть яблоки.

Предложения А и В имеют одинаковое значение.

A: Мне нравятся яблоки.

B: Мне нравятся манго.

Предложения А и В имеют разные значения.



Мы, люди, знаем, что многие слова могут означать одно и то же: хороший, приятный, великолепный, замечательный и т. д. Мы называем их синонимами. Кроме того, еще есть антонимы, омонимы, гиперонимы и т. д. Семантика охватывает все эти понятия, поскольку они непосредственно затрагивают значение предложения. Идеальная программа, понимающая ЕЯ, может безупречно понимать семантику предложения. Однако решение этой задачи не является тривиальным в искусственном интеллекте.

Прагматика связана с тем, как контекст вносит свой вклад в общее значение. Контекст может основываться на говорящем, слушателе или окружающем мире. Обычно человек передает сообщение косвенно, в результате чего его истинное значение может быть выведено логически, а не истолковано непосредственно. Представьте, что вы разговариваете с товарищем на оживленном вокзале.

Вы: Смотри, кто-то уронил свой кошелек на пол.

Товарищ: Вон тот человек, кажется, что-то ищет.

Исходя из этого разговора, людям нетрудно разобраться, что человек, на которого указывает ваш товарищ, является тем, кто, по его мнению, уронил свой кошелек, несмотря на то что ваш товарищ явно ничего об этом не говорит. В таких

разговорах смысл высказывания – в данном случае высказывания вашего товарища – подразумевается посредством контекстной информации. Как отдельное предложение, высказывание вашего товарища означает немного. Но с контекстом оно получает совершенно новое значение. Идеальная программа, понимающая ЕЯ, должна не только понимать значение отдельных предложений, но и учитывать контекст вокруг них, чтобы точнее истолковывать значения.

Сложности не заканчиваются синтаксисом, семантикой и прагматикой. Вот неисчерпывающий список других элементов, которые должна учитывать идеальная программа с элементами понимания естественного языка:

- слова, которые имеют многочисленные значения (сравните «вы правы» и «поворачивать направо»);
- предложения, которые требуют понимания окружающего мира («выбери второй блок» – второй по отношению к чему?);
- неопределенные предложения («Я опоздал на встречу» – на сколько опоздал?);
- предложения, которые можно истолковать более чем одним способом. Например: «Я видел человека на холме с телескопом». Речь о человеке на холме, на котором был установлен телескоп. Или я видел человека на холме, который нес свой телескоп?

И затем следует учитывать значительную разницу в письменном и устном языке. Когда люди пишут электронные письма, эссе, статьи, рассказы и т. д., они склонны соблюдать формальности и грамматические правила. Однако когда они говорят друг с другом, они проявляют тенденцию выражаться неформальным языком, не особо ограничивая себя правилами. Сравните предложение: «Я собираюсь это закончить к следующей неделе» с предложением «Сделаю это к следующей». К этому еще следует добавить проблему диалектов (локальных вариантов языка) и жаргонизмов (неформальные слова). У одного только английского языка имеются десятки диалектов, главными из которых являются британский, американский и австралийский варианты английского языка.

Мы надеемся, что вы теперь лучше оцените сложные проблемы, возникающие при проектировании идеальной программы с элементами понимания естественного языка.

Статистические модели в качестве решения недостаточны

Самые ранние попытки понимания естественного языка, в особенности SHRDLU, довольно точно справлялись с задачей понимания. Они были построены с использованием лингвистических правил синтаксиса и семантики. Однако лингвистические данные были не единственными элементами, наделявшими их глубоким пониманием. Эти ранние системы были сосредоточены на ограниченных, малых мирах и поэтому могли понимать команды, имеющие отношение к конкретным областям. Программа ELIZA показывала хорошие результаты только в психотерапии. Весь мир программы SHRDLU состоял всего лишь из нескольких геометрических блоков. Простой логики на основе подстановки и сопоставления с образцом было достаточно, чтобы придать им кажущуюся способность понимать своих пользователей. Вместе с тем такую простую логику нельзя было просто применить при проектировании мощных и универсальных систем понимания ЕЯ.

1990-е годы стали свидетелями революционных изменений в обработке ЕЯ. Исследователи смогли использовать статистику как более подходящий способ понимать естественные языки. В машинном обучении до статистической революции обычно использовались такие методы, как деревья решений. Такие системы приводили к многосложным правилам «если – то», которые было трудно поддерживать и отлаживать. Канун XXI века засвидетельствовал появление сложных статистических моделей, которые были способны анализировать большие объемы информации, содержащейся в текстах, и получать результаты с максимально возможной точностью.

Эти модели положили начало статистическим методам машинного обучения, которые могли использоваться для понимания естественного языка в более общем плане. Большие вычислительные мощности, которые требовали такие статистические модели, стали теперь доступны частично благодаря закону Гордона Мура.

Первоначально тренировка текстовых данных (корпусов текстов) осуществлялась вручную. Позднее в странах Европейского союза в качестве корпусов текстов для тренировки систем понимания естественного языка стали использовать общедоступные документы правительенного делопроизводства.

В главе 1 вы познакомились с двумя главными подходами к машинному обучению: обучением с учителем и обучением без учителя. Обучение с учителем означает, что машина учится понимать текст, речь или изображения благодаря меткам (или аннотациям), вносимым людьми вручную. В обучении без учителя машина обучается самостоятельно посредством переработки мегатонн обучающих данных. На данный момент обучение с учителем, по сравнению с обучением без учителя, приводит к более точным результатам. Однако, несмотря на все достижения, которые были достигнуты в области искусственного интеллекта, даже с учетом того факта, что это обучение с учителем, чисто статистическое машинное обучение имеет большие резервы для совершенствования.

Статистические модели пытаются решить проблему обработки ЕЯ посредством анализа на основе грамматики зависимостей или словарных векторов.

Во времена анализа на основе грамматики зависимостей в предложении определяются части речи (существительные, глагол, прилагательные, предлоги и т. д.), и между ними устанавливаются грамматические отношения. Это не то же самое, что и сквозное понимание значения предложения. Рисунок 4.5 показывает анализ предложения на основе грамматики зависимостей.

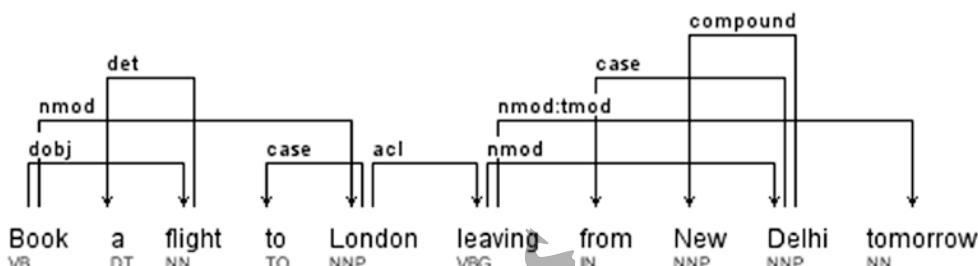


Рис. 4.5 ♦ Разбор предложения на основе грамматики зависимостей, сгенерированный с использованием Стэнфордского анализатора

Надпись на рисунке: Закажи билет на самолет в Лондон с отбытием завтра из Нью-Дели

Еще один подход к решению этой задачи представлен словарными векторами, где в предложении порождаются взвешенные векторы слов. Перси Лянг, доцент информатики в Стэнфордском университете, утверждает, что «словарные векторы верно улавливают некоторые аспекты семантики, но вместе с тем они представляют неполную картину».

i В случае если вам интересно, какова эффективность моделей глубокого обучения в области понимания естественного языка, напомним, что большинство методов глубокого обучения по своей сути являются статистическими. Рекурсивная нейронная сеть (recursive neural network, RNN) – это популярная модель глубокого обучения, которая особенно эффективна в приложениях машинного перевода. Нейросети RNN могут использоваться для построения моделей понимания ЕЯ, однако, чтобы произвести довольно точные результаты, они требуют огромной вычислительной мощности и невероятно больших объемов обучающих данных. А такие ресурсы имеются только в крупных корпорациях. Обратитесь к реализации алгоритма word2vec на основе Google Tensorflow и реализации fastText (который основан на мелких нейронных сетях) компании Facebook, оба из которых работают путем генерирования словарных векторов.



Многообещающее будущее

Мы увидели преимущества и недостатки ранних систем понимания естественного языка 1980-х годов и статистических моделей 1990-х и 2000-х годов. Научные исследования энергично пытаются достигнуть равновесия: точности (глубины) ранних систем и общности (широкоты) нынешних статистических систем.

Pat.ai представляет собой развивающийся проект, который претендует на то, чтобы стать уравновешенным решением задачи понимания ЕЯ. Джон Болл, когнитивист и основатель Pat Inc., в своей статье «Наука понимания ЕЯ» (The Science of NLU) пишет, что в то время как большинство решений полагаются сегодня на статистический анализ порядка и частоты слов, проект Pat.ai объединяет в себе теорию Patom (моделирующую функцию человеческого мозга) и грамматику ролей и ссылок (Role and Reference Grammar) (основанную на языковых моделях, разработанных в результате анализа разнообразных языков). Он утверждает, что эта комбинация теории мозга и лингвистики (а не статистики) представляет собой мощную смесь, которая позволила им интерпретировать истинное сквозное значение предложений. Он отделяет проект Pat.ai от существующих решений задачи понимания ЕЯ, обозначая его как *сопоставитель значений* (meaning matcher). Болл утверждает, что, в то время как существующие решения задачи понимания ЕЯ часто не понимают пространных высказываний, таких как «Позвони Бет, Нет Джон», Pat.ai с такой задачей успешно справляется.

В техническом отчете утверждается, что Pat.ai способен решить задачу схемы Винограда¹, т. е. теста с множественным выбором для проверки на искусственный интеллект, предложенный Гектором Левеском (Университет Торонто) и названный в честь Терри Винограда (он является создателем нашей любимой программы SHRDLU). Например, Pat.ai может сказать, к чему относится слово «они» в не-

¹ Схема Винограда – это пара предложений, которые различаются только одним или двумя словами и содержат ссылочную неоднозначность, разрешающуюся в противоположных направлениях в двух предложениях. – Прим. перев.

однозначном предложении: «Члены городского совета отказали демонстрантам в разрешении, потому что они опасались насилия».

В момент написания этой книги Pat.ai находился на этапе предварительного ознакомления, доступ к которому можно получить, подписавшись на частный список ожидания.



Облачная служба LUIS

LUIS – это облачная служба Microsoft понимания естественного языка и составная часть комплекта когнитивных служб (Cognitive Services). В ней используется схема «намерение – сущности» и несколько других структурных компонентов, позволяющих, применяя технологии понимания естественного языка, создавать мощные и функциональные приложения. С помощью простого веб-сайта можно создать приложение на основе облачной службы LUIS и обучить ее понимать все варианты взаимодействия, которые пользователи будут применять в вашем приложении. Приложение на основе облачной службы LUIS – это составное программное приложение, которое вы предоставляете своим пользователям. LUIS – это внешняя, облачная часть вашего приложения, установленного на вашем компьютере. Ваше приложение обращается к LUIS как облачной службе. Другими словами, ваше приложение использует онлайновый API, предоставляемый приложением LUIS.

Во время написания этой книги облачная служба LUIS поддерживала понимание естественного языка в девяти языковых культурах: на английском, китайском, французском, испанском, итальянском, немецком, японском, бразильском варианте португальского языка и корейском языках¹. В одном приложении LUIS нельзя использовать многочисленные языковые культуры. Культуру, выбранную во время создания приложения LUIS, нельзя изменять в дальнейшем. То, как облачная служба LUIS разбивает высказывания, в некоторой степени зависит от языковой культуры. Китайские и корейские высказывания разбиваются на символные лексемы, тогда как высказывания в любой другой культуре разбиваются на словарные лексемы.

Облачная служба LUIS может использоваться не только в текстоориентированных разговорах. Предусмотрены варианты применения, включающие распознавание речи. Служба LUIS беспрепятственно интегрируется в Microsoft API распознавания речи в составе комплекта когнитивных служб. Поэтому, вместо того чтобы заставлять ваших пользователей печатать сообщения на клавиатуре, вы можете сделать так, чтобы при помощи API распознавания речи они общались с вашим приложением устно, а облачная служба LUIS будет распознавать их речь. На внутреннем уровне речь перед отправкой в LUIS преобразуется в текст, при этом интеграция с API распознавания речи делает этот процесс плавным. Обработка речи будет рассматриваться в главе 7.

Как упоминалось ранее, каждое приложение на основе облачной службы LUIS предлагает конечную точку API, работающую по протоколу HTTP, как средство,

¹ На момент перевода данной книги (февраль 2018 г.) облачная служба LUIS уже поддерживала 12 языков. Были добавлены канадский вариант французского, мексиканский вариант испанского и датский языки. – Прим. перев.

позволяющее вашим приложениям реализовывать понимание естественного языка. Этот API принимает высказывание и возвращает его в виде разбиения на намерение–сущности. LUIS предлагает еще один API для программного управления приложением LUIS. Это RESTful-ориентированный API, который может использоваться для программного выделения списка намерений и сущностей, их добавления, редактирования и удаления, назначения меток, тренировки высказываний, публикации приложения и т. д. Это очень полезно в сценариях, где вы хотите создавать индивидуализированные интерфейсы для тренировки LUIS или не хотите предоставлять прямой доступ к вашей учетной записи LUIS.

Архитектура программного приложения на основе LUIS

Приложение на основе LUIS редко создается как всецело понимающий искусственный интеллект. Наоборот, проектирование приложения с использованием LUIS строится вокруг решаемой вашим программным обеспечением задачи (задач). Например, если ваше приложение создается для приема заказов еды, то часть вашего приложения, в котором применяется LUIS, следует натренировать на понимание высказываний, связанных с заказами еды, а не, скажем, на заказ авиабилетов. Каждое приложение LUIS сопровождается предварительно подготовленным намерением, и ни одно не может использоваться для обработки не относящихся к нему высказываний.

В качестве примера давайте рассмотрим приложение DefinitionApp, единственная цель которого состоит в том, чтобы предоставить определение слова или словосочетания. Даже такое простое приложение потребует возможностей понимания ЕЯ, чтобы правильно понимать тысячу способов, которыми пользователи могут предложить свои запросы.

«what is the definition of __» (каково определение)

«define __» (дай определение)

«tell me what is __» (скажи, что такое)

«dude, could help me understand the definition for __?» (чувак, помоги понять определение)

Этот список можно продолжать до бесконечности. Все вышеупомянутые высказывания отличаются по структуре, но похожи по значению, т. е. выражают намерение пользователей узнать определение  какого-то понятия. Это нечто, выраженное прочерком, является ключом; давайте назовем его «Phrase» (фразой).

Ниже приведены минимальные действия, которые необходимо предпринять, чтобы применить LUIS для приложения DefinitionApp:

1. Создать новое приложение в облачной службе LUIS, возможно, под таким же именем, что и у вашего прикладного программного приложения.
2. Добавить в приложение LUIS новое намерение под названием GetDefinition.
3. Добавить новую сущность под названием Phrase.
4. Под намерением GetDefinition добавить как можно больше различных вариантов высказываний, которые могут быть истолкованы как запросы на получение определения какого-то понятия.

5. В каждом высказывании подчеркнуть сущность `Phrase`. В случае «define sumptuous» (дай определение роскошный) слово «sumptuous» будет помечено как сущность `Phrase`.

6. Нажать кнопку Train (Натренировать), чтобы начать тренировку.

После того как ваше приложение LUIS будет натренировано на разбитие соответствующих высказываний на намерения и сущности, вы можете его опубликовать как конечную точку HTTP (API). Эта точка будет принимать высказывания из вашего прикладного программного приложения в виде строковых значений и отправлять назад разбиения «намерение–сущности» в виде документа JSON, которые будут выглядеть примерно так:

```
{
  "query": "define sumptuous",
  "topScoringIntent": {
    "intent": "GetDefinition",
    "score": 0.9820167
  },
  "intents": [
    {
      "intent": "GetDefinition",
      "score": 0.9820167
    },
    {
      "intent": "None",
      "score": 0.03545451
    },
    {
      "intent": "Hello",
      "score": 0.00300396071
    }
  ],
  "entities": [
    {
      "entity": "sumptuous",
      "type": "Phrase",
      "startIndex": 7,
      "endIndex": 15,
      "score": 0.851388454
    }
  ]
}
```

Эти данные JSON в значительной степени самоочевидны. Здесь надо отметить один момент: чем больше мы тренируем, тем степень уверенности больше. Об этом следует помнить. Так или иначе, но ниже мы упомянем об этом еще несколько раз.

Рисунок 4.6 является снимком экрана веб-интерфейса LUIS. Рисунок 4.7 является диаграммой архитектуры прикладного программного приложения на основе LUIS.

DefinitionApp

GetDefinition

Here you are in full control of this intent; you can manage its utterances, used entities and suggested utterances ... Learn more

Dashboard

Intents

Entities

Features

Train & Test

Publish App

Utterances (3) Entities in use (1) Suggested utterances

Type a new utterance & press Enter ...

Save Discard Delete Labels view (Ctrl+E): Entities Search in utterances

Reassign Intent

Utterance text Predicted Intent

what does [\$Phrase] mean 0.69 GetDefinition

what is the definition of [\$Phrase]? 0.81 GetDefinition

define [\$Phrase] 0.69 GetDefinition

Рис. 4.6 ♦ Приложение LUIS под названием DefinitionApp, которое доставляет определение слова или фразы. Намерение GetDefinition в настоящее время имеет три помеченных высказывания (utterances), но можно добавить столько, сколько потребуется. Для получения точных результатов высказывания, помеченные как намерение (intent), должны быть похожими или иметь одинаковое значение. \$Phrase является сущностью (entity), представляющей слово или словосочетание, для которых доставляется определение

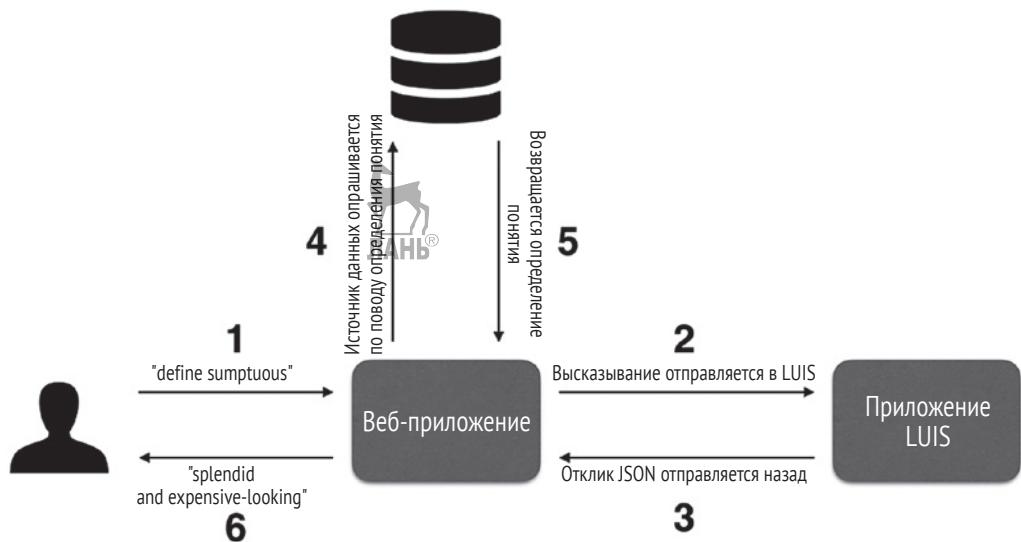


Рис. 4.7 ♦ Диаграмма последовательной архитектуры, иллюстрирующая взаимодействие между пользователем и веб-приложением с поддержкой LUIS, которое возвращает определение слова или словосочетания

Последовательность операций вашего веб-приложения может быть описана в следующих шагах.

1. Пользователь отправляет в веб-приложение сообщение «define sumptuous» (дай определение роскошный).
2. Веб-приложение передает сообщение как высказывание в соответствующее ему приложение LUIS.
3. LUIS выполняет свой анализ смысла естественного языка и возвращает свои предсказания в отношении намерения и сущности. Результат возвращается в виде данных JSON.
4. Веб-приложение извлекает из возвращенного документа JSON намерение с самой высокой степенью уверенности. Если оно помечено как `GetDefinition`, то отыскивает в документе JSON значение сущности `Phrase`. В вышеупомянутом примере это значение «sumptuous». Используя эту информацию, теперь приложение знает, что сделать. Оно должно обратиться к источнику данных, чтобы извлечь определение слова «sumptuous». Источник данных может быть базой данных, специально созданной для этого приложения, либо онлайновым API (таким как словарь или Википедия).
5. Источник данных передает краткое определение целевого слова/фразы назад в веб-приложение.
6. Веб-приложение представляет полученное определение пользователю и заканчивает свою работу.



За кадром

На данный момент вы, несомненно, убедились в том, что облачная служба LUIS применяет обучение с учителем, так как высказывание требуется помечать намерениями и сущностями до того, как служба LUIS сможет определять намерения и сущности в ранее встречавшихся высказываниях.

На внутреннем уровне облачной службы LUIS применяется два метода: для определения намерений используются логистические регрессионные классификаторы (LRC), а для определения сущностей – условные случайные поля (conditional random field, CRF). Как разработчику вам не требуется их знать. Но понять механизм их действия может дать вам представление о преимуществах и недостатках используемой технологии. LRC и CRF – это статистические модели: чем больше приложение LUIS тренируется на помеченных высказываниях, тем оптимальнее становятся эти модели. Более натренированные модели приводят к более точным предсказаниям, которые, в свою очередь, приводят к улучшению оценок уверенности в отношении намерений и сущностей.

Логистическая регрессия (LR) – это метод моделирования, используемый для генерирования предсказаний на основе ранее известных данных. Такие предсказания могут касаться прогноза погоды на завтра на основе анализа погодных данных за прошедшие семь дней или прогноза уровня младенческой смертности в регионе на основе анализа зарегистрированных случаев смертей среди младенцев за предыдущие годы и на основе существующих условий. Логистическая регрессия может также использоваться в качестве классификатора для предсказания классов тех или иных элементов данных на основе ранее известных классифицированных данных. Это именно то, что логистические регрессионные классификаторы делают с высказыванием: они предсказывают намерение на основе помеченного высказывания.

Условные случайные поля (CRF) – это вид моделирования последовательности, в котором слова классифицируются или помечаются тегами, после того как прияты в расчет соседние слова. Поэтому во время тренировки CRF-модели большую важность имеет порядок слов в предложении. Ранее в разделе «Сложности естественного языка» вы увидели, что значение слова может варьироваться в зависимости от контекста. Рассматривая соседние слова, CRF-модель помогает понимать контекст. CRF-модели обычно используются в частеречной разметке (part-of-speech, POS) и распознавании именованных сущностей (named-entity mapping, NEM), двух важных задачах обработки естественного языка.

При частеречной разметке предложение разбивается на синтаксические единицы. Например:

«Майк играет в футбол на соседнем стадионе»
становится



«Майк [СУЩЕСТВИТЕЛЬНОЕ] играет [ГЛАГОЛ] в футбол [СУЩЕСТВИТЕЛЬНОЕ] на [ПРЕДЛОГ] соседнем [ПРИЛАГАТЕЛЬНОЕ] стадион [СУЩЕСТВИТЕЛЬНОЕ]».

При распознавании именованных сущностей (NEM) из предложения извлекаются сущности на основе ранее помеченных сущностей. Например:

«Microsoft была соучреждена в 1975 Биллом Гейтсом»

получается

«Microsoft [КОМПАНИЯ] была соучреждена в 1975 [ВРЕМЯ] Биллом Гейтсом [ЧЕЛОВЕК]».



Масштабная тренировка – основа всему

Достижение службой LUIS того уровня понимания, которого вы ожидаете от своего приложения, требует выполнения тренировки, объем которой зависит от вариантов использования. Результатирующие данные JSON ранее показали, что для одного высказывания могут быть возвращены многочисленные конкурирующие намерения с разными уровнями уверенности. И это несмотря на точное высказывание, тренируемое относительно всего одного намерения. Работа LUIS основывается на статистических моделях, и такие модели являются вероятностными.

Если же вашему приложению требуется обрабатывать многочисленные варианты использования, которые похожи друг на друга, то при отсутствии достаточной тренировки по каждому из похожих вариантов служба LUIS окажется в тупике. Рассмотрим два следующих ниже высказывания, которые были помечены двумя разными намерениями:

BookFlight: «book me a flight from [\$Origin] to [\$Destination] leaving today»
(закажи мне рейс из [\$Origin] в [\$Destination] с отправлением сегодня).

BookReturnFlight: «book me a return flight [\$Origin] to [\$Destination]»
(закажи мне обратный рейс из [\$Origin] в [\$Destination]).

Эти высказывания тонко различаются по структуре и значению. При недостаточной натренированности LUIS может вернуть намерение BookFlight для втор-

рого высказывания, и наоборот. В качестве решения следует обрабатывать оба высказывания одним намерением (идеальное решение) либо предоставить достаточный объем обучающих данных, чтобы удалить из LUIS неоднозначность.

Облачная служба LUIS не может давать правильных результатов в 100% случаев, хотя при достаточной натренированности ее точность/прецизионность будет увеличиваться. Служба LUIS использует так называемое «активное обучение», которое позволяет ей самосовершенствоваться с течением времени. Она отслеживает все высказывания, для которых она не смогла предсказать намерение с высокой уверенностью. Вы найдете такие высказывания в категории **Suggested Utterances** (Предлагаемые высказывания) на странице **Intent** (Намерение). Используйте эту возможность, чтобы соответствующим образом помечать высказывания и давать LUIS подтверждения, что предсказание было верным или ложным.

Страница **Overview/Dashboard** (Обзор/панель мониторинга) в приложении LUIS показывает несколько ключевых статистических показателей и графиков, связанных с его тренировкой и использованием с течением времени. Эта информация полезна для тонкой настройки приложения в соответствии с тем, как пользователи его используют (виды высказываний, из которых они отправляют намерения чаще всего) и насколько однородной является натренированность (какой процент помеченных высказываний дошел до каждого намерения). Рисунок 4.8 показывает панели мониторинга для приложения DefinitionApp. В случае реально работающего приложения эта статистика будет намного плотнее.

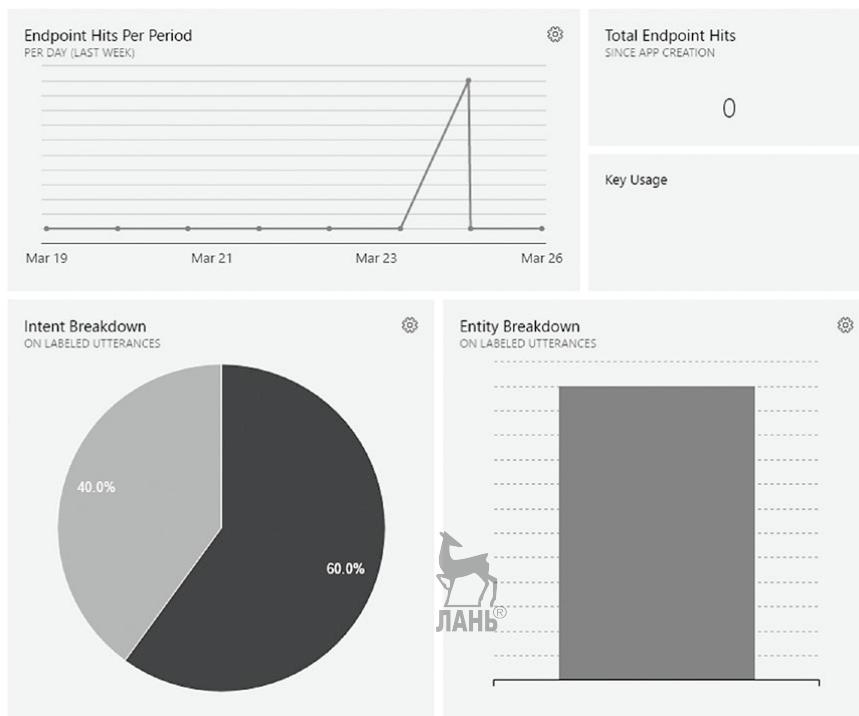


Рис. 4.8 ♦ Статистика для приложения DefinitionApp на основе LUIS

В главе 6 вы узнаете о структурных компонентах приложения LUIS, таких как намерения, сущности, списки фраз и т. д. Вы научитесь разрабатывать интеллектуальные разговорные интерфейсы с использованием LUIS в приложении Bot Framework. Но прежде чем вы сможете начать использовать LUIS в своем приложении, вам потребуется ключ подписки Azure.

ПОЛУЧЕНИЕ ПОДПИСКИ AZURE ДЛЯ ОБЛАЧНОЙ СЛУЖБЫ LUIS

Давайте создадим новую учетную запись когнитивных служб на портале Azure для облачной службы LUIS, получим ключи подписки и применим их к вашей учетной записи LUIS на <https://www.luis.ai>.

Получение ключей подписки на портале Azure

1. Откройте портал Azure. Если вы забыли, портал находится по адресу <https://portal.azure.com>.
2. Выберите из левого бокового меню **Создать ресурс > AI + Cognitive Services > Language Understanding**.
3. Заполните форму, как показано на рис. 4.9. Следует выбрать API интеллектуальной службы понимания языка (LUIS). Выберите вид подписки и ценовую категорию согласно своим потребностям. На начальной стадии можно выбрать бесплатную пробную версию и ценовую категорию F0. Позже всегда можно ее обновить до платной ценовой категории.
4. Отправьте форму, нажав кнопку **Создать**. Для создания своей учетной записи когнитивных служб может потребоваться от нескольких секунд до одной минуты. Процесс регистрации можно отследить в меню **Уведомлений** в правом верхнем углу окна.
5. После создания учетной записи выберите из левого бокового меню команду **Все ресурсы > <ваша-учетная-запись-LUIS> > Keys (Ключи)**. На экране появятся поля, в которых вы увидите ваши ключи (рис. 4.10). Запишите значение ключа 1 и ключа 2 в удобном для себя месте.

Применение ключей подписки в LUIS

1. Откройте веб-сайт LUIS, расположенный по адресу <https://www.luis.ai/>.
2. Авторизуйтесь, используя ту же учетную запись Microsoft (почтовый адрес и пароль), которую вы использовали на портале Azure. Если это ваша первая авторизация, то вам будет рекомендовано принять условия лицензионного соглашения и предоставить облачной службе LUIS разрешение получать доступ к вашей учетной записи Microsoft.
3. После авторизации и просмотра необязательного начального руководства перейдите на страницу **My Keys (Мои ключи)**.
4. Под заголовком My Keys вы увидите свой программный ключ API. Этот ключ НЕ является вашим ключом подписки. Он используется вместе с программным API облачной службы LUIS, о которой мы говорили ранее. Этот ключ позволяет вам управлять своими приложениями LUIS программным образом.
5. В разделе **Endpoint Keys (Ключи конечной точки)** нажмите кнопку **Add a new key (Добавить новый ключ)**. В появившемся диалоговом окне введите значение ключа 1, которое вы получили на портале Azure. Назовите этот ключ «Key1» (рис. 4.11).

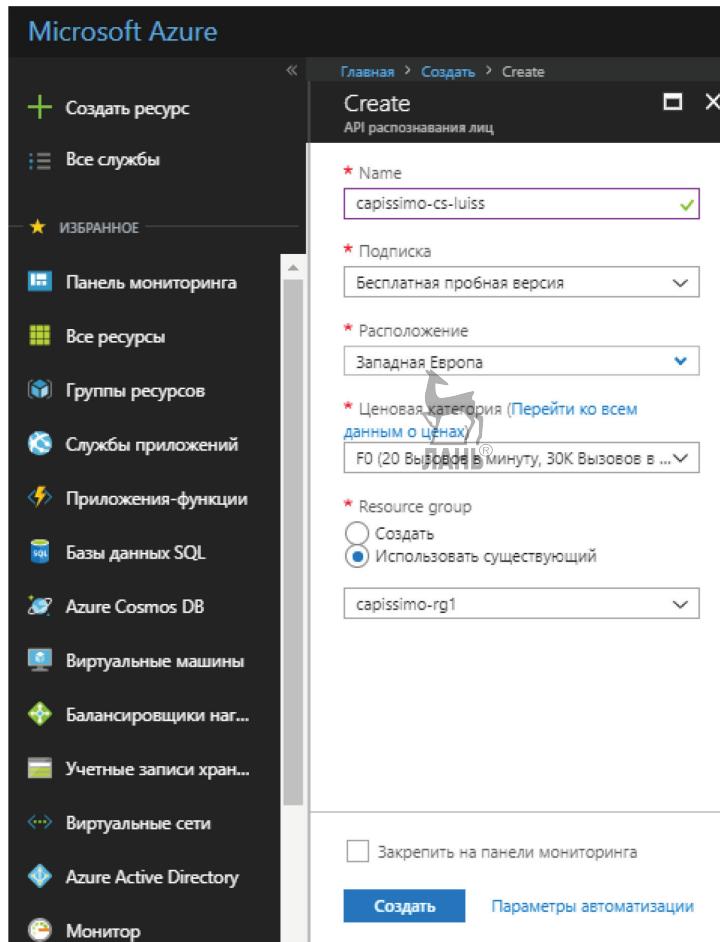


Рис. 4.9 ❖ Создание новой учетной записи Cognitive Services для LUIS на портале Azure

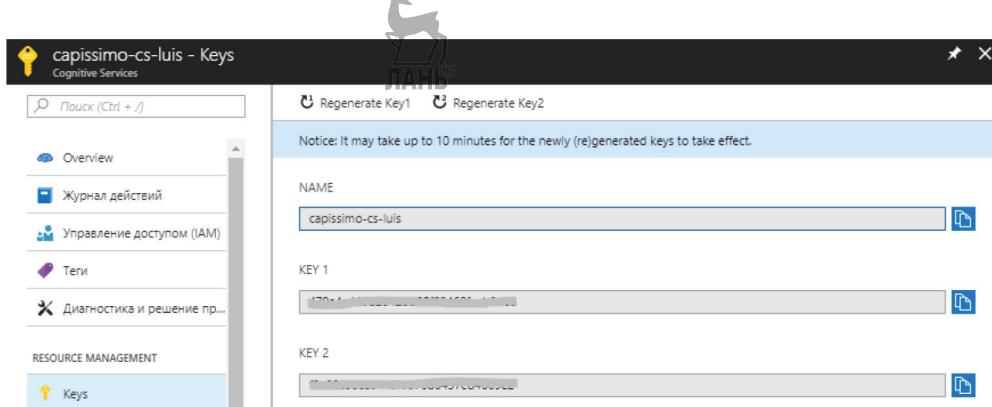


Рис. 4.10 ❖ Ключи подписки LUIS

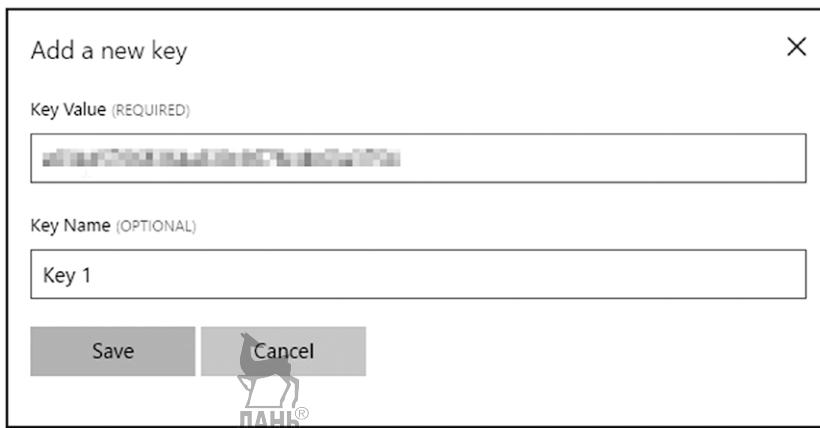


Рис. 4.11 ♦ Добавление ключа подписки в вашу учетную запись LUIS на <https://www.luis.ai>

6. Повторите пункт 5, чтобы добавить ключ 2.
7. На данном этапе все готово для создания новых приложений LUIS в нужном вам количестве. Если ваша учетная запись Azure для LUIS использует бесплатную ценовую категорию (F0), то ваше прикладное приложение может получать доступ к LUIS до 5 раз в секунду и в общей сложности до 10 000 раз в месяц.

Рядом с разделом Endpoint Keys имеется раздел External Keys (Внешние ключи). В этом разделе можно добавить ключ подписки на API Bing проверки орфографии, который тоже является составной частью комплекта когнитивных служб, чтобы позволить облачной службе LUIS автоматически выполнять проверку правописания.

ДЕМОНСТРАЦИОННЫЙ ПРИМЕР: ПРИЛОЖЕНИЕ DEFINITIONAPP

Мы закончим данную главу снимком экрана (рис. 4.12) и исходным кодом (листинг 4.1) вашего чрезвычайно простого приложения DefinitionApp. В главе 6 вы научитесь конструировать более продвинутые интеллектуальные разговорные приложения с использованием облачной службы LUIS и инфраструктуры Bot Framework.

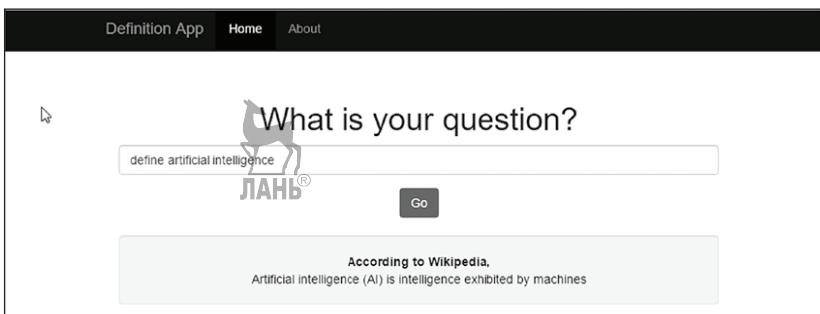


Рис. 4.12 ♦ Снимок экрана приложения DefinitionApp

Приложение DefinitionApp – это одностраничное веб-приложение, которое было создано с использованием приведенных ниже компонентов:

- Bootstrap (интерфейс пользователя, CSS);
- jQuery (AJAX-вызовы в ваше приложение на основе LUIS);
- Википедия (хранилище данных с определениями понятий);
- исходный код в листинге 4.1.

Листинг 4.1 ❖ Приложение DefinitionApp

HTML

```
-----
<div class="definition-app">
    <h1>What is your question?</h1>
    <form>
        <div class="form-group">
            <input type="text" class="form-control" id="utterance" placeholder='Eg.,>
            "define artificial intelligence'">
        </div>
        <button type="submit" class="btn btn-primary">Go</button>
    </form>
    <br>
    <div id="definition" class="well">Definition will appear here.</div>
</div>
```

JAVASCRIPT

```
-----
$(document).ready(function() {
    // Do this when the form is submitted;
    // that is, the "Go" button is clicked
    $('form').submit(function(e) {
        e.preventDefault(); // stop the default form submission behavior
        var utterance = $('#utterance').val(); // text entered by user in the textbox
        // Send the utterance to LUIS
        $.get("https://westeurope.api.cognitive.microsoft.com/luis/v2.0/
        apps/<appid>?subscription-key=<key>&timezoneOffset=0.0&verbose=true&q=" +
        utterance,
        function(responseLuis) {
            if (responseLuis.intents.length > 0) { // at least 1 intent was
                detected
                if (responseLuis.topScoringIntent.intent === "GetDefinition")
                    { // the top scoring intent is "GetDefinition"
                        if (responseLuis.entities.length > 0) { // at least
                            1 entity was detected
                            var phrase = responseLuis.entities[0].entity;
                            // ideally, we should check if entity type
                            is "Phrase"
                            // Get the definition from Wikipedia
                            var wikipediaData = {
                                "format": "json",
                                "action": "query",
                                "prop": "extracts",
                                "exintro": "",
                                "explaintext": "",
```



```

        "titles": phrase
    };
    $.get("https://en.wikipedia.org/w/api.php",
    wikipediaData, function(responseWikipedia) {
        var pages = responseWikipedia.query.
        pages;
        var summaryFirstLine = pages[Object.
        keys(pages)[0]].extract.split(". ")
        [0];
        $('#definition').html("<b>According
        to Wikipedia,</b><br>" +
        summaryFirstLine);
    });
}
else { // no entity was detected
    $('#definition').text("Sorry, I could not
    detect the phrase.");
}
}
else if (responseLuis.topScoringIntent.intent === "None") {
// the top scoring intent is "None"
    $('#definition').text("Did you ask the right
    question?");
}
else { // no intent was detected
    $('#definition').text("Sorry, I did not understand.");
}
});
});
});

```

Примечания

- Прежде чем использовать приведенный выше исходный код, вам нужно создать свое собственное приложение LUIS.
- Веб-браузер ограничен политикой одного источника из соображений безопасности. Это означает, что ваш браузер будет блокировать вызовы двух внешних API: облачной службы LUIS и Википедии. Чтобы обойти эту проблему, мы рекомендуем вам использовать браузер Google Chrome с установленным расширением «Allow-Control-Allow-Origin». По поводу политики одного источника вы можете узнать, погуглив в Интернете.
- Пределы определений ограничиваются тем, что находится в Википедии. Если в Википедии отсутствует страница с названием, которое является значением вашей сущности Phrase, то ваше приложение не сможет показать определение. Как результат данное приложение не поддерживает определения всех слов словаря.
- В качестве упражнения добавьте в это приложение поддержку словаря. Совет: выполните поиск «арі словаря» или «dictionary arī» в Сети, и вы увидите много бесплатных API.

Резюме

В этой главе вы подробно узнали о технологии понимания естественного языка (NLU), истории ее развития, приложениях и текущем состоянии дел. Вы увидели, почему машины с трудом понимают естественный язык. Наконец, вы получили всестороннее понимание среди Microsoft LUIS, предназначеннной для обработки ЕЯ.

Подводя итоги, в этой главе вы:

- познакомились с задачей понимания естественного языка;
- познакомились с историей развития методов понимания ЕЯ;
- узнали о сложности естественных языков, которые не дают машинам надлежащим образом их понимать;
- познакомились с облачной службой Microsoft LUIS;
- получили общее представление приложения с поддержкой LUIS;
- узнали, каким образом в таких приложениях различные компоненты общаются между собой;
- узнали, как получить ключ подписки на LUIS на портале Azure;
- создали работающее веб-приложение с поддержкой LUIS.

В **следующей главе** вы познакомитесь с четырьмя API когнитивных служб из категории «Язык».



Глава 5

Исследование когнитивной модели языка



На данном этапе у вас должно быть ясное понимание процесса обработки естественного языка. До сих пор все ваше внимание было сосредоточено на изучении области искусственного интеллекта, занимающейся пониманием языка и связанных с этим понятий. Задачи, связанные с обработкой естественного языка, затрагивают гораздо больший спектр вопросов, чем было показано в кратком обзоре задач обработки ЕЯ по ходу изложения. Так что же еще можно ожидать от машин, выполняющих обработку естественных языков? Давайте посмотрим.

После завершения этой главы вы:

- познакомитесь со следующими четырьмя API обработки языка и лежащими в их основе понятиями и применением, помимо облачной службы LUIS:
 - API Bing проверки орфографии;
 - API анализа текста;
 - API веб-модели языка;
 - API лингвистического анализа;
- узнаете, как Microsoft и другие компании используют основанные на языке когнитивные модели в реальном мире;
- познакомитесь с подзадачами, которые предлагает каждый API;
- получите подробное объяснение форматов запроса и отклика для API каждой задачи;
- познакомитесь с существующими и новыми идеями применения, которые могут побудить к написанию вашего следующего интеллектуального приложения.

API BING ПРОВЕРКИ ОРФОГРАФИИ

«bok a ticket»
«что такой атеист»
«кто такой мартин лютер кинг»
«в поисках утраченного кавчега»

Допущение орфографических ошибок – это наиболее распространенные ошибки в поведении пользователя. Орфографические ошибки имеют различные формы и разновидности, от легких и неумышленных до небрежных и совершенно возмутительных.

К счастью, вы не создаете текстовый редактор или текстовый процессор с использованием проверки орфографии для исправления ошибок в документах. Рынок уже насыщен такими редакторами. Кроме этого, есть много других областей, где требуется проверка орфографии. По сути дела, проверка ошибок в тексте иногда является главным техническим требованием, чтобы программное обеспечение работало правильно.

Возьмем, например, службу поиска. Большинство веб-сайтов и мобильных приложений предлагает поле ввода поискового запроса в виде текста, в котором пользователи могут напечатать все, что угодно и как угодно. При введении поискового запроса люди не ограничиваются какими-то правилами. Пользователи привыкли к качественным услугам поисковых систем, таких как Google и Bing, что ожидают такой же хорошей работы и от других веб-сайтов. Понимание службой поиска запросов, введенных в произвольной форме, сопровождается целым рядом дополнительных проблем. Например, нередки случаи, когда пользователи делают непроизвольные орфографические ошибки. Рассмотрим веб-сайт, позволяющий выполнять поиск кинофильмов.

Пользователь: «momento»

Веб-сайт: «Результаты для ‘momento’ не найдены».

Конечно же, веб-сайт не смог найти фильм с таким названием. «Momento» является часто встречающимся орфографическим искажением слова «Memento» (Помни). Это настоящее название популярного фильма, снятого в 1991 году Кристофером Ноланом. Стоит ли в действительности обвинять человека в том, что он допустил ошибку? Разумеется, нет! Звучное «нет» большими буквами! В свою очередь, стыдно, если веб-сайт не в состоянии обнаруживать такие элементарные орфографические ошибки. Подобных ситуаций можно полностью избежать при помощи простого орфографического корректора на основе словаря. А теперь почитайте следующий диалог:

Пользователь: «pursuit of happiness»

Веб-сайт: «Результаты для ‘pursuit of happiness’ не найдены».

Пользователь поклялся бы, что он написал каждое слово кинофильма Уилла Сmita, снятого в 2006 году, правильно. Так какие возникли затруднения? Настоящее название фильма, который ищет пользователь, – это «The Pursuit of Happyness» («В погоне за счастьем»). Проблема не с ключевыми словами поискового запроса пользователя, а с самим названием фильма. В слове *Happyness* имеется преднамеренное искажение слова *happiness*. Поэтому веб-сайт не смог выполнить поисковый запрос. Но в этом случае причина неудавшегося поиска не в орфографической ошибке, которую сайт не смог распознать. Причина – в том, что сайт не знал правильного названия фильма. И это не обычная проблема, поскольку мы говорим о создании более умного веб-сайта, который должен знать все названия фильмов.

Конечно, вы не хотите терять своих пользователей, потому что ваше приложение не умеет правильно выполнять поиск. Поэтому способность обрабатывать орфографические ошибки является основным техническим требованием. Веб-сайт, который не способен качественно обрабатывать поисковые запросы, обязательно потеряет свой наиболее ценный актив: своих пользователей. Пользователь, кото-

рый не может найти на веб-сайте то, что он ищет, не задумываясь, в одну секунду перейдет на другой веб-сайт, чтобы отыскать ответы там. А потерянный пользователь приводит к потере дохода.

Рисунок 5.1 показывает работу проверки правописания в [bing.com](#). Если бы в результатах поиска в Bing не принимались во внимание орфографические ошибки, задумались ли бы его пользователи о переходе в Google?

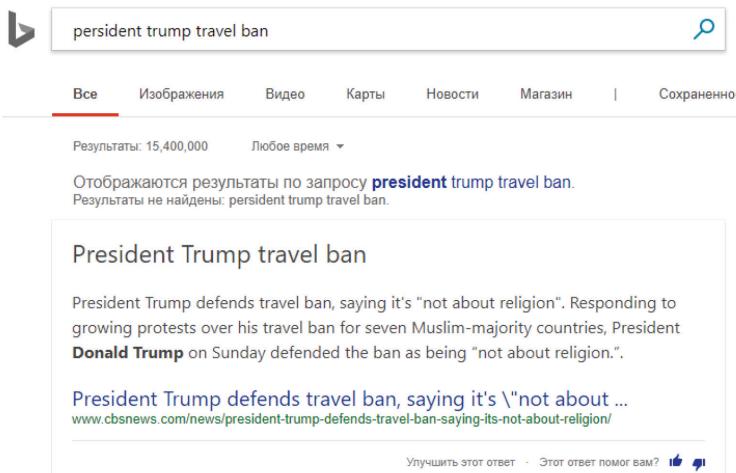


Рис. 5.1 ❖ Проверка правописания в Bing.com

Как можно забыть о разговорных интерфейсах? Единственный способ взаимодействовать с разговорными интерфейсами состоит в том, чтобы говорить с ними посредством текста или речи. Где присутствует разговор, обязательно произойдут орфографические ошибки. Приведенный ниже разговор с ботом можно по праву квалифицировать как один из самых постыдных прецедентов взаимодействия пользователя с программой.

Пользователь: Я неважно себя чувствую. Нужна помощь.

Медицинский бот: Какие симптомы?

Пользователь: Головная боль, боль в спине и знобит.

Медицинский бот: Вы принимали лекарство?

Пользователь: да. Парасетомол.

Медицинский бот: Извините, не понял.

Пользователь: Парасетомол.

Медицинский бот: Извините, не понял.

Пользователь вздыхает, затем думает.

Пользователь: Парасетамол.

Медицинский бот: Извините, не понял.

Пользователь закрывает приложение с ботом и вместо этого обращается за помощью к знакомым.

Что это такое?

API Bing проверки орфографии – это онлайновая служба проверки правописания, которая может просматривать фрагменты текста в поисках орфографических ошибок. При этом размеры текстового фрагмента значения не имеют. При обнаружении ошибки вам будет предложено несколько вариантов исправления. Варианты исправления будут показаны в порядке убывания уверенности. API пользуется многолетним опытом Bing в обработке пользовательских поисковых запросов и использует те же самые модели машинного обучения, что и в Bing. Эта служба является составной частью комплекта когнитивных служб Microsoft и, как вы можете предположить, умнее среднего орфографического корректора и делает намного больше, чем простая проверка орфографии.

Традиционные орфографические корректоры обнаруживают опечатки и предлагают варианты правильного написания, используя лежащие в их основе словари. Такие корректоры хоть и быстры, но малы по объему. Но они обычно не дают точного результата, могут распознавать лишь ограниченный набор имен собственных и обновляются лишь периодически с целью поддержки новых имен и слов. Эти корректоры обычно идут в комплекте с компактными приложениями, такими как веб-браузеры и текстовые редакторы.

Чуть более сложные корректоры могут принимать во внимание грамматические правила и даже дополнительно предлагать варианты исправления грамматических ошибок. Вы видели такие корректоры в текстовых процессорах, таких как MS Word, LibreOffice Writer и т. д. Они, как правило, точнее и крупнее по объему, чем простые корректоры на основе словаря. В остальном же к ним относятся все те же ограничения, что и у простых корректоров.

API Bing проверки орфографии работает на принципах машинного обучения. Как вы знаете, в случае достаточной натренированности модели машинного обучения систему можно научить понимать шаблоны (образы) данных. Это то, что происходит со всеми API Bing. Система Bing была запущена в 2009 году. Она не только была обучена ее создателями, но и все это время обучалась на анализе миллионов пользовательских запросов. API Bing предоставляет способ подключаться к обширной информации, которую эта поисковая система извлекла и еще извлечет. В главе 7 вы научитесь работать с API Bing обработки речи. Все другие API Bing (автоподстановки, поиска изображений, поиска новостей, поиска видео и поиска в Сети) будут рассмотрены в главе 8.

API Bing проверки орфографии выполняет контекстную проверку правописания в тексте. То есть этот API обнаруживает ошибки не на основе словарных правил, а на основе того, как и где слово используется в предложении. Рассмотрим следующее ниже предложение:

«Don't worry. I'll go home by Uber tonight»

(Не беспокойтесь. Сегодня вечером я поеду домой на Юбер).

Согласно правилам словаря, в этом предложении неправильно употреблено слово «uber». Слово Uber пришло из немецкого языка и означает «выше» или «над». В английском языке слово uber должно использоваться либо в качестве префикса (ubercool, uberstylish), либо в качестве прилагательного (uber intellectuals). Но спросите у любого человека, и он вам скажет, что слово, которое употреблено в предложении «I'll go home by...», относится к Uber, службе вызова такси. Обратитесь к на-

шему обсуждению темы прагматики в главе 4: контекстные ссылки в естественных языках настолько распространены, что полные предложения будут выглядеть не-натурально, а их постоянное использование будет утомительным.

Поэтому, в то время как другие корректоры пожалуются на использование слова *uber*, API Bing проверки орфографии не обнаружит никакой ошибки, потому что он будет учитывать контекст, в котором это слово употреблено. На самом деле он предостережет от неправильного употребления имени собственного *Uber*. См. приведенный ниже пример.

Пользователь: «I'll take an *uber* tonight».

Bing проверки орфографии: «I'll take an *uber* *Uber* tonight».

Мало того, что API Bing проверки орфографии распознает названия торговых марок, имена людей, названия географических мест и другие имена собственные, он также поддерживает сленг и разговорные выражения.

«Both kinda kinda look the same» (Оба типа того выглядят одинаково).

«Are you going to ch eck check the kitchen?» (Собираешься глянуть на кухне?)

«Clerk Clark Kent is Superman» (Служащий Кларк Кент – Супермен).

«The policemen had guns in there their hands» (У полицейских было там в руках оружие).

 На момент написания настоящей книги API Bing проверки орфографии поддерживал только английский (американский) текст. Поддержка большего количества английских локализаций, а также других языков находится в планах выпусков ближайшего будущего. На очереди находятся следующие локализации: британский вариант английского языка (*en-GB*), канадский вариант английского языка (*en-CA*), австралийский вариант английского языка (*en-AU*), Новозеландский вариант английского языка (*en-NZ*) и индийский вариант английского языка (*en-IN*) плюс китайский и испанский языки¹.

Как его использовать

Так же как и с другими API когнитивных служб, API Bing проверки орфографии основывается на механизме REST, к которому можно обратиться по URL-адресу. Но сначала вам нужен ключ подписки Azure. Обратитесь к описанию соответствующих шагов в главе 2 или 4, чтобы получить ключ подписки на API Bing проверки орфографии бесплатной ценовой категории.

 В этой главе мы не приводим исходный код на C# по каждой службе. Вместо этого мы подробно объясним структуру запроса и отклика каждой службы. При необходимости для вызова рассмотренных в этой главе служб вы можете повторно применить исходный код C# из главы 2. Однако мы настоятельно рекомендуем сначала воспользоваться клиентом REST API, таким как Postman, чтобы проверить различные комбинации параметров запроса и исследовать структуру отклика, получаемого из каждого API когнитивных служб.

Запрос

Конечная точка API Bing проверки орфографии активирована для GET- и POST-запросов. Различия между этими двумя методами незначительны. Метод GET

¹ На момент перевода настоящей книги (февраль 2018 г.) API Bing проверки орфографии поддерживал 40 языков, включая русский. – Прим. перев.

поддерживает больше вариантов и поэтому предпочтителен. Если нет возможности управлять форматом данных запроса, необходимо использовать метод POST (например, когда обрабатываются данные, полученные через HTML-форму).

Вот как выглядит образец GET-запроса:

```
GET /bing/v5.0/spellcheck/?mode=spell&mkt=en-us&text=whois persident trumpp HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Cache-Control: no-cache
```

URL-адрес конечной точки

<https://api.cognitive.microsoft.com/bing/v7.0/spellcheck/>

Все заголовки и параметры, кроме тех, которые отмечены звездочкой (*), являются необязательными.

Заголовки запроса

- Ocp-Apim-Subscription-Key*: этому ключу должен быть присвоен ваш ключ подписки Azure на API Bing проверки орфографии. Обратите внимание на то, что приведенный выше ключ в примере запроса используется только для демонстрации и недействителен.

Параметры запроса

- mode: может иметь всего два значения – Spell и Proof. API Bing проверки орфографии работает в двух режимах:
 - режим Spell оптимизирован для поисковых запросов и небольших предложений. Это обеспечивает быстрые и релевантные результаты;
 - режим Proof оптимизирован для длинных текстовых строк. Он аналогичен орфографическому корректору в MS Word.
- text*: текстовая строка, которая будет проверена на орфографические ошибки. В режиме Proof практически нет никаких ограничений на количество символов. В режиме Spell поддерживается до девяти слов.
- preContextText: хотя корректор понимает контекст автоматически, имеется возможность указывать контекст вручную, когда целевое слово или фраза известны заранее. Предварительный контекст preContextText представляет собой текст, который появляется перед целевым словом/фразой. Таким образом, «Stephen Spielberg» может быть действительным именем человека, но «Director Stephen Spielberg» обозначает популярного кинорежиссера Стивена Спилберга. Поэтому «Stephen» становится опечаткой при использовании с предварительным контекстом preContextText «Director».
- postContextText: этот параметр похож на preContextText, за исключением того, что он обеспечивает контекст после целевого слова/фразы, как, например, «inglorious bastards» по сравнению с «inglorious basterds movie» (к/ф «Бесславные ублюдки»).
- mkt: рынок, на который нацелено ваше приложение. Оптимизирует орфографию и проверку грамматики на основе целевого рынка. Он обнаруживается автоматически, но может быть проигнорирован вручную, если предоставить допустимое значение, такое как en-us, pt-br и т. д.

Пример POST-запроса выглядит следующим образом:

```
POST /bing/v5.0/spellcheck/?mode=spell&mkt=en-us HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache
Text=Bill+Gatas
```

Отклик

Давайте проанализируем отклик JSON, полученный из POST-запроса:

```
{
  "_type": "SpellCheck",
  "flaggedTokens": [
    {
      "offset": 0,
      "token": "whois",
      "type": "UnknownToken",
      "suggestions": [
        {
          "suggestion": "who is",
          "score": 1
        }
      ]
    },
    {
      "offset": 6,
      "token": "persident",
      "type": "UnknownToken",
      "suggestions": [
        {
          "suggestion": "president",
          "score": 1
        }
      ]
    },
    {
      "offset": 16,
      "token": "trumpp",
      "type": "UnknownToken",
      "suggestions": [
        {
          "suggestion": "trump",
          "score": 1
        }
      ]
    }
  ]
}
```

Свойства отклика

- `_type`: внутреннее свойство, которое представляет тип API когнитивных служб. Можно проигнорировать.

- flaggedTokens: массив всех обнаруженных орфографических ошибок, где каждая ошибка – это лексема, представленная объектом с дополнительными свойствами.
- offset: позиция первого символа в исходном тексте.
- token: слово с ошибкой. Значение этого свойства в комбинации со смещением offset используется для определения слова с орфографической ошибкой, которое следует заменить в исходном тексте.
- type: тип орфографической ошибки.
- suggestions: массив предложенных исправлений для слов с орфографической ошибкой. Каждый предложенный вариант сопровождается оценкой уверенности в диапазоне между 0 и 1. Неплохая идея отфильтровывать варианты с низкими оценками уровня.

Интеграция с облачной службой LUIS

Как было сказано в главе 4, API Bing проверки орфографии можно активировать в приложении с поддержкой LUIS. Рисунок 5.2 показывает, какой эффект это будет иметь на результат, полученный из LUIS.



```

{
  "query": "defnie rocket",
  "topScoringIntent": {
    "intent": "Hello",
    "score": 0.134179562
  },
  "intents": [
    {
      "intent": "Hello",
      "score": 0.134179562
    },
    {
      "intent": "GetDefinition",
      "score": 0.103685826
    },
    {
      "intent": "None",
      "score": 0.08644706
    }
  ],
  "entities": []
}

{
  "query": "defnie rocket",
  "alteredQuery": "define rocket",
  "topScoringIntent": {
    "intent": "GetDefinition",
    "score": 0.9508681
  },
  "intents": [
    {
      "intent": "GetDefinition",
      "score": 0.9508681
    },
    {
      "intent": "Hello",
      "score": 0.0463244952
    },
    {
      "intent": "None",
      "score": 0.0159818735
    }
  ],
  "entities": [
    {
      "entity": "rocket",
      "type": "Phrase",
      "startIndex": 7,
      "endIndex": 12,
      "score": 0.803182
    }
  ]
}

```

BEFORE

AFTER

Рис. 5.2 ♦ Результат JSON, полученный из приложения с поддержкой LUIS, до и после активации API Bing проверки орфографии

Чтобы в приложении с поддержкой LUIS активировать проверку орфографии, выполните приведенные ниже шаги.

- На домашней странице <https://www.luis.ai/home> перейдите на страницу My Keys.
- В разделе **External Keys** (Внешние ключи) нажмите кнопку **Add a new key** (Добавить новый ключ). Заполните форму, как на рис. 5.3.



Рис. 5.3 ♦ Добавление нового внешнего ключа в LUIS

- Теперь откройте приложение с поддержкой LUIS, в котором вам требуется активировать проверку орфографии. Перейдите на страницу **Publish App** (Опубликовать приложение).
- Прокрутите страницу вниз. Нажмите кнопку **Add key association** (Добавить связь с ключом), расположенную под разделом **External Key Associations** (Связи с внешними ключами), и выберите ключ, который вы ввели на шаге 2.
- Установите флажок напротив **Enable Bing spell checker** (Активировать корректор орфографии Bing) и нажмите **Publish** (Опубликовать). По поводу настроек вашего приложения обратитесь к рис. 5.4.

The screenshot shows the 'Publish App' interface. It includes fields for 'Endpoint url' (redacted), 'Add verbose flag' (checked), 'Enable bing spell checker' (checked), 'Timezone' (Western Europe Time, London), and 'Train' and 'Publish' buttons. Below is a table titled 'External Key Associations' with one row for 'BingSpellCheck'.

Key Type	Key	Actions
BingSpellCheck	[redacted]	[trash icon]

Рис. 5.4 ♦ Активация корректора орфографии Bing в приложении с поддержкой LUIS

API АНАЛИЗА ТЕКСТА

Данные или текст относятся к двум широким типам: структурированные и не-структурированные. Чтобы понять разницу между ними, давайте рассмотрим простой текстовый файл со следующим содержимым:

Я не верю в судьбу, которая настигает людей, что бы они ни делали; но я верю в такую судьбу, которая их настигает, когда они ничего не делают.

Такой текстовый файл, хранящийся в файловой системе, будет иметь метаданные, аналогичные приведенным ниже:

Имя: Quote.txt

Пользователь: Анураг Бандари



Дата создания: 01 апреля 2017

Дата изменения: 01 апреля 2017

Размер: 125 байт

Можете догадаться, какой из них является структурированным текстом: метаданные или содержимое файла? Если вы выбрали метаданные, то были правы. Метаданные имеют структуру: поля с именем, пользователем, размером и т. д. Эти метаданные повторяются для всех текстовых файлов; меняются только значения этих полей. Можете привести примеры других структурированных данных или текста? Подсказка: электронная таблица, база данных, журнальный файл.

С другой стороны, содержимое файла не структурировано; он может содержать любой текст, который существует под луной. Содержимое файла не имеет строгой формы или структуры. Этот факт сильно осложняет анализ неструктурированного текста. Предположим, что у вас 10 000 таких текстовых файлов и вам поручили программно найти размер файла, используя (a) метаданные и (b) содержимое файла. Даже ребенок может догадаться, что для такой задачи метаданные будут полезнее, и решение будет найдено быстрее.

Анализ неструктурированного текста – это именно то, чем занимается текстовая аналитика. Существуют сотни задач, решение которых сопряжено с анализом неструктурированного текста. По-настоящему трудные тексты нельзя эффективно решить с помощью традиционных программных алгоритмов. И вот тут на первый план выходит машинное обучение. API анализа текста когнитивных служб Microsoft предоставляет решения для четырех таких задач обработки естественного языка: идентификация языка, выделение ключевых фраз, анализ мнений и идентификация темы. Анализ текста в особенности полезен для точного определения реакций пользователя или потребителя на ваши продукты.

Идентификация языка

Рассмотрим представленный ниже разговор с чат-ботом.

Бот: Привет. Расскажите о своей проблеме, и я соединю вас с соответствующим отделом.

Пользователь: J'ai besoin d'aide pour la facturation.
(Мне нужна помощь по поводу счетов-фактур.)

Бот: Vous connecter avec notre expert en facturation francophone.
 (Перевожу вас на нашего франкоязычного эксперта по ведению счетов.)

Бот идентифицировал язык пользователя как французский и соединил его с франкоязычным оператором поддержки из отдела выписки счетов. Этот бот, конечно, был бы сногшибательным (и удобным) для многих пользователей благодаря своей способности работать на многочисленных языках. Такой бот имел бы большое преимущество над людьми, у которых способность понимать многочисленные языки не является данной от природы чертой.

При наличии слова, фразы или более крупного фрагмента текста API анализа текста может обнаружить их естественный язык с помощью HTTP-вызова. Чем длиннее фрагмент текста, тем точнее результат идентификации языка.

Функция обнаружения языка может быть полезной в десятках сценариев разговорного интерфейса. Она также может быть полезна, скажем, при определении первого, второго и третьего по популярности языков, используемых для отправки ответов в твит. Такая информация может быть неоценимой для компании, пытающейся понять, кто составляет аудиторию ее продукта.

Еще одним местом, где может пригодиться обнаружение языка, является поисковый механизм. Рисунок 5.5 показывает, каким образом Microsoft Bing его разумно использует, чтобы отфильтровать результаты поиска.

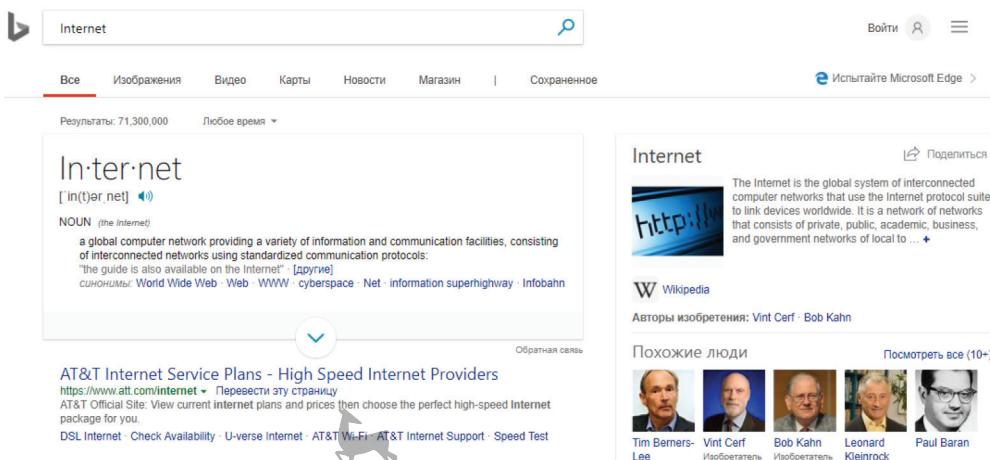


Рис. 5.5 ❖ Поисковая система Bing использует обнаружение языка,

чтобы отфильтровать результаты поиска в соответствии с обнаруженным языком.

Если интерфейсный язык (в данном случае русский язык) отличается от обнаруженного языка (в данном случае английского), то Bing также показывает резюме ключевых поисковых слов на обнаруженном языке

Запрос

API идентификации языка, выделения ключевых фраз и анализа тональности текста ожидают POST-запроса и используют общий JSON-формат тела запроса. Тело запроса в формате XML не поддерживается. Формат тела позволяет отправлять многочисленные фрагменты текста для пакетного анализа, при этом каждый фрагмент является «документом».

i Этот API начисляет 1 транзакцию в расчете на отправленный документ. Например, если вы запросите мнение (тональность) для 1000 документов в одном вызове, то будут вычтены 1000 транзакций. Это относится к API идентификации языка, выделения ключевых фраз и анализа тональности текста.

```
POST /text/analytics/v2.0/languages?numberOfLanguagesToDetect=5 HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/json
Cache-Control: no-cache

{
  "documents": [
    {
      "id": "1",
      "text": "Привет. Расскажите о своей проблеме, и я соединю вас с соответствующим
отделом."
    },
    {
      "id": "2",
      "text": "J'ai besoin d'aide pour la facturation."
    }
  ]
}
```

URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/analytics/v2.0/languages>

Все параметры, за исключением тех, которые помечены звездочкой (*), являются необязательными.

Параметры запроса

- **numberOfLanguagesToDetect**: необязательный параметр строки запроса для задания максимального количества языков для идентификации в расчете на документ. Это полезно для документов, которые содержат текст на нескольких языках. Это значение должно быть целым числом, чье значение больше или равно 1. Значение по умолчанию равняется 1.

Тело запроса

- **documents***: массив всех документов для анализа в одном вызове API. Максимальный размер одного документа, который можно передавать, составляет 10 КБ, и общий максимальный размер передаваемых входных данных составляет 1 МБ. В одном вызове можно передавать не более 1000 документов. Ограничение частоты запросов API установлено на уровне 100 вызовов в минуту. Поэтому рекомендуется передавать большое количество документов в одном вызове.
- **id***: строковое значение, которое однозначно определяет документ в списке. Оно же возвращается в отклике API вместе с обнаруженным языком. Разработчик может задать id по своему усмотрению при условии, что идентификатор каждого документа уникален. Возможно, самая простая схема идентификаторов будет состоять из 1, 2, 3, 4 и т. д. Для более сложных значе-

ний идентификаторов могут использоваться глобально-уникальные идентификаторы GUID.

- **text***: последовательность символов, которая должна быть проанализирована.

Отклик

```
{
  "documents": [
    {
      "id": "1",
      "detectedLanguages": [
        {
          "name": "English",
          "iso6391Name": "en",
          "score": 1
        }
      ]
    },
    {
      "id": "2",
      "detectedLanguages": [
        {
          "name": "French",
          "iso6391Name": "fr",
          "score": 1
        }
      ]
    }
  ],
  "errors": []
}
```



Свойства отклика

- **documents**: массив результатов по каждому документу, переданному в запросе.
- **id**: идентификатор документа, указанного в запросе.
- **detectedLanguages**: массив языков (или один язык), обнаруженных в тексте документа. Количество объектов в этом массиве может зависеть от параметра запроса `numberOfLanguagesToDetect`.
- **name**: полное название обнаруженного языка.
- **iso6391Name**: двухсимвольный сокращенный код языка, как определено организацией по стандартизации ISO. «en» для английского, «fr» для французского, «es» для испанского, «ru» для русского и т. д.
- **score**: уровень доверия обнаружения, где 0 – самый низкий и 1 – самый высокий.
- **errors**: если в переданном документе имеется ошибка (ошибки), то в этом массиве будет соответствующая ошибка.
- **id**: документ, в котором была обнаружена ошибка.
- **message**: подробное сообщение об ошибке.

i Следует уделить особое внимание при указании параметров и тела запроса. Неправильные значения в любом из них могут привести к ошибочному отклику JSON, который обычно будет иметь следующий ниже формат:

```
{ "statusCode": number, "message": string }
```

где statusCode – это код состояния HTTP и message – подробная строка сообщения.

Выделение ключевых фраз

Из того или иного длинного фрагмента беспорядочного текста иногда желательно извлечь только ключевые слова и фразы, главные темы, или «тезисы». Эта информация может использоваться, чтобы яснее понять контекст новостной статьи, записи в блоге или посте социальной сети. После того как известны главные темы текста, это знание можно применить с самыми разными целями.

Используя ключевые фразы, выделенные из многочисленных документов, можно генерировать облака слов для визуализации самых популярных ключевых фраз, присутствующих во всех документах. Рисунок 5.6 показывает облако слов, иллюстрирующее главные темы в отзывах пользователей в отношении Android-версии популярной игры для мобильных устройств Angry Birds Friends.



Рис. 5.6 ♦ Rovio, создатель франшизы Angry Birds, может увидеть, что пользователи главным образом говорят о power ups (бонусах), expensive (дороговизне), tournament (турнире) и т. д. Компания может использовать эту информацию в качестве неоценимой обратной связи, чтобы улучшить свой следующий релиз

Способность извлекать ключевые фразы аналогичным образом имеет чрезвычайное значение для цифровых рекламных сетей, таких как Google AdSense и DoubleClick. Такие сети используют извлеченные ключевые фразы из веб-страниц, на которых размещаются их объявления, чтобы показывать посетителям более релевантные объявления.

Запрос

```
POST /text/analytics/v2.0/keyPhrases HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/json
Cache-Control: no-cache

{
  "documents": [
    {
      "id": "1",
      "text": "Tesla will unveil an electric articulated lorry in September, chief executive Elon Musk has said. Additionally, he said an electric pick-up truck would be shown off in around 18-24 months.",
      "language": "en"
    },
    {
      "id": "2",
      "text": "La carta a Cristiano Ronaldo de la víctima de su presunta violación: 'Te dije no, te grité, rogué que parases'",
      "language": "es"
    }
  ]
}
```



URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/analytics/v2.0/keyPhrases>

Параметры запроса

Для этого API параметров нет.

Тело запроса

Тело запроса в формате JSON аналогично тому, которое вы видели в обнаружении языка. Дополнительным свойством в объекте документа выступает `language`. Свойство `language` является необязательным и должно указываться для неанглийских документов. В настоящее время этим API поддерживаются только английский (en), испанский (es), немецкий (de) и японский (ja) языки¹.

¹ На момент перевода настоящей книги (февраль 2018 г.) в API выделения ключевых фраз поддержка более 30 языков, включая русский, находится в состоянии пересмотра (см. <https://docs.microsoft.com/ru-ru/azure/cognitive-services/text-analytics/text-analytics-supported-languages>). – Прим. перев.

Отклик

```
{
  "documents": [
    {
      "keyPhrases": [
        "electric articulated lorry",
        "September",
        "chief executive Elon Musk®",
        "electric pick",
        "Tesla",
        "truck",
        "months"
      ],
      "id": "1"
    },
    {
      "keyPhrases": [
        "victima",
        "Cristiano Ronaldo",
        "carta",
        "presunta violación"
      ],
      "id": "2"
    }
  ],
  "errors": []
}
```

Свойства отклика:

- documents: список документов, передаваемых в запросе;
- id: уникальный идентификатор документа, указанный в запросе;
- keyPhrases: массив всех ключевых фраз, обнаруженных в соответствующем документе.

Анализ тональности текста

Анализ тональности текста – это процесс, используемый для определения, является ли тональность фрагмента текста положительной, нейтральной или отрицательной. Заданный текст анализируется с целью получения общей оценки тональности, где 0 означает крайне отрицательную тональность и 1 – крайне положительную.

Анализ мнений как инструмент получения отзывов о продукте использовался уже достаточно продолжительное время. Технологическая революция, связанная с большими данными, позволила крупным и малым компаниям подключиться к обширной информации с отзывами пользователей, доступной онлайн в форме прямых комментариев, постов/комментариев Facebook, постов в Twitter, блогов и т. д. Инструменты анализа больших данных и машинное обучение позволили выполнять экспоненциально ускоряющийся анализ больших объемов данных с целью определения критически важных откликов пользователей или клиентов на программное приложение или продукт.

Рисунок 5.7 показывает один из видов визуализации, который можно создать с помощью анализа мнений.

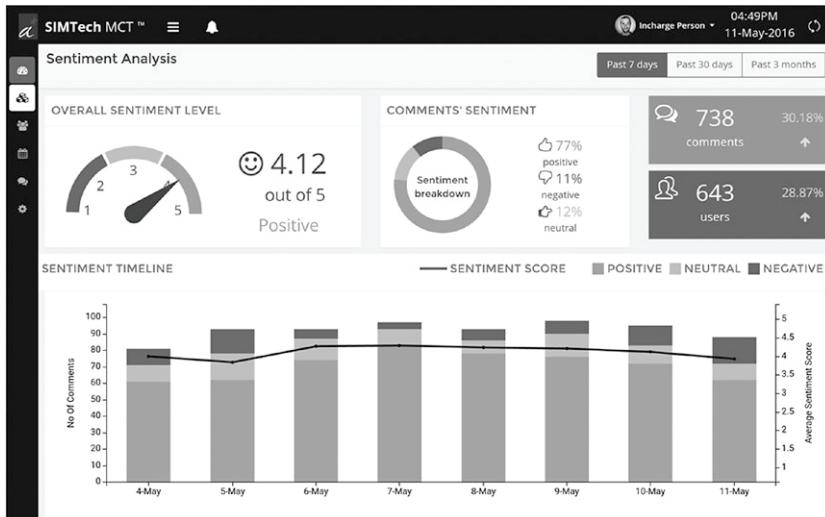


Рис. 5.7 ❖ Панель мониторинга, генерированная онлайновым инструментом SIMTech MCT

! Оценка тональности генерируется с использованием методов классификации. Признаки, подаваемые на вход классификатора, включают N-граммы, признаки, сгенерированные в результате частеречной разметки, и включенные слова. Классификатор был натренирован частично на основе данных Sentiment140. Поддерживаются английский (en), испанский (es), французский язык (fr) и португальский языки (pt)¹.

Запрос

```
POST /text/analytics/v2.0/sentiment HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/json
Cache-Control: no-cache

{
  "documents": [
    {
      "id": "1",
      "text": "You need to change it back! The new graphics are awful! Massive headache after a few minutes of play. Your graphic are what made candy crush different to other match 3 games now its a unplayable as the others. Im out till you fix it",
      "language": "en"
    },
  ]
}
```

¹ На момент перевода настоящей книги (февраль 2018 г.) в API анализа мнений поддержка более 30 языков, включая русский, находится в состоянии пересмотра (см. <https://docs.microsoft.com/ru-ru/azure/cognitive-services/text-analytics/text-analytics-supported-languages>). – Прим. перев.

```
{
  "id": "2",
  "text": "Great mental exercise and the variation of the levels keeps your interest.  
Never boring, just frustrating enough to make sure you come back to try again!"
}
]
```

URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/analytics/v2.0/sentiment>

Параметры запроса

Для этого API параметров нет.

Тело запроса

Здесь ничего необычного. Свойство `language`, как и ранее, необязательное.

Отклик

```
{
  "documents": [
    {
      "score": 0.0344527179662149,
      "id": "1"
    },
    {
      "score": 0.796423438202407,
      "id": "2"
    }
  ],
  "errors": []
}
```



Свойства отклика:

- `score` обозначает общую тональность текста документа. 0.03 – это весьма отрицательная тональность, в то время как 0.79 – довольно положительная.

Идентификация темы



Тема – это краткое, объемом в одну строку, резюме заданного текста. Это может быть одно слово или несколько связанных слов. Обнаружение темы не слишком отличается от выделения ключевых фраз. Так же как ключевые фразы, эти найденные слова обеспечивают главную тему или основной аргумент текста. Но вместо предоставления многочисленных ключевых фраз возвращается одна тема для всего текста.

i Этот API предназначен для работы с документами, содержащими краткий текст, такой как отзывы пользователей, комментарии и другие виды обратной связи. Данный API не будет хорошо работать с длинными текстами, такими как выдержки из книг, статьи и т. п.

Обнаружение темы имеет особую пользу, т. к. облегчает прочтение длинных отзывов покупателей о книге, кинофильме, приложении или продукте. Одна и та же тема может быть назначена нескольким текстам отзывов на основе их подобия. Обычно выделения ключевых фраз и анализа мнений достаточно, чтобы полу-

чить хорошее представление об общем мнении потребителей в отношении той или иной версии продукта. К обнаружению темы приходится обращаться дополнительно, чтобы подробно понять суть обратной связи, которую не могут обеспечить ключевые фразы.

Запрос

Этот API ожидает текстовых документов в формате, аналогичном другим трем API анализа текста. Правда, вызов API не столь прямолинеен. Обнаружение темы представляет собой емкий по времени процесс. Microsoft предоставляет вам право отправлять, по крайней мере, 100 текстовых документов в одном вызове API. Заданный предел максимального количества текстовых документов отсутствует, хотя максимальный размер 1 документа не может превышать 30 Кб, и максимальный размер всего тела запроса не должен превышать 30 Мб. Как напоминание каждый документ в вызове API считается одной транзакцией при расчете за оказанные услуги. Каждые 5 минут можно делать до пяти вызовов API.

Поскольку процесс обнаружения темы требует больших временных затрат, вы не получите ответ немедленно. Запрос к API передает документы в очередь и возвращает, с кодом состояния HTTP 202, URL-адрес, который должен периодически опрашиваться, чтобы извлечь результаты.

```
POST /text/analytics/v2.0/topics HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/json
Cache-Control: no-cache

{
  "documents": [
    {
      "id": "1",
      "text": "I love this product and have no complaints except that I cannot have 2 of them."
    },
    ...
    {
      "id": "100",
      "text": "The sound quality is not as good as my other speakers."
    }
  ],
  "stopWords": [
    "problem", "bug", "feedback"
  ],
  "topicsToExclude": [
    "create a ticket", "Amazon Echo"
  ]
}
```

URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/analytics/v2.0/topics>

Все параметры, кроме отмеченных звездочкой (*), являются необязательными.

Параметры запроса

- `minDocumentsPerWord`: целое число, задающее минимальное количество документов, для которых тематическое слово является одинаковым. Например, если назначить ему значение 2, то все тематические слова, которые присвоены всего 1 документу, будут исключены. Этот параметр помогает исключить редкие тематические слова.
- `maxDocumentsPerWord`: противоположность параметру `minDocumentsPerWord`. Используется для исключения очень общих тематических слов.

Тело запроса

Формат списка документов остается неизменным. Свойство `language` в объекте документа отсутствует, потому что поддерживается только английский язык. Документ, текст которого набран на языке, отличающемся от английского, будет проигнорирован. Таким же образом документы, длина которых меньше трех слов, будут проигнорированы.

- `stopWords`: список слов, которые не должны быть частью обнаруженных тем. Помогает исключить слова, которые широко распространены и выражены неявно в заданном сценарии. Например, если обнаружение темы будет выполнено на обзорах обратной связи, то такие слова, как `problem`, `bug`, `feedback` (проблема, системная ошибка, обратная связь) и т. д., не будут добавлять к значению темы. Кроме явно заданных стоп-слов, API также рассмотрит их форму во множественном числе.
- `topicsToExclude`: аналогично свойству `stopWords`. Позволяет указывать полные названия тем, которые будут исключены. Полные и частичные названия продуктов являются подходящими кандидатами для этого свойства.

Отклик

Непосредственный отклик на вышеупомянутый POST-запрос будет иметь пустое тело и код состояния HTTP 202. Он будет иметь заголовок, который приведен ниже:

```
'operation-location': 'https://westeurope.api.cognitive.microsoft.com/text/analytics/v2.0/operations/<operationId>'
```

Значение заголовка `operation-location` является URL-адресом конечной точки, которая в итоге будет возвращать результаты, когда они будут готовы. Время, требующееся для обработки запроса, зависит от количества и размера документов, передаваемых в первоначальном POST-запросе.

Этот URL-адрес должен периодически опрашиваться (рекомендуемая периодичность – один раз в минуту) с целью проверки результатов.

```
GET https://westeurope.api.cognitive.microsoft.com/text/analytics/v2.0/operations/<operationId>
```

Каждый раз вышеупомянутый вызов будет возвращать отклик в формате JSON, по крайней мере со свойством `status`, значение которого может быть `notstarted`, `running` или `succeeded` (обработка не начата, выполняется, выполнено успешно). Следует продолжать опрашивать до тех пор, пока вы не получите `succeeded` в качестве значения состояния.

Заключительный отклик будет иметь следующую форму:

```
{
    "status": "succeeded",
    "operationProcessingResult": {
        "topics": [
            {
                "id": "d8f62239-4ab5-4f95-8067-b2fca96d0c40"
                "score": "7"
                "keyPhrase": "no complaint"
            },
            ...
            {
                "id": "8af50d44-92d0-464f-886a-9690542f259b"
                "score": "2"
                "keyPhrase": "sound quality not good"
            }
        ],
        "topicAssignments": [
            {
                "topicId": "d8f62239-4ab5-4f95-8067-b2fca96d0c40",
                "documentId": "1",
                "distance": "0.354"
            },
            ...
            {
                "topicId": "8af50d44-92d0-464f-886a-9690542f259b",
                "documentId": "100",
                "distance": "0.758"
            }
        ],
        "errors": []
    }
}
```



ЛАНЬ®

Свойства отклика

- **topics:** массив всех уникальных тем, обнаруженных для переданных документов.
- **id:** идентификатор уникальной темы, сгенерированной API.
- **score:** количество документов, которые имеют эту тему.
- **keyPhrase:** обнаруженная тема.
- **topicAssignments:** массив присвоенных тем по каждому документу.
- **topicId:** уникальный идентификатор как массив тем.
- **documentId:** уникальный идентификатор документа, который передается в запросе.
- **distance:** оценка общности (афилированности) документа с темой. В диапазоне между 0 и 1. Чем ниже оценка, тем выше общность.



Идеи использования

- **Предсказание выборов:** выполнение анализа мнений на основе данных, собранных из социальных сетей, новостных статей, опросов общественного мнения и обзоров не является чем-то новым. Но в течение очень долгого времени используемые методологии в основном были статистическими. С по-

явлением машинного обучения, в особенности методов глубокого обучения, результаты предсказания победителя на выборах будут точнее.

- **Автоматическая генерация протокола сбражия.** Сбор и изложение протокола сбражия являются стандартной деятельностью, выполняемой во время формальных встреч. Такие методы, как обнаружение темы и выделение ключевых фраз, могут быть удачной находкой в автоматизации генерирования протоколов сбражий во время тексториентированных встреч на таких платформах, как Slack и IRC.
- **Анализ записей вызовов:** как правило, звонок пользователя в колл-центр записывается. Это делается в целях внутрифирменной аудиторской проверки. Используя технологию преобразования речи в текст, из этих записей при необходимости генерируются текстовые транскрипции. Далее эти транскрипции могут использоваться в качестве документов для выполнения анализа мнений и выделения ключевых фраз, чтобы лучше проанализировать уровни удовлетворенности клиентов.

API веб-модели языка (WebLM)

Создание модели языка для машинного обучения требует соответствующим образом подготовленного тренировочного корпуса текстов. Как было рассказано в предыдущих главах, результативность модели машинного обучения хороша настолько, насколько хороши ее тренировочные данные (корпус текстов). Чем больше корпус, тем больше тренировочные данные, тем лучше результирующая модель. Создание большого корпуса текстов вручную в масштабе, требуемом для тренировки модели, практически неосуществимо даже для крупных корпораций. Для создания систем машинного перевода использовались общедоступные протоколы заседаний в судах Европейского союза, переведенные на несколько языков. Документы ЕС являются не единственным бесплатным текстовым ресурсом, который мог использоваться для построения моделей языков. Разумеется, есть еще WWW.

Всемирная паутина содержит бесконечный объем текста, состоящего из миллиарда веб-страниц. Этот текст мог бы образовать невероятно большой корпус. Такая особенность Всемирной паутины дает возможность любому выполнять тренировку машинно-обучаемых моделей языка абсолютно бесплатно. Это именно то, что сделали в компании Microsoft с индексируемыми веб-страницами Bing. Они, используя текст, содержащийся в миллионах веб-страниц, создали обширные корпуса.

Корпус, созданный с использованием текста, собранного со всех концов сети, не подходит для таких приложений, как машинный перевод, так как в корпусе присутствуют некоторые аномалии: грамматические ошибки, сленг, ругательства, отсутствие пунктуации и т. п. Но корпус, созданный из веб-страниц, может дать нам общее представление о том, как обычные люди используют язык. В неправильном использовании слов и фраз могут быть найдены часто встречающиеся шаблоны. Корпусы достаточно хорошо подходят для создания моделей, выполняющих, например, предсказание следующего слова в последовательности и разбиение цепочки слов, в которой отсутствуют пробелы. Microsoft называет свою модель языка, созданную таким образом, веб-моделью языка, или WebLM.

Текст может присутствовать в разных частях веб-страницы: в теле, названии, якорных элементах и поисковых запросах. В тексте, извлеченном из каждой из этих четырех частей, имеются различия, которые варьируются от тончайших до видимых. В то время как слово «tryout» (испытание) обычно встречается в якорном тексте и поисковых запросах, в теле оно обычно присутствует в своей приемлемой форме «try out». Таким образом, Microsoft создала четыре разные модели, используя текст из этих четырех частей веб-страницы:

- модель Bing якорного элемента;
- модель Bing тела;
- модель Bing запроса;
- модель Bing названия.

В настоящее время единственным языком, который поддерживается API веб-модели языка, является американский вариант английского языка.

i Корпус WebLM – это не просто случайная коллекция огромного количества предложений. Вместо простых слов и фраз он содержит так называемые *n*-граммы. В Microsoft этот корпус называется веб-корпусом *n*-грамм.



N-грамма – это частичная последовательность из *n* элементов, полученных из большей, более полной последовательности. Элемент может быть символом, словом или фразой. В качестве примера рассмотрим приведенное ниже предложение:

«Собака играет с красным мячом».

N-граммные последовательности, которые могут быть получены из вышеупомянутой полной последовательности, будут следующими:

1-грамма (униграмма) – Собака, играет, с, красным, мячом;

2-грамма (биграмма) – Собака играет, играет с, с красным, красным мячом;

3-грамма (триграмма) – Собака играет с, играет с красным, с красным мячом.

Веб-корпус *n*-грамм Microsoft имеет до 5-граммных последовательностей. Эти *n*-граммы используются для выполнения таких задач, как разбиение на слова и предсказание следующего слова в последовательности.

Рисунок 5-8 показывает, как Microsoft Bing использует WebLM, чтобы показать результаты поиска.

happyeaster2018

happyeaster2018

happy easter 2016

Результаты: 4,870,000 Любое время ▾

Happy Easter 2018
<https://www.happyeaster2018.com> • Перевести эту страницу
 Happy Easter Wishes 2018: A Morning Wishes on Easter Day or touching Jesus Greetings on this special day will be somewhat touching the spirit of Jesus Christ. You might have touched with Inspiring Easter day 2018 images in our recent update.

Easter 2018 - When is Easter 2018, Easter Sunday 2018
www.easter2018.org • Перевести эту страницу
 Happy Easter 2018, 2017, 2019, 2020. When is Easter Sunday, What is, Dates, Easter Eggs, Happy Easter Images, Baskets, Pictures, Wishes, Greetings, Coloring Pages, Messages, Clip Art, Memes, Jokes, Drawings, Sayings, Easter Sunday Quotes, Greeting Cards, Pictures to Color, Images Jesus, Easter Weekend 2018, Holidays, Ideas, Celebration, Poems, Symbols, Easter Bunny Pictures, ...

Похожие поисковые запросы:
 when is easter 2018
 easter sunday 2018 date
 easter 2018 images
 easter vacation 2018
 happy easter pictures
 how is easter determined
 google easter sunday 2018
 easter 2017

Рис. 5.8 ❖ Разбиение на слова и предсказание следующего слова на странице результатов поиска в Bing в действии

Мало того, что поиск в Bing сумел разбить слово «happyeaster» на «happy easter», он также смог предоставить аналогичный поисковый запрос («happy easter 2016»), сверившись со своей моделью языка запросов в отношении наиболее распространенных следующих слов.

API веб-модели языка поддерживает следующие четыре задачи, или операции:

- разбиение последовательностей без пробелов на слова;
- совместная вероятность последовательности слов;
- условная вероятность одного слова, идущего после заданной последовательности слов;
- предсказание следующего слова (слов), которое может идти после заданной последовательности слов.

Разбиение на слова

Мы знаем, что разбиение непрерывной последовательности на отдельные слова является полезной операцией. Но где же можно найти примеры таких странных последовательностей? Другими словами, кто в здравом уме будет писать слова без разрывов?

Вы, возможно, встречались с такими последовательностями больше, чем могли себе представить. Хеш-теги в социальных сетях – весьма типичный пример.

#iamfeelinggood #germanyvsargentinafootballlive #EpicFail

Способность разбивать цепочки хеш-тегов на отдельные слова может представить системам анализа данных много скрытого материала. Непрерывные последовательности присутствуют и в URL-адресах и внутренних ссылках.

www.example.com/topsellers, www.example.com/tutorial#creatinganewproject

Запрос

```
POST /text/weblm/v1.0/breakIntoWords?model=body&text=germanyvsargentinafootballlive&order=5&maxNumOfCandidatesReturned=3 HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Cache-Control: no-cache
```

URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/weblm/v1.0/breakIntoWords>

Все параметры, за исключением отмеченных звездочкой (*), являются необязательными.

Параметры запроса

- **model***: используемая модель языка Bing. Одно из следующего: anchor, body, query и title (якорь, тело, запрос или название).
- **text***: последовательность, которая будет разбита на отдельные слова. Если пробелы в последовательности уже присутствуют, то в результате обработки они сохраняются без изменений.
- **order**: порядок n-грамм. Один из следующих: 1, 2, 3, 4 или 5. По умолчанию равняется 5.
- **maxNumOfCandidatesReturned**: максимальное количество возвращаемых совпадений. По умолчанию равняется 5.

Этот API не требует тела запроса.

Ответ

```
{
  "candidates": [
    {
      "words": "germany vs argentina football live",
      "probability": -12.873
    },
    {
      "words": "g ermny vs argentina football live",
      "probability": -19.08
    },
    {
      "words": "germany vs argentina football l i v e",
      "probability": -19.432
    }
  ]
}
```



Свойства отклика

- candidates: массив всех возможных совпадений, ограниченных параметром запроса maxNumOfCandidatesReturned.
- words: отдельные слова, полученные в результате разбиения, в одном совпадении.
- probability: менее отрицательные величины представляют высокую уверенность.

Совместная вероятность

Совместная вероятность показывает, как часто определенная последовательность слов встречается вместе. Например:

а вот и я => высокая вероятность
 привет чувак => средняя вероятность
 стиральная машина Microsoft => нулевая вероятность

Запрос

POST /text/weblm/v1.0/calculateJointProbability?model=body&order=5 HTTP/1.1

Host: westeurope.api.cognitive.microsoft.com

Ocp-Apm-Subscription-Key: abc123abc123abc123abc123abc123

Content-Type: application/json

Cache-Control: no-cache

```
{
  "queries": [
    [
      "but",
      "wide world",
      "natural language processing",
      "natural language dinner"
    ]
}
```



URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/weblm/v1.0/calculateJointProbability>

Все параметры, за исключением отмеченных звездочкой (*), являются необязательными.



Параметры запроса

- model*
- order

Тело запроса

- queries: массив последовательностей. Каждая строка представляет собой последовательность, совместную вероятность которой необходимо вычислить. Запросы из одного слова допустимы.

Отклик

```
{
  "results": [
    {
      "words": "but",
      "probability": -2.9
    },
    {
      "words": "wide world",
      "probability": -6.381
    },
    {
      "words": "natural language processing",
      "probability": -6.807
    },
    {
      "words": "natural language dinner",
      "probability": -12.939
    }
  ]
}
```

Свойства отклика

- results: массив результатов, один для каждого запроса, указанного в теле запроса.
- words: запрашиваемая последовательность в запросе.
- probability: чем ниже отрицательная оценка, тем более распространенным является слово или последовательность.



Условная вероятность

Условная вероятность говорит, как часто слово встречается после определенной последовательности слов. Например:

самый успешный художник => высокая вероятность
самый успешный автор => высокая вероятность

самый успешный парень => вероятность ниже среднего

самая продаваемая крыса => нулевая вероятность

Случаи возможного применения:

- восстановление перемешанного предложения;
- правильное угадывание короткого поискового запроса, в котором слова упорядочены неправильно.

Запрос

POST /text/weblm/v1.0/calculateConditionalProbability?model=body&order=5 HTTP/1.1

Host: westeurope.api.cognitive.microsoft.com

Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123

Content-Type: application/json

Cache-Control: no-cache

```
{
  "queries": [
    {
      "words": "top selling",
      "word": "artist"
    },
    {
      "words": "top selling",
      "word": "rat"
    },
    {
      "words": "game of",
      "word": "thrones"
    }
  ]
}
```



URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/weblm/v1.0/calculateConditionalProbability>

Все параметры, за исключением отмеченных звездочкой (*), являются необязательными.

Параметры запроса

- model*
- order

Тело запроса

- queries
 - words: последовательность слов, которые предшествуют целевому слову.
 - word: целевое слово.



Отклик

```
{
  "results": [
    {
      "score": 0.9999999999999998,
      "query": "top selling artist",
      "word": "artist"
    }
  ]
}
```

```
"words": "top selling",
"word": "artist",
"probability": -2.901
},
{
  "words": "top selling",
  "word": "rat",
  "probability": -5.848
},
{
  "words": "game of",
  "word": "thrones",
  "probability": -0.475
}
]
}
```

Предсказание следующего слова

Ранее на рис. 5.8 мы видели, как работает предсказание следующего слова, предлагая слова, которые могут идти после поискового запроса. Используя этот API, есть возможность предлагать варианты следующего слова для определенной последовательности слов. Например:

Нью-Йорк
Нью-Йорк таймс
Нью-Йоркский университет

Запрос

```
POST /text/weblm/v1.0/generateNextWords?model=body&words=Microsoft Windows&order=5&maxNumOfCandidatesReturned=5 HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Cache-Control: no-cache
```

URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/text/weblm/v1.0/generateNextWords>

Все параметры, за исключением отмеченных звездочкой (*), являются необязательными.

Параметры запроса

- model*
- order
- maxNumOfCandidatesReturned
- words*: строка из последовательности слов, для которых должны быть предложены варианты следующего слова. Эта последовательность может содержать всего одно слово.

Этот API не требует тела запроса.

Отклик

```
{
  "candidates": [
```

```
{
    "word": "7",
    "probability": -0.892
},
{
    "word": "xp",
    "probability": -0.998
},
{
    "word": "currentversion",
    "probability": -1.047
},
{
    "word": "server",
    "probability": -1.123
},
{
    "word": "8",
    "probability": -1.201
}
}
```



API лингвистического анализа

Операции, предлагаемые этим API, занимаются анализом языковой структуры заданного текста. Результат этого анализа может быть полезным для более четкого понимания значения и контекста текста.

API лингвистического анализа поддерживает три представленные ниже операции:

- **разбиение на предложения и лексемизация:** разбиение текста на предложения и предложений на лексемы;
- **частеречная разметка:** категоризация каждого слова предложения в качестве грамматического компонента (существительного, глагола, прилагательного и т. д.);
- **синтаксический анализ на основе грамматики составляющих:** разбиение текста на все возможные фразы.

Этот вид лингвистического анализа обычно выполняется на первом этапе других задач обработки естественного языка, таких как понимание ЕЯ, преобразование текста в речь, машинный перевод и т. д. Кроме пользы лингвистического анализа при предварительных шагах в области обработки ЕЯ, лингвистический анализ применяется в когнитивных приложениях, работающих с потребителями.

Чтобы система смогла выполнить эти три исследования, она должна быть настроена на основе банка синтаксических деревьев. Банк синтаксических деревьев (treebank) – это корпус, который содержит широкую коллекцию слов с частеречной разметкой и разобранных фраз. Для тренировки своих систем компания Microsoft использовала для этого API банка синтаксических деревьев Penn Treebank 3.

Разбиение на предложения и лексемизация

Разбиение больших фрагментов текста на отдельные предложения и разбиение предложения на лексемы являются часто встречающимся техническим условием при разработке программного обеспечения. Но разве у нас нет решения для этих задач? Рассмотрим приведенный ниже текст:



Было приятно с вами познакомиться. До свидания и до встречи.

Первое, что приходит на ум при разбиении этого текста на предложения, – разбить его по символу точки (.). Вполне разумно. Давайте применим это правило к приведенному выше тексту. В результате мы получим два предложения.

Предложение 1: Было приятно с вами познакомиться.

Предложение 2: До свидания и до встречи.

А как насчет такого текста:

Он сказал: "Иду на концерт." И бац, он ушел!

Разделение по символам точки дает нам следующее:

Предложение 1: Он сказал: "Иду на концерт.

Предложение 2: "И бац, он ушел!

Конечно же, это не то, что вы хотели бы получить. На данном этапе мы понимаем, что не все предложения заканчиваются точкой. Например:

Как у меня получилось? Хорошо или нет?

Да ты с ума сошел! Не смей к этому прикасаться.

Что?! Вы в своем уме?

Вот где на выручку приходит когнитивное разделение на предложения. Этот API может использоваться для выделения всех предложений из очень длинной статьи или книги и их сохранения для дальнейшего анализа. Либо, возможно, где-нибудь в своем приложении или на веб-сайте вы просто ищете способ разделить большой фрагмент текста на несколько предложений.

Еще одна операция – это лексемизация или разбиение предложения на отдельные слова и другие символы. Так же как разбиение на предложения, лексемизация не может быть достигнута за счет простого разделения предложения по пробелам. Среди всего прочего должны учитываться такие языковые явления, как сокращения и притяжательные формы в английском языке. К тому же в таких языках, как китайский, не всегда между словами имеются пробелы.

Предложение «I don't have anything to do with Ronnie's broken glasses, I swear» («Я не имею никакого отношения к разбитым очкам Ронни, клянусь») содержит следующие символы:

I – do – n't – have – anything – to – do – with – Ronnie – 's – broken – glasses – , – I – swear – .

n't – это *стяженная форма* для слова not. 's – это *притяжательная форма*. Точка и запятая – это *знаки препинания*. Вот лишь несколько лексем, помимо обычных слов, которые должны быть обнаружены для надлежащего дальнейшего анализа.

Запрос

```
POST /linguistics/v1.0/analyze HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/json
Cache-Control: no-cache

{
    "language": "en",
    "analyzerIds": ["08ea174b-bfdb-4e64-987e-602f85da7f72"],
    "text": "How did I do? Was I good?"
}
```

URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/linguistics/v1.0/analyze>

Все параметры, за исключением отмеченных звездочкой (*), являются необязательными.



Параметры запроса

Этот API не имеет параметров запроса.

Тело запроса

- `language`*: два буквенных кода ISO 639-1 для языка анализируемого текста.
- `analyzerIds`*: массив глобально-уникальных идентификаторов GUID используемых анализаторов (Tokens для лексем, POS_Tags для частеречных меток либо Constituency_Tree для дерева составляющих).
- `text`*: фрагмент анализируемого текста.

И К одному фрагменту текста можно применять все три анализатора сразу. Чтобы получить список всех поддерживаемых анализаторов (во время написания книги их было три) вместе с их идентификаторами GUID, выполните вызов GET по <https://westeurope.api.cognitive.microsoft.com/linguistics/v1.0/analyzers>, используя ключ подписки на API лингвистического анализа.

Отклик

```
[
  {
    "analyzerId": "08ea174b-bfdb-4e64-987e-602f85da7f72",
    "result": [
      {
        "Len": 13,
        "Offset": 0,
        "Tokens": [
          {
            "Len": 3,
            "NormalizedToken": "How",
            "Offset": 0,
            "RawToken": "How"
          },
          {
            "Len": 3,
            "NormalizedToken": "did",
            "Offset": 4,
            "RawToken": "did"
          }
        ]
      }
    ]
  }
]
```

```
        "Offset": 4,
        "RawToken": "did"
    },
    {
        "Len": 1,
        "NormalizedToken": "I",
        "Offset": 8,
        "RawToken": "I"
    },
    {
        "Len": 2,
        "NormalizedToken": "do",
        "Offset": 10,
        "RawToken": "do"
    },
    {
        "Len": 1,
        "NormalizedToken": "?",
        "Offset": 12,
        "RawToken": "?"
    }
]
},
{
    "Len": 11,
    "Offset": 14,
    "Tokens": [
        {
            "Len": 3,
            "NormalizedToken": "Was",
            "Offset": 14,
            "RawToken": "Was"
        },
        {
            "Len": 1,
            "NormalizedToken": "I",
            "Offset": 18,
            "RawToken": "I"
        },
        {
            "Len": 4,
            "NormalizedToken": "good",
            "Offset": 20,
            "RawToken": "good"
        },
        {
            "Len": 1,
            "NormalizedToken": "?",
            "Offset": 24,
            "RawToken": "?"
        }
    ]
}
```



```

    }
]
}
]
```

Обратите внимание, результатом является не объект, а массив объектов. Это вызвано тем, что данный API дает вам возможность выполнять одновременно несколько анализов на одном фрагменте текста. Результат каждого анализатора возвращается как объект корневого массива.

Свойства отклика:

- **analyzerId**: идентификатор GUID анализатора, используемого для соответствующего набора результатов;
- **result**: результат анализа. Массив предложений:
 - **Len**: общее количество символов в предложении;
 - **Offset**: начальная позиция предложения в тексте. Начинается с 0;
 - **Tokens**: массив лексем в предложении:
 - **Len**: количество символов в лексеме;
 - **Offset**: начальная позиция лексемы в предложении;
 - **RawToken**: обнаруженная лексема;
 - **NormalizedToken**: лексема, представленная в формате, который пригоден для использования в дереве синтаксического разбора.

Частеречная разметка

Помните, как в школе вы учились определять грамматическую структуру предложения? Учитель говорил о подлежащем, сказуемом и дополнении и затем переходил к определению подлежащего и дополнения в терминах имен собственных и имен нарицательных. После этого вы постепенно учились определять в предложении прилагательные, наречия, предлоги и союзы. Метод идентификации грамматической структуры предложения, который преподавал ваш учитель, можно представить как упрощенную версию частеречной разметки.

Такие слова, как существительные, глаголы и прилагательные, называются частями речи. Частеречная разметка (POS tagging) – это общепринятый метод разрешения неоднозначности, который делает запутанные/неоднозначные слова яснее. Этот метод используется для идентификации частей речи в заданном предложении на основе соответствующей им позиции и контекста. Иными словами, для каждого слова в предложении частеречная разметка определит, какой частью речи является это слово. Рассмотрим приведенное ниже предложение:

He is going to contest the election (Он собирается побороться на выборах).

Интерес в этом предложении вызывает слово «*contest*», которое может быть существительным или глаголом. В этом случае оно используется в качестве глагола. Частеречный разметчик должен умело выявлять контекст и окружающие слова, чтобы помечать слово правильно. Вот версия приведенного выше предложения с частеречной разметкой:

He (PRP) is (VBZ) going (VBG) to (TO) contest (VB) the (DT) election (NNS).

Были выявлены следующие метки (теги):

- PRP = местоимение, личное;

- VBZ = глагол, настоящее время, третье лицо, единственное число;
- VBG = глагол, причастие настоящего времени или герундий;
- TO = «to» в качестве маркера предлога или инфинитива;
- VB = глагол, основная форма;
- DT = определитель;
- NNS = существительное, распространенное, множественное.

Полный список всех возможных меток можно найти онлайн, а также в документации Microsoft по частеречной разметке.

Частеречная разметка в особенности полезна в области понимания естественного языка и машинном переводе.

Запрос

```
POST /linguistics/v1.0/analyze HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/json
Cache-Control: no-cache

{
    "language": "en",
    "analyzerIds": ["4fa79af1-f22c-408d-98bb-b7d7aef7f04"],
    "text": "He is going to contest the election"
}
```

URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/linguistics/v1.0/analyze>

Все параметры, за исключением отмеченных звездочкой (*), являются необязательными.

Параметры запроса

Этот API не имеет параметров запроса.

Тело запроса

- language*
- analyzerIds*
- text*

Отклик

```
[
  {
    "analyzerId": "4fa79af1-f22c-408d-98bb-b7d7aef7f04",
    "result": [
      [
        "DT",
        "VBZ",
        "JJ",
        "."
      ],
      [
        "PRP",
        "VBP",
        "NNP"
      ]
    ]
  }
]
```

```

    "PRP",
    "."
]
}
]
]
```



Свойства отклика

Результатом является массив меток (тегов) в каждом предложении. Метки находятся в той же последовательности, что и слова в предложении.

Синтаксический анализ на основе грамматики составляющих

Синтаксический анализ на основе грамматики составляющих используется для выявления словосочетаний или фраз в заданном предложении. Отличие синтаксического анализа от метода выделения ключевых фраз, показанного в API анализа текстов, – в том, что анализ на основе грамматики составляющих возвращает не только ключевые, но и все возможные фразы. Такая информация может быть полезна для анализа по методу выделения ключевых фраз как такового.

I want to buy a laptop with graphics card (Я хочу купить ноутбук с графической картой).

Сколько словосочетаний можно идентифицировать в вышеупомянутом предложении?

Давайте посмотрим. «I want to buy a laptop», «буу a laptop», «laptop with graphics card» и «graphics card» – все они являются допустимыми словосочетаниями. Можно ли найти еще? Взгляните на рис. 5.9, после того как попробуете это сделать.

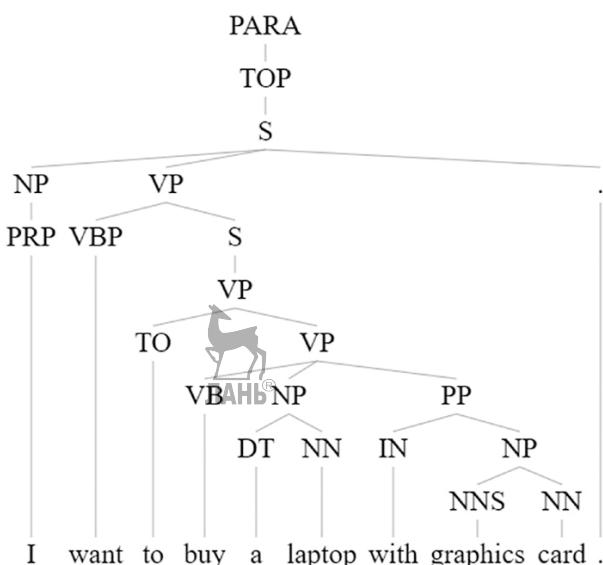


Рис. 5.9 ♦ Разобранное дерево для вышеупомянутого предложения.
Слова, включая и между каждой парой узлов в вышеупомянутом двоичном дереве, являются допустимым словосочетанием

Важно отметить, что словосочетание не является просто любой группой слов. Словосочетание – это то, что в предложении можно заменить или переместить как единое целое без потери грамматической правильности и общего смысла. Например, «I want to buy» и «a laptop with graphics card» являются двумя главными словосочетаниями в вышеупомянутом предложении. Если их поменять местами, то предложение остается понятным:

a laptop with graphics card I want to buy

Поскольку «to buy» не является словосочетанием, изменение позиции этих слов делает предложение непонятным:

to buy I want a laptop with graphics card

Словосочетания обычно вкладываются друг в друга, и поэтому их бывает сложно найти без понимания грамматической структуры предложения.

Запрос

```
POST /linguistics/v1.0/analyze HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: abc123abc123abc123abc123abc123
Content-Type: application/json
Cache-Control: no-cache

{
    "language": "en",
    "analyzersIds": ["22a6b758-420f-4745-8a3c-46835a67c0d2"],
    "text": "I want to buy a laptop with graphics card."
}
```



URL-адрес конечной точки

<https://westeurope.api.cognitive.microsoft.com/linguistics/v1.0/analyze>



Все параметры, за исключением отмеченных звездочкой (*), являются необязательными.

Параметры запроса

Этот API не имеет параметров запроса.

Тело запроса

- language*
- analyzersIds*
- text*

Отклик

```
[
  {
    "analyzerId": "22a6b758-420f-4745-8a3c-46835a67c0d2",
    "result": [
      "(TOP (S (NP (PRP I)) (VP (VBP want) (S (VP (T0 to) (VP (VB buy) (NP (DT a) (NN laptop))
        (PP (IN with) (NP (NNS graphics) (NN card))))))) (. .)))"
    ]
  }
]
```

Свойства отклика

Результатом является массив разобранных предложений. Попытайтесь сопоставить результат разбора в формате с круглыми скобками с рис. 5.9, чтобы разобраться в том, как его истолковывать.



Резюме

В этой главе вы, кроме задачи понимания естественного языка, познакомились с другими задачами обработки ЕЯ. Задача понимания ЕЯ была рассмотрена в предыдущей главе.

Подводя итоги, в этой главе вы:

- познакомились с четырьмя API из категории «Язык» когнитивных служб: API Bing проверки орфографии, анализа текстов, веб-модели языка и лингвистического анализа;
- познакомились с теоретическим материалом и приложениями по каждому API и всеми его операциями;
- получили подробный обзор применения каждого API;
- увидели работающие примеры реализации и получили новые идеи использования.

В **следующей главе** вы примените свои знания облачной службы LUIS и инфраструктуры Bot Framework для создания чат-бота, готового к внедрению на предприятии.



Глава 6

Потребление и применение облачной службы LUIS

Добро пожаловать в заключительную часть! В этой главе вы на практике задействуете знания, полученные в предыдущих главах.

В главе 2 вы узнали, как настроить среду разработки.

В главе 3 вы познакомились с разговорным интерфейсом пользователя (РИП, CUI), новой парадигмой для создания интеллектуальных приложений на основе естественного языка. В этой главе вы также научились создавать очень простой, хотя и не совсем интеллектуальный, чат-бот с использованием среды Bot Framework. Правда, вы не развернули свой бот на одном или нескольких поддерживающих каналах связи.

В главе 4 вы детально изучили задачу понимания естественного языка, т. е. то, что позволяет компьютерным программам истолковывать вводимые пользователями фразы так, как люди понимают друг друга. Вы получили представление об облачной службе Microsoft LUIS, а также узнали, как в приложении можно использовать LUIS для более естественного понимания пользователей.

В этой главе мы продолжим работу над медицинским ботом записи на прием к врачу. Вы воспользуетесь облачной службой LUIS, чтобы усовершенствовать ранее созданный глупый бот и сделать его намного умнее. Вы создадите приложение с поддержкой LUIS в пошаговом режиме, чтобы использовать эту службу вместе с ботом. Наконец, вы интегрируете приложение с поддержкой LUIS в среду Bot Framework. К концу этой главы вы:

- познакомитесь с анатомией приложения с поддержкой LUIS;
- познакомитесь с намерениями, сущностями, списками фраз и другими компонентами LUIS;
- узнаете, как создавать с нуля приложение с поддержкой LUIS систематически;
- узнаете, как тестировать, тренировать и публиковать приложение с поддержкой LUIS;
- узнаете, как сделать так, чтобы инфраструктура Bot Framework использовала LUIS;
- узнаете, как использовать канал связи, например Skype и Facebook Messenger, для опубликования вашего бота.

ПЛАНИРОВАНИЕ СВОЕГО ПРИЛОЖЕНИЯ

Создавая диалоговые приложения, крайне важно иметь формальный план. Для приложений с графическим интерфейсом пользователя (ГИП) разработчики используют макеты и другие дизайнерские документы для визуальной сверки. Это облегчает разработку различных компонентов приложения с ГИП, позволяя визуализировать их заключительный дизайн. Разговорные приложения не имеют многочисленных визуальных компонентов. Поток действий должен быть понят с точки зрения текстоориентированных разговоров между приложением и его пользователями.

Такая очередность действий может быть представлена самыми разнообразными способами, в частности с использованием блок-схем. Какой бы способ вы ни выбрали, крайне важно с самого начала иметь ответы на следующие вопросы:

- Что бот должен уметь делать? **Задачи.**
- Какая информация нужна боту от пользователя? **Пользовательский контекст.**
- Что должна делать облачная служба LUIS? **Намерения и сущности.**
- Что требуется от среды Bot Framework? **Логика и ответы.**

Давайте взглянем на каждый компонент медицинского бота записи на прием к врачу.

Что бот должен уметь делать?

Созданная вами в главе 3 версия бота могла делать всего одну вещь: планировать визит к врачу. Что еще он должен уметь делать? Напомним, что это медицинский бот записи на прием к врачу. А может ли он проdiagностировать пациента? Иными словами, имея список симптомов, перейти к списку потенциальных заболеваний или медицинских показаний. Это своего рода вид самопомощи, которую ваш бот может предоставить сидящему дома пользователю. Давайте добавим это в список задач вашего бота.

Пользователь: Чувствую боль в нижней части живота и также ощущаю небольшое вздутие.

Бот: У вас, возможно, аппендицит, запор или пищевое отравление.

Теперь вернемся к планированию записи на прием к врачу. При разработке бота в главе 3 предполагалось, что у нас один врач. В жизни так не бывает, если только бот не проектировался для клиники, в которой работает только один врач. Бот может обслуживать больницу, сеть больниц или сеть врачей. В любом случае, пользователь имеет дело с врачами нескольких узких специальностей и многочисленными врачами одной и той же узкой специальности. Ваш бот должен из запроса пользователя уметь определять узкую специальность врача: «запишите меня к **стоматологу**», «хочу консультироваться с **гинекологом**» или «как записаться к **кардиологу**».

Пользователь также может указать удобный ему промежуток времени: «запишите к стоматологу между 9:00 и 12:00». Бот должен уметь обнаруживать требуемые промежутки времени.

Резюмируя, бот должен уметь справляться со следующими двумя задачами:

- 1) определить основной диагноз, предоставив список заболеваний/медицинских показаний на основе заданного набора симптомов;
- 2) назначить визит к врачу или специалисту на необходимую дату и в желаемый промежуток времени.

Какая информация нужна боту от пользователя?

Теперь, когда у вас есть четкое представление о возможностях вашего бота, очень легко перечислить все данные, которые ему нужны от пользователя для выполнения своей задачи.

Двумя общепринятыми фрагментами данных, которые пользователь должен предоставить боту, являются его **имя** и **возраст**. Ведь одинаковый набор симптомов может по-разному проявляться у людей различных возрастных групп. Во всяком случае, эта информация крайне важна в **планировании** визита к врачу.

 Вы можете дополнительно запросить контактный номер и адрес электронной почты.

Для определения основного диагноза требуется **список всех симптомов**. Пользователь может перечислить эти симптомы одним предложением или каждый симптом в отдельном предложении.

Для планирования визита к врачу необходимо знать **специализацию врача, дату** и удобный для пользователя **промежуток времени**.

Что должно быть сделано в облачной службе LUIS?

На этот вопрос ответить легко. LUIS в вашем приложении является компонентом, понимающим естественный язык. Он может только помочь вашему боту понять значение заданного предложения. И не больше. Служба LUIS не сохраняет и не предоставляет ответы. Она не выполняет вычисления и не обрабатывает логику. Но зато эта служба может разделить предложение на намерение (смысл) и сущности.

Теперь неплохо было бы заранее дать имена вашим намерениям и сущностям. Это даст вам общее представление о точном количестве намерений и сущностей, которые понадобятся при создании приложения с поддержкой LUIS. Таблица 6-1 показывает удобный способ проектирования приложения с поддержкой LUIS. Вы сможете руководствоваться им позже, работая с компонентом, связанным с LUIS, либо передать его коллеге в качестве проектного документа для создания приложения с поддержкой LUIS.

Таблица 6.1. Словарь данных для приложения LUIS с ботом записи к врачу

Высказывание	Ответ	Намерение	Сущности
Привет Добрый день Послушайте Помогите	Добрый день! Сообщите ваши симптомы, и я дам вам возможные медицинские показания. Либо же просите записать к врачу. Но сперва укажите ваше имя и возраст	Greet	--
Меня зовут Анураг Бандари. Мне 29 лет	Здравствуйте, Анураг. Чем я могу вам помочь?	PatientDetails	PatientName::FirstName = Anurag, PatientName::LastName = Bhandari, PatientAge = 29 лет

Окончание табл. 6.1

Высказывание	Ответ	Намерение	Сущности
Ощущаю спутанность и тревогу. Меня вырвало несколько минут назад	У вас, возможно, гипогликемия	GetCondition	Symptom = confusion, Symptom = anxiety, Symptom = vomited
Чувствую боль в нижней части живота и также ощущаю небольшое вздутие	У вас, возможно, аппендицит, запор или пищевое отравление	GetCondition	Symptom = pain, Symptom = swelling BodyPart = lower abdomen
Запишите меня к дантисту на завтра между 9 утра и 12 вечера	Я записал вас к доктору Джону Доу на воскресенье 7 мая в 11 утра	ScheduleAppointment	DoctorType = dentist, datetime = tomorrow, AppointmentTime::StartTime = 9 am, AppointmentTime::EndTime = 12 pm
Хочу посетить врача	Я записал вас к доктору Джону Доу на воскресенье 7 мая в 11 утра	ScheduleAppointment 	-- (Поскольку в высказывании не указана ни одна сущность, принять значения по умолчанию.) DoctorType = physician, datetime & AppointmentTime = первый свободный промежуток времени

Все возможные случаи применения в приведенной выше таблице не предусмотрены. Таблица предназначена только для того, чтобы показать формат для вашего проектного документа. Расширьте приведенный выше словарь данных, чтобы охватить максимально необходимое количество вариантов использования. Это будет вашим практическим заданием. Обратите внимание: в первом проекте вы не сможете предусмотреть все варианты обращения пациента к боту и ответы самого бота пользователю. Вам, вероятно, придется пересматривать словарь несколько раз во время разработки бота. Облачная служба LUIS хорошо отрабатывает вновь добавленные и измененные намерения и сущности. Также служба хорошо проводит повторную тренировку ранее помеченных высказываний.

Что должно быть сделано в Bot Framework?

Вся остальная работа в значительной степени ложится на Bot Framework. Прочитав главу 3, вы узнали, как создается бот. Там же было рассказано, что среда Bot Framework предоставит вам свои инструменты (то есть API для разработчиков) для создания бота. То, как вы используете эти API, полностью зависит от вас. То есть всю логику для бота, включая взаимодействие с облачной службой LUIS, выполняет Bot Framework. Отсюда после выполнения ваших вычислений и обмена с LUIS пользователь получает ответы на задаваемые ими вопросы. Ответы могут поступать как из базы данных, так и напрямую из программного кода, в котором они статически хранятся.

И здесь возникают два очень важных вопроса:

- как логически определить заболевания из названных симптомов?
- как проверить, доступен ли врач в определенное время суток?

Вы же не думаете о создании базы данных всех возможных симптомов и медицинских показаний? Вы же не являетесь экспертом в медицинской области, а поддержка такой базы данных стала бы для вас цефалгией (головной болью). Базовое эмпирическое правило при создании приложений заключается в том,

чтобы никогда не изобретать заново колесо. Оперативный поиск в Google или Bing приведет вас к веб-сайту www.apimedic.com, который предоставляет вам API проверки симптомов. Этот API был бы оптимальным вариантом в данном сценарии. Хотя этот API является платным, у них есть бесплатная ценовая категория, которая позволяет делать 100 вызовов API в месяц.

Точно так же можно воспользоваться чем-то вроде API Google Календарь, чтобы создавать назначения к врачу и получать доступные часы приема. Как вариант API drchrono предоставляет полноценную систему управления в формате доктор–пациент.

Подводя итоги, задачи, которые будут выполняться с использованием среды Bot Framework, включают:

- получение данных, вводимых пользователем;
- отправку введенных пользователем данных в LUIS для анализа;
- выполнение действия (действий) на основе намерения и сущностей, полученных из LUIS;
- отправку ответа назад пользователю.

Создание приложения с поддержкой LUIS

Теперь у вас есть план. Вы знаете, каких входных данных ожидать от пользователей и какие ответы предоставлять. Также известно, какие действия требуется выполнять по ходу. У вас также есть четко определенный словарь данных для использования с LUIS.

Создание приложения с поддержкой LUIS не представляет труда. Вы, возможно, уже это сделали во время практических упражнений в главе 4. Давайте создадим новое приложение с поддержкой LUIS для медицинского бота записи на прием к врачу.

Во время авторизации на веб-сайте <https://www.luis.ai> перейдите к **My Apps** (Мои приложения) и нажмите кнопку **New App** (Новое приложение). Воспользуйтесь подсказками из рис. 6.1, чтобы заполнить форму нового приложения.



Рис. 6.1 ♦ Создание нового приложения с поддержкой LUIS

i Без ключа вы не сможете опубликовать это приложение. Обратитесь к главе 2 и посмотрите, какие шаги следует выполнить, чтобы получить ключ подписки на портале Azure, и к главе 4 – чтобы закрепить ключи подписки за вашей учетной записью LUIS.

Поздравляем. Вы только что создали новое приложение с поддержкой LUIS. Свое время добавить намерения.

ДОБАВЛЕНИЕ НАМЕРЕНИЙ

После создания нового приложения с поддержкой LUIS вы будете автоматически перенаправлены к обзорной странице приложения. Если этого не произошло, щелкните по недавно созданному приложению из списка на странице **My Apps**.

Чтобы добавить новое намерение, в левой боковой панели перейдите к странице **Intents** (Намерения). Сверяясь с табл. 6.1, вы увидите, что самым первым намерением является **Greet** (Приветствие). Нажмите кнопку **Add Intent** (Добавить намерение), укажите «Greet» в качестве имени намерения и нажмите **Create** (Создать). Должно произойти перенаправление на страницу намерения **Greet**.

ДОБАВЛЕНИЕ/РАЗМЕТКА ВЫСКАЗЫВАНИЙ

Начните на этой странице добавлять образцы высказываний. Постарайтесь предусмотреть максимальное количество всех возможных вариантов, как пользователь может начать разговор с вашим ботом. Можно начать с высказываний типа «привет», «добрый день» и «послушайте», которые являются общепринятыми вариантами начала разговора. После добавления первоначальных высказываний нажмите кнопку **Save** (Сохранить), чтобы зафиксировать ваши изменения. Рисунок 6.2 показывает пример внесенных изменений. На данном этапе ваше приложение с поддержкой LUIS еще не может создавать предсказаний для введенных пользователем данных, потому что оно еще не натренировано. Добавление высказываний в намерение НЕ равняется выполнением тренировки облачной службой LUIS.

The screenshot shows the 'Utterances' section of the LUIS interface. At the top, there are three tabs: 'Utterances (3)', 'Entities in use', and 'Suggested Utterances'. Below the tabs is a search bar with placeholder text 'Type a new utterance & press Enter ...'. A small icon of a person sitting at a desk is visible next to the search bar. On the right side of the interface, there are buttons for 'Save', 'Discard', 'Delete', and 'Reassign Intent'. A dropdown menu labeled 'Labels view (Ctrl+E)' is open, showing 'Entities' selected. There is also a search bar for 'Search in utterances ...' with a magnifying glass icon. The main table lists four utterances:

Utterance text	Predicted intent
hey	Not trained
hello	Not trained
hi	Not trained

Рис. 6.2 ❖ Добавление высказываний в намерение

Начнем тренировку приложения. Перейдите из левой боковой панели на страницу **Train and Test** (Тренировка и тестирование) и нажмите кнопку **Train Application** (Натренировать приложение). После завершения тренировки через несколько секунд протестируйте приложение, которое вы создали на данный момент. В разделе **Interactive Testing** (Интерактивное тестирование) попробуйте «hi» и «hello». На панели результатов справа вы получите Greet в качестве намерения с самой высокой оценкой уверенности вместе с величиной оценки уверенности. Результаты показаны на рис. 6.3.

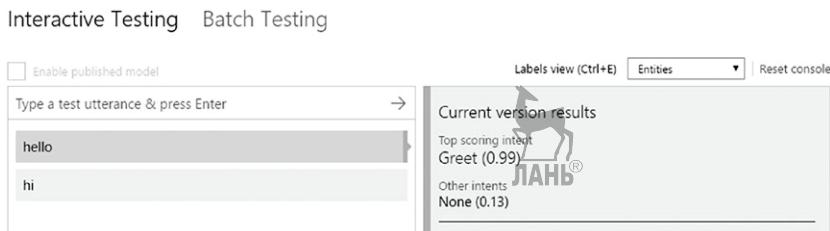


Рис. 6.3 ♦ Тестирование высказываний на странице Train and Test

На данном этапе уровень уверенности будет высоким (идеально 1, в противном случае всегда больше 0.9). На самом деле вы заметите, что LUIS будет возвращать Greet с высокой уверенностью даже для всех остальных высказываний. Попробуйте высказывание «как далеко Париж от Лондона» – и получите все тот же ответ. Это вызвано тем, что здесь у вас всего одно намерение.

Если быть точным, прямо сейчас ваше приложение имеет в общей сложности два намерения: Greet и None. Намерение None создается службой LUIS по умолчанию. Это встроенное намерение, которое нельзя удалить. Задача None состоит в том, чтобы уменьшить беспорядок, о котором мы только что говорили. Все высказывания, которые не обрабатываются вашим приложением, следует добавлять в намерение None.

Давайте вернемся к странице намерений Intents и откроем страницу намерения None из списка. Добавьте в этот список высказывание «как далеко Париж от Лондона». Сохраните и перетренируйте приложение. Если теперь протестируировать это высказывание, то будет возвращено намерение None в качестве намерения с самым высоким уровнем уверенности. Попробуйте другое подобное высказывание типа «как далеко Дели от Мумбаи». И вы по-прежнему увидите, что намерение None побеждает.

Протестируйте еще несколько высказываний: «кто вы такой», «привет, бот» и «как дела?». Если LUIS возвращает None, снова перейдите на страницу намерения Greet и добавьте высказывания, обозначающие приветствия, в список. Как можно увидеть на рис. 6.4, недавно добавленные высказывания появятся в другом цвете.

Натренируйте свое приложение снова и удостоверьтесь, что новые высказывания возвращают правильное намерение.

Utterances (3) Entities in use Suggested utterances

The screenshot shows the Microsoft LUIS Utterances interface. At the top, there's a search bar with placeholder text "Type a new utterance & press Enter ...". Below it is a toolbar with "Save", "Discard", "Delete", and "Reassign Intent" buttons, along with a dropdown menu "Labels view (Ctrl+E): Entities" and a search bar "Search in utterances ...". The main area displays a table of utterances:

Utterance text	Predicted Intent
hello bot	0.83 Greet
how are you ?	0.59 None 0.16 Greet
who are you	0.52 None 0.34 Greet
hey	0.99 Greet
hello	0.99 Greet
hi	0.99 Greet

Рис. 6.4 ❖ Добавление новых высказываний в существующее намерение

ПУБЛИКАЦИЯ СВОЕГО ПРИЛОЖЕНИЯ

Публикация своего приложения будет логическим и заключительным шагом в разработке приложения с поддержкой LUIS. Но поскольку вам хочется гарантий, что тренировка ведется правильно, сейчас крайне важно опубликовать свое приложение. Публикация просто означает развертывание самой недавней натренированной модели своего приложения на URL-адресе конечной точки HTTP.

Конечная точка – это веб-служба для доступа к приложению с поддержкой LUIS. Конечная точка получает высказывание через параметр строки запроса и возвращает результат в виде документа JSON с соответствующим разбиением в формате «намерение–сущности». Когда вы приложение публикуете впервые, URL-адрес конечной точки генерируется на основе идентификатора вашего приложения и ключа подписки. Этот URL-адрес остается постоянным на протяжении всей жизни вашего приложения с поддержкой LUIS.

Сейчас приложение следует опубликовать не для того, чтобы увидеть, как будет выглядеть результат в виде JSON, а для проверки уровня его натренированности. Бывает, когда результаты для одного и того же высказывания, полученные из конечной точки HTTP, отличаются от результатов, которые мы видели на странице интерактивного тестирования. В конечном итоге нам важны результаты, полученные от конечной точки, так как ваш бот будет получать именно их.

Перейдите из левой боковой панели на страницу **Publish** (Опубликовать). Прокрутите вниз и нажмите кнопку **Publish**. При этом удостоверьтесь, что в качестве места конечной точки выбрано **Production** (Производство). Как и в случае с тренировкой, на исполнение публикации уйдет несколько секунд. Результат вы можете увидеть на рис. 6.5.



Рис. 6.5 Публикация приложения с поддержкой LUIS

Вас может заинтересовать, почему кнопка **Train** находится рядом с кнопкой **Publish**, когда для тренировки существует специальная страница. Это сделано для экономии времени. Во время работы с рабочими (производственными) приложениями с поддержкой LUIS обычно больше опираются на результаты опубликованной конечной точки, чем на результаты интерактивного тестирования. Поэтому после разметки высказываний общепринято пропускать тренировку и тестирование и сразу переходить непосредственно на страницу публикации приложения Publish App.

Скопируйте URL-адрес конечной точки и вставьте его в строке поиска вашего браузера. Параметр q строки запроса является тем местом, где должно быть задано высказывание. Вот результат в виде документа JSON для высказывания «привет, бот», который мы получили при его тестировании:

<https://westeurope.api.cognitive.microsoft.com/luis/v2.0/apps/<id>?subscription-key=<key>&verbose=true&timezoneOffset=0&q=hello%20bot>

```
{
  "query": "hello bot",
  "topScoringIntent": {
    "intent": "Greet",
    "score": 0.9999995
  },
  "intents": [
    {
      "intent": "Greet",
      "score": 0.9999995
    },
    {
      "intent": "None",
      "score": 0.0768341646
    }
  ],
  "entities": []
}
```



Мы находим это в результирующих данных, полученных из конечной точки. Намерение Greet по-прежнему имеет самый высокий (99%-й) уровень уверенности. Это совпадает с результатами интерактивного тестирования, которые мы видели ранее. Пока неплохо.

ДОБАВЛЕНИЕ СУЩНОСТЕЙ

Для простых приветственных высказываний сущности вам не требовались. Со-гласно словарю данных в табл. 6.1, сущности будут нужны для всех остальных намерений. Давайте начнем с создания нового намерения. Перейдите на страницу намерений Intents и создайте намерение ScheduleAppointment (Наметить визит).

Добавьте высказывание без сущности «запишите меня». И затем еще одно с сущностью «запишите меня к стоматологу». В только что добавленном высказывании установите указатель мыши на слове «стоматолог». Выбранное слово выделится квадратными скобками. Щелкните мышью на выделенном слове. Появится меню с вариантами, как пометить это слово: как существующую сущность либо как новую сущность. На рис. 6.6 показан процесс создания новой сущности под названием DoctorType.

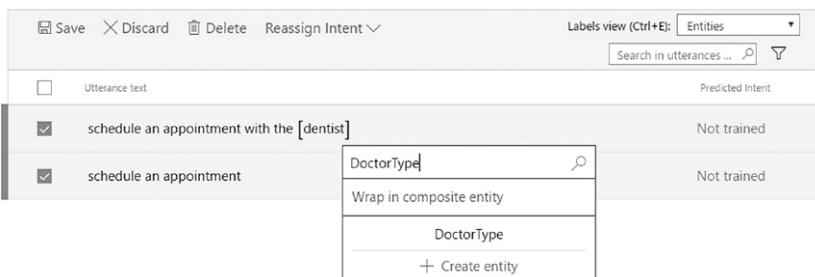


Рис. 6.6 ❖ Создание новой сущности на странице намерения

Сущность, созданная таким образом, называется простой сущностью. Имеются два других типа сущностей: *иерархические* и *составные*. Кроме того, имеются *предварительно подготовленные* сущности. Вы познакомитесь с ними чуть ниже.

Добавьте дополнительные варианты высказывания «запишите», такие как «я хочу записаться», «хочу записаться к стоматологу» и т. д. Кроме того, добавьте высказывания, охватывающие врачей других узких специальностей: «Я хочу посетить дерматолога», «запишите меня к кардиологу» и т. д. На рис. 6.7 показан результат добавления новых высказываний.

Обратите внимание, что в вышеупомянутом списке невозможно отличить «стоматолога» от «дерматолога». Обе сущности помечены как \$DoctorType. Щелкните на открывающемся списке просмотра меток **Labels view** и выберите метку **Tokens** (Лексемы). На рис. 6.8 показано обновленное представление с возможностью различить значения сущностей.

The screenshot shows the Rasa NLU interface with the 'Entities' tab selected. There are six rows of utterance variants:

Utterance text	Predicted Intent
schedule an appointment with the [\$DoctorType]	Not trained
i want to see a [\$DoctorType]	Not trained
set an appointment	Not trained
schedule an appointment with the [\$DoctorType]	Not trained
schedule an appointment	Not trained

Рис. 6.7 ❖ Добавление дополнительных вариантов одного высказывания

The screenshot shows the Rasa NLU interface with the 'Tokens' tab selected. There are six rows of tokens:

Utterance text	Predicted Intent
schedule an appointment with the [cardiologist]	Not trained
i want to see a [dentist]	Not trained
set an appointment	Not trained
schedule an appointment with the [dentist]	Not trained
schedule an appointment	Not trained

Рис. 6.8 ❖ Представление лексем для списка высказываний

Теперь откройте страницу тренировки и тестирования и выполните новую тренировку. После завершения тренировки в разделе интерактивного тестирования установите флажок **Enable published model** (Активировать опубликованную модель). Это позволит сравнить результаты модели интерактивного тестирования и опубликованной модели, как показано на рис. 6.9.

Interactive Testing Batch Testing

The screenshot shows the Rasa NLU Interactive Testing interface. On the left, there is a text input field with the placeholder "Type a test utterance & press Enter". Below it, a list box contains the utterance "schedule an appointment with the [\$DoctorType]". On the right, there are two sections: "Current version results" and "Published version results". Both sections show the same results:

- Top scoring intent:** ScheduleAppointment (1)
- Other intents:** None (0.06) Greet (0)

Below these sections is a link "Raw JSON view".

Рис. 6.9 ❖ Тестирование результатов с активированной опубликованной моделью

Важно отметить, что в результате активации опубликованной модели будут выполняться обращения к опубликованной конечной точке, и поэтому они будут добавляться к лимиту использования ключа подписки на LUIS. Документ JSON, полученный в результате вышеупомянутого тестирования, приводится ниже:

```
{
  "query": "schedule an appointment with the dentist",
  "topScoringIntent": {
    "intent": "ScheduleAppointment",
    "score": 0.9981268
  },
  "intents": [
    {
      "intent": "ScheduleAppointment",
      "score": 0.9981268
    },
    {
      "intent": "None",
      "score": 0.0598709881
    },
    {
      "intent": "Greet",
      "score": 7.16493842E-09
    }
  ],
  "entities": [
    {
      "entity": "dentist",
      "type": "DoctorType",
      "startIndex": 33,
      "endIndex": 39,
      "score": 0.95532
    }
  ]
}
```



Простые сущности

Сущность DoctorType, которую вы создали ранее, является примером простой сущности. Простая сущность – это тип специальной сущности в LUIS, которая используется для разметки проблемно-ориентированных ключевых слов вашего приложения. Она является самой простой из всех типов специальных сущностей.

Другие специальные сущности делятся на иерархические и составные. Они являются специальными, потому что, в отличие от предварительно подготовленных сущностей, они должны быть правильно натренированы на высказываниях и признаках (подробно о них позднее), тогда как предварительно подготовленные сущности заранее натренированы и готовы к применению.

Давайте создадим простую сущность под названием Symptom. Выберите в левой боковой панели страницу сущностей Entities и нажмите кнопку **Add custom entity** (Добавить специальную сущность). В диалоговой панели **Add Entity** (Добавить сущность)



вить сущность) присвойте сущности имя «Symptom». Далее выберите тип сущности **Simple** (Простая) и нажмите кнопку **Save** (Сохранить). Сущность Symptom появится вместе с сущностью DoctorType.

Вот и все. Больше ничего здесь делать не нужно. В отличие от намерения, вы не можете добавлять специально для сущности высказывания. Еще один важный момент касается возможности многократного использования сущности. Сущность не привязана к намерению. Она может и должна использоваться во всех намерениях.

Сверившись с табл. 6.1, мы увидим, что единственным намерением, где используется сущность Symptom, является GetCondition. Это намерение еще не существует. Создайте это намерение и добавьте в него несколько высказываний, как показано на рис. 6.10.

<input type="checkbox"/> i am feeling [pain] in my [lower abdomen], and there is some [swelling] as well.	Tokens View
<input type="checkbox"/> i am feeling [confusion] and [anxiety]. i [vomited] a few minutes ago	
<input type="checkbox"/> i have [pain]. [swelling] and [stiffness] in my [ankle]	
<input type="checkbox"/> i am feeling [\$Symptom] in my [\$BodyPart], and there is some [\$Symptom] as well	
<input type="checkbox"/> i am feeling [\$Symptom] and [\$Symptom]. i [\$Symptom] a few minutes ago	Entities View
<input type="checkbox"/> i have [\$Symptom], [\$Symptom] and [\$Symptom] in my [\$BodyPart]	

Рис. 6.10 ♦ Несколько примеров высказываний для намерения GetCondition

Добавьте еще больше связанных высказываний, подобных и отличающихся. Постарайтесь предусмотреть максимальное количество признаков и частей. Для справки сделайте запросы в Интернете и узнайте наиболее распространенные заболевания и их симптомы. На рис. 6.11 показаны результаты тестирования после тренировки намерения с числом высказываний около 10.

The screenshot shows the LAINB Labels view interface. On the left, there is a text input field with the placeholder "Type a test utterance & press Enter". Below it, several test utterances are listed:

- why is there [\$Symptom] in my [\$BodyPart] ?
- i have a sharp [\$Symptom] in my [\$BodyPart]
- feeling [\$Symptom] and nauseatic

To the right of the test utterances, the "Current version results" section displays the top scoring intent:

Top scoring intent
GetCondition (0.79)

Other intents
None (0.07) Greet (0.01) ScheduleAppointment (0.01)

Below this, the "Published version results" section shows the same top scoring intent:

Top scoring intent
GetCondition (0.79)

Other intents
None (0.07) ScheduleAppointment (0.01) Greet (0.01)

Рис. 6.11 ♦ Чем больше вы тренируете, тем лучше результаты.
Высказывание «почему у меня <symptom> в <body part>»
имеет низкий уровень уверенности. Кроме того, «nauseatic»
не идентифицирован как симптом. Промахи подобного рода могут быть
легко исправлены большим количеством тренировок

Составные сущности

Рассмотрим приведенный ниже документ JSON:

```
{  
  "query": "there is pain in my ankles, swelling and stiffness in my knees",  
  "topScoringIntent": {  
    "intent": "GetCondition",  
    "score": 0.9803361  
  },  
  "intents": [...],  
  "entities": [  
    {  
      "entity": "ankles",  
      "type": "BodyPart",  
      "startIndex": 20,  
      "endIndex": 25,  
      "score": 0.920250833  
    },  
    {  
      "entity": "knees",  
      "type": "BodyPart",  
      "startIndex": 57,  
      "endIndex": 61,  
      "score": 0.9399164  
    },  
    {  
      "entity": "pain",  
      "type": "Symptom",  
      "startIndex": 9,  
      "endIndex": 12,  
      "score": 0.9281677  
    },  
    {  
      "entity": "swelling",  
      "type": "Symptom",  
      "startIndex": 28,  
      "endIndex": 35,  
      "score": 0.9131367  
    },  
    {  
      "entity": "stiffness",  
      "type": "Symptom",  
      "startIndex": 41,  
      "endIndex": 49,  
      "score": 0.9450974  
    }  
  ]  
}
```



ЛАНЬ®



ЛАНЬ®

Для данного высказывания служба LUIS смогла успешно определить намерение и все сущности. Но этого недостаточно, чтобы правильно предсказать находящуюся в основе причину. Почему? Поскольку участвуют два набора симптомов, одни

из которых связаны с лодыжками, а другие – с коленями. Если не связать симптомы с частями тела, можно ли получить правильный диагноз? В конце концов, «отек в лодыжках» и «боль в коленях» могут относиться к совершенно разным медицинским показаниям.

Вам может прийти неплохая мысль. По каждой сущности в документе JSON возвращаются начальный индекс startIndex и конечный индекс endIndex. Можно задействовать индекс, чтобы в свой программный код поместить условие для связи всех симптомов, которые встречаются **перед** названием части тела в высказывании о части тела. На первый взгляд, неплохая мысль. Но что, если будет высказывание «my ankles have pain, my knees have swelling and stiffness» («у меня болят лодыжки, в коленях отек и скованность»)? Или, что еще хуже, «there is pain in my ankles, also my knees have swelling and stiffness» («болят лодыжки, и к тому же в коленях отек и скованность»)? Логика только на основе индекса здесь терпит полный провал.

Составная сущность помогает определять связи среди сущностей. Другими словами, она помогает создавать **группу** сущностей. Составная сущность – это логический контейнер/обертка для сущностей, существующих в вашем приложении. Она определяет отношение принадлежности (has-a). Например, вы можете иметь составную сущность SymptomGroup, состоящую из сущностей Symptom и BodyPart. Вернитесь к странице сущностей Entities и создайте составную сущность, как показано на рис. 6.12.

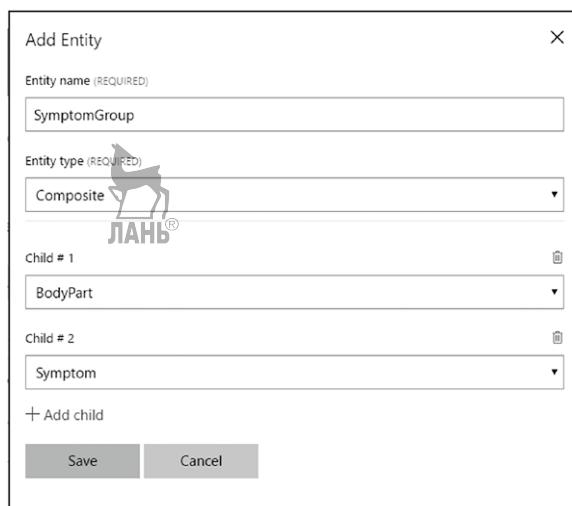


Рис. 6.12 ♦ Создание составной сущности в LUIS

Работа наполовину сделана. Сейчас основная работа состоит в разметке высказываний вашей новой составной сущности. Перейдите на страницу намерения GetCondition и добавьте высказывание «there is pain in my ankles, swelling and stiffness in my knees» («ощущается боль в лодыжках, отек и скованность в коленях»). Разметьте симптомы и части тела, как вы это делали обычно. Теперь щелкните по первому симптуму «pain» («боль»), затем нажмите **Wrap in composite entity**



(Обернуть в составную сущность) и выберите SymptomGroup. Начнется процесс обертывания. Вы увидите фигурную начальную скобку перед словом «pain». Щелкните мышью на слове «ankles» («лодыжки»), чтобы закончить обертывание. Вы успешно добавили составную сущность в высказывание. Проделайте то же самое для остальной части «swelling and stiffness in my knees» («отек и скованность в коленях»).

Натренируйте и опубликуйте приложение. Теперь LUIS вернет массив составных сущностей, каждый с дочерними сущностями. Написать логику вычисления правильного медицинского показания (показаний) теперь гораздо легче:

```
{
  "query": "there is pain in my ankles, also my knees have swelling and stiffness",
  "topScoringIntent": {
    "intent": "GetCondition",
    "score": 0.9933607
  },
  "intents": [...],
  "entities": [...],
  "compositeEntities": [
    {
      "parentType": "SymptomGroup",
      "value": "pain in my ankles",
      "children": [
        {
          "type": "BodyPart",
          "value": "ankles"
        },
        {
          "type": "Symptom",
          "value": "pain"
        }
      ]
    },
    {
      "parentType": "SymptomGroup",
      "value": "knees have swelling and stiffness",
      "children": [
        {
          "type": "BodyPart",
          "value": "knees"
        },
        {
          "type": "Symptom",
          "value": "swelling"
        },
        {
          "type": "Symptom",
          "value": "stiffness"
        }
      ]
    }
  ]
}
```



Иерархические сущности

Иерархические сущности похожи на составные тем, что в иерархических есть дочерние сущности. Но подобие на том и заканчивается. Вместо определения отношения принадлежности иерархическая сущность определяет отношение классификации (род–вид, is-a). Кроме того, эта сущность не опирается на существующие простые сущности. Вы создаете дочерние сущности, создавая родительскую сущность. Удаление сущности также удаляет все ее дочерние сущности. Однако с составными сущностями дело обстоит иначе. Составные сущности при удалении оставляют дочерние сущности нетронутыми – удаляется только отношение.

Например, иерархическая сущность **Location** (Местоположение) может иметь две дочерние: **Source** (Исходный пункт) и **Destination** (Пункт назначения). Точно так же **PatientName** (Имя пациента) может иметь **FirstName** (Имя) и **LastName** (Фамилия) в качестве дочерних сущностей. Используйте иерархические сущности, когда вы хотите иметь группу подобных сущностей, у которых с его родителем имеется отношение род–вид.

 Каждое приложение с поддержкой LUIS сопровождается максимальным пределом общего количества сущностей, которые оно может иметь. Во время написания данной книги предел равнялся 30. Одна иерархическая сущность с 5 дочерними приводит к добавлению к общему количеству сущностей всего 1 сущности. Каждая иерархическая сущность может иметь до 10 дочерних. При планировании своего приложения может возникнуть желание использовать иерархические сущности, чтобы уменьшить количество сущностей, если ожидается, что приложение будет иметь большое их количество.

Другая причина использовать иерархическую сущность состоит в том, чтобы принудить LUIS распознавать логические группы сущностей, такие как полное имя. Попробуйте на отдельном приложении: создайте две простые сущности под названием **FirstName** и **LastName**. Теперь соответствующим образом разметьте высказывания, которые приведены ниже:

- «мое имя – анураг бандари»;
- «я – нишит патак»;
- «привет, я – роит».

Теперь, когда облачная служба LUIS столкнется с выражением «я – анураг бандари», она правильно определит **FirstName** и **LastName**. Службе LUIS, однако, не удастся распознать сущности для имен, для которых она не была натренирована, таких как «я – пранка чопра». Тем не менее LUIS может распознавать широко распространенные английские имена и фамилии, но индийские, китайские, европейские имена и т. д. распознать не сможет.

Чтобы эту ситуацию исправить, используйте иерархическую сущность с именем и фамилией в качестве дочерних сущностей. Это заставит LUIS распознавать шаблоны, к примеру такие, где пара слов, которые следуют за «я –» или «мое имя –», будут распознаваться как сущности.

Перейдите на страницу сущностей и создайте новую иерархическую сущность под названием **PatientName**, как показано на рис. 6.13.

Сущность **PatientName** вам понадобится для намерения **PatientDetails**. Давайте создадим и натренируем намерение **PatientDetails**, как показано на рис. 6.14.

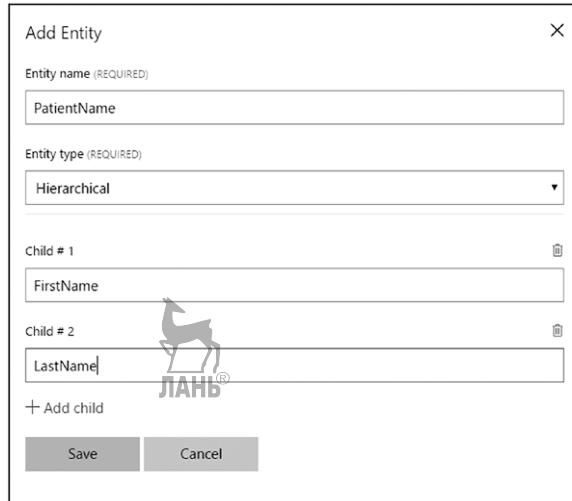


Рис. 6.13 ❖ Создание иерархической сущности в LUIS

		Save	Discard	Delete	Reassign Intent	Labels view (Ctrl+E): Entities	Search in utterances ...	Y
<input type="checkbox"/>	Utterance text					Predicted intent		
<input type="checkbox"/>	name [\$PatientName::FirstName] [\$PatientName::LastName], age 29 years old					Not trained		
<input type="checkbox"/>	my name is [\$PatientName::FirstName] [\$PatientName::LastName]. my age is 47 .					Not trained		
<input type="checkbox"/>	i am [\$PatientName::FirstName] [\$PatientName::LastName] . i am 29 years old .					Not trained		

Рис. 6.14 ❖ Разметка высказываний иерархической сущности

В документе JSON обнаруженные сущности возвращаются в формате Родитель::Потомок:

```
"entities": [
  {
    "entity": "апуаг",
    "type": "PatientName::FirstName",
    "startIndex": 11,
    "endIndex": 16,
    "score": 0.9718365
  },
  {
    "entity": "bhandari",
    "type": "PatientName::LastName",
    "startIndex": 18,
    "endIndex": 25,
    "score": 0.791057765
  }
]
```



i Теперь, когда вы знаете, как создавать иерархические сущности, создайте еще одну под названием **AppointmentTime** (Время приема) для использования с намерением **ScheduleAppointment** (Назначить визит). Обратитесь к табл. 6.1 для примера высказывания.

Предварительно подготовленные сущности

Во время тренировки намерения **PatientDetails** вы не обрабатываете возраст пациента. Для распознавания возраста нужна отдельная сущность. Можно создать простую сущность под названием **PatientAge** и затем ее натренировать метками, в частности «29 лет», «52 года» и т. д. Либо вместо этого можно использовать предварительно подготовленную сущность **age**.

Предварительно подготовленная сущность – это готовая к использованию сущность, которая уже была тщательно натренирована компанией Microsoft посредством платформы Bing. LUIS поставляется с большим количеством предварительно подготовленных сущностей, таких как:

- **age** (10-месячный, 19 лет, 58 лет);
- **datetime** (23 июня 1976, июль 11 2012, 7:00, 18:49, завтра в 7:00);
- **dimension** (2 мили, 650 квадратных километров, 9350 футов);
- **e-mail** (user@site.net, user_name@mysite.com.eg, user.Name12@website.nets);
- **geography** (Антарктида, Португалия, Дубай, округ Синьцзян, озеро Понтчар-трейн, CB3 0DS);
- **number** (десять, сорок два, 3.141, 10К).

Кроме того, есть еще **encyclopedia**, **money**, **ordinal**, **percentage**, **phonenumbers**, **temperature** и **url**. Настоятельно рекомендуем вместо создания специальных сущностей использовать предварительно подготовленные сущности везде, где это возможно.

✓ В одно приложение добавлять одну предварительно подготовленную сущность можно только один раз.

Добавьте в свое приложение с поддержкой LUIS предварительно подготовленную сущность **age**. Эту сущность вы найдете на странице сущностей. Вот и все. В дальнейшем размечать намерение **PatientDetails** не требуется. В следующий раз, когда вы будете тренировать и тестировать высказывания, возраст **age** будет распознаваться автоматически (см. рис. 6.15).

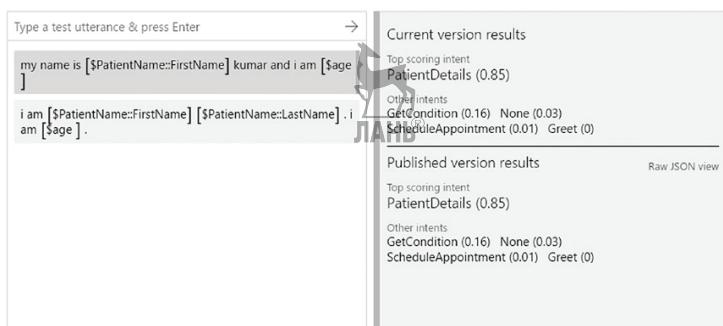


Рис. 6.15 ♦ После того как предварительно подготовленная сущность добавлена в приложение с поддержкой LUIS, все высказывания со словами и фразами, которые совпадают с шаблоном предварительно подготовленной сущности, обнаруживаются автоматически

Вот документ JSON сущности age:

```
{  
  "entity": "29 years old",  
  "type": "builtin.age",  
  "startIndex": 27,  
  "endIndex": 38,  
  "resolution": {  
    "unit": "Year",  
    "value": "29"  
  }  
}
```



ДОБАВЛЕНИЕ СПИСКА ФРАЗ

Помните, при создании табл. 6.1 вам было предложено добавить в таблицу дополнительные высказывания, чтобы охватить как можно больше узких специальностей врачей? Мы предложили расширить и список симптомов. С помощью списка фраз вам это делать не придется.

В зависимости от предметной области и требований могут иметься сотни вариантов высказываний и десятки значений сущностей, участвующих в этих высказываниях. Это равноценно тысячам перестановок и комбинаций во время тренировки LUIS. *Список фраз* – это список разделенных запятой возможных значений сущности или фраз в высказывании.

Например, список фраз под названием SeeDoctor (встретиться с врачом) может иметь значения «встретиться с врачом», «проконсультироваться с доктором», «запланировать визит к врачу», «посетить врача» и т. д. Все эти фразы имеют одно значение. Можно сказать, что эти фразы взаимозаменяемые.

Аналогичным образом список фраз Cities может иметь такие значения, как «Лондон», «Нью-Дели», «Париж», «Чикаго» и т. д., однако эти возможные значения не взаимозаменяемые. Лондон – это не то же самое, что и Нью-Дели.

Luis позволяет создавать и взаимозаменяемые, и взаимонезаменяемые списки фраз. Рисунок 6.16 показывает список фраз TypesOfDoctor (Специализации врача), и этот список намного улучшит качество результатов в вашем приложении. Из левой боковой панели перейдите на страницу **Features** (Признаки) и в разделе признаков списка фраз **Phrase list features** нажмите кнопку **Add phrase list** (Добавить список фраз).

Натренируйте и опубликуйте модель. Теперь, если протестируйте высказывание, например, «запишите меня к невропатологу», вы увидите, что «невропатолог» правильно идентифицирован как сущность DoctorType. Если этого не происходит, добавьте еще несколько высказываний с двумя или тремя значениями из этого списка фраз.



Рис. 6.16 ♦ Добавление списка фраз в LUIS

РЕКОМЕНДУЕМЫЕ ПОСЛЕДУЮЩИЕ ШАГИ

Приводимые ниже разделы предлагают несколько идей в отношении последующих шагов.

Активное обучение посредством рекомендованных высказываний

Когда вы начинаете создавать приложение с поддержкой LUIS, то используете группу подлежащих тренировке высказываний, о которых вы и/или некоторые другие подумали во время этапа планирования. Когда ваше рабочее приложение будет работать и люди начнут им пользоваться, это приложение получит много пользовательских запросов, сильно отличающихся от начальной группы высказываний.

Скорее всего, на большинство этих запросов служба LUIS ответит правильно и уверенно. Но для многих запросов правильного ответа не найдется. Со временем LUIS продолжит накапливать высказывания, не знакомые приложению. Именно этот обширный объем данных наводит разработчиков на мысль провести дальнейшую самостоятельную тренировку. Такой процесс тренировки называется *активным обучением*. Одно дело – натренировать модель на исходных высказываниях, и другое – выполнить тренировку на данных, полученных от настоящих пользователей.

Рисунок 6.17 показывает несколько рекомендованных высказываний для на-мерения GetCondition (Получить медицинское показание). Выбрав эти высказывания и нажав **Save** (Сохранить), вы дадите службе LUIS свое согласие, что результат этого высказывания правильный. Как вариант вы можете разметить пропущенные сущности, изменить неправильно идентифицированные либо полностью переназначить намерение. Эта работа должна выполняться вами и вашими коллегами-разработчиками регулярно, чтобы улучшить ответы вашего приложения и сделать его более естественным.

The screenshot shows the Microsoft LUIS Overview interface. At the top, there are tabs for 'Dashboard' and 'Suggested utterances'. On the right, there is a logo for 'ЛАНЬ®' (Lany) featuring a stylized deer head. Below the tabs, a dropdown menu says 'Suggest utterances for : GetCondition'. The main area displays a table of recommended utterances:

Utterance text	Suggested Intent
my name is [akshay] kumar and i am [49 - years - old]	0.74 PatientDetails
i am [akshay] [kumar] . i am [49 years old] .	0.7 PatientDetails
schedule an appoint with a [psychiatrist]	0.73 ScheduleAppointment
i have a [headache] and am feeling [nauseatic]	0.97 GetCondition

At the top of the table, there are buttons for 'Save', 'Discard', and 'Delete'. To the right, there is a 'Labels view (Ctrl+E)' dropdown set to 'Tokens', a search bar 'Search in utterances ...', and a magnifying glass icon.

Рис. 6.17 ❖ Рекомендованные высказывания на обзорной странице Overview

Доступ к рекомендованным высказываниям можно получить с обзорной страницы Overview, со страницы подробной информации о намерении либо со страницы сущностей.

Использование API LUIS для автоматизации

LUIS поставляется вместе с набором функционально богатых RESTful API, который позволяет разработчикам управлять приложением в полностью программном режиме. Такие задачи, как создание намерений, сущностей и списков фраз, добавление и разметка высказываний, а также тренировка и опубликование модели, могут быть автоматизированы посредством API.

Желание применить этот API может возникнуть по целому ряду причин:

- создать пользовательский веб-сайт с целью тренировки и активного обучения;
- извлечь подробную статистику о своем приложении с целью создания панелей мониторинга и графиков;
- создать совершенно новую версию веб-сайта с поддержкой LUIS, если по каким-то причинам сайт, предлагаемый Microsoft (<https://www.luis.ai>), вас не удовлетворяет.

Чтобы увидеть все варианты или исследовать все функциональные возможности, предоставляемые этим API, посетите страницу <https://www.luis.ai/home/help#api-docs> или выполните поиск в Интернете, введя поисковый запрос «luis programmatic api».

ИНТЕГРАЦИЯ LUIS со СРЕДОЙ Bot FRAMEWORK

Перед тем как продолжить, настоятельно призываем вернуться к главе 3, чтобы быстро освежить свои знания инфраструктуры Bot Framework. В той главе вы создали статический вид бота, т. е. который отвечал на пользовательские запросы, основываясь на простых условиях в формате если-то. Но при этом вы получили основное представление о том, как использовать Bot Framework SDK для быстрого создания чат-бота.

Приступим теперь к созданию интеллектуального бота, который использует возможности службы LUIS для понимания естественного языка, чтобы отвечать на вопросы пользователей. У вас есть готовое приложение с поддержкой LUIS, которое вы ранее сконструировали. Теперь пора его объединить со средой Bot Framework.

Добавление поддержки LUIS в среду Bot Framework потребует применения ее диалогов. В результате подход будет немного отличаться от подхода, применявшегося в главе 3. Поэтому вместо повторного использования кода бота HealthCheckupBot вы создадите с нуля еще один проект. И присвоим этому боту более красивое имя.

Итак, давайте сначала переименуем бота. В реальной жизни имя HealthCheckupBot не только звучит неестественно, но и показывает, что вы имеете дело с автоматизированным программным обеспечением. Как бы вы его назвали? **Dr. Checkup** (Доктор Медосмотр) кажется то, что надо!

Создание проекта в Visual Studio

Запустите Visual Studio и создайте новый проект, в качестве типа выбрав Bot Framework. Мы так делали в главе 3. Присвойте новому проекту имя DrCheckup. В результате будет создан полнофункциональный бот, который можно запускать и тестировать с использованием эмулятора Bot Framework. Однако пока бот лишь по умолчанию отправляет назад пользователю введенный им текст. Всю логику можно найти в **Контроллеры > MessagesController.cs**. Давайте оставим здесь все, как есть. Чтобы интегрировать приложение с поддержкой LUIS, вам нужно создать класс **LuisDialog**.

В обозревателе решений щелкните правой кнопкой мыши на проекте DrCheckup и выберите команду **Добавить > Новая папка**. Назовите вновь созданную папку **Dialogs** (Диалоги). Теперь в папке **Dialogs** создайте новый класс с именем **HealthCheckupDialog.cs**. Этот класс будет производным от **LuisDialog**. Листинг 6.1 показывает шаблонный код.

Листинг 6-1 ♦ Шаблонный код для **HealthCheckupDialog.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Bot.Builder.Dialogs;
using Microsoft.Bot.Builder.Luis;
using Microsoft.Bot.Builder.Luis.Models;
using System.Threading.Tasks;

namespace DrCheckup.Dialogs
{
    [LuisModel("luis-app-id", "luis-subscription-key")]
    [Serializable]
    public class HealthCheckupDialog : LuisDialog<object>
    {
    }
}
```

Передайте свой идентификатор приложения и свой ключ подписки в атрибут класса **LuisModel**. Получите идентификатор приложения со страницы **Overview** при-

ложения и ключ подписки – со страницы My Keys в LUIS. Теперь вы выполнили все настройки, чтобы приступить к обработке намерений в этом классе.

Обработка намерения без сущностей

Давайте начнем с простого. Какое намерение в вашем приложении является самым элементарным? Разумеется, намерение, связанное с приветствием, – Greet. Высказывания, которые поставлены в соответствие намерению Greet, просты. Кроме того, это намерение не имеет связанных с ним сущностей. Чтобы обрабатывать его в боте, в классе HealthCheckupDialog создайте приведенный ниже метод:

```
[LuisIntent("Greet")]
public async Task Greet(IDialogContext context, LuisResult result)
{
}
```

Вот как он работает:

1. Бот получает сообщение от пользователя.
2. Бот отправляет сообщение в сконфигурированное приложение LUIS.
3. LUIS возвращает намерение с самой высокой оценкой.
4. Бот проверяет наличие в классе LuisDialog метода, соответствующего обнаруженному намерению. Если он его находит, то исполняется программный код внутри этого метода.

Другими словами, метод с точно таким же именем, что и соответствующее ему намерение, или с именем намерения, переданным в атрибут LuisIntent, будет выступать в качестве обработчика этого намерения. Любое из условий должно быть истинным; для удобства мы приводим оба условия. Внутри метода Greet программный код отсутствует. Давайте добавим приведенный ниже исходный код, чтобы отправлять соответствующее сообщение назад пользователю в качестве ответа бота:

```
[LuisIntent("Greet")]
public async Task Greet(IDialogContext context, LuisResult result)
{
    string message = "Hello, there. Tell me your symptoms and I will tell you possible
    conditions. " +
        "Or ask me to schedule an appointment with a doctor. " +
        "But first I need to know your name and age. ";
    await context.PostAsync(message);
    context.Wait(MessageReceived);
}
```

Инструкция context.PostAsync(message) отправляет ваше сообщение пользователю. Инструкция context.Wait(MessageReceived) приостанавливает диалог до тех пор, пока не будет получено следующее сообщение пользователя.

Мы еще не исследовали аргумент result. Однако займемся этим чуть ниже, но пока знайте, что он обычно используется для извлечения сущностей.

Подготовка своего бота для использования HealthCheckupDialog

Другие намерения вы будете обрабатывать позже. Помните, всему свое время. Во-первых, давайте протестируем то, что было уже сделано. Чтобы бот вызывал

LUIS, необходимо настроить исходный код в `MessagesController.cs`, чтобы он направлял полученные сообщения в `HealthCheckupDialog`.

В методе POST контроллера `MessageController` замените весь исходный код в условии `if (activity.Type == ActivityTypes.Message)` на приведенный ниже:

```
if (activity.Type == ActivityTypes.Message)
{
    await Conversation.SendAsync(activity, () => new HealthCheckupDialog());
```

Теперь все действия с типом `Message` будут перенаправляться в `HealthCheckupDialog`.

Тестирование бота в эмуляторе

Выполните свой исходный код в Visual Studio, нажав клавишу **F5**. В результате ваш браузер по умолчанию будет открыт на странице <http://localhost:3979/>.

Откройте эмулятор Bot Framework из стартового меню. Введите в строку поиска <http://localhost:3979/api/messages>. Оставьте поля идентификатора приложения и пароля приложения пустыми и щелкните по кнопке **Connect** (Соединить). После того как связь с вашим ботом будет установлена, отправьте приветственное сообщение. Вы должны увидеть ответ, как на рис. 6.18.

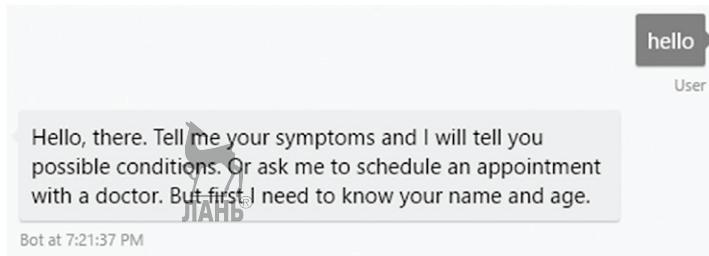


Рис. 6.18 ♦ Тестирование бота с поддержкой LUIS в эмуляторе Bot Framework

Работает!

Обработка намерения с полным комплектом сущностей

Давайте напишем метод для обработки `PatientDetails`, намерения, которое связано с сущностями. Но сначала добавим несколько переменных класса, которые будут выступать в качестве состояния диалога. В верху класса `HealthCheckupDialog` добавьте приведенный ниже фрагмент кода:

```
string patientFirstName, patientLastName, patientAge;
```

Эти три переменные, которые относятся к подробной информации о пациенте (пользователе), станут состоянием диалога. Под состоянием имеется в виду, что после присвоения им значений они останутся постоянными в течение сеанса пользователя с ботом. Поэтому, когда ваш бот получает высказывание, которое соответствует намерению `PatientDetails`, ваши переменные состояния получат

значения, извлеченные из соответствующих сущностей, PatientName и age. Подробная информация о пациенте должна сохраняться для последующих высказываний пользователя, чтобы их можно было использовать для записи его к врачу (самым важным из них является имя) либо для определения медицинского показания (показаний) на основе симптомов (тут будет полезна информация о возрасте).

Теперь в класс HealthCheckupDialog добавьте приведенный ниже метод:

```
[LuisIntent("PatientDetails")]
public async Task PatientDetails(IDialogContext context, LuisResult result)
{
    EntityRecommendation patientFirstNameEntity, patientLastNameEntity, patientAgeEntity;
    if (!result.TryFindEntity("PatientName::FirstName", out patientFirstNameEntity))
    {
        patientFirstNameEntity = new EntityRecommendation() { Entity = "Unknown" };
    }
    if (!result.TryFindEntity("PatientName::LastName", out patientLastNameEntity))
    {
        patientLastNameEntity = new EntityRecommendation() { Entity = "Unknown" };
    }
    if (!result.TryFindEntity("builtin.age", out patientAgeEntity))
    {
        patientAgeEntity = new EntityRecommendation() { Resolution = new Dictionary<string, object>() { { "value", "29" } } };
    }
    patientFirstName = patientFirstNameEntity.Entity;
    patientLastName = patientLastNameEntity.Entity;
    patientAge = patientAgeEntity.Resolution["value"].ToString();
    string message = $"Hi, {patientFirstName}. How can I help you today?";
    await context.PostAsync(message);
    context.Wait(MessageReceived);
}
```

Давайте рассмотрим его по частям:

```
EntityRecommendation patientFirstNameEntity, patientLastNameEntity, patientAgeEntity;
```

Класс EntityRecommendation представляет объект в массиве entities в результирующем документе JSON. Он имеет такие свойства, как Entity, Type, Score, StartIndex и т. д. Вы создали три экземпляра EntityRecommendation, представляющих три возможные сущности, которые вы можете получить с высказыванием PatientDetails:

```
if (!result.TryFindEntity("PatientName::FirstName", out patientFirstNameEntity))
{
    patientFirstNameEntity = new EntityRecommendation() { Entity = "Unknown" };
}
```

Метод TryFindEntity класса LuisResult отыскивает в результате сущность с заданным типом. В случае успешного поиска данный метод присваивает параметру out типа EntityRecommendation найденную сущность и возвращает true. В слу-

чае неуспеха он возвращается `false`, не затрагивая параметра `out`. Приведенное выше условие `if` является хорошим способом установки значения по умолчанию для сущности в случае, если она не присутствует в результате, полученном из LUIS.

Напомним, что `PatientName` является иерархической сущностью. Тип сущности `PatientName::FirstName` ссылается на своего потомка, `FirstName`. Чтобы найти простые сущности, передайте имя сущности без двоеточий (к примеру, `Symptom`).

```
if (!result.TryFindEntity("builtin.age", out patientAgeEntity))
{
    patientAgeEntity = new EntityRecommendation() { Resolution = new Dictionary<string,
        object>() { { "value", "29" } } };
}
```

Вычисленное значение предварительно подготовленной сущности возвращается в ее свойство `resolution` в JSON. Вычисленное значение удобнее значения, возвращенного в свойство `entity`. Значение `resolution.value` хранит числовую часть, тогда как `resolution.type` хранит единицы измерения. В C# `Resolution` представляет собой словарь в формате `<string, object>`.

```
patientFirstName = patientFirstNameEntity.Entity;
patientLastName = patientLastNameEntity.Entity;
patientAge = patientAgeEntity.Resolution["value"].ToString();
```

Вы присваиваете переменным состояния значения соответствующих им сущностей. И в заключение вы передаете назад пользователю сообщение, которое содержит извлеченную сущность: имя пациента.

Запустите бот и протестируйте его в эмуляторе. Вы должны получить ответ, который выглядит, как показано на рис. 6.19.

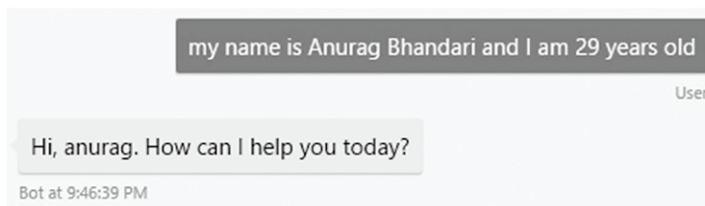


Рис. 6.19 ❁ Ответ, который содержит сущность (имя), извлеченную из сообщения пользователя

Обработка намерения с составными сущностями

Ранее вы видели, что составные сущности возвращаются в массиве `compositeEntities` в документе JSON. Объект составной сущности выглядит следующим образом:

```
{
    "parentType": "SymptomGroup",
    "value": "pain in my ankles",
    "children": [
        {
            "type": "BodyPart",
```

```

        "value": "ankles"
    },
    {
        "type": "Symptom",
        "value": "pain"
    }
]
}

```



C# управляет этой структурой. Результат экземпляра `LuisResult` имеет свойство под названием `CompositeEntities` и представляет собой массив составных сущностей, которые подчиняются вышеупомянутой структуре. Как таковая эта структура требует небольшой доработки, чтобы получать доступ к потомкам составной сущности. Неплохой идеей будет создание специальных моделей, которые будут представлять ваши составные сущности, и затем приданье этим моделям вышеупомянутой структуры.

В проекте `DrCheckup` добавьте новую папку под названием `Models`. В этой папке создайте приведенный ниже класс:

```

public class SymptomGroup
{
    public string BodyPart { get; set; }
    public string[] Symptoms { get; set; }
}

```

Теперь давайте создадим обработчика для намерения `GetCondition`. Поскольку это намерение сопряжено с составными сущностями, в первую очередь нужно поставить `result.CompositeEntities` в соответствие вашей модели.

```

[LuisIntent("GetCondition")]
public async Task GetCondition(IDialogContext context, LuisResult result)
{
    IList<SymptomGroup> symptomGroups = new List<SymptomGroup>();
    foreach (CompositeEntity sg in result.CompositeEntities)
    {
        if (sg.ParentType == "SymptomGroup")
        {
            symptomGroups.Add(new SymptomGroup()
            {
                BodyPart = sg.Children.Where(x => x.Type == "BodyPart").
                    FirstOrDefault().Value,
                Symptoms = sg.Children.Where(x => x.Type == "Symptom").
                    Select(x => x.Value).ToArray()
            });
        }
    }
}

```

Здесь вы задействуете немного LINQ, чтобы выборочно отобрать значения из объектов `CompositeEntity`. Если вы не знакомы с LINQ, представьте SQL-подобный синтаксис. LINQ значительно упрощает опрос объектов `IEnumerable` в C#, которые формируются прямо в программном коде либо в результате выполнения SQL-запроса.

Теперь, когда мы создали три группы симптомов в соответствии с частью тела, можно вызвать сторонний API, чтобы доставить возможные медицинские показания/заболевания:

```
string[] conditions = { };

// TODO: Вызвать ApiMedic API, используя извлеченную группу(ы) симптомов

string message;
if (conditions.Length < 1)
{
    message = "I'm sorry, I could not determine your condition.";
}
else
{
    message = $"You might be suffering from {string.Join(" or ", conditions)}.";
```

 Также напишите программный код обработки ситуаций, где вместо составных сущностей напрямую возвращаются простые сущности (Symptom и BodyPart). Вспомните, что вы первоначально добавляли высказывания с намерением GetConditions, у которого не было составных сущностей. Вполне возможны ситуации, когда имеются симптомы, которые либо не затрагивают одну определенную часть тела, либо неявно связаны с одной из них, как, например, тошнота, головная боль, диарея, беспокойство и т. д.

Рисунок 6.20 показывает сущности, выделенные из высказывания «у меня боль в лодыжках, также отек и скованность в коленях» во время выполнения программного кода в режиме отладки.



Рис. 6.20 ♦ Составные сущности в ответе, полученном из LUIS, были успешно соотнесены с вашими специальными моделями

Обработка намерения None

В завершение давайте также обработаем намерение None.

```
[LuisIntent("")]
public async Task None(IDialogContext context, LuisResult result)
{
    string message = "Sorry, I did not understand.";
    await context.PostAsync(message);
    context.Wait(MessageReceived);
}
```

УПРАЖНЕНИЕ

Завершите интеграцию с LUIS

Анализ раздела «Интеграция LUIS со средой Bot Framework» дает вам практические знания об обработке намерений и всех типов сущностей в среде Bot Framework. Используйте эти знания, чтобы закончить интеграцию вашего приложения HealthCheckup с поддержкой LUIS в проекте Bot Framework DrCheckup.

- Обработайте намерение `ScheduleAppointment` вместе со значениями по умолчанию для участующих сущностей. Используйте Google Календарь или API drchrono для создания назначений к врачу.
- Пользователи, повторно вышедшие на связь, часто забывают представиться. Они сразу задают свой вопрос. В этом случае их следует попросить представиться: указать свое имя и возраст.
- Попытайтесь добавить в ваш поток взаимодействий несколько элементов интерфейса пользователя. Вместо того чтобы ваш бот был чисто текстоориентированным, сделайте его гибридным приложением с разговорным интерфейсом. В официальной документации по Bot Framework обратитесь к **Connector Service** (Служба соединителя) ➤ страница **Attachments, Cards and actions** (Вложения, карты и действия). Раздел, посвященный действиям, связанным с нажатием кнопок или с картами, может вас особо заинтересовать. Например, чтобы показать несколько общих симптомов, вы можете установить кнопки, и в результате облегчите пользователям взаимодействие со своим ботом.

Наконец, узнайте больше о диалогах и подсказках в Bot Framework. Они значительно улучшают пользовательский опыт взаимодействия и сокращают усилия, связанные с программированием бота.

ДОБАВЛЕНИЕ БОТА В SKYPE

Пришло время переместить свой бот за пределы своего собственного компьютера. Давайте посмотрим, как он будет работать в ваших любимых каналах связи. В качестве первого шага необходимо опубликовать (развернуть) свой бот по доступному в сети URL-адресу. При наличии действительной подписки Azure опубликовать Bot Framework как службу очень просто. В этом нам поможет Visual Studio.

Публикация своего бота

В Обозревателе решений щелкните правой кнопкой мыши на проекте DrCheckup и выберите команду **Опубликовать**. Убедитесь с помощью подсказки **Выберите цель публикации**, что в **Microsoft Azure App Service** (Служба приложения) выбрана опция **Create New** (Создать новую). Нажмите **OK/Next** и следуйте инструкциям, чтобы создать новую службу приложения и развернуть в ней ваше приложение.

После завершения публикации запишите URL-адрес сайта вашей недавно созданной службы приложения. URL-адрес должен иметь вид `http://<имя_приложения>.azurewebsites.net`.

 Вы можете протестировать опубликованный бот в эмуляторе. Просто поместите сгенерированный URL-адрес сайта в строку поиска, оставьте поля идентификатора приложения и пароля пустыми и нажмите **Connect**. Чтобы иметь возможность тестировать удаленные URL-адреса в эмуляторе, может потребоваться установка ngrok.

Регистрация своего бота

Для регистрации своего бота на сайте Bot Framework выполните приведенные ниже шаги. Это вам позволит позже добавить свой бот в один или во все поддерживаемые каналы связи, включая Skype.

1. Обратитесь к dev.botframework.com и авторизуйтесь под своей учетной записью Microsoft.
2. Перейдите на страницу **Register a bot** (Зарегистрировать бот). Заполните раздел профиля бота по своему усмотрению.
3. В разделе **Configuration** (Конфигурация) в поле **Messaging endpoint** (Конечная точка обмена сообщениями) вставьте свой URL-адрес сайта службы развертываемого приложения, как указано ниже: *https://<имя_приложения>.azurewebsites.net/API/messages*.
4. Нажмите на **Create Microsoft App ID and password** (Создать идентификатор и пароль приложения Microsoft). В результате откроется новая вкладка/окно с недавно сгенерированным идентификатором приложения для вашего бота. В появившейся странице нажмите кнопку **Create password** (Создать пароль), чтобы сгенерировать новый пароль для вышеупомянутого идентификатора приложения. Сохраните свой идентификатор приложения и пароль в локальном текстовом файле. Microsoft повторно пароль не показывает никогда.
5. Вернитесь на страницу **Register a bot** и заполните поле идентификатора приложения.
6. С помощью вертикальной полосы прокрутки отобразите нижнюю часть страницы и прочитайте условия использования. Согласившись с условиями, поставьте флажок напротив **I agree...** и нажмите кнопку регистрации **Register**.

На данном этапе ваш бот зарегистрирован на портале разработчиков Bot Framework. Но он еще не опубликован ни на одном канале связи. Прежде чем это сделать, вернитесь в Visual Studio. В Web.config приложения DrCheckup обновите ключи `MicrosoftAppId` и `MicrosoftAppPassword` в `<appSettings>`, подставив значения, которые вы сохранили ранее. Для ключа `BotId` используйте любое значение, к примеру DrCheckup. Эти параметры конфигурации необходимы, чтобы надежно подтвердить подлинность бота на портале Bot Framework. Опубликуйте свой проект заново.

Вернитесь на веб-сайт Bot Framework. Откройте страницу **My bots** (Мои боты) и выберите DrCheckup. Если прокрутить страницу немного вниз, то вы увидите список активированных и других доступных каналов связи. По умолчанию Skype будет добавлен в список активированных каналов. Оставьте параметры настройки, предложенные по умолчанию.

С помощью полосы прокрутки отобразите верх этой страницы. Щелкните по кнопке **Test** (Протестировать) в разделе **Test connection to your bot** (Протестировать соединение со своим ботом). Через несколько секунд вы должны получить сообщение **Endpoint authorization succeeded** (Авторизация конечной точки прошла успешно). Если это не так, проверьте, что значения идентификатора и пароля приложения, которые вы использовали при регистрации бота, полностью соответствуют значениям в `Web.config` бота.

Нажмите кнопку **Publish** в правом верхнем углу, чтобы начать процесс публикации. Тщательно заполните предложенную форму. Чтобы понять, что происходит при опубликовании бота, представьте, что вы добавляете Android-приложение на Google Play Store. Все представляемые приложения должны пройти формальную проверку Google. В Play Маркет публикуются только те приложения, которые та-

ую проверку прошли. Компания Microsoft также проверяет все предоставляемые ей боты, чтобы убедиться, что все условия использования и нормы поведения выполнены. После заполнения всех полей формы нажмите кнопку **Submit for review** (Предоставить для проверки). При заполнении форм будьте особо внимательны! Когда проверка будет завершена, вам придет уведомление.

Резюме

В этой главе, воспользовавшись ранее полученными знаниями, связанными с применением естественного языка, вы создали приложение с поддержкой LUIS. Это приложение полностью готово для производственного применения. Вы также познакомились с тем, как интегрировать облачную службу LUIS в свой бот и публиковать его в справочнике Skype.

Подводя итоги, в этой главе вы:

- узнали, как создавать приложения с поддержкой LUIS;
- познакомились со структурными компонентами приложения с поддержкой LUIS, в частности намерениями, сущностями и списками фраз;
- познакомились с различными типами сущностей в LUIS;
- научились тренировать и публиковать приложения с поддержкой LUIS;
- узнали, как добавлять поддержку LUIS в приложение Bot Framework;
- научились публиковать бот на одном из поддерживаемых каналов связи.

В **следующей главе** вы познакомитесь с интеграцией речевых возможностей в приложение .NET.



Глава 7



Взаимодействие с API обработки речи

В последнее время способы взаимодействия людей и устройств меняются коренным образом. Каждый день появляются новые, революционные методы и входят в нашу жизнь. Еще десять лет назад такие методы взаимодействий воспринимались бы как чудо. Среди этих инноваций становится очень популярным речевое взаимодействие. Одна из основных причин, дающих преимущества голосовому способу взаимодействия, – в его быстроте и естественности. Сегодня все больше устройств, обладающих встроенными возможностями распознавания и синтеза речи. Взять, к примеру, персональные цифровые секретари наподобие Кортана и Сири с полностью измененным способом взаимодействия с мобильными приложениями. Или, к примеру, умные автомобили. Управляемые голосом автомобили существовали в течение долгого времени, но инновационность здесь заключается в естественном взаимодействии. Все эти интегрированные компоненты делают системы интеллектуальными и более интерактивными.

Если вы задумывались о создании общедоступного приложения, в котором будет использована речь, эта глава ответит на большинство ваших вопросов. Здесь же вы найдете ответы, как создать программное обеспечение с элементами распознавания речи, о сложностях распознавания голосов людей, о сложностях создания бесконтактного приложения и о беседах с машиной. Благодаря постоянно эволюционирующему миру когнитивных служб эти взаимодействия довольно легко осуществить в любом приложении. На самом деле в этой главе вы увидите несколько взаимодействий с использованием Microsoft Speech. Microsoft Speech API, т. е. API обработки речи, имеет более чем 20-летний опыт исследований в области анализа и синтеза речи.



Еще в 1995 г. первая версия речевого API (также именуемая SAPI) компании Microsoft стала частью Windows 95. С тех пор обработке речи уделялось огромное внимание, и были выпущены разновидности речевого API для самых разных ОС, предлагающие функциональные возможности преобразования устной речи в текст и текста в речь. С появлением API обработки речи в когнитивных службах компания Microsoft предоставила самые лучшие функциональные возможности, доступные в облаке, и они могут использоваться с помощью простых в использовании REST API.

Давайте разберемся в разновидностях API обработки речи (Speech API). В конце этой главы вы поймете, как:

- использовать API обработки речи для распознавания естественной речи;
- использовать API обработки речи, чтобы компьютер заговорил;
- использовать API обработки речи для распознавания голоса.

Способы взаимодействия с речью

Если вы хотите создать приложение с элементами речи и после его использовать, Visual Studio 2017 может предложить вам несколько вариантов обработки речи.

1. Если вы используете Windows 10 и хотите разработать и развернуть приложение на Универсальной платформе Windows, которое будет работать в разъединенных сценариях, вам необходимо использовать класс распознавателя речи Speech Recognizer.
2. Если ваше приложение требует использования каких-то других платформ либо вы хотели бы развернуть его в облачной среде, Microsoft предлагает два варианта:
 - a) скачать комплект разработчика для когнитивной обработки речи Cognitive Speech SDK;
 - b) использовать функционал, вызывая REST API когнитивных служб.

Чтобы использовать речевой функционал в клиентском приложении, вы должны установить пакет NuGet для API Bing обработки речи. Щелкните правой кнопкой мыши по своему проекту и выберите **Manage NuGet Packages**, как показано на рис. 7.1.

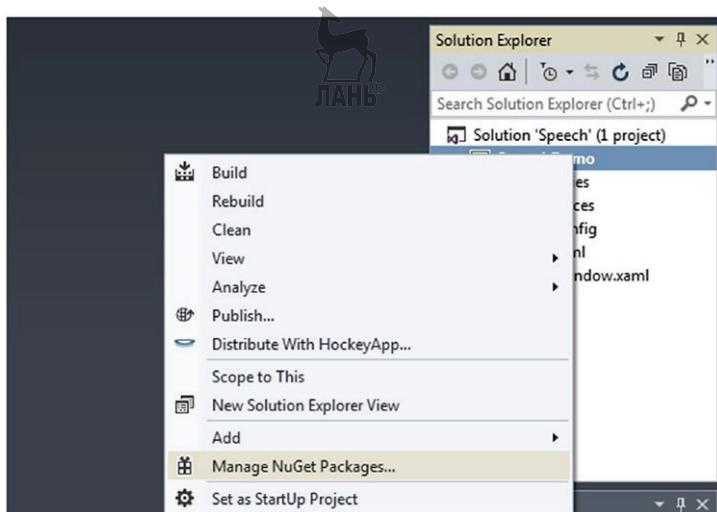


Рис. 7.1 ❖ Управление пакетом NuGet в Visual Studio

В показанном ниже окне установите источник пакета в Nuget.org. Отыщите «Bing speech». Вы должны установить пакет Microsoft.Bing.speech, как показано на рис. 7.2.

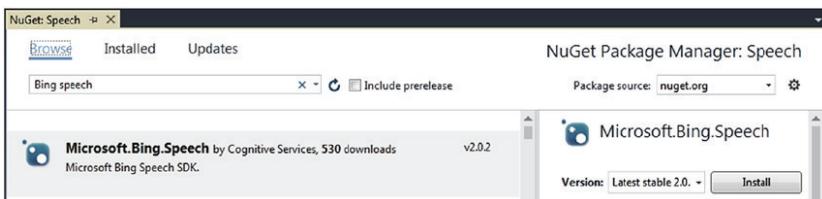


Рис. 7.2 ❖ Пакет Bing NuGet найден в Диспетчере пакетов NuGet



Когнитивные службы возникли из проекта «Project Oxford». Если выполнить поиск Oxford speech, то можно найти 32- и 64-разрядные версии пакета NuGet. Некоторые разработчики, которые экспериментируют с когнитивной обработкой речи, могут быть знакомы с пакетом NuGet для Oxford Speech. Мы настоятельно рекомендуем использовать Microsoft. Bing. speech.

Установка пакета NuGet обработки речи обеспечивает возможность создать речевое приложение, используя строго типизированный объект. После выбора проекта вся речевая библиотека будет находиться в пространстве имен `Microsoft. Bing.speech`. Доступ к клиентской библиотеке не раскрыт в подробностях, но подавляющая часть функциональности работает аналогично REST API, который мы подробно рассматриваем в этой главе.

API когнитивного поиска

API обработки речи (Speech API) обеспечивает самый легкий способ расширения возможностей вашего приложения. Это достигается с помощью управляемых речью сценариев. Speech API прост в использовании. Этот API не использует сложных речевых алгоритмов и представляет простой в применении REST API. В широком смысле функциональность Speech API может быть разбита на три главные категории:

- API Bing обработки речи:
 - распознавание речи: распознавание речи и преобразование его в текст;
 - синтез речи: преобразование текста в аудио;
 - понимание намерения (смысла) разговорного аудио;
- служба персонализированной обработки речи (ранее называвшаяся CRIS):
 - персонализированная модель языка;
 - персонализированная аудиомодель;
 - развертывание модели в конечной точке;
 - распознавание того, кто говорит.

Распознавание речи

Распознавание речи, также именуемое преобразованием речи в текст, позволяет вам обрабатывать слова, произносимые вашими пользователями в вашем приложении посредством механизма распознавания и преобразования его в текстовую форму. Эта текстовая транскрипция в конечном счете может использоваться во множестве целевых задач. Например, замечательным вариантом использования API преобразования устной речи в текст является создание мобильного решения на основе принципа общедоступности для глухих, в результате чего они смогут

читать то, что говорят другие. На самом деле области распознавания речи могут быть везде, где вы хотите, чтобы пользователь говорил, а не набирал на клавиатуре. Любое бесконтактное приложение требует наличия возможностей распознавания речи. Произносимые слова могут поступать непосредственно из микрофона, громкоговорителя или даже из звукового файла. И это не фантастика. Мы можем вводить речь, не печатая ни единого символа.

Но горькая правда в том, что большинство механизмов распознавания речи во всем мире не дают 100%-го правильного результата. Но, по сравнению с недавним прошлым, качество распознавания речи, безусловно, стало намного лучше. И ресурс для улучшения возможностей распознавания речи еще очень большой. На самом деле во время написания настоящей книги Марк Цукерберг (генеральный директор Facebook) работал над созданием Джарвиса (Jarvis).

«Системы распознавания речи в последнее время стали совершеннее, и все же ни одна система ИИ пока не настолько совершенна, чтобы понимать разговорную речь. Распознавание речи опирается на слушание того, что вы говорите, и предсказание того, что вы скажете затем, поэтому структурированную речь по-прежнему намного легче понять, чем неструктурированный разговор».

– Марк Цукерберг

Эта интересная цитата содержит серьезное конструктивное заявление, на которое стоит обратить внимание. Это сделано, чтобы вы не испугались, а задумались об обработке устной речи. Каждый день в этой области происходит новое исследование и изобретение, поэтому, возможно, к тому времени, когда вы будете читать эту книгу, данный комментарий будет неактуальным. Скрестим пальцы!



НАЧАЛО РАБОТЫ

Когнитивные службы делают удивительную работу по уходу от использования внутреннего механизма алгоритма. Все вызовы проходят через API, находящийся на облачном сервере Azure. Этот API поддерживается комбинацией работающего за кадром чрезвычайно мощного глубокого обучения и нейронной сети. Чтобы использовать REST API Bing обработки речи, необходимо выполнить следующие шаги.

1. Получить токен (маркер) доступа в формате JSON (JSON Web Token, JWT) путем вызова службы токенов.
2. Поместить JWT-токен в заголовок и вызвать API Bing обработки речи.
3. Выполнить разбор текста.

СНАЧАЛА ПОЛУЧЕНИЕ JWT-ТОКЕНА

Каждый вызов API нуждается в аутентификации перед отправкой ответа пользователю. Все запросы к конечной точке обработки речи требуют передачи токена доступа `access_token` в качестве аутентификационного заголовка. Первый шаг в потреблении конечной точки обработки речи состоит в получении своего токена доступа. `access_token` – это JWT-токен, переданный в виде строки, кодированной по стандарту Base64, в заголовке запроса на обработку речи. Чтобы получить JWT-

токен, пользователь должен отправить POST-сообщение в службу токенов вместе с ключом подписки, как показано ниже:

```
POST https://api.cognitive.microsoft.com/sts/v1.0/issueToken HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key:*****
```

Листинг 7.1 показывает способ получения нового токена в C#. Вы можете повторно использовать то же решение, которое было разработано в главе 2. В исходном коде сначала создается экземпляр HTTP-клиента, потом добавляется ключ подписки в качестве заголовка и затем POST-методом вызывается сайт токенов.

Листинг 7.1 ♦ Получение нового токена

```
private static string GetFreshToken ()
{
    using (var client = new HttpClient())
    {
        client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", ApiKey);

        var response = client.PostAsync("https://api.cognitive.microsoft.com/sts/v1.0/
issueToken", null).Result;

        return response.Content.ReadAsStringAsync().Result;
    }
}
```

Период действия токена равняется 10 минутам, поэтому ваше решение должно обеспечивать, чтобы этот символ извлекался и активировался через каждые 10 минут. Существуют самые разные способы реализации этой поддержки. Листинг 7.2 регенерирует текущий токен, прежде чем его период действия истечет. Эта стратегия также регенерирует существующий токен. В частности, будет запланировано автоматическое выполнение, когда до истечения срока действия недавно полученного токена останется одна минута.

Листинг 7.2 ♦ Регенерация текущего токена

```
private const int TokenExpiryInSeconds = 600;
private string Token;

private void getValidToken()
{
    this.token = GetFreshToken();
    this.timer?.Dispose();
    this.timer = new Timer(
        x => this.getValidToken(),
        null,
        // Указать, что токен должен выполниться по прошествии 9 мин.
        TimeSpan.FromSeconds(TokenExpiryInSeconds).Subtract(TimeSpan.FromMinutes(1)),
        TimeSpan.FromMilliseconds(-1));
    // Указывает, что эта функция выполнится всего один раз
}
```

Вы не добавили исходный код получения эксклюзивного доступа к текущему токену. Если у вас многопоточное приложение, то может возникнуть потребность заблокировать токен перед вызовом метода `getValidToken`.

Вызовы службы токенов с правильным ключом подписки будут возвращать JWT-токен как `text/Plain` с типом содержимого `application/jwt`. Возвращаемые значения JWT-токена затем могут использоваться для вызова конечной точки обработки речи для последующих вызовов.

Во время написания этой книги использование бесплатной ценовой категории продолжало варьироваться; доступно до 5К бесплатных транзакционных вызовов в месяц с оговоркой 20 вызовов в минуту. Для получения дополнительной информации посетите www.microsoft.com/cognitive-services/en-us/subscriptions.

API ПЕРСОНАЛИЗИРОВАННОЙ ОБРАБОТКИ РЕЧИ

Все аудио проходит через HTTP POST-запрос в конечную точку обработки речи. После получения своего токена доступа второй шаг состоит в выполнении еще одного POST-запроса к <https://speech.platform.bing.com/recognize> с дополнительными HTTP-параметрами запроса. Все эти параметры подробно описаны в табл. 7.1.

Таблица 7.1. Параметры, используемые для вызова API Bing обработки речи

Имя	Описание
Version	Версия API, используется клиентом. Мы использовали API версии 3.0, т. е. самую последнюю, для большинства потребностей
RequestId	Глобально-уникальный идентификатор для вызова API. Рекомендуется использовать идентификатор GUID
appId	Должен быть D4D52672-91D7-4C74-8AD8-42B1D98141A5
Format	Формат, в котором вы хотели бы возвращать данные. Мы предлагаем <code>format=json</code> для большинства сценариев. Можно также выбрать другие варианты, к примеру XML
Locale	Код языка для передаваемого аудио. Он регистронечувствительный. Поддерживается 28 локалей, количество которых будет увеличиваться с течением времени. Для большинства наших демонстрационных примеров мы используем <code>locale=en-us</code>
Device.os	Определяет ОС, из которой был сделан вызов. Вариантами могут быть Window OS, Window Phone OS, Android, iPhone OS и Xbox
InstanceId	Идентификатор GUID, который однозначно определяет устройство, делающее запрос

Листинг 7.3 показывает, как вызывать конечную точку обработки языка с соответствующими параметрами, которая принимает аудио, преобразует его в текст и возвращает его назад пользователю.

Листинг 7.3 ❖ Вызов конечной точки обработки языка

```
private GUID instanceID;
public async Task<string> GetTextFromAudioAsync(Stream audiostream)
{
    var requestUri = @"https://speech.platform.bing.com/recognize?scenarios
=smd&appid=D4D52672-91D7-4C74-8AD8-42B1D98141A5&locale=en-US&device.os=
WindowsOS&version=3.0&format=json&instanceid=instanceID3&requestid=" + Guid.
NewGuid();

    using (var client = new HttpClient())
    {
        var token = this.getValidToken();
        client.DefaultRequestHeaders.Add("Authorization", "Bearer " + token);

        using (var binaryContent = new ByteArrayContent(StreamToBytes(audiostream)))

```

```

    {
        binaryContent.Headers.TryAddWithoutValidation("content-type", "audio/
wav; codec=\"audio/pcm\"; samplerate=18000");

        var response = await client.PostAsync(requestUri, binaryContent);
        var responseString = await response.Content.ReadAsStringAsync();
        dynamic data = JsonConvert.DeserializeObject(responseString);
        return data.header.name;
    }
}
}
}

```

Если посмотреть на этот исходный код, то в первую очередь был создан URL-адрес, указывающий на конечную точку обработки речи с необходимыми параметрами. Важно гарантировать, что каждый параметр используется только один раз. В противном случае вы получите ответ с ошибкой HTTP 400. Кроме того, важно перед вызовом конечной точки обработки речи убедиться, что ваш ключ подписки действителен. В противном случае вы получите ответ с ошибкой HTTP 403. Как отмечалось ранее, JWT-токены истекают каждые 10 минут, поэтому важно вызывать метод `getValidToken`, созданный ранее перед вызовом приведенного выше метода, чтобы ваш токен гарантированно был действителен. После валидации JWT-токена его нужно передать как аутентификационный заголовок, начинающийся строковым значением `Bearer` (носитель). Все аудио необходимо конвертировать из аналоговой версии в цифровую. Для того чтобы конвертировать аудио из аналоговой версии в цифровую для передачи в API распознавания речи, используются кодеки. API распознавания речи поддерживает три типа кодеков:

- PCM;
- Siren;
- SirenSR.

В большинстве случаев вам следует использовать PCM, если только вам не нужна транскрипция, близкая к реальному времени, для таких сценариев, как конференц-связь, для которой вам потребуется кодек Siren. Затем вы вызываете API обработки речи, который возвращает ответ в формате документа JSON. Чтобы десериализовать его в объект .NET, можно применить любой конвертер JSON.

СИНТЕЗ РЕЧИ

Синтез речи, также именуемый преобразованием текста в речь (TTS), не является новым понятием. В машинах в той или иной форме, начиная с XVIII века, использовались функциональные элементы синтеза речи. TTS позволяет озвучивать слова или фразы с помощью механизма синтеза речи. Практически каждая машина на основе Windows в наше время имеет встроенный синтезатор речи, который преобразовывает текст в речь. Этот встроенный синтезатор особенно полезен для любого, кто не имеет возможности прочитать текст, напечатанный на экране. Однако здесь есть два аспекта. Во-первых, эти системы не обновляются. И во-вторых, они ресурсоемки и сильно зависят от количества свободной оперативной памяти компьютера. Поэтому они хорошо обрабатывают простые или не очень сложные сценарии. Для того чтобы строить корпоративные системы, рече-ориентированная модель должна удовлетворять следующим требованиям:

- простота в использовании;
- увеличение скорости с течением времени;
- система должна непрерывно обновляться, чтобы быть производительнее и точнее;
- должна уметь выполнять сложные вычисления;
- должна быть рентабельной;
- должна быть универсальной и доступной на всех платформах, даже на мобильных устройствах низшего ценового класса.

API Bing преобразования текста в речь обеспечивает все эти требования в легком в использовании REST API. К настоящему времени вы знакомы с использованием API преобразования текста в речь.

Аналогично преобразованию текста в речь, в API Bing преобразования речи в текст все взаимодействия выполняются с помощью HTTP POST-запросов. Все запросы к API преобразования речи в текст требуют наличия JWT-токена. В предыдущем разделе мы уже показали, как получать JWT-токен. Ключевое свойство API Bing преобразования речи в текст состоит в том, что API Bing поддерживает язык разметки синтеза речи Speech Synthesis Markup Language (SSML).

i В современном мире синтез речи применяется в различных сферах и самыми разными способами. Раньше большинство крупных компаний-разработчиков программного обеспечения занималось обработкой речи, используя собственные протоколы, которые были привязаны к конкретной платформе. Этому программному обеспечению не удавалось преодолеть проблему связанных с речью аспектов, таких как произношение и интонация, которые не были одинаково доступными. Способность задействовать связанные с речью системы является решающей потребностью в большом количестве организаций. Кроме того, недавно разработанные приложения должны сделать речь одинаково доступной на существующих платформах, и компании должны обеспечивать одинаковую доступность речи. Еще в начале 2000-х годов это было главной проблемой для всех крупных поставщиков программного обеспечения, и они все хотели использовать широко принятый и одобренный комплект протоколов. Чтобы решить эту проблему, лидеры отрасли, такие как Microsoft, IBM, BEA и Sun, разработали спецификацию W3C под названием язык разметки синтеза речи Speech Synthesis Markup Language. SSML обеспечивают однородный способ создания размеченного текста, основанного на речи. Обратитесь к официальной спецификации в отношении полного синтаксиса SSML на www.w3.org/TR/speech-synthesis.

Вызов API преобразования текста в речь аналогичен использованию API преобразования речи в текст. Сначала необходимо получить действительный токен. Для этого можно снова применить метод `getValidToken()`. Далее следует выполнить POST-запрос к <https://speech.platform.bing.com/synthesize>. Несколько раз в этой книге вы видели, как работают GET- и POST-запросы в C# при помощи класса `HttpClient`. В вашем примере исходного кода с преобразованием текста в речь вы воспользуетесь популярной сторонней библиотекой HTTP под названием `RestSharp`. Легкая установка `RestSharp` в Visual Studio поддерживается NuGet (см. листинг 7.4).

Листинг 7.4 ❖ Исходный код

```
public byte[] convertTextToSpeech(string ssml)
{
    var token = this.getValidToken();
    var client = new RestClient("https://speech.platform.bing.com/synthesize");
    var request = new RestRequest(Method.POST);
```

```

request.AddHeader("authorization", "Bearer " + token);
request.AddHeader("x-search-clientid", "8ae9b9546ebb49c98c1b8816b85779a1");
request.AddHeader("x-search-appid", "1d51d9fa3c1d4aa7bd4421a5d974aff9");
request.AddHeader("x-microsoft-outputformat", "riff-16khz-16bit-mono-pcm");
request.AddHeader("user-agent", "MyCoolApp");
request.AddHeader("content-type", "application/ssml+xml");
request.AddParameter("application/ssml+xml", ssml, ParameterType.RequestBody);
IRestResponse response = client.Execute(request);

return response.RawBytes;
}

```

Как можно убедиться выше, RestSharp имеет удобные API настройки объектов HTTP-запроса. Тело запроса представляет собой строку в формате SSML. Параметры запроса устанавливать не требуется. Вместо этого вы устанавливаете несколько заголовков. Ниже приводится перечень возможных заголовков:

- **Authorization:** *обязательный*. Действительный токен, сгенерированный API IssueToken;
- **x-search-appid:** *необязательный*. GUID без дефисов, который однозначно определяет ваше приложение; 
- **x-search-clientid:** *необязательный*. GUID без дефисов, который однозначно определяет ваше приложение в расчете на инсталляцию;
- **x-microsoft-outputformat:** *обязательный*. Формат выходного звукового файла. Ниже перечислены допустимые форматы файла:
 - ssml-16khz-16bit-mono-tts;
 - raw-16khz-16bit-mono-pcm;
 - audio-16khz-16kbps-mono-siren;
 - riff-16khz-16kbps-mono-siren;
 - riff-16khz-16bit-mono-pcm;
 - audio-16khz-128kbitrate-mono-mp3;
 - audio-16khz-64kbitrate-mono-mp3;
 - audio-16khz-32kbitrate-mono-mp3;
- **user-agent:** *обязательный*. Имя вашего приложения;
- **content-type:** *необязательный*. Единственное рекомендованное значение "application/ssml+xml".

Метод convertTextToSpeech() возвращает байтовый массив аудио, переданного назад в качестве отклика API преобразования текста в речь. Что нужно делать с байтовым массивом? Давайте сначала посмотрим, как вызывается приведенный выше метод и как выглядит допустимая строка SSML:

```

string ssml = "<speak version='1.0' xml:lang='en-US'><voice xml:lang='en-US'>
xml:gender='Female' name='Microsoft Server Speech Text to Speech Voice (en-US,
ZiraRUS)'>Hello, how may I help you?</voice></speak>";
byte[] TTSAudio = this.convertTextToSpeech(ssml);
SoundPlayer player = new SoundPlayer(new MemoryStream(TTSAudio));
player.PlaySync();

```

Самый распространенный вариант использования состоит в воспроизведении аудио сразу после того, как оно было получено в отклике. Приведенный выше исходный код преобразовывает текст «Hello, how may I help you?» («Привет, чем могу вам помочь») в речь. SoundPlayer является встроенным классом .NET, который вос-

производит звуковые файлы и потоки. Формат этого звукового файла определяется значением заголовка `x-microsoft-outputformat`. Поскольку `SoundPlayer` поддерживает только звуковые файлы `WAV`, то в качестве значения заголовка `outputformat` используется `"riff-16khz-16bit-mono-pcm"`.

Можно написать отдельный метод, который будет генерировать допустимые строки SSML. Листинг 7.5 показывает один из способов генерирования SSML.

Листинг 7.5 ❖ Генерирование SSML

```
private string GenerateSsml(string locale, string gender, string name, string text)
{
    var ssmlDoc = new XDocument(
        new XElement("speak",
            new XAttribute("version", "1.0"),
            new XAttribute(XNamespace.Xml + "lang", "en-US"),
            new XElement("voice",
                new XAttribute(XNamespace.Xml + "lang", locale),
                new XAttribute(XNamespace.Xml + "gender", gender),
                new XAttribute("name", name),
                text)));
    return ssmlDoc.ToString();
}
```

`text` – это текст, который будет преобразован в речь, `gender` может быть `Female` (женский) или `Male` (мужской). По поводу полного списка возможных значений локали и имени обратитесь к официальной справочной документации по API преобразования текста в речь на <https://docs.microsoft.com/en-us/azure/cognitive-services/speech/api-reference-rest/bingvoiceoutput>.

Механизм распознавания речи

На верхнем уровне механизм признания принимает аудиовход или речь из микрофона пользователя и его обрабатывает, пытаясь найти соответствия звуковым сигналам в шаблонах/образцах в различных базах данных. Эти шаблоны сигналов затем связываются с известными словами, и если механизм находит совпадающий образец в базе данных, он возвращает связанное с ним слово в виде текста (см. рис. 7.3). Речевой жargon кратко рассмотрен ниже:

- отдельные звуки в речи называются *фонемами*. Фонемы также именуются единицами языка. Например, слово «*Nishith*» состоит из трех фонем: «*Ni*», «*shi*» и «*th*»;
- процесс преобразования звуковых волн в фонемы называется *акустическим моделированием*. Он работает как отображение звуковых волн на одну или несколько фонетических единиц. Как видно на рис. 7.3, он является первым шагом в любом процессе распознавания речи;
- каталог слов языка называется *словарем*, или лексиконом;
- система правил, объединяющих эти слова в значимые предложения, называется *грамматикой*;
- *модель языка* представляет собой комбинацию словаря и грамматики. Модель языка также помогает идентифицировать правильный контекст между словами и фразами. Это имеет важное значение, т. к. иногда различные слова и фразы звучат одинаково. Например, фразы «*the stuffy nose*» («заложен-

ный нос») и «the stuff he knows» («то, что он знает») похожи по звучанию, но имеют разные значения.

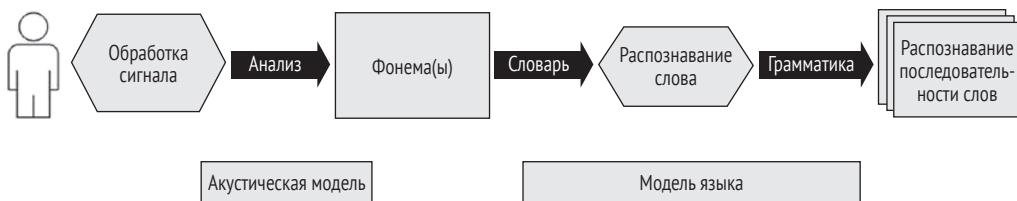


Рис. 7.3 ❖ Весь поток распознавания речи

Как вы видите, важно знать предметную область и контекст этой среды. Для того чтобы любая система распознавания речи показывала хорошую результативность, она должна быть хорошо натренирована. Механизм Microsoft преобразования текста в речь прошел массивный объем речевой тренировки. Он является лучшим в своем классе и работает чрезвычайно хорошо в универсальных сценариях. Однако иногда требуются системы распознавания речи в закрытой предметной области или для определенной среды. Например, есть потребность в распознавании речи в средах с фоновым шумом, специфическим жаргоном и разнообразными акцентами. Такие сценарии предписывают выполнение персонализированной настройки как акустики, так и языковой модели, с тем чтобы получить хорошую результативность. Именно здесь на первый план выступает служба персонализированной обработки речи.

СЛУЖБА ПЕРСОНАЛИЗИРОВАННОЙ ОБРАБОТКИ РЕЧИ

Служба персонализированной обработки речи (Custom Speech Service, CSS), которая ранее называлась Custom Recognition Intelligent Service (CRIS, т. е. интеллектуальная служба индивидуально настраиваемого распознавания), позволяет вам индивидуально настраивать системы распознавания речи, создавая индивидуализированные модели языка и акустические модели, которые характерны для вашей предметной области, окружающей среды и акцентов. Во время написания настоящей книги она была доступна в публичном предварительном просмотре. Большинство понятий вряд ли изменится, за исключением URL-адресов.

Персонализированная акустическая модель

Создание персонализированной акустической модели требует выполнения четырех шагов:

- создать набор аудиофайлов (или файл) речевых данных;
- создать текстовый файл, содержащий транскрипцию (расшифровку) каждого аудиофайла;
- закачать аудиофайл (файлы) и транскрипции на веб-портал Custom Speech Service (CSS), чтобы сделать акустический набор данных «готовым» к применению;
- создать персонализированную акустическую модель, используя акустический набор данных.



Аудиофайл должен быть записан в той же среде и той же группой людей, которые фактически используют эту систему персонализированного распознавания речи. Это имеет особую важность, потому что люди обладают разнообразными акцентами. Люди могут ее использовать в шумной окружающей среде. Фоновый шум представляет одну из самых трудных проблем для современных систем персонализированного распознавания речи. Запись аудио в той же самой окружающей среде помогает основной платформе распознавания освоиться с производственной средой и при этом оставаться результативной.

Закачиваемый на веб-портал аудиофайл (файлы) должен соответствовать некоторым основным правилам:

- весь аудиофайл должен иметь формат WAV. На данный момент это является ограничением, накладываемым порталом CSS. Надеемся, что в ближайшем будущем Microsoft сможет принимать аудио других типов;
- аудио должно быть закодировано с частотой дискретизации 8 кГц или 16 кГц. В большинстве коммуникаций VOIP, телефонов и беспроводной передачи данных используется одна из этих частот, поэтому для человеческой речи этого вполне достаточно;
- поддерживается всего один канал (моно);
- каждый аудиофайл не должен длиться более 1 минуты. На самом раннем этапе служба CRIS поддерживала всего 10 секунд. Надеемся, что в будущем эта длительность увеличится и при этом может основываться на продолжительности;
- каждый файл должен начинаться и заканчиваться тишиной как минимум в 1 миллисекунду;
- все аудиофайлы (файл) для конкретного набора данных должны иметь уникальное имя;
- все аудиофайлы (файл) должны быть заархивированы в одном архивном файле, который должен содержать только аудиофайл (файлы) и никакие папки/подпапки;
- размер файла после архивирования не должен составлять более 2 ГБ.



Служба Microsoft обработки речи не предоставляет записывающие инструменты для записи аудио. Вам придется использовать некоторые существующие инструменты Windows. Лично мы для записи аудио используем Audacity (www.audacityteam.org/download/). Этот инструмент также может экспорттировать в формат .WAV. Для записи не забудьте установить канал моно и частоту дискретизации 8 кГц, поскольку CSS не предоставляет подробного описания, кроме упоминания «Недопустимое аудио». Вы можете использовать любой аудиоинструмент, при условии что придерживаетесь упомянутых выше требований, предъявляемых к аудио.

После того как вы записали все аудиофайлы с приведенными выше правилами, следующий шаг состоит в создании простого текстового файла, содержащего транскрипцию всех аудиофайлов. Этот файл транскрипции должен соответствовать некоторым основным правилам написания и нормализации:

- для всех созданных файлов .WAV должен быть всего один файл транскрипции;
- каждая строка в файле транскрипции должна иметь имя звукового файла и транскрипцию, отделенную символом табуляции (/t). Например:

- Audio01.wav Это мой первый аудиофайл;
 Audio02.wav Я очень рад использовать API CRIS;
- в тексте транскрипции должен использоваться только набор печатаемых символов ASCII;
 - не используйте Юникод и замените его подстановкой ASCII везде, где был Юникод. Например, вместо написания **Nishith's work** (правая одинарная кавычка) должно быть **Nishith's work** (апостроф);
 - сокращения должны быть записаны словами. Например, Dr. Sanjay следует переписать как Doctor Sanjay;
 - слова без букв алфавита должны быть записаны прописью. Например, 363, SW следует переписать как триста шестьдесят три, юго-запад.

После подготовки аудиофайлов и файла транскрипции они могут быть импортированы на портал персонализированной обработки речи Custom Speech Portal. Для того чтобы им воспользоваться, необходимо перейти на сайт CRIS (<https://cris.ai>). (Поскольку название службы CRIS было изменено на Службу персонализированной обработки речи Custom Speech Service, URL-адрес сайта портала, по всей видимости, может поменяться. Мы советуем вам перейти на домашнюю страницу когнитивных служб и получить фактический URL-адрес службы CRIS.) Текущая домашняя страница показана на рис. 7.4.

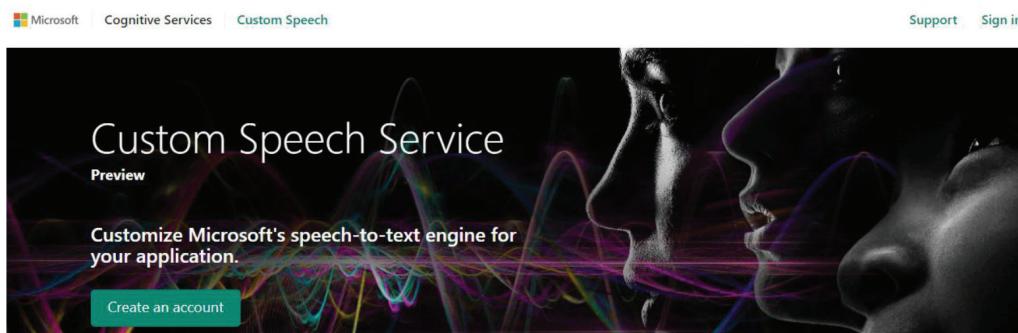


Рис. 7.4 ♦ Страница портала
Службы персонализированной обработки речи Custom Speech Service

Если вы впервые обращаетесь к этому ресурсу, вам необходимо сделать учетную запись. Для этого нажмите кнопку **Create an account** (Создать учетную запись), которая перенаправит вас на страницу регистрации. Если у вас есть доступ к речевому порталу, то вы должны авторизоваться, нажав кнопку **Sign In** (Авторизоваться) на верхней правой панели. После авторизации вы увидите страницу, показанную на рис. 7.5.

Щелкните на открывающемся меню **Custom Speech** (Персонализированная обработка речи) и выберите **Adaptation Data** (Данные адаптации). Поскольку вы находитесь на этой странице впервые, то увидите пустую таблицу под названием **Acoustic Datasets** (Наборы акустических данных), рис. 7.6.

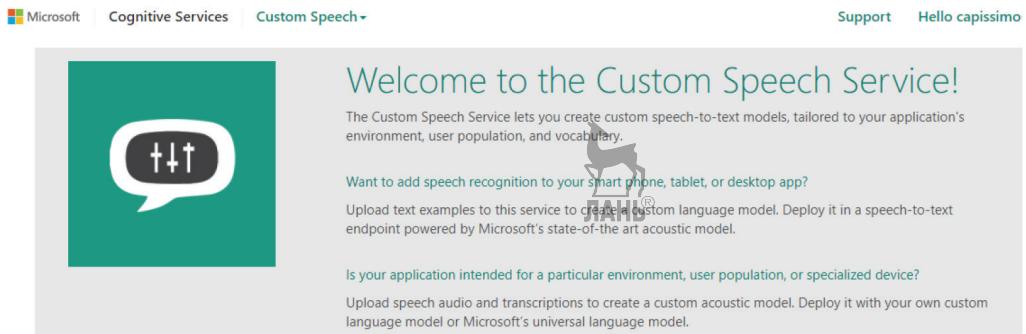


Рис. 7.5 ❖ Портал персонализированной обработки речи
Custom Speech Portal после авторизации

Рис. 7.6 ❖ Страница наборов данных Adaptation Datasets при первом посещении

По умолчанию используется локаль en-us. Во время написания настоящей книги Служба персонализированной обработки речи поддерживала две локали (en-US и zh-CN). Другая локаль выбирается из открывающегося списка, расположенного в правом верхнем углу. Обратите внимание, что локаль en-US поддерживает все три набора данных – **Acoustic** (Акустика), **Language** (Язык) и **Pronunciation** (Произношение), в то время как zh-CN поддерживает только наборы данных **Acoustic** и **Language**. Со временем это может поменяться. Чтобы импортировать данные аудио и транскрипции, нажмите кнопку **Import**, расположенную рядом с **Acoustic and Language datasets**. Затем будет проверен ваш ключ подписки, и вам будет предложено связать его с бесплатной подпиской или с любой существующей подпиской, как показано на рис. 7.7.

Рис. 7.7 ❖ Варианты подписки для выбора перед импортированием данных

После подтверждения вашего ключа подписки вы будете переведены на страницу импортирования данных, как показано на рис. 7.8.

Microsoft | Cognitive Services | Custom Speech+

Import Acoustic Data

Name

Description

Transcriptions file (.txt) No file chosen
The maximum size of the transcriptions file is 200 MB.

Audio files (.zip) No file chosen
The maximum size of the archive is 2048 MB.

Рис. 7.8 Страница импорта акустических данных

Введите имя и описание импортируемого файла. Выберите соответствующий файл транскрипции и zip-файл, содержащий все аудио, нажав на кнопку **Choose file** (Выбрать файл). Нажмите кнопку **Import**, чтобы закачать ваши данные в центр персонализированной обработки речи. Если вы загрузили файлы до допустимого формата и размера, то будете перенаправлены назад к таблице Acoustic Dataset, которая теперь будет содержать одну запись, как показано на рис. 7.9.

Datasets

Acoustic Datasets				
Name	Description	Created	Status	
Test Data	This is for testing our sample Data with 8 KHZ	5/13/2017 9:19:11 AM	Processing	<input type="button" value="Edit"/>

Language Datasets	
<input type="button" value="Import"/>	

Pronunciation Datasets	
<input type="button" value="Import"/>	

Рис. 7.9 Страница наборов данных с одной записью после импортирования данных

Чтобы поменять имя и описание аудио, щелкните мышью на ссылке **Edit** (Править). Если вы закачали большое количество аудио, то можете получить статусное сообщение **Waiting** (Ожидание данных) или **Processing** (Обработка данных). Статусное сообщение **Waiting** означает, что вы закачали аудио, но эти данные находятся в очереди для обработки. Статусное сообщение **Processing** означает, что механизм службы обработки речи выполняет внутреннюю проверку аудио и транскрипции, которая включает проверку длины файла, частоты дискретизации, нормализацию данных транскрипции и т. д. Если при закачке файла все правила были соблюдены, на экране появится **Complete** (Обработка завершена). Рисунок 7.10 показывает один из наборов данных, отмеченных статусным сообщением **Failed** (Обработка не удалась), и другой, отмеченный как **Complete**.

Datasets

Acoustic Datasets Import

Name	Description	Created	Status	Edit	Delete	Details
Test Data	This is for testing our sample Data with 8 KHZ	5/13/2017 1:44:34 PM	Failed			
Test Data	This is for testing our sample Data with 8 KHZ	5/13/2017 9:19:11 AM	Complete			

Language Datasets Import

Pronunciation Datasets Import

Рис. 7.10 ❖ Две записи в наборе акустических данных, один набор отмечен как **Failed** (Обработка не удалась), а другой отмечен как **Complete**

Теперь после нажатия на ссылке **Details** (Подробная информация) для набора данных, отмеченного как **Complete**, будет показано, что высказывание для записи было успешным. Это означает, что аудио успешно прошло проверку CSS, как показано на рис. 7.11.

Adaptation Data Details

Name	Test Data
Description	This is for testing our sample Data with 8 KHZ
Locale	en-US
Id	1b02210d-e2dc-49e5-b398-7f20cc95840f
Created	5/13/2017 9:19:11 AM

Results of the dataset import

Number of successful utterances: 2
Number of failed utterances: 0

Рис. 7.11 ❖ Успешные высказывания набора данных, отмеченного как **Complete**

Может так случиться, что вы забудете выполнить все правила при импорте аудиофайлов и транскрипции. Если в файле транскрипции и zip-файле будут обнаружены ошибки и эти файлы не будут подчиняться правилам, то вы получите сообщение об ошибке **An error occurred while processing your request** (Во время обработки вашего запроса произошла ошибка). Точную причину ошибки вы можете узнать из URL-адреса страницы, и в итоге вы увидите нечто вроде этого: <https://cris.ai/AcousticDatasets/Create?errorMessage=The%20name%20and%20the%20two%20files%20are%20required>. Состояние вашего набора данных будет отмечено как **Failed**, как показано на рис. 7.8. Чтобы получить дополнительные сведения о количестве неудавшихся и успешных высказываний, щелкните мышью на ссылке **Details**. Сообщение **Failed** означает, что ни одно из высказываний в аудиозаписи не было успешным. Для этого может иметься несколько причин. Основная причина заключается в том, что вы не создали файл с частотой 8 кГц. Другая потенциальная причина может состоять в том, что файл транскрипции не подчинялся установленным правилам, и поэтому вы можете получить ошибку, где построчно будет говориться: **The format of this line is incorrect, should be <wavefile><tab><transcription>** (Формат этой строки неправильный, должен быть <файл_wav><табуляция><транскрипция>).

Для того чтобы создать персонализированную акустическую модель данных, ваш набор акустических данных должен иметь состояние **Complete**. Чтобы создать персонализированную акустическую модель, нажмите **Custom Speech** и затем выберите **Acoustic model**. Чтобы создать любую персонализированную акустическую модель, вам нужно применить какую-то акустическую модель в качестве базовой модели для персональной настройки. Так как вы обратились к этой ссылке впервые, то увидите две базовые акустические модели, которые поставляются вместе с CSS (см. рис. 7.12).

Acoustic Models



Create New						
Base models						
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Microsoft Conversational Model</td> <td>Microsoft Conversational Model for recognizing speech spoken in a conversational style (i.e. speech directed at another person).</td> </tr> <tr> <td>Microsoft Search and Dictation Model</td> <td>Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.</td> </tr> </tbody> </table>	Name	Description	Microsoft Conversational Model	Microsoft Conversational Model for recognizing speech spoken in a conversational style (i.e. speech directed at another person).	Microsoft Search and Dictation Model	Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.
Name	Description					
Microsoft Conversational Model	Microsoft Conversational Model for recognizing speech spoken in a conversational style (i.e. speech directed at another person).					
Microsoft Search and Dictation Model	Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.					

Рис. 7.12 ❖ Две базовые предварительно определенные акустические модели, которые поставляются вместе с CSS

Microsoft Search and Dictation AM-1.1 является базовой акустической моделью, которая в большинстве случаев действует как отправная точка для персональной настройки. Эта модель имеет конечную точку REST службы распознавания речи. Данная модель более-менее похожа на ту, которая используется в Cortana. Базовая модель Microsoft Conversational Model используется для распознавания разговорной речи. Этот модуль применяется для построения любого приложения, основанного на разговорном интерфейсе, которое требует персонализированные речевые сценарии. Эти модели нельзя удалить. У вас всегда есть возможность создать дополнительную персонализированную акустическую модель и использовать ее в качестве базовой модели для других персонализированных акустических моделей. Чтобы создать новую модель, щелкните мышью на кнопке **Create New**, как показано на рис. 7.12. Вы будете перенаправлены на новый экран, показанный на рис. 7.13.

Create Acoustic Model

Name	<input type="text"/>
Description	<input type="text"/>
Base Acoustic Model	Microsoft Conversational Model
Acoustic Data	Test Data
Subscription	Nishith Pathak
Accuracy Testing	<input type="checkbox"/>
Back	Create

Рис. 7.13 ❖ Форма для создания новой акустической модели

Введите имя и описание. Поскольку вы создаете акустическую модель впервые, ваш открываящийся список базовых акустических моделей будет иметь два варианта: **Microsoft Conversational Model** и **Microsoft Search and Dictation Model**. После создания дополнительной персонализированной модели вы увидите, что она тоже будет внесена в этот открываящийся список. Выберите **Test Data** в качестве набора акустических данных, чтобы вычислить персонализированную модель. Это единственный набор данных с состоянием **Complete**. Все ваши подписки также будут показаны в открываящемся меню **Subscription** (Подписка). Если вы просто используете бесплатную подписку, то также получите возможность для бесплатной подписки для службы персонализированной обработки речи.

Нажмите кнопку **Create**, чтобы создать модель. На создание акустической модели уйдет примерно такое же время или больше, что и на создание вашего набора данных. После нажатия кнопки **Create** вы будете перенаправлены на домашнюю страницу акустической модели **Acoustic Model**, показанную на рис. 7.14. Теперь вы увидите три записи.

Acoustic Models

Name	Base model	Description	Created	Status	Actions
Testing Model for Speech Chapter	Microsoft Conversational Model	This is for testing model for our Speech Chapter	5/13/2017 3:44:13 PM	Processing	Edit
Microsoft Conversational Model	Microsoft Conversational Model	Microsoft Conversational Model for recognizing speech spoken in a conversational style (i.e. speech directed at another person).		Complete	Edit
Microsoft Search and Dictation Model	Microsoft Search and Dictation Model	Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.		Complete	Edit

Рис. 7.14 ♦ Создание новой акустической модели

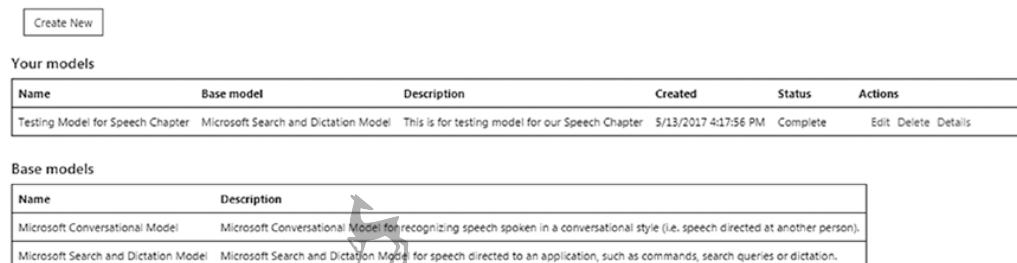
На рис. 7.14 показано состояние **Processing**. Если же данные находятся в очереди, то состояние должно быть **Waiting**. После завершения обработки состояние показано как **Complete**, см. рис. 7.15. Чтобы изменить имя и состояние, нажмите **Edit**. Вариант **View Result** (Просмотреть результат) используется, когда вы хотите просмотреть результаты офлайнового тестирования.

Чтобы получить подробную информацию о создании персонифицированной модели, нажмите **Details**. На рис. 7.16 показана новая акустическая модель.

Веб-портал Custom Speech Service (CSS) также позволяет вам протестировать свою «новую» модель адаптации относительно некоторых тестовых данных. Если вы хотите протестировать свою модель, поставьте флажок напротив **Accuracy Testing** (Тестирование точности). Это позволит протестировать вашу модель напрямую. Вам не нужно отправлять данные в свою новую конечную точку, на которой размещена ваша адаптированная модель. Ранее флажок **Accuracy Testing** назывался **Offline Testing** (Оффлайновое тестирование). Я был одним из тех разработчиков обратной связи, которые предложили изменить название Оффлайновое тестирование, поскольку конечный пользователь мог бы посчитать, что *оффлайно-*

вое означает локальное. На самом деле для персонализированной обработки речи всегда требуется подключение к Интернету.

Acoustic Models



Name	Base model	Description	Created	Status	Actions
Testing Model for Speech Chapter	Microsoft Search and Dictation Model	This is for testing model for our Speech Chapter	5/13/2017 4:17:56 PM	Complete	Edit Delete Details

Name	Description
Microsoft Conversational Model	Microsoft Conversational Model for recognizing speech spoken in a conversational style (i.e. speech directed at another person).
Microsoft Search and Dictation Model	Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.

Рис. 7.15 ♦ Акустическая модель, отмеченная как Complete

Acoustic Model Details

Name	Testing Model for Speech Chapter
Description	This is for testing model for our Speech Chapter
Locale	en-us
Id	92c36b56-7cfa-4814-a109-71dd3df3a6e3
Created	5/13/2017 4:17:56 PM

Base model

Name	Microsoft Search and Dictation Model
Description	Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.
Id	a3d8aab9-6f36-44cd-9904-b37389ce2bfa
Created	11/4/2016 8:23:42 AM

Adaptation data

Name	Test Data
Description	This is for testing our sample Data with 8 KHZ
Id	1b02210d-e2dc-49e5-b398-7f20cc95840f
Created	5/13/2017 9:19:11 AM

Back

Delete

Рис. 7.16 ♦ Подробная информация о новой акустической модели

При установке флагка **Accuracy Testing** вам потребуется выбрать модель языка и тестовые данные для тестирования. Если вами не было создано ни одной модели языка, то из открывавшегося списка вы сможете выбрать универсальную модель Microsoft Universal LM как единственную модель языка. Microsoft Universal LM – это исходная модель языка, которая также используется в других когнитивных службах. Вам также нужно выбрать тестовые данные для тестирования своей модели, как показано на рис. 7.17. Важно выбрать данные, которые не использовались для создания акустической модели. Это также поможет получить истинные результаты тестирования результативности вашей модели. Если вы выбрали ва-

риант с офлайновым тестированием, то заметите, что во время вычисления модели будет показано состояние **Testing**.

Create Acoustic Model

Name	Testing Model for Speech Chapter
Description	This is for testing Language model for our Speech Chapter
Base Acoustic Model	Microsoft Search and Dictation Model
Acoustic Data	Test Data
Subscription	Nishith Pathak
Accuracy Testing	<input checked="" type="checkbox"/>
Language Model	Microsoft Search and Dictation Model
Test Data	Test Data for Accuracy testing

The test will only run on the first 1000 utterances if the data contains more than that.

Рис. 7.17 ❖ Создание акустической модели с флагжком напротив тестирования точности

После создания акустической модели вы получите дополнительные результаты тестирования точности, которые будут показаны в подробной информации об акустической модели (рис. 7.18).

Acoustic Model Details

Name	Testing Model for Speech Chapter
Description	This is for testing Language model for our Speech Chapter
Locale	en-us
Id	f50635db-e1c6-4589-b7a3-52117899ab46
Created	5/13/2017 4:38:00 PM
Base model	
Name	Microsoft Search and Dictation Model
Description	Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.
Id	a3d8aab9-6f36-44cd-9904-b5778846e07a
Created	11/4/2016 8:23:42 AM
Adaptation data	
Name	Test Data
Description	This is for testing our sample Data with 8 KHZ
Id	1b02210d-e2dc-49e5-b398-7f20cc95840f
Created	5/13/2017 9:19:11 AM
Accuracy tests using this model	

Language Model	Dataset	Created	State	Word Error Rate (WER)	
Microsoft Search and Dictation Model	Test Data for Accuracy testing	5/13/2017 4:38:02 PM	Complete	30.77 %	Details

Рис. 7.18 ❖ Дополнительная информация

Как показано выше, коэффициент словарных ошибок составил 30.77%. Нажав на ссылку **Details**, вы получите дополнительную информацию о результатах тестирования точности. Можно также увидеть все тесты на точность или создать новый тест, непосредственно выбрав в меню **Custom Speech** команду **Accuracy Tests** (Тесты на точность). Меню **Custom Speech** располагается слева вверху на странице **Custom Speech** (Персонализированная обработка речи).

Персонализированная модель языка

Как было отмечено ранее, создание модели языка имеет большое значение для понимания правильного контекста слов. Например, возьмем фразу «slow children». Ее можно произнести разными способами, и при этом она будет иметь разные значения:

Slow, Children.
Slow, Children?
Slow Children!
Slow Children?



В зависимости от той или иной окружающей обстановки и контекста приведенные выше слова могут иметь разные значения. Персонализированная модель языка придерживается более простого процесса, чем создание персонализированной акустической модели. Вам больше не нужно создавать аудиофайлы; напротив, вы работаете всего с одним файлом транскрипции. Вот соответствующие шаги:

- подготовить языковые данные;
- закачать языковые данные на портал персонализированной обработки речи Custom Speech Service, чтобы сделать языковой набор данных полным;
- создать персонализированную модель языка, используя языковой набор данных.

Создание языкового файла данных требует от вас соблюдения набора правил:

- ваши данные языковой модели должны содержаться в обычном текстовом файле;
- если вы используете локаль en-US, то ваша модель языка поддерживает кодировку US-ASCII или UTF-8. При использовании локали zh-CN вам нужно иметь текстовый файл в кодировке UTF-8;
- каждая строка в файле должна содержать всего одно высказывание. Причина, почему я называю его высказыванием, а не предложением, состоит в том, что оно должно точно представлять фактически произнесенный вход, а не быть грамматически правильным;
- каждая строка имеет вес, по умолчанию равный 1. Вы можете по выбору назначить больший вес любой строке. Вес представляет собой целое число больше 1. Данное число позволяет повторять строку соответствующее количество раз;
- чтобы добавить любой вес, просто допишите число, отделенное символом табуляции (t), как показано ниже. Приведенный ниже пример с файлом языковых данных будет успешно принят:

Это мой первый аудиофайл\t5

Я очень рад использовать API CRIS

Этот текст используется здесь много раз

- максимальный размер файла не должен превышать 200 МБ.

После того как ваши языковые данные готовы, их следует закачать на сервер. Так же, как и для добавления акустического набора данных, выберите в меню **Custom Speech** команду **Adaptation Data**. Введите соответствующее имя и описание, чтобы идентифицировать языковые данные. Выберите правильный языковой файл данных, как показано на рис. 7.19, и нажмите **Import**, чтобы импортировать ваш набор данных.



Рис. 7.19 ❖ Импортирование языковых данных

Вы будете перенаправлены назад на страницу языковых данных и увидите одну запись, как показано на рис. 7.20. Вашим языковым данным будет назначен уникальный идентификатор; языковые данные имеют состояние **Waiting**, **Processing** или **Complete**. Рисунок 7.20 показывает состояние **Complete**, что означает, что ваш файл языковых данных успешно прошел проверку допустимости данных и готов к созданию персонализированной модели языка. Вы можете нажать на ссылку **Details**, чтобы увидеть список успешных высказываний.

Language Datasets				
Name	Description	Created	Status	
Language Test Data	This is for testing Language model for our Speech Chapter	5/13/2017 5:54:43 PM	Complete	Edit Delete Details

Рис. 7.20 ❖ Подмножество страницы Dataset, показывающей набор языковых данных

Как и в случае с персонализированной акустической моделью, чтобы создать персонализированную модель языковых данных, ваш набор языковых данных должен иметь состояние **Complete**. Выберите из меню **Custom Speech** команду **Language Model**, чтобы создать персонализированную модель языка. Поскольку вы переходите по этой ссылке впервые, то аналогично акустической модели вы увидите две записи: **Microsoft Conversational Model** и **Microsoft Search and Dictation Model**. Эти модели служат базовыми моделями для создания новых персонализированных моделей. Эти модели удалить нельзя. У вас всегда есть возможность создать дополнительную персонализированную модель языка и использовать ее в качестве базовой модели для других персонализированных моделей языка. Нажмите кнопку **Create New**, чтобы создать новую модель языка.

Как и в случае с персонализированной акустической моделью, вы получите экран, показанный на рис. 7.21, для создания модели языка. Введите имя

и описание, чтобы однозначно идентифицировать вашу модель языка. Выберите одну из двух базовых моделей языка в качестве базовой модели. Выберите ранее созданные языковые данные в качестве набора языковых данных. Нажмите **Create**, чтобы создать модель языка. У вас также есть возможность протестировать модель языка на точность, не обращаясь к персонализированной конечной точке.

Create Language Model

Name	Language Test Data
Description	This is for testing Language model for our Speech Chapter
Base Language Model	Microsoft Search and Dictation Model
Language Data	Language Test Data
Pronunciation Data (optional)	You may want to add pronunciation data to be able to use it in adaptations.
Subscription	Custom Speech Service - Free subscription
Accuracy Testing	<input type="checkbox"/>
<input type="button" value="Back"/> <input type="button" value="Create"/>	

Рис. 7.21 ♦ Страница создания тестовой модели языка

Вы будете перенаправлены на домашнюю страницу **Language Model** и получите дополнительную запись. Создавая модель языка, удостоверьтесь, что у вас достаточно данных. Если это не так, ваша модель языка получит значение **Failure** с объяснением, что **Language dataset doesn't contain enough words to form a language model** (Набор языковых данных не содержит достаточно слов, чтобы сформировать модель языка). Если все пройдет хорошо, вы получите состояние **Complete**, как показано на рис. 7.22.

Your models					
Name	Base model	Description	Created	Status	Actions
Language Test Data	Microsoft Search and Dictation Model	This is for testing Language model for our Speech Chapter	5/13/2017 6:08:15 PM	Complete	Edit Delete Details

Base models	
Name	Description
Microsoft Conversational Model	Microsoft Conversational Model for recognizing speech spoken in a conversational style (i.e. speech directed at another person).
Microsoft Search and Dictation Model	Microsoft Search and Dictation Model for speech directed to an application, such as commands, search queries or dictation.

Рис. 7.22 ♦ Страница модели языка
после создания персонализированной модели языка

Данные произношения

При создании языковой модели у вас будет возможность добавить набор персонализированных данных произношения. Этот набор предназначен для понимания новых терминов или акронимов. Чтобы создать набор данных произношения, не-

обходится создать простой текстовый файл, у которого должно быть два столбца (написание и речевая форма), как показано ниже:

Написание	Речевая форма
Cavity	kay vity
Cephalic	see pha lic
Cholesterol	kho les trol
Generic	Jeneric
Xerosis	Zerosis

Сохраните приведенный выше или подобный файл в файле .txt. Затем выберите команду меню **Custom Speech ▶ Adaptation Data** и нажмите кнопку **Import**, расположенную рядом с набором данных произношения. Вы получите форму, показанную на рис. 7.23. Введите имя и описание, выберите ранее созданный файл данных произношения и нажмите кнопку **Import**.

Import Pronunciation Data

Name	medical terms
Description	This pronunciation is been used in Medical space
Language data file (.txt)	<input type="button" value="Choose File"/> pronunciation.txt <small>The maximum size of the pronunciation data file is 200 MB.</small>
<input type="button" value="Back"/>	<input type="button" value="Import"/>

Рис. 7.23 ❖ Страница импорта данных произношения

Если файл данных произношения сформирован согласно вышеупомянутым рекомендациям, то вас автоматически перенаправят на домашнюю страницу набора акустических данных, где в наборе данных произношения вы увидите одну запись, отмеченную как **Complete** (рис. 7.24).

Pronunciation Datasets				Import
Name	Description	Created	Status	
medical terms	This pronunciation is been used in Medical space	5/13/2017 6:56:40 PM	Complete	<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Details"/>

Рис. 7.24 ❖ Набор данных произношения

Как только верификационная обработка вашего набора данных произношения будет завершена, т. е. отмечена как **Complete**, его можно использовать при создании моделей языка.

Персонализированная конечная точка преобразования речи в текст

Одна из функциональных возможностей программного интерфейса когнитивной обработки речи Microsoft Cognitive Speech API заключается в том, что этот API позволяет создавать свою собственную конечную точку преобразования речи в текст, отвечая вашим конкретным требованиям. Когда ваши персонализированные акустические модели и персонализированные модели языка будут готовы, вы

сможете создавать персонализированные конечные точки преобразования речи в текст. При этом вы можете объединить эти недавно созданные персонифицированные модели как друг с другом, так и с базовой моделью. Чтобы создать новую персонализированную конечную точку преобразования речи в текст, выберите из меню **Custom Speech** (Персонализированная обработка речи) команду **Deployments** (Развертывания). Далее нажмите кнопку **Create New**, чтобы создать новую персонализированную конечную точку. Введите уникальное имя и описание, которыми будут идентифицировать вашу персонализированную конечную точку. Выберите базовую модель – **Microsoft Search and Dictation Model** либо **Microsoft Conversational Model**. В зависимости от выбранного варианта вы получите список ранее созданных персонализированных акустических моделей и моделей языка. У вас также есть возможность использовать любую из базовых моделей, как показано на рис. 7.25. Нажмите кнопку **Create**, чтобы создать новую конечную точку.

Create Deployment

* These fields are required

Name *	<input type="text" value="TestEndpoint"/>
Description	<input type="text" value="This endpoint is been used for our speech Chapter"/>
Base Model	<input type="text" value="Microsoft Search and Dictation Model"/>
Acoustic Model	<input checked="" type="radio"/> Testing Model for Speech Chapter <input type="radio"/> Testing Model for Speech Chapter <input type="radio"/> Microsoft Search and Dictation Model
Language Model	<input checked="" type="radio"/> Language Test Data <input type="radio"/> Microsoft Search and Dictation Model (Base)
Subscription	<input type="text" value="Custom Speech Service - Free subscription"/>
Back	Create

Рис. 7.25 ❖ Создание персонализированной конечной точки для развертывания

Когда конечная точка будет создана, вас автоматически перенаправят на страницу развертывания **Deployment**. Как и в случае с вашей персонализированной моделью, у вас теперь будет одна запись в таблице, которая однозначно идентифицирует ваше развертывание. Создание конечной точки развертывания занимает некоторое время. После того как ваша конечная точка будет создана, ваше состояние развертывания должно быть отмечено как **Complete** (рис. 7.26).

Deployments

[Create New](#)

Locale	Name	Description	Id	Created	State	Actions
en-us	TestEndpoint	This endpoint is been used for our speech Chapter	24a8ae1b-ae88-4876-9b14-6dc8b3338d87	5/13/2017 7:16:41 PM	Complete	Delete Details

Рис. 7.26 ❖ Список конечных точек развертывания

Щелкните мышью на ссылке **Details**. Вас автоматически перенаправят на новую страницу, где будет показан URL-адрес персонализированной конечной точки (рис. 7.27). В конечном итоге вы увидите три URL-адреса. Один URL-адрес – это HTTP URL, который должен использоваться для HTTP-запросов. Два других URL предназначены для веб-сокетов режима короткой фазы и режима длинной фазы. Эти URL-адреса прежде всего применяются для создания пользовательского приложения с использованием библиотеки Microsoft Cognitive Client Service, которая на внутреннем уровне применяет веб-сокеты.

Deployment Information

Id	24a8ae1bae88487690146dc0b3338d87
Name	TestEndpoint
Locale	en-US
Description	This endpoint is been used for our speech Chapter
Language Model	Language Test Data
Acoustic Model	Testing Mode for Speech Chapter
Created	5/13/2017 7:16:41 PM
Subscription Key	47b84ed306ca47ecaf6e732f5d31b01f

Protocol	Url
Http	https://24a8ae1bae88487690146dc0b3338d87.api.cris.ai/cris/speech/vQuery
WebSocket for ShortPhrase mode	https://24a8ae1bae88487690146dc0b3338d87.api.cris.ai/ws/cris/speech/recognize
WebSocket for LongDictation mode	https://24a8ae1bae88487690146dc0b3338d87.api.cris.ai/ws/cris/speech/recognize/continuous

Test your endpoint

Upload an example file to test your deployed endpoint. The audio must be single channel, less than 15 seconds, with an 8 kHz or 16 kHz sampling rate, and stored as uncompressed 16-bit PCM in WAV format.

Audio file
 No file chosen

Рис. 7.27 ❖ Ссылки развертывания конечной точки

На момент написания настоящей книги в CSS имелось ограничение, которое заключалось в поддержке всего двух параллельных запросов к этим персонализированным конечным точкам. Если прокрутить страницу вниз, то вы увидите опцию тестирования вашей конечной точки, как показано на рис. 7.28. У вас есть возможность закачать аудиофайл, который создан согласно индивидуальным правилам для аудиофайла, показанного жирным шрифтом. После того как вы выберете соответствующий аудиофайл и нажмете **Test**, персонализированный речевой механизм будет использовать ваш недавно созданный образец речи, чтобы проверить конечную точку, выполнив распознавание переданного аудио. Если персонализированный механизм обработки речи его распознает, то в текстовом поле **Recognition** (Распознавание) вы увидите, что он будет преобразован в текст, как показано на рис. 7.28.

Test your endpoint

Upload an example file to test your deployed endpoint. The audio must be single channel, less than 15 seconds, with an 8 kHz or 16 kHz sampling rate, and stored as uncompressed 16-bit PCM in WAV format.

Audio file
 audio01.wav

This is my 1st audio file.

Recognition

Рис. 7.28 ❖ Результаты тестирования аудиофайла из конечной точки

Распознавание говорящего



Теперь у вас появилось представление, как работает распознавание речи и как эта функция используется. API распознавания речи помогает идентифицировать пользователей по их голосу. Обратите внимание, что имеется четкое различие между *распознаванием говорящего* и *распознаванием того, что он сказал*. Распознавание речи отвечает на вопрос «что», тогда как распознавание говорящего отвечает на вопрос «кто». Проще говоря, распознавание речи – это процесс, во время которого ведется работа с тем, что уже было сказано. Распознавание говорящего – это метод идентификации того, кто говорит. Этот метод также называется *биометрией на основе голосовых сигналов*.

Любое корпоративное приложение требует подтверждения достоверности пользователя. Если вы знакомы с традиционной моделью на основе ASP.NET, то знаете термины *аутентификация* и *авторизация*. Аутентификация – это процесс идентификации пользователя. Пользователь может представить одно или несколько доказательств, объявляя о себе как об истинном пользователе. Почти все приложения требуют, чтобы пользователь передал имя пользователя и пароль как одно из средств идентификации. Эта форма идентификации продолжает служить основной моделью устойчивой аутентификации. Вместе с тем появились различные дополнительные модели идентификации, чтобы вывести обеспечение безопасности на следующий уровень. Один такой способ аутентификации предлагаются когнитивной моделью Microsoft посредством распознавания говорящего. Такое обеспечение безопасности может выступать в качестве второй формы аутентификации, дополняя вашу стандартную аутентификацию.

Распознавание голоса говорящего не является новым понятием. По сути дела, оно существует уже более 40 лет. В историческом плане с течением времени различные алгоритмы эволюционировали в сторону обеспечения точных результатов распознавания говорящего. Все началось с систем частотного анализа и дискретной векторной квантизации. В прошедшее десятилетие системы распознавания говорящего были адаптированы к модели смеси нормальных распределений (Gaussian Mixture Model, GMM). В Microsoft применяются системы распознавания говорящего с использованием GMM на основе *i*-вектора.

Верификация говорящего против идентификации говорящего

API распознавания говорящего помогает идентифицировать пользователей по их голосу. Используя API распознавания говорящего, можно значительно увеличить качество обслуживания клиентов не только путем верификации пользователя, но и помогая в идентификации пользователя. Следует проводить различие между *верификацией* и *идентификацией*. Смысл слов верификация и идентификация похож, но есть некоторые отличия. *Верификация говорящего*, или его аутентификация, – это процесс проверки запроса на основе шаблона или записи в хранилище. При верификации говорящего его голос вместе с фразой сопоставляется с зарегистрированным голосом и фразой, чтобы определить, является ли он тем же самым человеком или каким-то другим. Верификация говорящего работает на основе отображения один-к-одному (1:1). В реальных сценариях предоставление вашего паспорта или какого-то цифрового удостоверения личности называется аутентификацией.

Идентификация говорящего, с другой стороны, – это проверка запроса посредством всех возможных записей в хранилище и прежде всего используется для установления личности неизвестного человека из группы людей. Идентификация говорящего работает на основе отображения один-ко-многим (1:N). Прежде чем использовать API распознавания говорящего для верификации или идентификации, этот человек подвергается процессу под названием регистрации (enrollment). Поскольку идентификация говорящего является отображением один-ко-многим и на нее требуется больше времени, верификация говорящего работает быстрее, чем идентификация говорящего.



Регистрация для верификации говорящего

Любой голос имеет отличительные характеристики, которые могут быть использованы для распознания человека. Во время регистрации голос говорящего записывается, и при этом, как правило, извлекается ряд акустических характеристик, таких как тембр, высота, частота и т. д., чтобы сформировать голосовой отпечаток. Комбинация всех этих характеристик формирует голосовой отпечаток или голосовую модель, которая однозначно определяет индивидуальный голос. Как показано на рис. 7.29, процесс регистрации требует итерационного подхода, в котором каждый человек повторяет определенные фразы три раза или более. Всякий раз, когда пользователь говорит определенную фразу, извлекаются акустические характеристики его голоса. Комбинация извлеченных характеристик и фраз формирует модель уникального отпечатка голоса соответствующего человека. Эта модель затем сохраняется в хранилище моделей голосовых отпечатков для дальнейшего использования с целью верификации или идентификации говорящего.

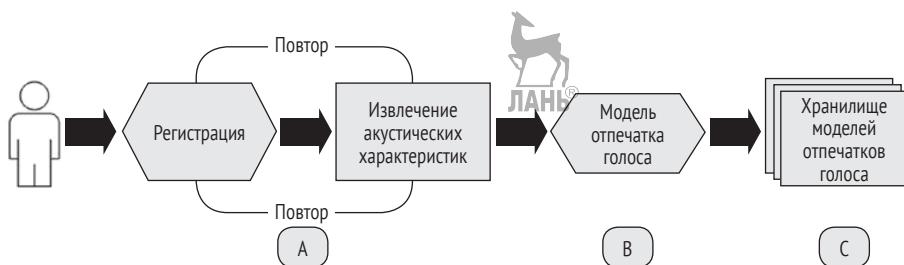


Рис. 7.29 ♦ Шаги, выполняемые во время процесса регистрации

Регистрация говорящего является трехшаговым процессом:

- создать профиль верификации;
- предложить пользователю сказать одну из верификационных фраз и сохранить аудио;
- вызвать API регистрации путем передачи идентификатора профиля (созданного на шаге 1) и речевого аудиопотока (созданного на шаге 2).

Создание профиля верификации является прямолинейным и простым процессом. Вам нужно использовать HTTP POST API, доступный на <https://westeurope.api.cognitive.microsoft.com/spid/v1.0/verificationProfiles>, передав ему правильную локаль и ключ подписки, как показано ниже:

```
POST https://westeurope.api.cognitive.microsoft.com/spid/v1.0/verificationProfiles HTTP/1.1
Content-Type: application/json
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
{
  "locale": "en-us",
}
```

Если вызвать API с недействительным ключом подписки, то вы получите ошибку доступа 401 с сообщением «Доступ запрещен из-за недействительного ключа подписки. Удостоверьтесь, что вы подписаны на API, который вы пытаетесь вызвать, и предоставьте правильный ключ». Во время написания этой книги единственной поддерживаемой локалью являлась en-us. Мы уверены, что этот список поддерживаемых локалей будет со временем расширяться. Если вызвать API с правильным ключом подписки, то вы получите идентификатор GUID с именем verificationProfileID, как показано ниже:

```
Pragma: no-cache
apim-request-id: c245b8ec-2bfb-48a3-b562-cafb4b01789f
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
x-content-type-options: nosniff
Cache-Control: no-cache
Date: Sun, 14 May 2017 08:23:31 GMT
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Content-Length: 71
Content-Type: application/json; charset=utf-8
Expires: -1

{
  "verificationProfileId": "05822be0-5c41-4055-9890-6fa9219baaa6"
}
```

Этот идентификатор в конечном счете будет использоваться для создания идентификатора регистрации. На момент написания этой книги индивидуальная подписка рассчитана на создание максимум 1000 верификационных профилей.

После создания идентификатора verificationProfileID следующий шаг состоит в том, чтобы предложить пользователю сказать одну из верификационных фраз и сохранить аудио. Процесс регистрации говорящего зависит от текста. Чтобы проговорить фразы, сначала необходимо получить список всех поддерживаемых верификационных фраз. Это может быть обеспечено за счет вызова API, как показано ниже, передачей правильной локали и ключа подписки:

```
GET https://westeurope.api.cognitive.microsoft.com/spid/v1.0/verificationPhrases?locale=en-US
HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
Calling with the right subscription key and locale will return all of the phrases. As of writing, the following 10 phrases were returned by the API:
[{"phrase": "i am going to make him an offer he cannot refuse"}, {"phrase": "houston we have had a problem"}, {"phrase": "my voice is my passport verify me"}]
```

```
{
  "phrase": "apple juice tastes funny after toothpaste" },
  { "phrase": "you can get in without your password" },
  { "phrase": "you can activate security system now" },
  { "phrase": "my voice is stronger than passwords" },
  { "phrase": "my password is not your® business" },
  { "phrase": "my name is unknown to you" },
  { "phrase": "be yourself everyone else is already taken" }
```

Вы можете выбрать одну из вышеупомянутых фраз, записать ее как минимум три раза и сохранить в файле .WAV. Давайте назовем этот файл аудиофайлом регистрации, чтобы ссылаться на него в дальнейшем. Каждая запись должна иметь продолжительность от 1 до 15 секунд и быть оцифрована с использованием импульсно-кодовой модуляции PCM с частотой 16 кГц и моноканалом. После того как вы сделали запись согласно вышеупомянутым правилам, вы готовы для регистрации пользователя вызвать API регистрации говорящего и передать записанную речь и идентификатор verificationProfileID, созданный на шаге 1. Регистрация пользователя выполняется путем вызова API, как показано ниже:

```
POST https://westeurope.api.cognitive.microsoft.com/spid/v1.0/verificationProfiles/05822be0-5c41-4055-9890-6fa9219baaa6/enroll HTTP/1.1
Content-Type: multipart/form-data
Host: westeurope.api.cognitive.microsoft.com

[BinaryData]
```

Как показано выше, ранее созданный идентификатор verificationProfileID передается в параметре запроса, в то время как аудиофайл регистрации передается в теле запроса.

- Вызов API с недействительным идентификатором verificationProfileID вернет ошибку 401 «Доступ запрещен».
- Вызов API с действительным идентификатором verificationProfileID и неправильным аудиофайлом вернет ошибку 400 «Неверный запрос» с сообщением «*Недопустимый формат аудио: не является файлом WAVE – отсутствует заголовок RIFF*».
- Вызов API с действительным идентификатором verificationProfileID и надлежащим аудиофайлом вернет отклик в виде документа JSON со следующими полями:
 - enrollmentStatus: указывает одно из трех состояний регистрации: **Enrolling** (В процессе регистрации), **Training** (Выполнение тренировки), **Enrolled** (Зарегистрирован);
 - EnrollmentCounts: указывает количество регистраций говорящего;
 - remainingEnrollments: указывает количество необходимых регистраций;
 - Phrase: распознает фразу, используемую в аудиофайле регистрации.

Если говорящий был успешно зарегистрирован с помощью как минимум трехкратного вызова этого API, то состояние enrollmentStatus получит значение **Enrolled** (Зарегистрирован). Сразу после того, как состояние enrollmentStatus получит значение **Enrolled**, оно может использоваться в сценариях верификации. Если говорящий сразу же сделал вызов, то будет возвращено состояние **Enrolling** (В процессе регистрации), количество регистраций EnrollmentCounts получит значение 1, и количество оставшихся регистраций remainingEnrollments должно получить значение 2.

Верификация говорящего

Процесс верификации говорящего зависит от текста. Это означает, что для регистрации и верификации используется одна и та же фраза. Процесс верификации аналогичен процессу регистрации. Вы должны использовать ту же самую верификационную фразу, которую вы записали на аудио для регистрации. Так же как и в процессе регистрации, вы должны вызвать API с записанной речью и идентификатором verificationProfileID следующим образом:

```
POST https://westeurope.api.cognitive.microsoft.com/spid/v1.0/verify?verificationProfileId={verificationProfileId} HTTP/1.1
Content-Type: application/octet-stream
Host: westeurope.api.cognitive.microsoft.com

[BinaryData]
```

Этот запрос вернет ответ в виде документа JSON со следующими полями:

- **Result**: указывает, была верификация принята (Accept) или отклонена (Reject);
- **Confidence**: показывает уровень уверенности результата. Он может быть низким (Low), нормальным (Normal) или высоким (High).

```
{
  "result" : "Accept",
  "confidence" : "Normal",
  "phrase": "my voice is my passport verify me"
}
```

Результат имеет два варианта: Accept или Reject. Accept означает, что верификация была принята. Reject означает, что проверка была отклонена. Confidence указывает на уровень уверенности верификации; он может быть низким, нормальным или высоким. Очень важно опираться на комбинацию результатов и оценок уверенности. Если результат отмечен как Accept с низкой оценкой уверенности, то это будет сигнализировать о необходимости провести дополнительную тренировку модели. Уровень уверенности, как минимум, должен быть нормальным или высоким. В приведенном выше примере результат сообщает о том, что верификация была принята с нормальной оценкой уверенности, имея в виду, что все в порядке и можно переходить к дальнейшей работе.

Регистрация для идентификации говорящего

Так же как и в первом шаге регистрации, для верификации говорящего пользователь должен сначала создать идентификационный профиль. URL-адрес для вызова регистрации для идентификации отличается от регистрации для верификации. Процесс создания идентификационного профиля является прямолинейным и простым. Вам нужно получить отклик от HTTP POST API, доступный на <https://westeurope.api.cognitive.microsoft.com/spid/v1.0/verificationProfiles>, передав правильную локаль и ключ подписки. На момент написания этой книги идентификационный профиль поддерживал дополнительную локаль китайского языка (zn-CN).

```
POST https://westeurope.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles HTTP/1.1
Content-Type: application/json
Host: westeurope.api.cognitive.microsoft.com
```

```
Ocp-Apim-Subscription-Key:.....
```

```
{
  "locale": "en-us",
}
```

Если вызвать API с правильным ключом подписки и локалью, API вернет идентификатор GUID с именем `IdentificationProfileID`. Этот идентификатор однозначно идентифицирует говорящего и будет использоваться позже для создания идентификатора регистрации и распознавания голоса из группы беседующих людей.

Система идентификации говорящего не зависит от текста. Это означает, что при регистрации и проверке пользователь в качестве речевого образца может использовать любой текст. В этом кроется причина, почему, в отличие от регистрации для верификации говорящего, регистрация для идентификации является двухступенчатым процессом. Отсутствуют какие-либо ограничения на то, что именно пользователи наговаривают в аудиозаписи, поэтому шаг, связанный с получением всех фраз, не требуется.

Как показано ниже, идентификатор `IdentificationProfileID` передается в качестве параметра запроса, и аудиофайл регистрации передается в теле запроса. Каждая запись должна иметь продолжительность от 5 секунд до 5 минут и должна быть оцифрована с использованием импульсно-кодовой модуляции PCM с частотой 16 кГц. После удаления тишины минимальная рекомендуемая сохраненная речь для регистрации составляет приблизительно 30 секунд. У вас есть возможность назначить полю `shortAudio` значение `истина/ложь` в зависимости от продолжительности аудио, которое вы желаете предоставить. Это дает сигнал службе отказаться от рекомендуемого предела размера аудио, необходимого для регистрации, и состояние регистрации профиля соответствующим образом изменится с **Enrolling** (В процессе регистрации) на **Enrolled** (Зарегистрирован). Так вы можете отправлять аудиофайлы продолжительностью с 1 секунды, но не дольше 5 минут. Регистрация для идентификации пользователя выполняется путем вызова API по адресу:

```
https://westeurope.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles/
{identificationProfileId}/enroll[?shortAudio]
Content-Type: multipart/form-data
Host: westeurope.api.cognitive.microsoft.com

[BinaryData]
```

- вызов API с недействительным идентификатором `IdentificationProfileID` вернет ошибку 401 «Доступ запрещен»;
- вызов API с действительным идентификатором `IdentificationProfileID` и неправильным аудиофайлом вернет ошибку 400 «Неверный запрос» с сообщением «*Недопустимый формат аудио: не является файлом WAVE – отсутствует заголовок RIFF*»;
- вы получите ошибку 500 с сообщением «*SpeakerInvalid*» (Недопустимый говорящий), если при регистрации произойдет любая другая ошибка;
- вызов API с действительным идентификатором `IdentificationProfileID` и надлежащим аудиофайлом вернет отклик в виде документа JSON со следующими полями:

- Accepted: служба успешно приняла запрос;
- Operation URL: используется для извлечения состояния результата позже в операционном API.

Идентификация говорящего

Как отмечалось ранее, идентификация человека помогает распознавать и идентифицировать человека из группы говорящих. Сразу после того, как вы зарегистрировали всех говорящих и для всех них имеете отклик Accepted, вы можете распознать одного человека из группы говорящих путем вызова API идентификации говорящего следующим образом:

```
https://westeurope.api.cognitive.microsoft.com/spid/v1.0/identify?identificationProfileIds={identificationProfileIds} [&shortAudio]
Content-Type: multipart/form-data
Host: westeurope.api.cognitive.microsoft.com
[BinaryData]
```

Как показано выше, вам нужно передать все (созданные ранее) идентификационные профили в качестве значений, отделенных запятой. Аналогично регистрации для идентификации говорящего, вам нужно передать аудио в теле запроса и назначить необязательному полю shortAudio значение истина/ложь в зависимости от продолжительности аудио. Вы получите результат из операционного URL-адреса, который затем можно использовать для отслеживания состояния операции.

Состояние операции

Теперь вы понимаете, как создавать идентификационные профили. Иногда возникают ситуации, когда требуется проверить точное состояние операции. Например, позже может возникнуть желание проверить состояние операции или состояние регистрации профиля идентификации говорящего. Может возникнуть желание проверить, прошла ли регистрация успешно или же по-прежнему находится в тренировочном режиме. Процесс распознавания говорящего в Microsoft в значительной степени упрощает работу с любой операционной задачей; все это можно проделать одним-единственным вызовом операционного URL-адреса:

```
GET https://westeurope.api.cognitive.microsoft.com/spid/v1.0/operations/{operationID} HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
```

Вызов вышеупомянутого API с правильным идентификатором operationID и ключом подписки вернет отклик в виде документа JSON. Упомянутый здесь идентификатор operationID доступен либо путем запроса на создание регистрации для идентификационного профиля, либо путем идентификации говорящего, которую мы рассмотрим чуть позже. Следующий ниже фрагмент кода показывает отклик в виде документа JSON, если состояние регистрации успешно:

```
{
  "status": "succeeded", // [notstarted|running|failed|succeeded]
  "createdDateTime": "2017-01-13T01:28:23Z",
  "lastActionDateTime": "2017-01-15T01:37:23Z",
  "processingResult":
  {
    "enrollmentStatus" : "Enrolled", // [Enrolled|Enrolling|Training]
```

```

    "remainingEnrollmentSpeechTime" : 0.0,
    "speechTime" : 0.0,
    "enrollmentSpeechTime":0.0
}
}

```

После получения отклика в виде JSON вам сначала нужно проверить состояние. Оно может иметь одно из четырех значений:

- Not Started** (Не начато);
- Running** (Выполняется);
- Failed** (Неуспешно);
- Succeeded** (Успешно).

Если состояние равняется **Succeeded**, то обратитесь к результату обработки в формате JSON, как показано выше. Если состояние имеет любое другое значение, то, возможно, вам нужно проверить другие поля. Например, если состояние говорит о неуспешности, т. е. равно **Failed**, то стоит проверить поле сообщения, чтобы увидеть причину неудачи. Кроме этого, можно выполнять и другие операционные задачи; см. табл. 7.2, в которой приведена некоторая информация.

Таблица 7-2. Другие операционные задачи



Имя API	Описание
Удалить профиль	DELETE https://westeurope.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles/{identificationProfileId} Безвозвратно удаляет из службы все профили идентификации говорящего вместе со всеми связанными с ними регистрациями
Получить все профили	GET https://westeurope.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles Получает все профили идентификации говорящего в пределах подписки. Вернет отклик в виде документа JSON со всеми идентификационными профилями в пределах подписки с дополнительной информацией, включая состояние регистрации, локаль, количество секунд, использованных для идентификации, количество секунд, оставшихся для успешной регистрации, и т. д.
Получить конкретный профиль	GET https://westeurope.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles/{identificationProfileId} Передает URL-адрес с конкретным идентификатором идентификационного профиля, чтобы предоставить вам профиль идентификации говорящего, связанный с этим идентификатором. Этот API вызывается, чтобы получить состояние регистрации идентификационного профиля
Удалить все регистрации	POST https://westeurope.api.cognitive.microsoft.com/spid/v1.0/identificationProfiles/{identificationProfileId}/reset Безвозвратно удаляет из службы все регистраций, связанные с заданным профилем идентификации говорящего

Резюме

В этой главе вы узнали, как использовать API Bing обработки речи (Bing Speech API) с целью получения функциональности преобразования речи в текст и текста в речь. В этой главе также была подробно рассмотрена необходимость службы персонализированной обработки речи Custom Speech Service (ранее называвшейся CRIS). В конце главы вы также исследовали API распознавания говорящего и узнали, как его использовать для выполнения идентификации, верификации и регистрации говорящего. В **следующей главе** вам будет предложен обзор использования API Bing поиска и других разновидностей поиска.

Глава 8

Применение поисковых предложений

Только представьте жизнь без поисковых систем. Всякий раз, когда вам нужен ответ, вы автоматически обращаетесь к сайтам наподобие Google и Bing. За последнее десятилетие или около того история развития поисковых систем прошла длинный путь. Прошло то время, когда результаты поиска ограничивались выбором данных из базы данных. С увеличивающимся объемом данных, производимым каждой секундой, и с изобретением новых алгоритмов поиск вышел за пределы текстового поля. Поиск перешел от эры обратной отправки результатов к частичным обратным отправкам и теперь даже к автозаполнению. Более того, поиск также перешел от простого предоставления ссылок и текстовых результатов к новостям и контекстному поиску, изображениям и видео; он перешел от реактивного поиска к превентивному и от обобщенного поиска к персонализированному опыту. Google, без сомнения, придал всей поисковой системе новое измерение. В течение прошедшего десятилетия или около того вашей первой поисковой системой оставался Google. Эта поисковая система помогала вам получать информацию от погоды на ближайшие выходные до счета в матче и последних новостей, видео или чего-либо еще, что вы хотите знать.

Вы можете удивиться, почему это мы обсуждаем поиск в Google, когда тема нашей главы – когнитивный поиск, предлагаемый компанией Microsoft. Дело в том, что Google – это гигант в области поиска. Сравнение поисковой системы Google со своими конкурентами может вызвать у многих мороз по коже. Google – это впечатляющая компания, на которую работает самая лучшая команда специалистов в области науки о данных и инженеров по программному обеспечению. Однако компания Microsoft, несомненно, сделала большие успехи в области поиска, внедрив в свою платформу поисковую систему Bing. Microsoft Поиск, который основан на Bing, является альтернативой Google. Одна из миссий API Bing поиска состоит в том, чтобы противостоять монополии Google и обеспечивать альтернативу. В настоящее время Bing является второй по мощности поисковой системой после Google. Вы когда-нибудь думали о реализации возможностей Google или Microsoft Bing в своем корпоративном приложении? В этой главе вы познакомитесь с новыми предложениями от Bing Поиск. В конце этой главы вы:

- познакомитесь с историей и эволюцией поисковой системы Microsoft Bing;
- узнаете, как Bing конкурирует с Google;
- познакомитесь с тремя «П» поиска;

- узнаете о предлагаемых решениях API Bing Поиск;
- узнаете, как задействовать предлагаемые решения API Bing Поиска в своем приложении.

Поиск повсюду

Поиск присутствует повсюду. Мы используем поиск на наших настольных компьютерах и в мобильных телефонах. Мы применяем веб-поиск, чтобы выполнять рутинную работу в своей повседневной жизни. С новыми сетевыми устройствами, которые появляются каждый день, нарастает потребность в более умном поиске. По последним отчетам к концу 2020 г. ожидается появление 50 миллиардов сетевых устройств, и каждому из них требуется сетевой поиск. Вне всяких сомнений, поиск, кроме всего прочего, стал неотъемлемой частью наших любимых приложений. Возьмем покупки в сети. Вы, вероятно, часто заходите на Amazon.com, не только из-за большого ассортимента товаров, но и потому, что поиск в Amazon более изящен, по сравнению с его конкурентами. И именно по этой причине Stackoverflow.com стал одним из дежурных мест, куда заходят в поисках технических решений. Поиск подразделяется на две широкие категории:

- явный поиск;
- неявный поиск.



Явный поиск – это когда вы набираете что-то в поле поиска и получаете результаты. Неявный поиск – это очень умная форма поиска, в котором приложение производит опережающий неявный предсказательный поиск, опережая ваши желания. Например, всякий раз, когда вы выбираете любой товар на Amazon, вы также получаете список похожих товаров. Вы также получаете список товаров, к которым обратились другие пользователи, когда они рассматривали выбранный товар (рис. 8.1). Это означает, что иногда поиск становится более опережающим, проактивным, и от вас не требуется выполнять какой-то определенный поиск.

Почти все корпоративные и социально-медийные приложения имеют одну общую черту. И это поиск. Представьте, как несколько лет назад вы набирали запрос в поле поиска, и ваши приложения ASP.NET выполняли обратную отправку результатов (возможно, частичных). Поиск состоит из трех аспектов:

- 1) повсеместность (везде на любом устройстве);
- 2) предсказательность;
- 3) проактивность (опережающий поиск).

ПОВСЕМЕСТНОСТЬ, ПРЕДСКАЗАТЕЛЬНОСТЬ, ПРОАКТИВНОСТЬ (ТРИ «П» ПОИСКА)



Информационные технологии все больше становятся доступными всем и каждому в любое время, где угодно и на любом устройстве. У вас есть доступ к Интернету по телефону. Ваш телефон больше не находится в нескольких метрах от вас. Любопытно, что большинство из нас обращается к своему телефону несколько сотен раз за день. Поиск должен иметься в распоряжении где угодно в любое время и на любом устройстве.

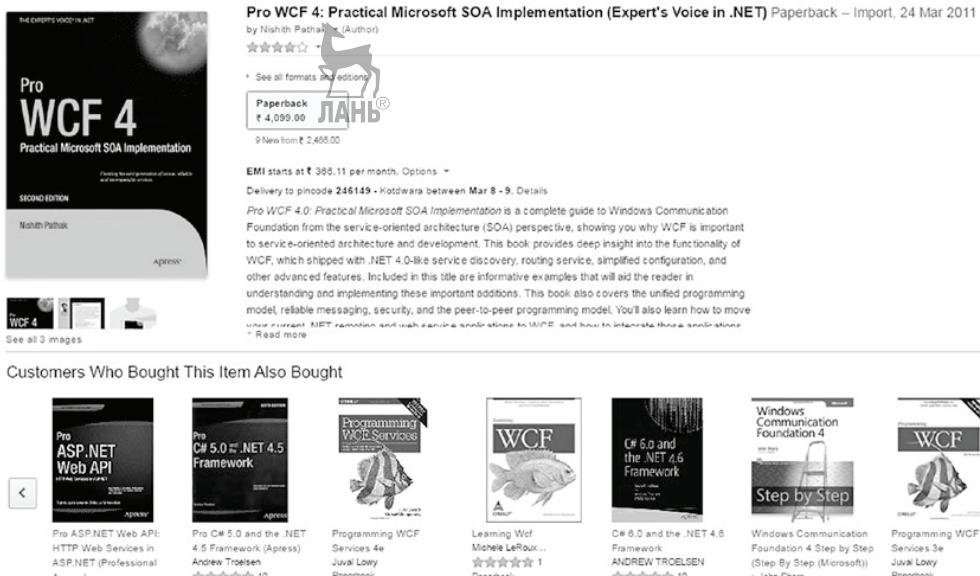


Рис. 8.1 ❖ Страница подробной информации о товаре на Amazon

Предсказательность – это процесс, с помощью которого система может предугадать, что произойдет в будущем. Это можно обеспечить за счет интеграции поиска с большим объемом данных. Предсказательность очень близко связана с проактивностью; на самом деле они работают рука об руку. Если мы способны предсказывать, то это помогает нам делать это в опережающем режиме, т. е. проактивно. *Проактивность* означает использование поиска для предоставления ответов, прежде чем будет задан вопрос или даже прежде чем о нем задумаются. Выполнение такого поиска в корпоративных приложениях придает поиску большую размерность (объемность).

Разве не было бы здорово, если бы мы могли получать рекомендации и результаты неявного поиска в наших поисковых системах точно так же, как в Google или Bing? На данный момент в Google не может выполнять неявный поиск в своей поисковой системе, кроме предоставления данных автозаполнения на основе исторических данных. Поиск в Google прежде всего опирается на контекстный поиск, и поэтому ваши результаты поиска зависят от того, что вы искали в прошлом.

Теперь давайте рассмотрим Bing. Если вы используете Bing в Windows 10 или ее более поздней версии, откройте домашнюю страницу когнитивных служб, выделите текст «Компьютерное зрение», щелкните правой кнопкой мыши и выберите «спросить Кортану», как показано на рис. 8.2.

Результаты показывают вам не только определение, но и всю информацию, связанную с научной конференцией, курсами, людьми и т. д. Это классический пример предсказательного и проактивного поиска в Microsoft Bing. Теперь вы знаете об одной из хитростей, доступных в Bing. Попробуйте это в Google. Вам нужно открыть Google, набрать поисковое ключевое слово, и вы получите результаты в различных поисковых ссылках. Правда, неявный, проактивный поиск луч-

ше? Microsoft хочет гарантировать, чтобы поиск в Bing двигался с опережением в направлении к большей предсказательности и проактивности.

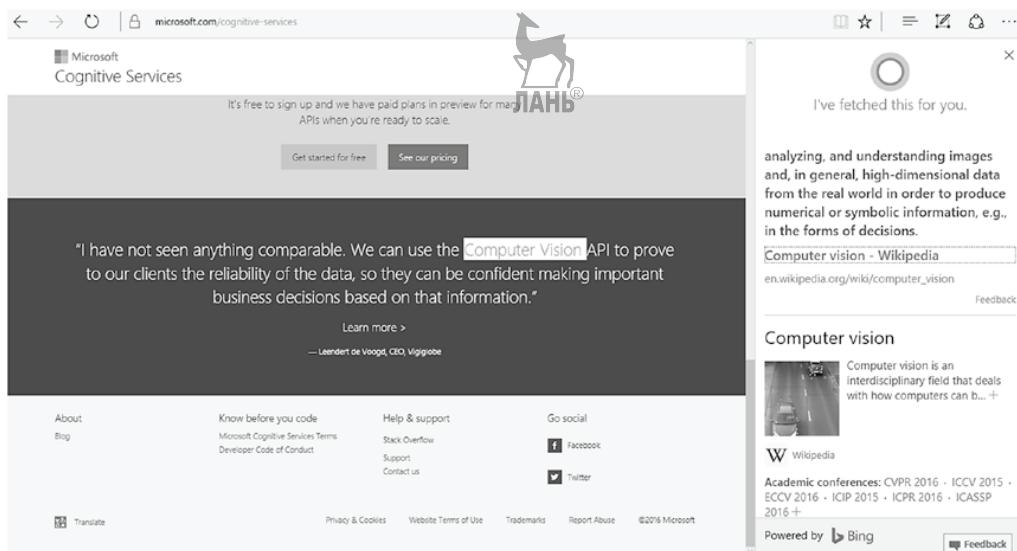


Рис. 8.2 ❖ Информация, которую Кортана показывает после выбора текста в браузере Edge

i Кортана (Cortana) – это поисковый интеллект и цифровой помощник, который помогает вам выполнять свои задачи в Windows. Он появился в Windows 10, но теперь доступен на различных устройствах, таких как телефоны, планшеты и машины Windows. В будущем вы увидите, что Кортана будет интегрирована в другие более умные устройства, такие как HoloLens, Xbox и т. д., что позволит сделать Кортану полностью доступным поисковым средством. Это классический пример повсеместного поиска.

История Bing

Мы полагаем, что лучший способ разобраться в любой технологии – это узнать ее истоки. Давайте сначала разберемся в причине, почему Bing появился. Microsoft доминировала в создании ОС на основе ГИП в течение прошедших двух-трех десятилетий. В конце 1990-х годов Microsoft выпустила свою первую поисковую систему под названием MSN Поиск (рис. 8.3). MSN Поиск первоначально получал результаты поиска, выполнявшиеся другими компаниями, такими как Inktomi и AltaVista, и только потом начал использовать собственную поисковую систему Microsoft. В течение этого времени компания Microsoft по-прежнему опиралась на третьих лиц, для того чтобы получать результаты в виде изображений.

Во время этого периода одна из задач поиска состояла в обеспечении пользователям более быстрых результатов. В середине 2006 г. Microsoft сделала главный рывок в поиске, заменив свою версию MSN Поиска на Windows Live Search. Это позволило сделать поиск частью Windows Live. Windows Live Search предоставлял дополнительные возможности, например такие, как поиск новостей, музыки и поиск на компьютере. Одним из радикальных изменений, которое привнес Windows

Live Search, было использование внутренней поисковой системы Microsoft и алгоритмов для возвращения результатов. Также имелись главные радикальные изменения в продвижении некоторых предлагаемых решений Microsoft Live.



Рис. 8.3 ♦ MSN Поиск (бета)

В Microsoft понимали, что в следующем десятилетии поиск станет одним из развивающихся рынков. Однако, несмотря на продвижение такого количества предложений, поиск Microsoft Live Search далеко отставал от подобных ему систем Google и Yahoo. По сути дела, в это время доля Microsoft в области поиска неуклонно падала. После долгих дискуссий, встреч и исследований в области продвижения своей торговой марки Microsoft удалила слово «live» из поиска и затем придумала новый фирменный знак: Bing (рис. 8.4).



Рис. 8.4 ♦ Первоначальная эмблема Bing

2009 год стал для поисковой системы Microsoft ключевым годом. В этом году на рынок не только был выпущен Bing, но и подписано десятилетнее соглашение с Yahoo, чтобы заменить на сайте Yahoo поисковую систему Yahoo на поисковую систему Bing. С момента принятия названия «Bing» компания Microsoft ни разу не пожалела об этом и стала продвигать все новые и новые предложения.

i Во время поиска нового фирменного имени обсуждались самые разные названия, и самым ближайшим конкурентом Bing было слово Bang. Слово Bing было выбрано в качестве окончательного имени, потому что оно было небольшим, односложным, легко произносимым и узнаваемым.

Сначала компания Microsoft стремилась, чтобы в каждом доме были настольные компьютеры. В последнее время компания Microsoft пересмотрела свою цель, которая теперь выражается как «наделение каждого человека и каждой организации на планете возможностями достичь большего». Эта цель более человекоцентрична.

У этой цели есть много оговорок, в которых прописывается предоставление большого количества услуг везде и повсюду. Доведение API поисковой системы Bing до простых людей и предоставление доступа к некоторым службам посредством механизма REST API являются одним из путей реализации новой цели Microsoft. Теперь, когда вы узнали историю возникновения Bing, давайте рассмотрим некоторые уникальные особенности поисковой системы Bing Поиск.

В ЧЕМ УНИКАЛЬНОСТЬ BING?

В настоящее время Bing представляет собой механизм Microsoft поиска в вебе. Конечно же, для некоторых фанатов Microsoft Bing – это больше, чем строка поиска. И разумеется, в нем существуют особенности, которые делают Bing уникальным. Хотя в центре настоящей главы находится API когнитивных служб, очень важно сообщить об особенностях поисковой системы Microsoft Bing, так как данная информация позволит понять некоторые API Bing, которые Microsoft может выпустить в ближайшем будущем.

Bing и Google предлагают контекстный поиск, но в Bing есть свои уникальные особенности. Например, Bing позволяет пользователю выполнять предварительные просмотры видео, прежде чем выбрать его для полного просмотра. Эта функция экономит время, позволяя пользователю быстро найти нужное видео. Чтобы его просмотреть, достаточно установить курсор над видео в разделе поиска в Bing. Довольно часто мы ищем информацию об известных личностях и хотим знать их биографию. Для таких известных личностей Bing предлагает уникальную функцию временной шкалы. Bing также предлагает вознаграждения. На данный момент этот функционал доступен только в некоторых регионах наряду с США. Если вы оказались в этом регионе, не забудьте посетить Bing.com/rewards и получить справку на счет вознаграждений за свои поисковые запросы в Bing. Вы также можете сохранять в Bing закладки, посетив Bing.com/saves. Одна из областей, где Bing обогнал Google, – это обширный набор API когнитивных служб для конечных пользователей.

API поиска

API Bing поиска является одним из пяти столбов когнитивных служб, предлагаемых Microsoft. API Bing поиска помогает задействовать мощь поискового механизма Bing. Среди них:

- API Bing автозаполнения;
- API Bing поиска изображений;
- API Bing поиска новостей;
- API Bing поиска видео;
- API Bing поиска в сети.

Давайте теперь подробно рассмотрим каждый из этих API.

API Bing автозаполнения

API Bing автозаполнения помогает пользователю печатать меньше и делать больше. Как следует из названия API автозаполнения обеспечивает поиск в вашем приложении опережающим интеллектуальным вводом, когдалагаются варианты поиска непосредственно из Bing поиска. То есть пользователь набирает

текст запроса внутри поля поиска, а система предлагает готовые варианты запроса. Как правило, к этому API обращаются, когда требуется задействовать автозаполнение в поле поиска вашего приложения. По мере того как пользователь набирает текст в поле поиска, ваше приложение вызывает API, чтобы показать список предлагаемых вариантов. А пользователь получает возможность выбрать вариант, устраивающий его. Рисунок 8.5 показывает разные варианты автозаполнения, основываясь на том, что пользователь набрал в поле поиска. В идеальном сценарии вы будете вызывать API автозаполнения по мере ввода пользователем каждого символа в поле поиска. Основываясь на каждом введенном символе, API Bing автозаполнения будет выводить разные релевантные результаты поиска. На внутреннем уровне это достигается посредством частичного поискового запроса в Bing и получения результатов. Этим результатом будет комбинация контекстного поиска с тем, что другие пользователи искали в прошлом.

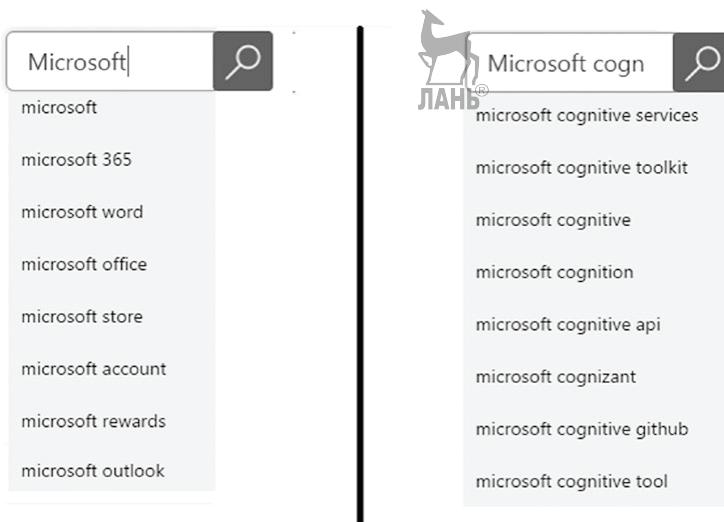


Рис. 8.5 ❖ Разные результаты на основе того, что пользователь печатает

Как использовать API Bing автозаполнения

Все API Bing требуют наличия ключа подписки для доступа к службе. На данный момент мы предполагаем, что у вас есть под рукой ключ подписки. Если вы сразу перешли к этой главе или же хотите знать шаги по созданию подписного ключа, обратитесь к главе 2.

Процесс использования службы Bing API автозаполнения является простым; вам нужно обратиться с GET-запросом на <https://api.cognitive.microsoft.com/Bing/v5.0/suggestions>, передав поисковый запрос в качестве параметра и ключ подписки в заголовке Ocp-Apim-Subscription-Key. Например, если вы хотите искать Microsoft, то ваш запрос должен показать что-то вроде этого:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/suggestions/?q=Microsoft HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....  
.....
```

Обратите внимание, что вышеупомянутый URL-адрес использует протокол HTTPS. HTTP-вызовы, конечно же, не поддерживаются ни для одного из запросов к когнитивным службам. Кроме того, посмотрите на версию в URL-адресе. При написании настоящей книги последней версией API автозаполнения была V5.0¹. Важно понимать, что Microsoft продолжит выпускать новые версии своих API. Microsoft, безусловно, оставляет за собой право внести изменения в версию V6.0 или более позднюю, поэтому убедитесь, какую версию службы вы используете. Вместе с тем Microsoft будет поддерживать существующую версию API, поэтому если окажется, что вы получили эту книгу после того, как была выпущена V7.0, то вы по-прежнему сможете вызывать и использовать V5.0 API без внесения каких-то изменений. Также обратите внимание, что ключевое поисковое слово передается через параметр q.

Если принять, что ваш ключ подписки правильный, то вы получите отклик в виде документа JSON. Приведенный ниже фрагмент кода показывает подмножество документа JSON, полученного при поиске ключевого слова Microsoft:

```
{
  "_type": "Suggestions",
  "queryContext": {
    "originalQuery": "microsoft"
  },
  "suggestionGroups": [
    {
      "name": "Web",
      "searchSuggestions": [
        {
          "url": "https://www.bing.com/cr?IG=D267DC92EC52478D86A10652DCDE2015&CID=297FEC8A42866B4712E1E6F743636A98&rd=1&h=VmqXfTxdcTzsDb12vh_sG9LXP9iKyoAwgw5oqUaf60&v=1&r=https%3a%2f%2fwww.bing.com%2fsearch%3fq%3dmicrosoft%2baccount%26FORM%3dUSBAPI&p=DevEx_5003.1",
          "displayText": "microsoft account",
          "query": "microsoft account",
          "searchKind": "WebSearch"
        },
        {
          "url": "https://www.bing.com/cr?IG=D267DC92EC52478D86A10652DCDE2015&CID=297FEC8A42866B4712E1E6F743636A98&rd=1&h=oEkAB-ob-fIDhkOiJU7ekQdYucx-voM5hWz_ZtijbFQ&v=1&r=https%3a%2f%2fwww.bing.com%2fsearch%3fq%3dmicrosoft%26FORM%3dUSBAPI&p=DevEx_5004.1",
          "displayText": "microsoft",
          "query": "microsoft",
          "searchKind": "WebSearch"
        },
        {
          "url": "https://www.bing.com/cr?IG=D267DC92EC52478D86A10652DCDE2015&CID=297FEC8A42866B4712E1E6F743636A98&rd=1&h=l_0yDf4fwfCsVS6TzqIT1Vz-IX8Z15JkxF3VLz48&v=1&r=https%3a%2f%2fwww.bing.com%2fsearch%3fq%3dmicrosoft%2b365%26FORM%3dUSBA"
        }
      ]
    }
  ]
}
```



¹ На момент перевода (февраль 2018 г.) текущей версией API Bing автозаполнения была версия 7. – Прим. перев.

```

    "PI&p=DevEx,5005.1",
    "displayText": "microsoft 365",
    "query": "microsoft 365",
    "searchKind": "WebSearch"
}

```

Если рассмотреть результат JSON, то вы увидите, что каждый предлагаемый вариант поиска является объектом, который содержит не только результат поиска, но и URL-адрес. Как правило, вы берете результат, как есть, и показываете эти результаты поиска как гиперссылки. Щелчок мышью на любом показанном тексте перенаправит вас к фактическому поиску в Bing с этим ключевым словом. Также обратите внимание, что все предлагаемые варианты поиска сгруппированы под одним объектом с именем SearchSuggestions. Вверху вы видите объект queryContext, где хранится строка запроса, которую Bing использовал для поиска. Давайте исследуем, как можно сделать результаты поиска более релевантными, добавив некоторые необязательные параметры и заголовки.

1. `mkt` – это параметр запроса, который будет использоваться большую часть времени. Он представляет рынок, откуда прибудут результаты. Рынок представляется как код языка (дефис), после которого идет код страны. Классический пример – `en-US`. Код рынка нечувствителен к регистру. Используйте этот параметр, если ваши результаты указывают на конкретную страну и конкретный язык. Для Bing есть предварительно заданный список кодов рынков. Чтобы узнать о нем больше, поищите в Интернете, потому что все время добавляются новые коды. Если упомянутый параметр `mkt` не соответствует ни одному элементу списка кода рынков, либо если результаты для этого рынка не доступны, Bing интеллектуально возвращает самый близкий соответствующий код и указывает используемый рынок в заголовке отклика `BingAPIs-Market`. Вам следует рассмотреть заголовок в возвращенном результате, чтобы убедиться, что результаты являются точными. Рисунок 8.6 показывает разные результаты, возвращаемые по одинаковому ключевому слову. Слева находятся результаты для рынка `en-us`. Справа – результаты для рынка `ja-jp`.

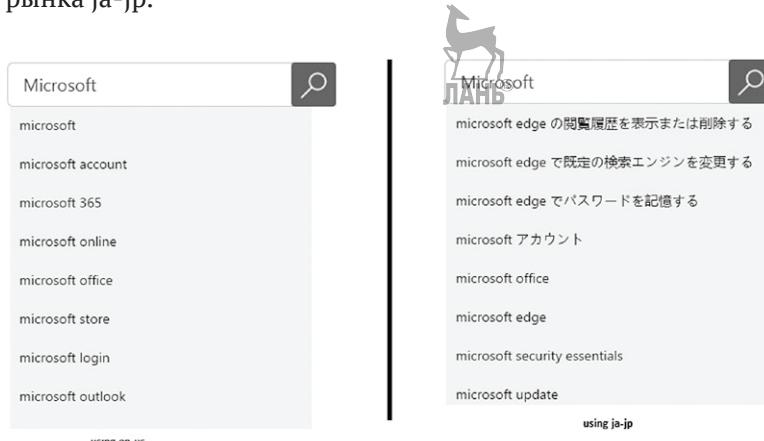


Рис. 8.6 ❖ Разные результаты для разных рынков для поиска с одинаковыми ключевыми словами

2. *cc* – это параметр запроса, который представляет двухсимвольный код страны. Используйте этот параметр, чтобы представлять результаты, поступающие из конкретной страны. Не путайте с параметром *mkt*. На самом деле *cc* и *mkt* взаимоисключают друг друга. Параметр *mkt* следует использовать, если вы работаете со страной и одним языком. Если одна страна имеет несколько языков, то вам следует использовать параметр *cc*. Передайте список языков в виде значений, разделенных запятыми, в параметр заголовка *Accept-Language*. Обратите внимание на порядок следования языков; список должен быть упорядочен в порядке убывания.
3. Параметр запроса *setLang* обычно работает с параметром *mkt*, чтобы задать язык. Как правило, если вы хотите, чтобы ваша страница пользовательского интерфейса была показана на другом языке, вам следует задать язык, отличающийся от языка, указанного в параметре *mkt*.
4. Иногда бывают ситуации, когда запрос может быть правильным, но результат не возвращается. В таких случаях отклик от Bing будет содержать идентификатор *BingAPI-traceID*, который содержит запись системного журнала, связанную с запросом. Отправьте этот идентификатор трассировки в команду поддержки обратной связи в Bing, чтобы получить дополнительную информацию.
5. *Retry-After* появляется в заголовке отклика, если превышено количество запросов, предусмотренных в секунду (QPS) или в месяц (QPM). Эта числовая величина показывает, сколько секунд пользователь должен выждать, прежде чем отправить следующий запрос.
6. Параметр *User-Agent* может использоваться в качестве заголовка запроса, чтобы сообщить Bing об устройстве и ОС, из которых вы отправляете запрос. Это позволит Bing понимать ресурс запроса и оптимизировать результаты.
7. По умолчанию любой запрос в API Bing рассматривается как уникальный, даже если вы передаете его с тем же самым ключом подписки. Когда Bing отправляет отклик, он генерирует идентификатор нового клиента. Это иногда приводило к тому, что пользователь получал другой отклик в последующих вызовах или результатах, прибывающих из Bing. Сейчас Bing позволяет пользователю во время запроса передавать тот же самый идентификатор клиента, чтобы поисковая система понимала, что запрос поступает от одного человека. Идентификатор клиента можно передать, добавив его в заголовок *X-MSEdge-ClientID*. Единственный случай, когда не следует передавать идентификатор клиента, – это при первом запросе, когда Bing производит уникальный идентификатор. Последующие запросы должны содержать идентификатор, сгенерированный в первом отклике Bing.



Очень важно понимать, что некоторые заголовки и параметры характерны для большинства когнитивных API Bing. Мы не будем слишком вдаваться в подробности, но вам всегда следует указывать эти параметры.

8. Существует множество сценариев, в которых требуется получить информационное содержимое с привязкой к месту поиска. Bing позволяет передавать IP-адрес клиентского устройства, которое может использоваться для идентификации местоположения. Затем Bing возвращает отклик с соответствующим информационным содержимым, касающимся места поиска, откуда был произведен запрос. Чтобы отправить IP-адрес, используйте дополнительный заголовок отклика *X-Search-ClientIP*. Если вы не имеете IP-

адреса, но имеете геолокационную привязку, то можете передать эту информацию в качестве пары ключ/значение с помощью заголовка X-Search-Location.

И Если вы будете пользоваться автозаполнением очень часто, то заметите разницу между Google и Bing. В момент написания этой книги Microsoft Bing поддерживал восемь вариантов автозаполнения, по сравнению с четырьмя вариантами в Google.

Как и все остальные когнитивные службы Microsoft, API автозаполнения также имеет бесплатные и платные категории подписки. Во время написания этой книги бесплатная категория позволяла выполнять 10K вызовов в месяц и была доступна в течение трех месяцев. Платные подписки имеют три категории: S1, S2 и S3. Категории S1, S2 и S3 позволяют выполнять соответственно 10K, 100K и 1M вызовов в месяц. Категория S1 доступна с минимальной стоимостью \$3 и не предусматривает никаких дополнительных платежей за превышение установленной верхней границы. Категория S2 доступна за \$30, и категория S3 – за \$270 в месяц. Все они предусматривают дополнительный платеж в размере \$30 за дополнительные 100K транзакций.

API BING ПОИСКА ИЗОБРАЖЕНИЙ

За прошедшее десятилетие поиск изображений получил широкое распространение. API Bing поиска изображений предоставляет разработчикам возможность внедрять в свои приложения аналогичный опыт, что и Bing.com/images. Здесь необходимо отметить одну особенность. API Bing поиска изображений следует использовать, лишь когда требуется найти лишь изображение. API Bing поиска изображений не только предоставляет возможность получать результаты поиска на основе изображений, но и возможность выбора параметров запроса, а также с помощью заголовка индивидуально настраивать желаемые результаты. Сейчас одной из причин популярности поиска на основе изображений является то, что этот поиск более естественный и потому более эффективен. Его результаты легко визуализируются, позволяя смотреть не на текст, а на содержимое изображений. Рисунок 8.7 показывает пример демонстрации изображений, возвращенных в результате поиска того же самого ключевого слова, но с разными параметрами.



Поиск на основе изображений с ключевым словом Cute Animal



Поиск на основе изображений с ключевым словом Cute Animal («Симпатичное животное») и типом «штриховой» (line)

Рис. 8.7 ♦ Разные изображения, полученные в результате одинакового запроса, но с разными параметрами

Как использовать API Bing поиска изображений

Так же как и для API Bing автозаполнения, для использования API Bing поиска изображения следует отправить GET-запрос на <https://api.cognitive.microsoft.com/Bing/v5.0/images/search>, передав ключевое слово в параметре q для получения информации в виде изображений и ключ подписки в заголовке с ключом Ocp-Apim-Subscription-Key.

Например, если вы хотите получить изображения, связанные с Microsoft, то ваш обобщенный запрос должен выглядеть следующим образом:

```
POST https://api.cognitive.microsoft.com/bing/v5.0/images/search?q=Microsoft HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
```

Обратите внимание, так же как и в API автозаполнения, в URL-адресе используются протокол HTTPS и версия V5.0. Параметр, на основе которого будет выполняться поиск, является точно таким же. Единственная разница состоит в URL-адресе, который вызывается. Если ключ подписки будет правильным, то вызов, такой же, что приведен выше, в качестве отклика доставит следующее ниже множество документа JSON:

```
{
  "_type": "Images",
  "instrumentation": {
    "pageLoadPingUrl": "https://www.bingapis.com/api/ping/pageload?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=28C0E7D43623671417E9EDA937C66659&Type=Event.CPT&DATA=0"
  },
  "readLink": "https://api.cognitive.microsoft.com/api/v5/images/search?q=Microsoft",
  "webSearchUrl": "https://www.bing.com/cr?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=28C0E7D43623671417E9EDA937C66659&rd=1&h=sfoMvg6zTPpJcZkSlNz-bTguuG0uLuDscR5LiHjNc&v=1&r=https%3a%2f%2fwww.bing.com%2fimages%2fsearch%3fq%3dMicrosoft%26FORM%3d0IIARP&p=DevEx,5245.1",
  "totalEstimatedMatches": 926,
  "value": [
    {
      "name": "Microsoft - Stretch",
      "webSearchUrl": "https://www.bing.com/cr?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=28C0E7D43623671417E9EDA937C66659&rd=1&h=iniLs-zQ90gF1Mx35V-3tPiQh-6x2qK90ka3Qi_AkRE&v=1&r=https%3a%2f%2fwww.bing.com%2fimages%2fsearch%3fview%3ddetailv2%26FORM%3d0IIIRPO%26q%3dMicrosoft%26id%3dDA06857EE5270F0FA81C85FC6A432E3E1E7FD020%26simid%3d608003027952795739&p=DevEx,5006.1",
      "thumbnailUrl": "https://tse3.mm.bing.net/th?id=OIP.4gTvU2SEGruWNTza0oa_nQE&c=1&pid=Api",
      "datePublished": "2017-04-08T03:34:00",
      "contentUrl": "http://www.bing.com/cr?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=28C0E7D43623671417E9EDA937C66659&rd=1&h=PPUkxm_E770dpe1mCL2fURxcgvAfrFB6aUbZq9Zy3uc&v=1&r=http%3a%2f%2fwww.stretch.se%2fwp-content%2fuploads%2f2014%2f03%2fmicrosoft-logo-NYA.jpg&p=DevEx,5008.1",
      "hostPageUrl": "http://www.bing.com/cr?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=28C0E7D43623671417E9EDA937C66659&rd=1&h=_gC_9Zf6p0BB4P1CqByS7Vjss-hnVGyZSf1sXeShHU&v=1&r=http%3a%2f%2fwww.stretch.nu%2fpartners%2fmicrosoft-3%2f&p=DevEx,5007.1",
      "contentSize": "29128 B",
      "encodingFormat": "jpeg",
      "hostPageDisplayUrl": "www.stretch.nu/partners/microsoft-3",
      "width": 1242,
```

```

    "height": 643,
    "thumbnail": {
        "width": 300,
        "height": 155
    },
    "imageInsightsToken": "ccid_4gTvU2SE*mid_DA06857EE5270F0FA81C85FC6A432E3E1E7FD020*sim
id_608003027952795739",
    "insightsSourcesSummary": {
        "shoppingSourcesCount": 0,
        "recipeSourcesCount": 0
    },
    "imageId": "DA06857EE5270F0FA81C85FC6A432E3E1E7FD020",
    "accentColor": "CB9200"
},
{
    "name": "All Logos: Microsoft Logo",
    "webSearchUrl": "https://www.bing.com/cr?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=
28C0E7D43623671417E9EDA937C66659&rd=1&h=_1rpMWtxKVUr0MEAuXyC8ZRefIfVDWdPvwt8Jnrgw
a0&v=1&r=https%3a%2f%2fwww.bing.com%2fimages%2fsearch%3fview%3ddetailv2%26FORM%3d
0IIRPO%26q%3dMicrosoft%26id%3d0BD9C219266D3C08C71B244E90DB86EA1AE2675D%26simid%3d
608001382978096-683&p=DevEx,5012.1",
    "thumbnailUrl": "https://tse1.mm.bing.net/th?id=OIP.JdXWz-9R1LRCPBVmGjFtvwEsDh&pid=Api",
    "datePublished": "2014-06-02T20:55:00",
    "contentUrl": "http://www.bing.com/cr?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=28C
0E7D43623671417E9EDA937C66659&rd=1&h=d0mMvw0woksCSWQdMnQi6CqVOuHsVBej2UHj8JB4-pQ&
v=1&r=http%3a%2f%2f1.bp.blogspot.com%2f-go0yfq407Hg%2fURx8Dz6asmI%2fAAAAAAAUAw%2fT
yaM8lhH-DY%2fs1600%2fNew-Microsoft-Logo-PPT-Backgrounds.jpg&p=DevEx,5014.1",
    "hostPageUrl": "http://www.bing.com/cr?IG=2662ED1019ED4D41990A8AC61216DB7E&CID=28C0E7D43623671417
E9EDA937C66659&rd=1&h=zdBhSwmMJmtYPE6txoc6PwR2qeQsJBXvOooRxAbqvM&v=1&r=http%3a%2f
%2falllogos7.blogspot.com%2f2013%2f02%2fmicrosoft-logo.html&p=DevEx,5013.1",
    "contentSize": "78238 B",
    "encodingFormat": "jpeg",
    "hostPageDisplayUrl": "alllogos7.blogspot.com/2013/02/microsoft-logo.html",
    "width": 1600,
    "height": 1200,
    "thumbnail": {
        "width": 300,
        "height": 225
    },
    "imageInsightsToken": "ccid_JdXWz+9R*mid_0BD9C219266D3C08C71B244E90DB86EA1AE2675D*
simid_608001382978096683",
    "insightsSourcesSummary": {
        "shoppingSourcesCount": 0,
        "recipeSourcesCount": 0
    }
},

```

Давайте подробнее разберемся в некоторых ключевых пунктах документа JSON, чтобы их лучше понять. Результат JSON содержится в объекте с именем `Images`. Помимо изображения, он также содержит одно крайне важное поле под названием `totalEstimateMatches`, которое, как следует из его имени, показывает количество всех изображений, относящихся к выполненному запросу.

В отклике JSON также стоит обратить внимание на еще один объект. Это объект `image`, поскольку он содержит всю информацию о конкретном изображении. Давайте взглянем на некоторые важные свойства объекта `image`, который вы будете использовать в своем приложении (см. табл. 8.1).



Таблица 8.1. Поля объекта `image`

Имя	Описание
<code>Imageid</code>	Идентифицирует изображение
<code>imageInsightsToken</code>	Используйте этот токен, если вам нужна дополнительная информация об изображении, которая не предоставляется в отклике по умолчанию, таком как источник покупок. Вам нужно скопировать этот идентификатор и передать его в параметр запроса <code>insightstoken</code>
<code>contentSize</code>	Задает размер файла изображения. Вы можете отфильтровать отклик на основе размера изображения
<code>insightsSourcesSummary</code>	Показывает количество источников, где можно выполнять действия, такие как подсчет покупок. Обычно используется в приложении B2C. Например, если вы получили изображение сумки, то этот параметр покажет вам список сайтов, где предлагаются аналогичные сумки
<code>thumbnailUrl</code>	URL-адрес миниатюры изображения
<code>Thumbnail</code>	Ширина и высота изображения миниатюры
<code>Height</code>	Высота исходного изображения в пикселях
<code>Width</code>	Ширина исходного изображения в пикселях
<code>encodingFormat</code>	Тип изображения, такой как <code>png</code>
<code>contentUrl</code>	URL-адрес изображения на исходном сайте
<code>datePublished</code>	Дата и время, когда Bing обнаружил изображение
<code>hostPageUrl</code>	Содержит URL-адрес страницы, на которой размещено изображение

Очень важно хорошо разбираться в документе JSON, полученном в качестве отклика, чтобы обеспечить фильтрацию результатов, перед тем как отобразить их в клиентской части приложения. После получения отклика можно поэкспериментировать с полями документа. API Bing поиска изображений также предоставляет фильтрующий параметр запроса, который позволяет отфильтровывать и получить отклик только для тех изображений, которые удовлетворяют критериям. Некоторые параметры, такие как `cc`, `mkt`, `q`, `setlang` и `id`, которые были рассмотрены в разделе про API автозаполнения, применимы к API поиска изображений. Мы не будем на них останавливаться, т. к. они были описаны ранее, но для вас важно понимать, что эти параметры могут применяться и с API поиска изображений. Давайте разберемся в нескольких новых параметрах для проактивной фильтрации (см. табл. 8.2).

Таблица 8.2. Фильтрующие параметры запроса

Имя	Описание
<code>Count</code>	Количество изображений, возвращенных в отклике
<code>moduleRequested</code>	Показывает дополнительную информацию по запросам изображений, которые доступны по умолчанию. Например, можно добавить модуль <code>shoppingSources</code> , чтобы получить список всех торговцев, которые предлагают товар, изображенный на фото. Информация <code>moduleRequested</code> должна быть передана в виде разделенных запятой значений
<code>Offset</code>	Задает количество изображений, которые нужно пропустить в возвращаемом результате. Это лучше всего используется в соединении со счетчиком для постраничного показа

Окончание табл. 8.2

Имя	Описание
SafeSearch	Это имя используется для фильтрации изображений с содержимым для взрослых. Поддерживаются следующие варианты: Off (фильтрация выключена), Strict (строгая) и Moderate (умеренная). При строгой фильтрации никаких изображений для взрослых не возвращается. При выключенном фильтрации возвращаются изображения с содержимым для взрослых. По умолчанию используется умеренная фильтрация, при которой содержимое для взрослых не возвращается; содержимое для взрослых будет возвращаться в API поиска в виде расплывчатых миниатюр
Aspect	Используя этот фильтр, вы можете фильтровать изображения по соотношению сторон. Предусмотрены следующие варианты: Square (квадратный), Tall (по высоте), Wide (по ширине) и All. По умолчанию используется All, что означает, что соотношение сторон не указано
Color	Этот вариант используется для фильтрации изображения по цвету. Поддерживается двенадцать основных цветов плюс монохромный, который возвращает черно-белые изображения
Freshness	Этот вариант используется, если требуется фильтрация изображений по времени обнаружения изображений в Bing. Предусмотрены варианты Day (день), Week (неделя) и Month (месяц)
Height	Изображения фильтруются по высоте
Width	Изображения фильтруются по ширине
ImageType	Изображения фильтруются по типу изображения. Предусмотрены варианты AnimatedGIF, Clipart, Line (штриховое), Photo и Shopping
License	Используется для фильтрации изображения по применяемой лицензии. Этот вариант самый востребованный; мы его использовали для получения нелицензированных изображений, на которые можно сослаться в книгах и публикациях
Size	Этот вариант используется для фильтрации изображений по размеру. Предусмотрены варианты Small (малый), Medium (средний), Large (крупный), Wallpaper (обои) и All. Размеры основаны на пикселях. Например, вариант Small возвращает изображения размером менее 200×200 пикселов

API Bing поиска изображений также предоставляет возможность получать детальную информацию по определенному изображению. В этом случае необходимо передать POST-запрос на <https://api.cognitive.microsoft.com/Bing/v5.0/images/search>, передав изображение в теле POST-запроса. Этот запрос используется, когда требуется подробная информация относительно автоматически генерируемых заголовков для получения изображений аналогичного типа и т. д.

Помимо поиска изображений на основе заданного фильтра и подробной информации о них, Bing также может отыскивать изображения, популярность которых определяется поисковыми запросами, выполненными другими пользователями. Чтобы получить такие изображения, вам нужно вызвать API GET-методом по <https://api.cognitive.microsoft.com/Bing/v5.0/images/trending>, как показано ниже:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/images/trending HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apm-Subscription-Key: .....
```

После вызова API вы получите приведенный ниже документ JSON:

```
{
  "_type": "TrendingImages",
  "instrumentation": {
    "pageLoadPingUrl": "https://www.bingapis.com/api/ping/pageload?IG=524A0BEF27844894B8CDA1
900FA3A6C1&CID=39A93253FBFC66811D0F382EFA1967FC&Type=Event.CPT&DATA=0"
```

```

},
"categories": [
{
  "title": "Popular people searches",
  "tiles": [
    {
      "query": {
        "text": "Nicki Minaj",
        "displayText": "Nicki Minaj",
        "webSearchUrl": "https://www.bing.com/cr?IG=524A0BEF27844894B8CDA1900FA3A6C1&CID=39A93253FBC66811D0F382EFA1967FC&rd=1&h=hgzU1-YGXX61W8XxJcBbdmSRH_XABJ028PovVG6i118&v=1&r=https%3a%2f%2fwww.bing.com%2fimages%2fsearch%3fq%3dNicki%2bMinaj%26FORM%3dISTRTH%26id%3dE9B866C2A8AE6043DAE307D0490386AA278BB419%26cat%3dPopular%2520people%2520searches%26lpversion%3d&p=DevEx,5000.1"
      },
      "image": {
        "thumbnailUrl": "https://tse3.mm.bing.net/th?id=OET.796b8441539c4ad0af4875cf911eb824&pid=Api",
        "contentUrl": "http://ime.ulximg.com/image/300x300/artist/1330095247_6dc51fbfeddedd01d0e78650d286e2f0.png/ce576f26cc31f554fa906b8ba115802e/1330095247_nickiminaj2.png",
        "thumbnail": {
          "width": 300,
          "height": 300
        },
        "imageId": "E9B866C2A8AE6043DAE307D0490386AA278BB419"
      }
    }
  ],
}
],

```



Одна из любопытных особенностей приведенного выше фрагмента отклика JSON состоит в том, что поисковый результат был разделен на различные категории, в частности поиск изображений популярных людей, популярных животных, популярных видов природы и т. д.

Интересно отметить, что Bing не только делает поиск по заданному тексту, но и предлагает разумные варианты, которые сужают первоначальный поисковый запрос. Например, если вы ищете ключевое слово «Microsoft search», то Bing разумно его разбьет на различные варианты, такие как «Microsoft Desktop search», «Microsoft Windows Search» и «Microsoft search 4.0». У вас есть возможность выбрать одно из расширений поискового запроса и получить результаты поиска. Как показано на рис. 8.8, точно такой же опыт взаимодействия можно также получить в Bing.com/images.

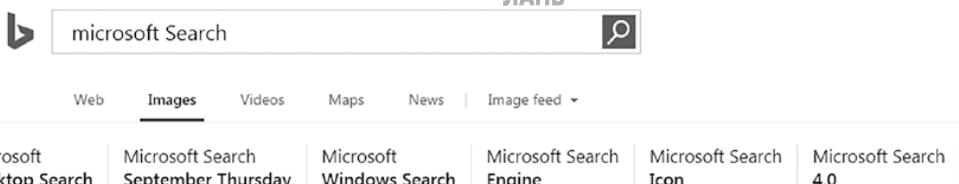


Рис. 8.8 ❖ Расширение поискового запроса Microsoft search

Bing таким же образом стремится разбить исходный запрос на различные сегменты, именуемые осями (pivots), чтобы показать, какие запросы делают другие пользователи. Это особенно полезно при предоставлении пользователю поисковых предложений. В целом API Bing поиска предоставляет обширный набор свойств и параметров запроса для персональной настройки поиска изображений.

В API Bing поиска изображения предусмотрены бесплатная и платная ценовые категории. Ценовая категория с бесплатной подпиской позволяет делать 1000 вызовов в месяц, и срок действия бесплатной подписки составляет три месяца. С точки зрения платных подписок в API Bing поиска изображений предусмотрено шесть стандартных ценовых категорий от S1 до S6. В зависимости от объема можно выбрать категорию S1 для 1000 поисковых запросов в месяц до категории S6, которая позволяет делать 10M поисковых запросов в месяц. За исключением категории S1, остальные стандартные категории предусматривают дополнительные расходы за превышение верхней границы.

API BING ПОИСКА НОВОСТЕЙ

Bing также выступает как агрегатор новостей, собирающий, консолидирующий и классифицирующий новости из тысяч газет и новостных статей во всем мире. Эту возможность предоставляет [Bing.com/news](https://bing.com/news). API Bing поиска новостей позволяет вам создать в своем приложении агрегатор новостей. На верхнем уровне API Bing поиска новостей дает возможность получать статьи/заголовки главных новостей на основе заданной категории, возвращает новостные статьи на основе поискового запроса пользователя или позволяет возвращать темы новостей, которые набирают популярность в социальной сети. Давайте разберем API Bing поиска новостей подробнее. Как в случае двух других рассмотренных ранее API, вам нужно вызвать <https://api.cognitive.microsoft.com/Bing/v5.0/news/> посредством GET-запроса, передав ключ подписки в заголовке Ocp-Apim-Subscription-key, как показано ниже:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/news/search HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....
```

В отличие от API автозаполнения и поиска изображений, API Bing поиска новостей можно вызывать, передавая любой параметр, и он вернет главные новостные статьи в каждой категории. При вызове API поиска новостей без какого-либо параметра вы увидите подмножество документа JSON. По умолчанию отклик будет включать до 12 заголовков статей и до 4 статей в каждой категории новостей. У вас всегда есть возможность применить параметр запроса headlineCount, чтобы изменить количество возвращаемых заголовков статей.

```
{
  "_type": "News",
  "readLink": "https://api.cognitive.microsoft.com/api/v5/news/search?q=",
  "value": [
    {
      "name": "Kapil Mishra to lodge FIR , hand over 'evidences' against Kejriwal to CBI",
      "url": "http://www.bing.com/cr?IG=67AEC659B7A646BC95C49DF8347532E5&CID=3C649CB66ABA67
FE16BF96CB6B5F6610&rd=1&h=Q_PKXZziVhM9IhE50ocONWr3e20WBeB1RW0SuuFxxg&v=1&r=http%3a%
```

```

2%2ftimesofindia.indiatimes.com%2findia%2fin-letter-to-guru-kapil-mishra-says-will
file-an-fir-against-kejriwal%2farticleshow%2f58587624.cms&p=DevEx,5006.1",
"image": {
    "thumbnail": {
        "contentUrl": "https://www.bing.com/th?id=ON.4B95481C9E5C3BB13058E8D39DB3BFCA&pid=
        News",
        "width": 700,
        "height": 525
    }
},
"description": "NEW DELHI: The war within AAP intensified on Tuesday with sacked Delhi
minister Kapil Mishra meeting CBI officials to file FIR against Arvind Kejriwal. Kapil
Mishra will also handed over three packets® of \"evidences\" against Kejriwal to the
CBI officials.",
"about": [
{
    "readLink": "https://api.cognitive.microsoft.com/api/v5/entities/03ee339a-5811-c624-
    d751-de0c2b8aaa9f",
    "name": "Kapil Mishra"
},
{
    "readLink": "https://api.cognitive.microsoft.com/api/v5/entities/54ac3847-14fa-4877-
    bbea-8428099abf2d",
    "name": "School of International Relations, University of Economics in Prague"
},
{
    "readLink": "https://api.cognitive.microsoft.com/api/v5/entities/a00227e3-ffd7-b1d1-
    b5e1-0d69bafbfa37",
    "name": "India News"
}
],
"provider": [
{
    "_type": "Organization",
    "name": "Times of India"
}
],
"datePublished": "2017-05-09T07:31:00",
"category": "Politics",
"headline": true
},

```

Давайте сначала разберемся в отклике JSON. Таблица 8.3 показывает некоторые поля, которые вы будете использовать в новостной статье.

Таблица 8.3. Поля объекта статьи Article

Имя	Описание
Name	Название статьи
Headline	Булево значение, сообщающее, является новость заголовком (headline) или нет
Category	Задает категорию, которой принадлежит новостное сообщение
Id	Однозначно идентифицирует новостную статью
Image	Содержит все изображения из этой новостной статьи в виде полей с миниатюрами

Окончание табл. 8.3

Имя	Описание
Description	Содержит краткое описание новостной статьи
Provider	Список организаций, которые выпустили статью
Date published	Дата и время обнаружения статьи в Bing
clusteredArticle	Список новостных статей аналогичного типа
URL	URL-адрес статьи

По умолчанию новости поступают из универсального рынка и могут быть недостоверными. Чтобы получить отклик с соответствующими и действующими новостями, мы рекомендуем использовать параметр `mkt`, чтобы указать рынок. Приведенный ниже URL-адрес вернет сообщения о главных новостях из Индии:

<https://api.cognitive.microsoft.com/Bing/v5.0/news/?mkt=en-in>

Одним из широко используемых параметров является параметр `category`. Bing предоставляет список категорий на основе каждого рынка. Почти все новостные сообщения относятся к той или иной категории. Приведенный ниже URL-адрес вернет новости, связанные со спортом для Индии:

<https://api.cognitive.microsoft.com/Bing/v5.0/news/?mkt=en-in&category=sports>

Не все категории привязаны к каждому рынку, поэтому важно знать список категорий, принятых на каждом рынке. Например, если вы искали под медицинской категорией вместе с Индией, то вы получите ошибку 400 «Неверный запрос» с показанным ниже откликом JSON:

```
{
  "_type": "ErrorResponse",
  "errors": [
    {
      "code": "RequestParameterInvalidValue",
      "message": "Parameter has invalid value. The category parameter is invalid",
      "parameter": "category",
      "value": "health"
    }
  ]
}
```

Чтобы производить поиск новостей на основе некоторого заданного ключевого слова, используйте параметр `q`. Приведенный ниже URL-адрес будет искать новости, связанные с Microsoft:

<https://api.cognitive.microsoft.com/Bing/v5.0/news/search?q=microsoft>

Обратите внимание на изменение в URL-адресе; теперь помимо параметра `q` URL-адрес дополнен словом `search/`. Если перейти на домашнюю страницу Bing.com, то можно увидеть, что в баннере внизу страницы появится список новостей, как показано на рис. 8.9.

Список тех же самых набирающих популярность новостей можно получить путем вызова приведенного ниже URL-адреса:

<https://api.cognitive.microsoft.com/Bing/v5.0/news/trendingtopics>

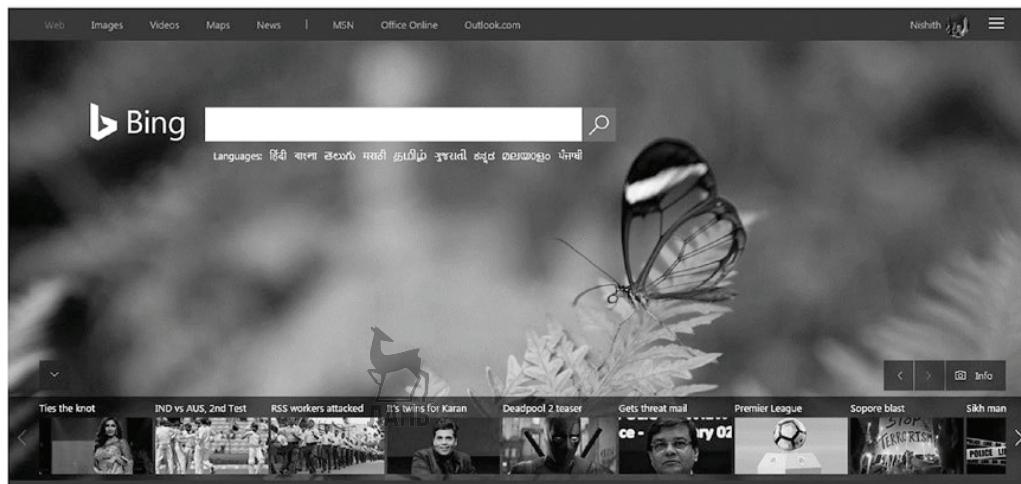


Рис. 8.9 ♦ Набирающие популярность новости в баннере домашней страницы Bing

Как можно увидеть, API Bing поиска новостей предоставляет пользователям опыт взаимодействия с новостной информацией с эффектом присутствия и представляет собой самый легкий способ создания опыта взаимодействия с новостным порталом, используя для этого всего несколько строк кода. В API Bing поиска новостей имеются бесплатные и платные ценные категории подписки, аналогичные API поиска изображений. Обратитесь к последнему разделу по API поиска изображений, чтобы разобраться в указанных ценных категориях. Теперь давайте зайдемся поиском видео.

API BING ПОИСКА ВИДЕО

Поиск видео в Bing является одной из самых захватывающих особенностей Bing. API Bing поиска видео и Bing.com/videos предоставляют много интересных возможностей, которые делают поиск видео в Bing одним из любимых занятий. Такие возможности, как предварительные просмотры видео и воспроизведение видео, не покидая сайт Bing, находят широкую популярность среди пользователей. API Bing поиска видео предоставляет такие же возможности и в Bing.com/videos. Рисунок 8.10 показывает некоторые из возможностей, доступных в Bing.com/videos.

На верхнем уровне API Bing поиска видео позволяет вам получать видео на основе разнообразных фильтров, таких как ключевые слова, разрешение экрана и т. д., и в результате получать больше информации о конкретном видео или выставочных видео, набирающих популярность в социальной сети. Теперь давайте подробнее разберемся в API Bing поиска видео.

КАК ИСПОЛЬЗОВАТЬ API BING ПОИСКА ВИДЕО

Как и в случае с большинством других API Bing, вам нужно вызвать <https://api.cognitive.microsoft.com/Bing/v5.0/videos/search> посредством GET-запроса, передав

ключ подписки в заголовке Ocp-Apim-Subscription-key и искомое видео в параметре q, как показано ниже:

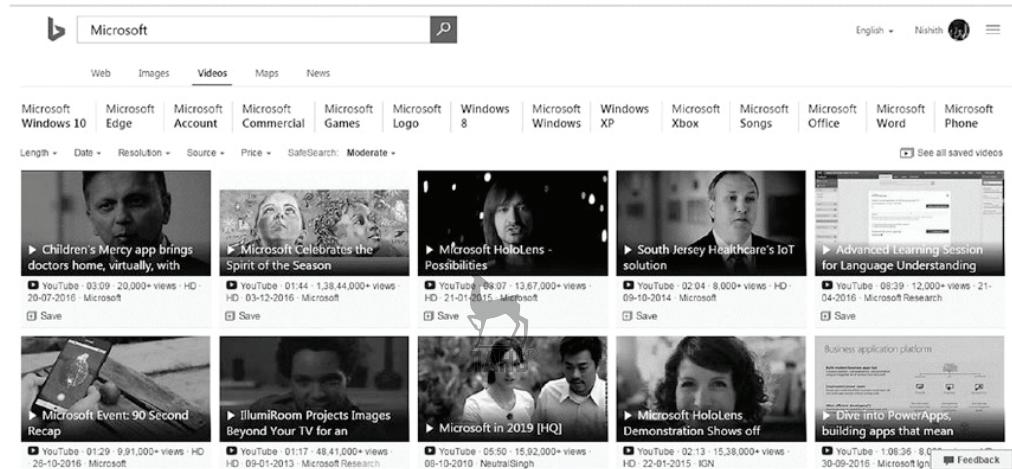


Рис. 8.10 ♦ Снимок Bing.com/videos

GET <https://api.cognitive.microsoft.com/bing/v5.0/videos/search?q=Microsoft> HTTP/1.1

Host: api.cognitive.microsoft.com

Ocp-Apim-Subscription-Key:
.....

У вас также есть возможности передать дополнительные параметры запроса, чтобы уточнить ваш запрос для поиска видео. Приведенный ниже фрагмент кода уточняет поисковый запрос видео, передавая дополнительные параметры запроса:

GET <https://api.cognitive.microsoft.com/bing/v5.0/videos/search?q=Microsoft&count=50&offset=0&mkt=en-us&safeSearch=Moderate> HTTP/1.1

Host: api.cognitive.microsoft.com

Ocp-Apim-Subscription-Key:
.....



Ниже приведено подмножество документа JSON, возвращенного при вызове API поиска видео:

```
{
  "_type": "Videos",
  "instrumentation": {
    "pageLoadPingUrl": "https://www.bingapis.com/api/ping/pageLoad?IG=80D340D10D4A4986ACFCAE
D8781E1011&CID=079C1AB0315763FF32D110CD30B262B8&Type=Event.CPT&DATA=0"
  },
  "readLink": "https://api.cognitive.microsoft.com/api/v5/videos/search?q=Microsoft",
  "webSearchUrl": "https://www.bing.com/cr?IG=80D340D10D4A4986ACFCAED8781E1011&CID=079C1AB03
15763FF32D110CD30B262B8&rd=1&h=efHRLYPwMuf0NAYRYlm_kL65gvQMXF9zel8xU8F9KUw&v=1&r=https%3a%
2f%2fwww.bing.com%2fvideos%2fsearch%3fq%3dMicrosoft&p=DevEx,5387.1",
  "totalEstimatedMatches": 1000,
  "value": [
    {
      "name": "Çekinmeyin, web'e dokunun",
    }
  ]
}
```

```

"description": "Βρείτε απαντήσεις στις απορίες σας στο φόρουμ του Internet Explorer.",
"webSearchUrl": "https://www.bing.com/cr?IG=80D340D10D4A4986ACFCAED8781E1011&CID=079C1AB0315763FF32D110CD30B262B8&rd=1&h=qhXQDtX7quHivkKPnI5mprr_VB03vcOBKeFLP9oPMkE&v=1&r=h
https%3a%2f%2fwww.bing.com%2fvideos%2fsearch%3fq%3dMicrosoft%26view%3ddetail%26mid%3d32
CCEE6E00380AACAB9E32CCEE6E00380AACAB9E&p=DevEx,5388.1",
"thumbnailUrl": "https://tse2.mm.bing.net/th?id=OVP.bs8aHGNwE0DVJFtjIsJEWQDIDI&pid=Api",
"publisher": [
    {
        "name": "Microsoft"
    }
],
"contentUrl": "https://support.microsoft.com/en-us/products/internet-explorer",
"hostPageUrl": "https://www.bing.com/cr?IG=80D340D10D4A4986ACFCAED8781E1011&CID=
079C1AB0315763FF32D110CD30B262B8&rd=1&h=gzFge4ilBwp0cF9Uun0frqf-oD-ZRBhUexJflryd
PT8&v=1&r=https%3a%2f%2fsupport.microsoft.com%2fen-us%2fproducts%2finternetexplo
rer&p=DevEx,5011.1",
"encodingFormat": "",
"hostPageDisplayUrl": "https://support.microsoft.com/en-us/products/internet-explorer",
"width": 200,
"height": 200,
"allowHttpsEmbed": false,
"thumbnail": {
    "width": 200,
    "height": 200
},
"videoId": "32CCEE6E00380AACAB9E32CCEE6E00380AACAB9E",
"allowMobileEmbed": false,
"isSuperfresh": false
},

```

Давайте сначала разберемся в отклике JSON. Наверху находится объект Videos, который содержит все видео в форме массива объектов Video. Таблица 8.4 показывает некоторые поля, которые можно использовать в объекте Video.

Таблица 8.4. Поля объекта видео

Имя	Описание
Name	Название видео
Description	Краткое описание видео
thumbnailUrl	URL-адрес миниатюрного изображения видео
Publisher	Имя издателя видео
videoId	Прямо идентифицирует видео
contentURL	URL-адрес видео на сайте, где размещено видео
encodingFormat	Задает формат кодирования (например, h264)
Width	Ширина видео
Height	Высота видео
allowHTTPsEmbed	Булево значение, которое определяет, может ли видео быть внедрено в страницу HTML или нет
allowMobileEmbed	Булево значение, которое определяет, может ли видео быть размещено на мобильных устройствах или нет
isSuperFresh	Булево значение, которое определяет, было ли видео обнаружено в Bing недавно или нет
viewCount	Количество просмотров видео, согласно подсчету по источникам

Эта таблица показывает несколько основных полей объекта Video. Помимо них, есть и другие свойства, которые могут показаться немного запутанными, если не разъяснить их назначение. Одним таким полем является идентификатор объекта Video. Каждый объект Video имеет два поля, videoID и ID. Оба поля определено идентифицируют видео. В идеальном плане videoID – это поле, которое четко идентифицирует видео. У вас также есть возможность назначить значение поля videoID полю ID, чтобы поэкспериментировать с видео.

По умолчанию API Bing поиска видео возвращает все виды видео, которые относятся к вашему запросу. Некоторые параметры, которые вы использовали ранее, такие как mkt, count, freshness, offset, safesearch и setlang, также могут применяться и к поиску видео. Ниже перечислены новые дополнительные параметры запроса, которые могут использоваться для уточнения результатов поиска.

- Используйте параметр запроса pricing, чтобы отфильтровать видео по ценовой информации. Предусмотрены варианты Free (бесплатное), Paid (платное) и All (все). По умолчанию используется All.
- У вас также есть возможность получать видео по разрешению экрана путем задания параметра запроса resolution. Предлагаемые варианты – 480p, 720p, 1080p и All. Любые выбранные варианты будут показывать изображение с более высоким разрешением, чем выбранный вариант. Например, если выбран вариант с 720p, то это означает, что будет возвращено видео только с 720p или выше. По умолчанию результаты по разрешению экрана не фильтруются и имеют вариант All.
- Можно возвращать видео по длине видео путем установки параметра videoLength. Предусмотрены варианты Short (короткое), Medium (среднее), Long (длинное) и All (все). Short означает видео продолжительностью менее 5 минут. Medium означает видео продолжительностью от 5 и до 20 минут. Long вернет видео продолжительностью больше 20 минут. Вариант All задан по умолчанию и не имеет ограничения по продолжительности.

В приведенном выше материале, посвященном дополнительным параметрам, вы увидели, что параметр videoId четко идентифицирует каждое видео. Этот параметр далее может использоваться для получения дополнительной информации в отношении конкретного видео, за пределами того, что было доступно при обычном поиске видео в Bing. Для того чтобы получить дополнительную информацию, вам нужно выполнить еще один вызов API поиска видео и назначить параметру videoId значение идентификатора и параметру modulesRequested присвоить значение All, как показано ниже:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/videos/details?id=42656D88F511F16E9CFD42656D  
8BF511F16E9CFD&modulesRequested=All HTTP/1.1  
Host: api.cognitive.microsoft.com  
Ocp-Apim-Subscription-Key: .....  
.....
```

Если вы посмотрите на этот фрагмент кода, то увидите, что вызывается другой URL-адрес API Bing <https://api.cognitive.microsoft.com/Bing/v5.0/videos/details>. Параметры запроса videoID и modulesRequested являются обязательными для получения дополнительной информации. Параметр modulesRequested является специфическим параметром, который поддерживает список значений, разделенных запятыми с одним или несколькими запрашиваемыми элементами. Значение All, как

показано выше, доставит всю информацию. У вас также есть возможность задать варианты `RelatedVideos` и `VideoResult`. `RelatedVideos` доставить все видео, связанные с видео, заданным в идентификационном параметре. Параметр `VideoResult` возвращает видео, о котором вы запрашиваете дополнительную информацию. Иногда могут возникать ситуации, когда вы запрашиваете дополнительную информацию и в Bing ее нет; если это так, то объект отклика этого поля содержать не будет.

Так же как и в случае с набирающими популярность изображениями, вы можете получить список наиболее популярных видео, обратившись по приведенному ниже URL-адресу:

<https://api.cognitive.microsoft.com/Bing/v5.0/videos/trending>

Как можно видеть, API Bing поиска видео обеспечивает возможности поиска видео с эффектом присутствия и представляет собой самый легкий способ создать опыт взаимодействия с видеопорталом, воспользовавшись для этого всего несколькими вызовами API. В API Bing поиска видео есть бесплатные и платные ценовые категории, аналогично API поиска изображений. Обратитесь к предыдущему разделу по API поиска изображений, чтобы разобраться в бесплатных и платных предложениях.

API BING ПОИСКА В СЕТИ

Вы познакомились с разнообразными API Bing. Каждый из этих API специализируется на предоставлении определенной информации. Например, API Bing поиска изображений применяется, если требуется осуществить поиск изображений. Точно так же API поиска новостей применяется для поиска новостей. Кроме того, имеется довольно много сценариев, в которых потребуется объединение результатов запросов, чтобы получить изображения и видео вместе в одном наборе поисковых результатов. Это похоже на поиск в Bing.com/search, где вы получаете результаты, включающие изображения, видео и другие типы информации. API Bing поиска в сети предоставляет решение для всех подобных сценариев.

Как использовать API Bing поиска в сети

Как и большинство других API Bing, вам нужно с помощью GET-запроса вызвать <https://api.cognitive.microsoft.com/Bing/v5.0/search>, передав ключ подписки в заголовке `Ocp-Apim-Subscription-key` и поисковый запрос в параметре `q`:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/search?q=Cognitive service HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....
```

Ниже показано подмножество возвращенного документа JSON, который получаем при вызове API Bing поиска в сети:

```
{
  "_type": "SearchResponse",
  "webPages": {
    "webSearchUrl": "https://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=2711158
76DD9657922A81FFA6C3C645D&rd=1&h=QVjaswr8AICW0SzfYCFLtx3gSvGUR5zF6Uxs9z8U30M&v=1&r=https
```

%3a%2f%2fwww.bing.com%2fsearch%3fq%3dCognitive&p=DevEx,5363.1",
"totalEstimatedMatches": 14000000,
"value": [
 {
 "id": "https://api.cognitive.microsoft.com/api/v5/#WebPages.0",
 "name": "Cognition - Wikipedia",
 "url": "https://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=271115876DD9
657922A81FFA6C3C645D&rd=1&h=-FncTcyJrdq3ayy4aX4tTVV70wIX1zUpPtkDMfxBcCw&v=1&r=https%
3a%2f%2fen.wikipedia.org%2fwiki%2fCognition&p=DevEx,5077.1",
 "about": [
 {
 "name": "Cognition"
 }
],
 "displayUrl": "Wikipedia > wiki > Cognition",
 "snippet": "The sort of mental processes described as cognitive are largely influenced by research which has successfully used this paradigm in the past, likely starting with Thomas Aquinas, who divided the study of behavior into two broad categories: cognitive (how we know the world), and affective (how we understand the world via feelings and emotions) [disputed - discuss]",
 "snippetAttribution": {
 "license": {
 "name": "CC-BY-SA",
 "url": "http://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=271115876DD9
657922A81FFA6C3C645D&rd=1&h=tJYnHtBaEymlKo_fRl0wo_Zhu4dAaCwVMH_Bvcix0gM&v=1&r=http%
3a%2f%2fcreativecommons.org%2flICENSES%2fbY-sa%2f3.0%2f&p=DevEx,5362.1"
 },
 "licenseNotice": "Text under CC-BY-SA license"
 },
 "deepLinks": [
 {
 "name": "Load a Random Article",
 "url": "https://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=271115876DD9
657922A81FFA6C3C645D&rd=1&h=q1Bn_0G8fm7uQkimqCtaj19x7I491UAUQPmzrYtjo&v=1&r=
https%3a%2f%2fen.wikipedia.org%2fwiki%2fSpecial%3aRandom&p=DevEx,5067.1"
 },
 {
 "name": "Contact Page",
 "url": "https://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=271115876DD9
657922A81FFA6C3C645D&rd=1&h=6WyXHuw6VvpEvlqirdvJB9bZ8g86C42gTpWTmaeXGM8&v=1&r=
https%3a%2f%2fen.wikipedia.org%2fwiki%2fWikipedia%3aContact_us&p=DevEx,5068.1"
 },
 {
 "name": "Citation Needed",
 "url": "https://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=271115876DD9
657922A81FFA6C3C645D&rd=1&h=0HnjXTBDeLsXjIzd1Ve70gmwak7KfLaJJFmaniZcPB0&v=1&r=https%
3a%2f%2fen.wikipedia.org%2fwiki%2fWikipedia%3aCitation_needed&p=DevEx,5069.1"
 },
 {
 "_type": "SearchResponse",
 "webPages": {
 "webSearchUrl": "https://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=271115876DD9
657922A81FFA6C3C645D&rd=1&h=QVjaswr8AICh0SzfYCfLtx3gSvGUR5zF6Uxs9z8U30M&v=1&r=https%3a%2f%

```
%2fwww.bing.com%2fsearch%3fq%3dCognitive&p=DevEx,5363.1",
"totalEstimatedMatches": 14000000,
"value": [
{
  "id": "https://api.cognitive.microsoft.com/api/v5/#WebPages.0",
  "name": "Cognition - Wikipedia",
  "url": "https://www.bing.com/cr?IG=73EEE703FDE24263A07FA788B5251242&CID=271115876DD9657922A81FFA6C3C645D&rd=1&h=-FnCTcyJrdq3ayy4aX4tTYV70wIX1zUpPtkDMfxBcCw&v=1&r=https%3a%2f%2fen.wikipedia.org%2fwiki%2fCognition&p=DevEx,5077.1",
  "about": [
    {
      "name": "Cognition"
    }
  ],
}
```

Как видно, API Bing поиска в сети не только попытается извлечь полностью отвечающие требованию поискового запроса (релевантные) результаты поиска для всех API Bing, таких как изображения, новости и видео, имеющих в конечном счете отдельные конечные точки, но и извлечет поисковые результаты и предложит варианты, не имеющие отдельной конечной точки. То есть API Bing поиска в сети вызывается в случае, когда требуется извлечь многочисленные наборы информации. В противном случае следует использовать отдельные API. У вас всегда есть возможность отфильтровать и получить подмножество API путем передачи их в виде разделенных запятыми значений в параметре запроса `responseFilter`, как показано ниже:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/search?q=cognitive service&responseFilter=co
mputation,images HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....
```

Одна из любопытных особенностей API Bing поиска в сети состоит в том, что в отклике он предлагает ранжирование результатов. Ранжирование доступно в параметре `rankingResponse`. В идеальном случае следует выводить результаты поиска на основе данного ранжирования. Такой поиск предоставит вам опыт взаимодействия с любой страницей результатов поиска. Данный параметр имеет три варианта значений, в т. ч. `mainline` (магистральная часть страницы), `pole` (верхушка страницы) и `sidebar` (боковая панель). Значения `mainline` и `sidebar`, как следует из названий, показывают результаты соответственно в теле и в боковой панели страницы. Значение `pole` показывает результат поиска вверху магистральной части и боковой панели страницы. В случае других сценариев, если вы не хотите следовать ранжированию, убедитесь, что вы придали магистральной части содержимого большую видимость, чем боковой панели.

Поскольку речь идет о веб-поиске, пользователь может задавать любой тип вопросов, начиная с математических выражений и заканчивая запросами, связанными с часовым поясом. API Bing поиска в сети обеспечивает объекты для обработки таких откликов. Как показано ниже, для любых запросов, связанных с вычислениями и преобразованиями из одной системы мер в другую, отклик будет включать объект `computation`:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/search?q=How many meters in a mile HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
```

Пользователь спросил, сколько метров в милю. Если вы посмотрите на отклик JSON, то увидите, что он содержит один объект computation:

```
"computation": {
    "id": "https://api.cognitive.microsoft.com/api/v5/#Computation",
    "expression": "1 mile",
    "value": "1609.344 meters"
},
"rankingResponse": {
    "mainline": {
        "items": [
            {
                "
```



Отклик показывает два элемента в объекте: выражение expression и значение value. Элемент expression содержит выражение сделанного запроса. Если запрос касается преобразования, как в нашем случае, то он будет показывать единицы, из которых выполняется преобразование в выражении, и значение будет содержать фактический ответ или единицы, в которые выполняется преобразование. Как показано в приведенном выше примере, value показывает фактически преобразованное значение. Кроме того, вы можете задавать вопросы, связанные с часовыми поясами. Для времени в часовом поясе EST фрагмент кода будет следующим:

```
GET https://api.cognitive.microsoft.com/bing/v5.0/search?q=time in EST zone HTTP/1.1
Host: api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
```

Когда делаются запросы наподобие приведенного выше, в отклике JSON наряду с фактическим временем и датой будет находиться объект timeZone, который будет содержать поле ID и имя главного города:

```
"timeZone": {
    "id": "https://api.cognitive.microsoft.com/api/v5/#TimeZone",
    "primaryCityTime": {
        "location": "Eastern Time Zone",
        "time": "2017-05-09T04:09:00.989604Z",
        "utcOffset": "UTC-4"
}
```

Bing также предлагает объекты RelatedSearchAnswer, QueryContext и SpellSuggestions в зависимости от выполняемого запроса. Объект RelatedSearchAnswer показывает самые популярные похожие запросы, сделанные другими пользователями. Объект QueryContext используется, когда выполняемый конкретный запрос содержит орфографические ошибки. Bing[®] автоматически исправляет ошибки и в объекте QueryContext показывает не только исходный запрос, но и исправленный запрос. Вы также можете получить объект SpellSuggestions, если API Bing найдет варианты исправления орфографических ошибок.



Резюме

Подведем итоги. В этой главе вы познакомились с поисковым механизмом Bing и связанными с ним API. Вы также научились вызывать каждый API и выполнять персонализированную настройку и фильтрацию результатов поиска на основе параметров запроса и заголовков. Вы так же получили глубокое понимание возвращаемых из этих API документов JSON, которые помогают вам выполнять настройку результатов согласно вашим требованиям. В **следующей главе** вы узнаете о внутреннем устройстве API рекомендаций когнитивных служб Microsoft.



Глава 9

Работа с рекомендациями



Машинное обучение и его применение сегодня можно найти повсюду. Многие из нас используют машинное обучение в нашей повседневной жизни намного больше, чем мы сами об этом догадываемся. Одно из классических применений и грандиозных предназначений машинного обучения – это рекомендательная система. Мы встречаемся с рекомендациями в Интернете повсюду. И эти рекомендательные системы широко внедрены в некоторые лидирующие глобальные веб-сайты. Давайте сначала рассмотрим несколько примеров, где вы могли бы использовать рекомендательные системы:

- YouTube использует рекомендательную систему, чтобы рекомендовать видео;
- Netflix использует рекомендательную систему, чтобы рекомендовать интересные видео/истории;
- Amazon не только использует рекомендации для показа результатов поиска товаров, но и для рекомендации товаров, продаваемых своим конечным пользователям;
- Google не только использует рекомендательные системы, чтобы ранжировать веб-ссылки, но и предлагает веб-ссылки и новостные ссылки своим пользователям;
- новостные сайты, такие как Нью-Йорк таймс, предоставляют рекомендации по поводу новостей, которые необходимо посмотреть/прочитать;
- Facebook предоставляет рекомендации не только для обновления новостных лент, но и рекомендует людей, с которыми вы могли бы познакомиться;
- Твиттер показывает вам рекомендуемые результаты и предложения, чьи твиты читать.

Все сайты, связанные с вовлечением пользователей, имеют рекомендательный механизм. Как отмечено выше, рекомендательные системы широко используются в различных прикладных областях. Сила рекомендаций придала мощный импульс некоторым успешным компаниям. По сути дела, классическим примером применения рекомендаций является компания Netflix, одна из самых популярных компаний индустрии развлечений. Согласно недавним данным, порядка 80% ее потокового контента просматривается благодаря рекомендациям и персонализации. Остальной контент (20%) получен в результате поиска. Эти данные говорят об эффективности рекомендательной системы, потому что рекомендации

не только помогли конечным пользователям выбрать поток, но и помогли компании сэкономить большую сумму денег. Рекомендательный механизм Netflix является ядром бизнеса этой компании. Если вы следили за деятельностью Netflix в течение последних 10 лет, то вы, возможно, вспомните про конкурс, в котором был призовой фонд в миллион долларов. Тогда компания предложила командам разработчиков усовершенствовать рекомендательный механизм под названием Cinematch на 10%. Именно сила рекомендательного механизма позволила Netflix увеличить свою клиентскую базу с 6 миллионов американских пользователей в 2006 г. до 83 миллионов пользователей более чем в 190 странах мира. Возьмем еще один случай: Amazon. Компания Amazon заявляет, что почти 30% ее продаж осуществляются посредством рекомендаций. Обратите внимание, что большинство сегодняшних успешных компаний в качестве ядра своего бизнеса содержат рекомендательную систему. Компании продолжают инвестировать, исследовать и обновлять свои рекомендательные системы с целью их модернизации и обеспечения конечному пользователю максимально возможного опыта персонализированного взаимодействия.

Создание рекомендательных систем является нелегкой задачей, и компании активно вкладывают в нее не только свои финансовые ресурсы, но и время. Решение этой задачи требует наличия глубоких экспертных знаний в области науки о данных. Microsoft предоставляет два пути создания рекомендательной системы:

- рекомендации на основе механизмов машинного обучения в Azure посредством рекомендателя Matchbox;
- API рекомендаций когнитивных служб Microsoft.

Каждый из этих путей взаимодействия выполняет различные цели и отличается функциональностью. Анализ работы механизма машинного обучения Azure с помощью рекомендателя Matchbox заслуживает отдельной главы. Однако важно понять основное отличие. Рекомендатель Matchbox предлагает рекомендации на основе ранжирования, в то время как API Рекомендаций больше ориентирован на транзакции. Для предоставления рекомендаций API рекомендаций идентифицирует шаблоны транзакций в ваших данных. API рекомендаций позволяет создавать персонализированные рекомендации всего в нескольких строках кода, улучшать качество обслуживания клиентов и, как результат, увеличивать продажи товаров. К концу этой главы вы:

- поймете, зачем нужны рекомендации;
- ощутите силу API рекомендаций компании Microsoft;
- познакомитесь с различными возможностями по настройке рекомендаций, которые предоставляет API рекомендаций;
- познакомитесь с управлением моделями в API рекомендаций;
- познакомитесь с несколькими оптимальными практическими приемами использования API рекомендаций;
- научитесь применять интерфейс пользователя рекомендаций.



ПОНИМАНИЕ ОСНОВ

К настоящему времени вы уже знаете одну из систем рекомендаций. Эта система выполняется на фоне нескольких лидирующих сайтов для показа товаров, которые вы, скорее всего, приобретете. Рекомендации можно представить как вид на-

вигации. Вы, скорее всего, приобретете тот товар, который видите, и большинство компаний показывает вам свой товар исключительно посредством рекомендаций. Это означает, что рекомендательная система в значительной степени облегчает навигацию в поисках какого-то конкретного товара. Кроме того, следует помнить одну вещь: API рекомендаций компании Microsoft не является решением для всех типов рекомендаций. Как отмечалось ранее, этот API предназначен главным образом для рекомендаций, связанных с транзакциями. Как показано на рис. 9.1, API рекомендаций компании Microsoft на верхнем уровне поддерживает три вида рекомендаций.



Рис. 9.1 ❖ Типы рекомендаций, поддерживаемых API рекомендаций

Давайте разберемся с каждым из этих типов рекомендаций по отдельности.

РЕКОМЕНДАЦИЯ ТОВАРОВ, ЧАСТО ПРИОБРЕТАЕМЫХ ВМЕСТЕ

Посетите Amazon.com. Отыщите роман Льва Толстого «Война и мир», одну из моих любимых книг. Как показано на рис. 9.2, вы увидите, что помимо этого романа Amazon порекомендует вам другие книги. Эти товары часто приобретались вместе с романом «Война и мир».

Frequently bought together

<input checked="" type="checkbox"/> This item: War and Peace (Vintage Classics) by Leo Tolstoy Paperback \$12.19
<input checked="" type="checkbox"/> Crime and Punishment by Fyodor Dostoyevsky Paperback \$5.50
<input checked="" type="checkbox"/> The Brothers Karamazov by Fyodor Dostoevsky Paperback \$14.57

Рис. 9.2 ❖ Пример рекомендаций FBT на Amazon

API рекомендаций товаров, часто приобретаемых вместе (Frequent Brought Together, FBT), предлагает товарные позиции, которые покупаются или используются в сочетании друг с другом. Обычно лежащий в основе алгоритм выполняет

анализ всех товарных позиций, которые приобретаются вместе с товарной позицией, разыскиваемой вами (скорее всего, как часть той же самой транзакции), и потом предоставляет рекомендации как FBT. Возьмем еще один пример: если вы являетесь фанатом изделий Apple, то вы, возможно, приобрели iPhone 7. Но когда вы покупаете такой дорогостоящий телефон, неужели вы купите его без защитной пленки? Разумеется, нет! Рисунок 9.3 показывает рекомендации на Amazon.



Рис. 9.3 ❖ Рекомендации FBT при покупке iPhone 7

К вашему восхищению, Amazon порекомендует защитное стекло как часть своего плагина рекомендации товаров, часто приобретаемых вместе. Такие рекомендации создают для сбытовых компаний «вау-эффект» в отношении их товаров. Рекомендации FBT являются неотъемлемой частью некоторых розничных торговых площадок в сети. Обычно большинство товаров, заказанных как часть рекомендаций FBT, являются дополняющими по отношению к основному товару, и, как следствие, конечный пользователь чаще всего покупает их вместе. Давайте рассмотрим рекомендации «от предмета к предмету».

РЕКОМЕНДАЦИЯ «ОТ ПРЕДМЕТА К ПРЕДМЕТУ»

Рекомендация «от предмета к предмету», также широко известная как совместная фильтрация по схожести предметов, является еще одним способом предоставления рекомендации конечному пользователю в отношении товарных позиций. Возьмем пример с романом «Война и мир». Разве вы не хотели бы увидеть книги, приобретенные людьми, которые приобрели роман «Война и мир»? Это позволяет вам получить легко понятную информацию о книгах, которые могли бы вас заинтересовать. Amazon.com делает это в разделе под названием «Вместе с этим товаром также приобретают...», как показано на рис. 9.4.

Возьмем наш пример с iPhone. Если вы приобрели iPhone, то вы, вероятно, захотите приобрести другие аксессуары (не только защитную пленку), такие как адаптер для подключения наушников, чехол и т. д. Рано или поздно вы приобретете некоторые из них. Amazon показывает все эти рекомендуемые изделия в одном разделе. Преимущество таких рекомендаций состоит в том, что для поиска сопутствующих товаров вам не нужно посещать другие сайты. Эти продукты расположены на той же самой странице. Пользователи это оценят.

Customers who bought this item also bought

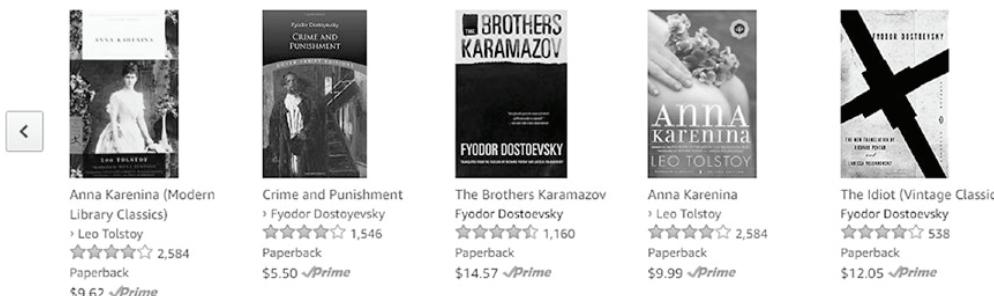


Рис. 9.4 ♦ Больше идей для покупок

РЕКОМЕНДАЦИЯ НА ОСНОВЕ ПРЕДЫСТОРИИ

Рекомендация на основе предыстории, так называемая коллаборативная фильтрация на основе пользователя, или рекомендация «от покупателя к предмету», является еще одним способом предоставления рекомендаций конечному пользователю в отношении товарной позиции. В наших предыдущих примерах я, конечно же, хотел бы знать, когда было опубликовано новое издание книги Льва Толстого или имеется ли новая версия iPhone. Обычно для получения таких рекомендаций не требуется производить никаких действий. Например, некоторые компании присылают на электронную почту письмо с сообщением о поступлении товара, который может вас заинтересовать. Еще один вариант заключается в том, чтобы показывать эти товары на вашей домашней странице в виде новинок. Мы надеемся, что вы уже поняли полезность API рекомендаций компании Microsoft.

Давайте познакомимся с API рекомендаций поближе. Как и в случае со всеми остальными когнитивными службами, сначала вам необходимо подписаться на API рекомендаций. Процесс подписки на API рекомендаций такой же, что и для остальных когнитивных API. Об этом было рассказано в главах 2 и 3. Там же приводятся примеры подписки на когнитивные службы. Двигаясь дальше, мы предполагаем, что вы уже подписались на API рекомендаций и имеете основной и вторичный ключи.

КАК ЭТИ РЕКОМЕНДАЦИИ РАБОТАЮТ?

К настоящему времени вы знакомы с большей частью жаргонизмов машинного обучения и ИИ. По существу, целями машинного обучения являются обучение компьютерной программы на основе исторических и прошлых данных, анализ шаблонов (образов) и использование этих шаблонов для решения каких-то конкретных задач. При этом ключевым элементом понимания является сущность исторических и прошлых данных. В главах 4 и 5 мы изложили важность моделей и исторических данных.

Сначала вы должны создать модель. В API рекомендаций термин «модель» отличается контекстом. Для API рекомендаций модель – это контейнер для ваших данных об использовании модели рекомендаций и каталога. Данные об использовании представляют собой исторические транзакционные данные. Каталог, как

следует из названия, является набором данных о ваших товарных позициях. Под термином «модель», который мы использовали в предыдущих главах, здесь подразумевается модель рекомендаций. По нашему мнению, здесь стоило бы применить более подходящий термин, чтобы избежать путаницы, но давайте остановимся на этом слове. Для того чтобы создать модель (или контейнер), следует с помощью POST-запроса вызвать API <https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models> и передать в заголовке ключ подписки, а имя модели и ее описание – в теле запроса, как показано ниже:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models HTTP/1.1
Content-Type: application/json
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****

{
  "modelName": "BookModel",
  "description": "This is a demo model for our chapter"
}
```



Если ваш ключ подписки действителен, то вы быстро создадите свою модель под названием BookModel и получите приведенный ниже отклик JSON:

```
{
  "id": "6db5116c-977c-4a6d-b1d0-e7b968a8901f",
  "name": "BookModel",
  "description": "This is a demo model for our chapter",
  "createdDateTime": "2017-04-21T09:59:00Z",
  "activeBuildId": -1
}
```

Любую неиспользуемую модель вы всегда можете удалить, вызвав тот же самый API с использованием метода Delete и передав id, как показано ниже:

```
DELETE https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
```

Для любой заданной подписки можно в общей сложности создать до 10 моделей. Всякий раз, когда вам нужно получить список всех моделей, следует вызвать тот же самый API с использованием GET-запроса, передав ключ подписки, как показано ниже:

```
GET https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
```

Как отмечалось ранее, API рекомендаций основывается на исторических данных. Чем точнее ваши исторические данные, тем точнее рекомендации. Поскольку API рекомендаций опирается на транзакции, для того чтобы создать модель рекомендаций (независимо от вариантов, которые вы выбираете), все сводится к тому, что вы по существу предоставляетете два типа исторических данных:

- данные, связанные с товарной позицией;
- данные предыдущих транзакций.

Данные, связанные с товаром, так называемый *каталог*, представляют собой набор данных о ваших товарных позициях (позиции). Его следует рассматривать как список товарных позиций, которые вы предлагаете своим конечным пользователям. Он не только содержит список товарных позиций, но и метаданные и некоторую другую ключевую информацию, такую как идентификатор позиции, описание, категория (если таковая имеется) и т. д. Ваш каталожный файл может быть очень простым в виде значений, отделенных запятыми. Например, приведенный ниже фрагмент кода, сохраненный как CSV-файл, может использоваться в качестве каталожного файла:

```
itemid,item name,Category
P1001,Monitor,Laptop
P1002,Samsung Galaxy S,Phone
P1003,CPU,PC
P1004,iPhone 7,Phone
```

На самом деле, чтобы получить базовый рабочий каталожный файл, он должен находиться в формате, приведенном на рис. 9.5.

Item ID, Item Name, Category

Рис. 9.5 ♦ Базовый формат каталожного файла

Такие типы каталожных файлов всего с тремя полями называются каталожными файлами без признаков. Каталожные файлы без признаков могут использоваться для демонстрационных и пилотных проектов, но в реальной эксплуатационной среде каталожные файлы без признаков могут не работать из-за недостатка информации о товаре. Каталожный файл можно сделать более обстоятельным и полезным, добавив список признаков. Чтобы добавить признак, вам нужно соблюсти формат, показанный на рис. 9.6.

Item ID, Item Name, Category, Description, Feature List

Рис. 9.6 ♦ Формат каталога с признаками

Добавление признаков и необязательных описаний делает ваш каталог более содержательным и привлекательным. На внутреннем уровне API рекомендаций создает статистическую модель на основе данных каталога. Вернитесь назад к главе 4, чтобы больше узнать о статистической модели. Чем больше будет признаков в каталоге и чем больше будет записей, тем более точную статистическую модель для рекомендаций мы получим. Согласно этому формату, образец вашего каталога будет выглядеть следующим образом:

P1001,Monitor,Laptop,,“Color=Black, availability =US/UK, Size =15.6”“”

P1002,Samsung Galaxy S,Phone,,“Color=Gray, availability=All, Screen size=5”“, RAM=4 GB”

P1003,CPU,PC,,“Color=White, availability=APAC”

P1004,iPhone 7,Phone,,“Color=Gray, availability=US, Screen size=5”“, RAM=16GB”

Еще одно преимущество от использования списка признаков состоит в возможности рекомендовать конечным пользователям новые, недавно поступившие товары. При наличии признаков новые товарные позиции можно сравнивать с интересующими конечного пользователя. И, прежде чем предлагать конечному пользователю новые товарные позиции, следует проверить, имеются ли у вновь поступивших товаров признаки товаров, приобретенных пользователем ранее. Кроме того, посмотрите на приведенный выше фрагмент кода: в приведенный выше образец мы не добавили никаких полей описания. Поэтому после названия товара вы видите две запятые. На момент написания этой книги каталог имел несколько ограничений:

1. Он может иметь не более 20 признаков.
2. Размер каталожного файла не должен превышать 200 МБ.
3. Максимальное количество позиций в каталожном файле равняется 100K.

 Если у вас каталожный файл большого размера, в котором более 20 признаков, попробуйте ранжировать эти признаки, чтобы модель могла взять только те признаки, которые имеют более высокую ранговую величину. В идеальном случае ваши данные об использовании будут определять рекомендацию в фактическом смысле. Однако добавление дополнительных признаков будет включать в рекомендации мало востребованные товарные позиции. Термин «мало востребованная позиция» (cold item) используется для обозначения тех товаров, которые редко покупаются и имеют очень немного транзакций в прошлом.

Для того чтобы закачать каталожный файл, обратитесь к API по <https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/>, передав идентификатор созданной ранее модели и отображаемое имя каталога вместе с закачиваемым файлом в теле запроса в виде двоичного файла, как показано ниже:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f/catalog?catalogDisplayName=BookCatalog HTTP/1.1
Content-Type: application/octet-stream
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....
```

Допустим, что ваш идентификатор модели и ключ подписки правильные и переданный каталожный файл находится в правильном двоичном формате. В этом случае мы получим отклик с кодом состояния 201, показанный ниже:

```
{
  "processedLineCount": 5,
  "errorLineCount": 0,
  "importedLineCount": 5,
  "errorSummary": [],
  "sampleErrorDetails": []
}
```

Как отмечалось ранее, помимо каталожного файла, еще требуются исторические данные или данные об использовании в прошлом. Данные об использовании играют важную роль в создании модели статистического анализа, так как они обеспечивают сведения об интересах пользователей. Эти данные показывают взаимодействия пользователя в историческом плане с товарными позициями каталога. Подобно созданию вашего каталожного файла, процесс создания файла с данными об использовании достаточно простой: имеются всего четыре поля (рис. 9.7).

User ID, Item Id, Timestamp, Event(optional)

Рис. 9.7 ♦ Базовый формат для файла с данными об использовании

Первые три поля обязательны. Из них только поле идентификатора товарной позиции должно быть одним из полей в каталожном файле. Ваш простой файл с данными об использовании должен выглядеть следующим образом:

```
User1001, P1001,2017/08/04T11:02:52
User1005, P1002,2017/08/04T09:02:52
User1001, P1003,2017/08/01T11:01:10
User1003, P1004,2017/08/04T08:02:52
```

Каждая строка в этом файле может рассматриваться как одно взаимодействие между пользователем и товарной позицией и может называться транзакцией. Событие Event является необязательным полем и может иметь одно из перечисленных ниже значений, где Purchase является значением по умолчанию:

- Click (нажатие);
- RecommendationClick (нажатие на рекомендациях);
- AddShopCart (добавить корзину с покупками);
- RemoveShopCart (удалить корзину с покупками);
- Purchase (покупка).

Файл с данными об использовании должен содержать, по крайней мере, 10–20 записей, даже если вы используете его для демонстрационного примера. **Если в файле будет меньше количество записей**, то вы получите ошибку с сообщением, что «в связанную модель не закачивается файл с данными об использовании». Помимо этого, во время написания этой книги имелся верхний порог в размере 200 МБ данных для файла, закачиваемого в одном POST-событии. Кроме того, предварительным условием является закачивание сначала каталожного файла и только потом закачивание файла с данными об использовании. Если ваш объем данных об использовании превышает верхний порог, то его следует разделить на несколько файлов CSV. При этом объем каждого файла должен превышать 200 МБ. Ваш вызов API для увязывания с моделью файла с данными об использовании должен иметь HTTP-запрос, как показано ниже:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f//usage?usageDisplayName=BookUsage HTTP/1.1
```

Content-Type: application/octet-stream
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key:
.....

Если ваш идентификатор модели и ключ подписки действительные, а файл с данными об использовании соответствует указанным рекомендациям, то в итоге вы создадите свой файл, получив отклик с кодом состояния 201 «Создано». До этого времени были показаны способы создания модели и добавления в нее каталожного файла и файла с данными об использовании. Следующий шаг состоит в использовании этих трех компонентов для создания модели рекомендаций на основе машинного обучения. Прежде чем мы займемся фактическим созданием модели рекомендаций, давайте разберемся в различных типах моделей рекомендаций, поддерживаемых API рекомендаций когнитивных служб.



МОДЕЛИ И ТИПЫ РЕКОМЕНДАЦИЙ

Чтобы создать машинно-обучаемую модель рекомендаций, необходимо запустить сборку модели. На внутреннем уровне Microsoft использует ее как процесс тренировки, в результате которой получается модель рекомендации на основе машинного обучения. Единственной предпосылкой для запуска сборки является наличие созданной вами модели (или контейнера) и закачанного каталожного файла и файла с данными об использовании. Весь процесс сборки модели является асинхронным и может потребовать от нескольких минут до нескольких часов в зависимости от объема каталожного файла и файла с данными об использовании. Прежде чем мы займемся подробным анализом внутреннего механизма запуска сборки, сначала давайте разберемся с типами сборок, поддерживаемых в API рекомендаций. Для создания или запуска сборки вам нужно обратиться с POST-запросом на <https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/>, передав `modelId` в качестве параметра запроса и ключ подписки в заголовке. Составной частью любой сборки должны быть два основных параметра:

- описание;
- тип сборки.

Основной элемент сборки – это описание, которое можно рассматривать и как имя сборки. Сейчас необходимо уяснить, какой тип рекомендаций следует создать во время сборки. В настоящее время поддерживается четыре типа рекомендаций:

- сборка обычной модели рекомендаций (Recommendation Build);
- сборка модели рекомендаций на основе FBT (Frequently Bought Together, на основе товарных позиций, часто приобретаемых вместе);
- сборка модели рекомендаций на основе ранжирования (Rank Build);
- сборка модели рекомендаций на основе SAR (Smart Adaptive Recommendation, на основе механизма интеллектуальной адаптивной рекомендации).

В качестве типа рекомендаций необходимо назначить один из перечисленных выше типов. Другая информация, такая как соответствующие для этой сборки параметры, должна быть частью тела запроса. Каждый тип сборки имеет несколько уникальных параметров. Давайте перечислим эти параметры и попробуем в них разобраться.

СБОРКА ОБЫЧНОЙ МОДЕЛИ РЕКОМЕНДАЦИЙ

Сборка обычной модели рекомендаций (Recommendation Build) используется, когда требуются либо рекомендации «от предмета к предмету», либо рекомендации «от покупателя к предмету». В предыдущих разделах этой главы мы уже рассказали о каждом из основных понятий. Сборка обычной модели рекомендаций поддерживает ряд уникальных параметров, которые подробно изложены ниже:

1. Во время выполнения тренировки на этих сборках API рекомендаций по умолчанию делает 20 итераций модели. Итерация – это организация обработки данных, при которой действия повторяются многократно, не приводя при этом к вызовам самих себя. Чем больше количество итераций, тем дольше длятся вычисления и тем выше точность модели. Если вы хотите изменить количество итераций, назначьте параметру `NumberOfModelIterations` конкретное число. Это число может принимать значения от 10 до 50.
2. Во время выполнения тренировки модель попытается найти в ваших данных 20 признаков. В большинстве сценариев этого количества достаточно, и его менять не следует. Однако это значение всегда можно изменить, назначив параметру `NumberOfModelDimensions` значение от 10 до 50. Увеличение числа позволит выполнить более точную настройку, но при этом затруднит обнаружение корреляции (взаимосвязи) между товарными позициями. Пометьте это значение как неизменяемое, если только у вас нет веских причин для выполнения точной настройки ваших результатов.
3. Также можно изменить величины минимального и максимального количества использования, служащие, чтобы та или иная товарная позиция рассматривалась в модели, изменив соответственно величины `ItemCutOffLowerBound` и `ItemCutOffUpperBound`. Величина должна быть от 0 до 30. Параметр `ItemCutOffLowerBound` имеет значение по умолчанию, равное 20.
4. Вы также можете изменить минимальное и максимальное количества precedентов использования, которые требуются для рассмотрения в модели, изменив величину соответственно нижней границы отсечения `UserCutOffLowerBound` и верхней границы отсечения `UserCutOffUpperBound`. Эта величина должна быть от 0 до 10. По умолчанию эта величина равна 2.
5. Большую часть случаев данные об использовании являются ключевым фактором при принятии решения о рекомендации. Вы также можете принять решение об использовании признаков для усиления модели, установив значение `UseFeaturesInModel`. По умолчанию значение этого параметра равняется `True`. Это значение следует оставить без изменения, если только у вас есть сомнения в перспективности вашей модели. Вы также можете предложить признаки, которые будут использоваться в модели рекомендации, указав в `ModelingFeatureList` имена признаков в виде списка отделенных запятыми значений. Важность задания списка имен признаков обусловлена тем, что модель рекомендации не будет использовать признаки, если этот список будет пустым.
6. Вы также можете принять решение, чтобы модель рекомендации показывала мало востребованные товарные позиции при отображении аналогичных позиций, назначив параметру `AllowColdItemPlacement` значение `True`.

7. Присвойте параметру `EnableFeaturesCorrelation` значение `True`, если вы хотите, чтобы признаки использовались для демонстрации логического вывода. Если вы присваиваете параметру `EnableFeaturesCorrelation` значение `True`, укажите в `ReasoningFeatureList` имена признаков в виде списка отделенных запятыми значений, которые будут использоваться для демонстрации логического вывода.
8. По умолчанию модель рекомендации будет показывать рекомендации «от предмета к предмету». Присвойте параметру `EnableU2I` значение `True`, чтобы получить рекомендацию «от покупателя к предмету».

Пример POST-запроса модели рекомендации с использованием нескольких параметров сборки будет выглядеть следующим образом:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f/builds HTTP/1.1
Content-Type: application/json
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****
```

```
{
  "description": "Simple recommendations build",
  "buildType": "recommendation",
  "buildParameters": {
    "recommendation": {
      "numberOfModelIterations": 25,
      "itemCutOffLowerBound": 1,
      "userCutOffLowerBound": 0,
      "userCutOffUpperBound": 0,
      "enableModelingInsights": false,
      "useFeaturesInModel": false,
      "modelingFeatureList": "string",
      "allowColdItemPlacement": false,
      "enableFeatureCorrelation": true,
      "reasoningFeatureList": "string",
      "enableU2I": true
    }
  }
}
```

Поскольку запуск сборки является асинхронной операцией, в результате исполнения приведенной выше инструкции вы получите код состояния 202 «Принято» с приведенным ниже откликом:

```
Pragma: no-cache
Transfer-Encoding: chunked
Operation-Location: https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/operations/1623835
x-ms-request-id: f1e5bcb5-660c-4489-b72c-4197fa5d7969
X-Frame-Options: deny
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
apim-request-id: 587594af-b0c4-473b-baa3-09fe75a12313
Cache-Control: no-store, no-cache
```

```
Date: Sun, 23 Apr 2017 04:56:08 GMT
Location: https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/
operations/1623835
Content-Type: application/json; charset=utf-8
Expires: -1

{
  "buildId": 1623835
}
```

Обратите внимание на идентификатор operationsid, упомянутый в состоянии заголовка. В приведенном выше фрагменте кода operationsid равняется 1623835. Значение operationsid потребуется для отслеживания или отмены сборки. Это значение может пригодиться в случае, если вы случайно создали операцию и хотите удалить существующую сборку. Для отслеживания состояния сборки обратитесь на страницу по адресу <https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/operations/> и передайте с помощью GET-запроса operationsid в параметре запроса и ключ подписки в заголовке. Допустимый запрос с operationsid будет выглядеть следующим образом:

```
GET https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/operations/1623835
HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....
```

Вызов того же самого API с методом Delete отменяет операцию. Успешный HTTP-запрос об отслеживании состояния сборки вернет HTTP-отклик, как показано ниже:

```
Pragma: no-cache
Transfer-Encoding: chunked
x-ms-request-id: 1bc423b1-8942-4fb6-a047-cda7a535ad4e
X-Frame-Options: deny
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
apim-request-id: a1861134-4ee7-4ef8-91a9-6d63a5955fdf
Cache-Control: no-store, no-cache
Date: Mon, 24 Apr 2017 13:07:51 GMT
Content-Type: application/json; charset=utf-8
Expires: -1

{
  "type": "BuildModel",
  "status": "Succeeded",
  "createdDateTime": "2017-04-23T04:56:09",
  "percentComplete": 0,
  "resourceLocation": "https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/
operations/1623835?buildId=1623835",
  "result": {
    "id": 1623835,
    "type": "Recommendation",
    "modelName": "BookModel",
    "modelId": "6db5116c-977c-4a6d-b1d0-e7b968a8901f",
    "status": "Succeeded",
```

```

    "startDateTime": "2017-04-23T04:56:09",
    "endDateTime": "2017-04-23T04:57:55"
}
}

```

Как видно, отклик HTTP ясно определяет имя модели, ее тип, идентификатор modelId, состояние и время ее создания и завершения.

Иногда ваши параметры могут быть неправильными. Например, если назначить параметру NumberOfModelIterations значение меньше 10, то будет получено сообщение 401. Если вы оказываетесь в такой ситуации, внимательно просмотрите сообщение JSON. Как правило, сообщение innerError является достаточно описательным, чтобы предложить возможные причины ошибки. В нашем примере значение NumberOfModelIterations менее 10 приведет к следующему отклику JSON:

```

Pragma: no-cache
Transfer-Encoding: chunked
x-ms-request-id: ed216733-9634-4abc-bd6d-ad3d077499ee
X-Frame-Options: deny
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
apim-request-id: 94c53473-f078-45bd-a3ea-3294519f2e99
Cache-Control: no-store, no-cache
Date: Sun, 23 Apr 2017 05:01:14 GMT
Content-Type: application/json; charset=utf-8
Expires: -1

{
  "error": {
    "code": "BadArgument",
    "message": "(EXT-0108) Passed argument is invalid.",
    "innerError": {
      "code": "EXT-0310",
      "message": "Failed to trigger build for model 6db5116c-977c-4a6d-b1d0-e7b968a8901f.
Invalid build parameter.",
      "innerError": {
        "code": "EXT-0046",
        "message": "Build parameter 'NumberOfModelIterations' has invalid value '5'.
ModelId: 6db5116c-977c-4a6d-b1d0-e7b968a8901f"
      }
    }
  }
}

```



СБОРКА МОДЕЛИ РЕКОМЕНДАЦИИ НА ОСНОВЕ FBT

Сборка модели рекомендации на основе FBT, по сути, предлагает товарные позиции, которые часто приобретаются или используются в сочетании друг с другом. Это становится возможным путем анализа данных об использовании, в результате которого можно увидеть, какие товарные позиции встречаются в покупке вместе, и затем применив анализ подобия на основе функций подобия Jaccard (на основе коэффициента Жаккара), Lift (на основе показателя лифта) или со-ос-

currence (на основе совместной встречаемости). Таблица 9.1 показывает примеры функции подобия на основе совместной встречаемости.

Таблица 9.1. Метрики совместной встречаемости между товарными позициями

	Item001	Item002	Item003	Item004	item005
Item001	3	7	1	2	6
Item002	7	7	5	4	3
Item003	1	5	6	5	1
Item004	2	4	5	1	1
Item005	6	3	2	1	3

Приведенный выше пример представляет собой классический способ определения данных для анализа подобия на основе функции совместной встречаемости. Глядя на таблицу, вы видите, сколько раз та или иная товарная позиция встречалась совместно. Это самый легкий метод анализа подобия; вместе с тем он довольно предсказуем. Есть большая вероятность, что все время будут генерироваться рекомендации в отношении популярных товарных позиций. Для решения этой проблемы поддерживаются два других алгоритма: Lift и Jaccard.

Функция подобия на основе лифта (Lift) – это процесс нахождения товарной позиции посредством интуитивной прозорливости (serendipity) или эвристики (discovery). В табл. 9.1 позиция 2 и позиция 4 имеют одинаковую с позицией 3 совместную встречаемость, при этом позиция 4 обычно встречается менее часто, чем позиция 2, следовательно, функция подобия Lift отдаст предпочтение позиции 4 над позицией 2. Функция подобия на основе лифта, Lift, используется для FBT-рекомендаций по умолчанию. Функция подобия на основе совместной встречаемости, co-occurrence, всецело связана с предсказуемостью, а метод Jaccard берет лучшее из функций подобия Lift и co-occurrence. Поскольку FBT-сборка основывается на упомянутом выше процессе, она является самой традиционной сборкой модели рекомендаций. Поскольку рекомендации на основе FBT связаны с совместной покупкой пользователем двух товаров в составе одной и той же транзакции, сборка модели рекомендаций на основе FBT не поддерживает персонализированных рекомендаций и мало востребованных позиций. FBT-сборка поддерживает всего несколько типов параметров:

1. Как отмечалось ранее, FBT-сборка рекомендует товарные позиции, когда в данных об использовании эти товары встречаются совместно несколько раз. По умолчанию, чтобы две позиции были рассмотрены моделью, в FBT-сборке ожидается, что две позиции будут встречаться совместно, по крайней мере, 6 раз. Однако эту величину можно изменить, присвоив параметру порога `FbtSupportThreshold` любое число от 3 до 50. Чем больше величина порога `FbtSupportThreshold`, тем консервативнее созданные вами FBT-сборки. Рекомендуем держать первоначальное значение порога максимально низким (скажем, 3) для исходных данных. После того как у вас появилось большое количество транзакционных данных, ему следует назначить более высокое значение. Если задать числа, иные, чем указанные величины, то вы получите состояние сборки со значением «Неправильный запрос» и ошибкой в виде «кода»: «EXT-0046», «message»: «Build parameter 'FbtSupportThreshold' has invalid value '0'. ModelId: 6db5116c-977c-4a6d-b1d0-

`e7b968a8901f»` (Параметр сборки ‘FbtSupportThreshold’ имеет недопустимое значение ‘0’...).

2. По умолчанию FBT-сборки привязывают количество товарных позиций к набору, состоящему из двух элементов. Это срабатывает в большинстве случаев. В определенных сценариях может возникнуть желание иметь в наборе 3 товарные позиции, и поэтому вы можете поменять значение размера набора `FbtMaxItemSetSize` на 3. Однако важно помнить, что в `FbtMaxItemSetSize` поддерживается только два значения: 2 и 3. Любое другое значение набора вернет внутреннюю ошибку `<innerError>: { «code»: «EXT-0046», «message»: «Build parameter 'FbtMaxItemSetSize' has invalid value '0'. ModelId: 6db5116c-977c-4a6d-b1d0-e7b968a8901f».`
3. После того как вы задали количество позиций в FBT-наборе, может понадобиться установить минимальный уровень сработки в FBT-наборе. Это можно сделать, присвоив параметру минимального уровня `FbtMinimalScore` любое значение, выраженное целым числом. По умолчанию это значение равняется 2. Чем выше величина этого значения, тем лучше рекомендация, но эта величина должна быть установлена в соединении с объемом данных об использовании. Если данные об использовании небольшие, то стоит оставить значение по умолчанию.
4. По умолчанию функция подобия `FbtSimilarityFunction`, которая применяется в FBT-сборке, имеет значение `Lift`. Если вы хотите поменять функцию подобия, то параметру `FbtSimilarityFunction` следует присвоить значение `Jaccard` (функция подобия на основе коэффициента Жаккара) или `co-occurrence` (функция подобия на основе показателя совместной встречаемости).

Успешный POST-запрос на FBT-сборку должен иметь приведенный ниже HTTP-отклик:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f/builds HTTP/1.1
Content-Type: application/json
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: *****

{
  "description": "Simple frequent build",
  "buildType": "fbt",
  "buildParameters": {
    "fbt": {
      "supportThreshold": 4,
      "maxItemsetSize": 2,
      "minimalScore": 2,
      "similarityFunction": "Jaccard",
      "enableModelingInsights": true,
    }
  }
}
```

В случае успешности вы получите соответствующий отклик с идентификатором `operationId`, как показано ниже:

Pragma: no-cache
 Transfer-Encoding: chunked
 Operation-Location: <https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/operations/1623876>
 x-ms-request-id: 4dbf70be-af6f-41ca-b849-d9c8b51c3a6
 X-Frame-Options: deny
 X-Content-Type-Options: nosniff
 Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
 apim-request-id: 1ebade34-a252-4440-9e30-683cd1f9588f
 Cache-Control: no-store, no-cache
 Date: Sun, 23 Apr 2017 16:43:36 GMT
 Location: <https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/operations/1623876>
 Content-Type: application/json; charset=utf-8
 Expires: -1

```
{
  "buildId": 1623876
}
```



Следует напомнить, что каждое из этих действий по сборке представляет собой асинхронную операцию. Как отмечалось выше, идентификатор operationsid может использоваться для отслеживания или отмены состояния сборки.

СБОРКА МОДЕЛИ РЕКОМЕНДАЦИЙ НА ОСНОВЕ РАНЖИРОВАНИЯ

Ранжирование играет важную роль в результатах отклика. На самом деле большинство возвращаемых откликов основано на ранге. Мы вкратце коснулись ранжирования, когда рассматривали поиск. Взгляните на результаты поиска видео в Bing, которые приведены на рис. 9.8.

The screenshot shows the Bing search results for the query "Microsoft cognitive". At the top, there's a search bar with the query, a magnifying glass icon, and a ranking indicator labeled "Ranking". Below the search bar, the results are displayed in a grid format. A large grey arrow points horizontally across the results, indicating the ranking of the videos from left to right. The results include various Microsoft Cognitive Services-related videos, such as "Microsoft Cognitive Services: Introducing the Seeing AI project", "Fun with Microsoft Cognitive Services", "How Uber is using driver selfies to enhance security, powered by ...", "Build 2016 : Microsoft Cognitive Services Give Your Apps a Human", "Microsoft Cognitive Services: An Emotional Demo", "Microsoft Cognitive Services: Give your apps a human side", "An Overview of Microsoft Cognitive Services", "Develop applications with Intelligence using Microsoft", "Deep Learning with Microsoft Cognitive Toolkit CNTK – Microsoft", and "Microsoft Cognitive Services: Build smarter and more engaging". Each result card includes a thumbnail, title, channel name, duration, views, and a "Save" button.

Рис. 9.8 ♦ Результат поиска видео в Bing с примером ранжирования

Возвращенные в Bing результаты поиска видео на самом деле основаны на ранг-ре, причем первое видео имеет самый высокий ранг. Ранжирование идет слева направо и сверху вниз. Ранжирование является ядром рекомендаций во многих сценариях, таких как результаты поиска, лента новостей и т. д. Поэтому всякий раз, когда вы заходите на YouTube, Bing, Google или Netflix и выполняете поиск, помните, что результаты поиска основаны на ранге.

Каким же образом выполняется ранжирование результатов? Дело в том, что ответ на этот вопрос зависит от объема ваших данных об использовании. Если объем ваших данных об использовании большой, то вы можете опереться на создание модели на основе ранжирования, и модель Microsoft ранжирования рекомендаций поделится с вами результатами. Но что происходит, когда объем данных слишком мал? В этом случае ключевую роль играют признаки. Напомним, что вы добавили признаки в каталожные данные, а также указали важность признака. Эти признаки и их важности принимаются в расчет на первоначальном этапе, если данные об использовании имеют небольшой объем. Когда объем данных об использовании увеличивается, ранжирование признака соответствующим образом изменяется.

Параметры ранжирования аналогичны параметрам рекомендации и приведены ниже. Мы не раскрываем их полностью, потому что эти параметры уже рассматривались ранее. Чтобы освежить свою память, вернитесь назад к обычной модели рекомендации.

- NumberOfModelIterations;
- NumberOfModelDimensions;
- ItemCutOffLowerBound и ItemCutOffUpperBound;
- UserCutOffLowerBound и UserCutOffUpperBound.

Типичный HTTP POST-запрос на выполнение сборки модели рекомендации на основе ранжирования будет выглядеть следующим образом:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f/builds HTTP/1.1
Content-Type: application/json
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apm-Subscription-Key: *****

{
  "description": "Simple Ranking build",
  "buildType": "ranking",
  "buildParameters": {
    "ranking": {
      "numberOfModelIterations": 15,
      "numberOfModelDimensions": 10,
      "itemCutOffLowerBound": 10,
      "itemCutOffUpperBound": 20,
      "userCutOffLowerBound": 5,
      "userCutOffUpperBound": 9
    }
  }
}
```

Как и любая другая сборка модели рекомендации, успешная сборка модели рекомендации на основе ранжирования вернет идентификатор operationsid, как показано в приведенном ниже HTTP-отклике:

```

Pragma: no-cache
Transfer-Encoding: chunked
Operation-Location: https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/
operations/1623950
x-ms-request-id: ca874586-a650-4422-811a-308cef8c4838
X-Frame-Options: deny
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
apim-request-id: 84a1c08b-3b80-43a5-af19-239c19138aad
Cache-Control: no-store, no-cache
Date: Mon, 24 Apr 2017 05:10:17 GMT
Location: https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/
operations/1623950
Content-Type: application/json; charset=utf-8
Expires: -1
{
  "buildId": 1623950
}

```



СБОРКА МОДЕЛИ РЕКОМЕНДАЦИИ НА ОСНОВЕ SAR

Возьмем самый первый пример со сборкой обычной модели рекомендации. Сборки обычной модели рекомендации хорошо работают для рекомендаций «от предмета к предмету» и «от пользователя к предмету». Хорошой особенностью модели является то, что ответ конечному пользователю персонализирован согласно модели рекомендации. Однако, когда добавляются новые товарные позиции, сборка обычной модели рекомендации не показывает тенденцию генерировать отличные ответы, за исключением показа некоторых мало востребованных позиций. Именно в таких ситуациях на выручку приходит модель рекомендации на основе интеллектуальной адаптивной рекомендации Smart Adaptive Recommendation (SAR). Эта модель представляет собой новейшее дополнение к моделям рекомендации, поддерживаемым API рекомендаций. Она отличается от сборки обычной модели рекомендации тем, что SAR-сборка основана на взаимодействиях пользователя с системой. SAR-сборка не опирается на ранжирование товарных позиций пользователями, которое зачастую ненадежно, а больше опирается на общность пользователей с системой. Таблица 9.2 определяет матрицу «от пользователя к предмету».

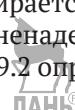


Таблица 9.2. Матрица «от пользователя к предмету»

	<i>Item001</i>	<i>Item002</i>	<i>Item003</i>	<i>Item004</i>	<i>item005</i>
<i>User001</i>	2	1		4	6
<i>User002</i>	1		5	4	1
<i>User003</i>	1	2	1	1	4
<i>User004</i>	3	2		1	
<i>User005</i>	6		2		

Если взглянуть на матрицу, то можно безопасно предположить, что User001 максимально раз связан с предметом Item006, после которого идет

предмет Item004, и ни разу не связан с предметом Item003. Кроме того, User004 ни разу не связывался с предметом Item005, но имеет максимальную связь с предметом Item001. Эта ситуация в действительности может измениться на основе событий, когда имелось взаимодействие пользователя и предмета. Кроме того, каждое событие может иметь разный вес. Например, событие «Покупка» будет иметь максимальный вес. Объединение этого веса с приведенной выше матрицей генерирует матрицу общности пользователей (user affinity matrix), которая затем может использоваться для предоставления рекомендаций. Модель SAR производит два типа рекомендаций:

- пользовательские рекомендации;
- рекомендации FOT (Frequent Occurring Together, т. е. на основе предметов, часто встречающихся вместе).

Для пользовательских рекомендаций, как мы видели ранее, SAR зависит от транзакционных данных, общности пользователей и подобия признаков, чтобы получить матрицу рекомендаций, имеющую для предметов балльную оценку и взаимодействие пользователя. Рекомендации на основе SAR больше выступают как динамическая сборка обычной модели рекомендации, и ее показатели продолжают изменяться на основе взаимодействия пользователя. Рекомендации на основе FOT, с другой стороны, являются надмножеством рекомендаций на основе FBT и основаны на рекомендациях «от предмета к предмету» и анализе подобия. По сравнению с фактической FBT-сборкой, функция подобия Jaccard рассматривается как заданная по умолчанию функция анализа подобия. Исходя из начальных экспериментов, результативность SAR-сборки превышает результативность предыдущих сборок моделей рекомендаций. Мы, конечно же, рекомендуем вам попробовать рекомендации на основе SAR-сборки, если вы до настоящего времени использовали сборку обычной модели рекомендации. SAR-сборка сопровождается множеством параметров, некоторые из которых являются общими для сборок обычной модели и модели на основе FBT:

1. Аналогично порогу `FbtSupportThreshold`, рассмотренному в разделе о FBT-сборке, вы можете предоставить пороговое число, чтобы выбрать пороговую совместную встречаемость двух товарных позиций, назначив параметру `SupportThreshold` заданное значение. Значение по умолчанию и допустимые значения точно такие же, что у `FbtSupportThreshold` и для `SupportThreshold`.
2. Вы можете задать функции подобия, которые будут применяться в SAR-сборке, назначив параметру `SimilarityFunction` значение `Jaccard`, `Lift` или `Cooccurrence`. В SAR-сборке значением по умолчанию является `Jaccard`.
3. Вы можете группировать встречаемость на основе пользователя или на основе временной метки. По умолчанию группирование выполняется на основе пользователя. Это можно изменить, назначив параметру `CooccurrenceUnit` значение `Timestamp`.
4. По умолчанию мало востребованные товарные позиции не пропускаются через анализ подобия. Назначьте параметру `EnableColdItemPlacement` значение `True`, чтобы выполнялся анализ подобия признаков. После этого вы можете по выбору задействовать рекомендацию «от мало востребованного предмета к мало востребованному предмету». Она не задействована по

умолчанию. Чтобы ее задействовать, назначьте параметру `EnableColdToColdRecommendations` значение `True`.

5. Назначьте параметру `EnableBackfilling` значение `True`, чтобы предоставлять рекомендации в отношении популярных товарных позиций, когда отсутствуют подходящие рекомендации.

Ниже приведен пример HTTP POST-запроса на создание SAR-сборки:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f/builds HTTP/1.1
Content-Type: application/json
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....
{
  "description": "Simple SAR build",
  "buildType": "sar",
  "buildParameters": {
    "sar": {
      "supportThreshold": 5,
      "cooccurrenceUnit": "User",
      "similarityFunction": "Jaccard",
      "enableColdItemPlacement": true,
      "enableColdToColdRecommendations": true,
      "enableModelingInsights": true,
      "enableU2I": true,
    }
  }
}
```

Как и все наши сборки, приведенный выше фрагмент кода при его исполнении вернет идентификатор `operationsid` в HTTP-отклике, который будет выглядеть следующим образом:

```
Pragma: no-cache
Transfer-Encoding: chunked
Operation-Location: https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/
operations/1624000
x-ms-request-id: 8dc96af3-a20a-46c8-a8c4-99b40e0f9580
X-Frame-Options: deny
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
apim-request-id: 9e3f69f4-b150-4431-889a-37440eada16a
Cache-Control: no-store, no-cache
Date: Mon, 24 Apr 2017 12:56:46 GMT
Location: https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/
operations/1624000
Content-Type: application/json; charset=utf-8
Expires: -1
{
  "buildId": 1624000
}
```

УСТАНОВЛЕНИЕ ПРАВИЛ В СБОРКЕ

Все эти сборки и рекомендации хороши, но время от времени у вас может возникнуть желание изменить некоторые правила. Например, у вас есть новая товарная позиция и вы хотите, чтобы она была показана как одна из позиций в рекомендации во всех рекомендациях, которыми вы делитесь, или вы знаете, что одна из позиций в вашем списке отсутствует, и поэтому вы хотите заблокировать включение этой позиции в рекомендации. Для таких сценариев поверх сборки создаются бизнес-правила. С помощью бизнес-правил вы можете сделать следующее:

- 1) заблокировать включение одного или нескольких списков товарных позиций в рекомендации по имени либо по признаку;
- 2) задействовать поэлементное блокирование включения в рекомендательный список;
- 3) заставить товарную позицию быть частью рекомендации по имени либо признаку.

Простое бизнес-правило, которое блокирует включение одной из товарных позиций в рекомендацию, выглядит следующим образом:

```
POST https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f/rules HTTP/1.1
```

```
Content-Type: application/json
```

```
Host: westeurope.api.cognitive.microsoft.com
```

```
Ocp-Apim-Subscription-Key: .....  
{
```

```
    "type": "blocklist",
    "parameters": {
        "blockList": {
            "itemIds": [
                "item001"
            ]
        },
    }
}
```



ОФЛАЙНОВОЕ ОЦЕНИВАНИЕ

Теперь у вас должно быть достаточное понимание типов сборок и связанных с ними признаков. Есть вероятность, что в итоге вы можете принять одну из сборок, которая не будет рекомендовать правильные товарные позиции фактическому конечному пользователю. Как в этом случае определить правильную сборку для ваших решений? В действительности лучшим способом будет проведение экспериментов путем развертывания разных сборок, основанных на разных моделях и алгоритмах. В этом случае вы создаете сборку, разворачиваете решение в производственной среде и затем отслеживаете уровень конверсии. Уровень конверсии имеет несколько параметров и может зависеть от вашего бизнес-решения. Но обычно уровень конверсии измеряется нажатиями на рекомендации и товарами, приобретенными в результате нажатия на рекомендации. Чем выше уровень конверсии, тем удачнее сборка. Не все виды бизнеса могут предоставить

вам возможность развертывать рекомендации в производственной среде и затем тестируировать с участием фактических пользователей. Такие эксперименты вы можете провести офлайн и прийти к конкретным первоначальным заключениям в результате тестирования на типовых данных, перед тем как развернуть сборку в производственной среде. Как раз здесь в игру вступает офлайновое оценивание.

Процесс офлайнового оценивания довольно прост. Он требует, только чтобы имелся достаточный объем данных об использовании. В процессе офлайнового оценивания данные об использовании разделяются на две части: тренировочные данные и тестовые данные. Вы бегло ознакомились с использованием тренировочных и тестовых данных в предыдущих главах. Чтобы задействовать офлайновое оценивание, вам нужно во время процесса сборки это указать. Чтобы онлайновое оценивание задействовать, достаточно назначить параметру `enableModelingInsights` значение `True`. После этого вам нужно решить, как разделить данные на тестовые и тренировочные. Для этого существует три способа разбиения:

- произвольный разделитель;
- разделитель по последнему событию;
- разделитель по дате.

Произвольный разделитель разбивает данные об использовании на основе процентных долей и начального значения случайного числа. Разделитель по последнему событию разбивает данные на основе последней транзакции по каждому пользователю. Разделитель по дате разбивает данные на основе указанной даты. После того как сборка сгенерирована, вы можете получить метрические показатели разнообразия и прецизионности, вызвав API метрических показателей, как показано ниже:

```
GET https://westeurope.api.cognitive.microsoft.com/recommendations/v4.0/models/6db5116c-977c-4a6d-b1d0-e7b968a8901f/builds/1624000/metrics HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Ocp-Apim-Subscription-Key: .....12.....
```

Обратите внимание, что наблюдаемая модель и идентификатор сборки передаются в качестве параметров запроса. HTTP-отклик вернет подробную информацию с метрическими показателями, в том числе прецизионность рекомендации товарной позиции, прецизионность рекомендации пользователя, разнообразие рекомендации пользователя и т. д.

ВЕБ-ИНТЕРФЕЙС API РЕКОМЕНДАЦИЙ

К настоящему времени вы получили достаточное представление об API рекомендаций и о том, как его применять. Одна из причин, почему мы подчеркнуто придерживались такого изложения понятий с фрагментами кода и запросов API, состоит в том, чтобы удостовериться, что вы как разработчик имеете полное представление о внутреннем устройстве приложения. Компания Microsoft также разработала специальный интерфейс пользователя API рекомендаций (на момент написания книги этот интерфейс вышел в бета-версии). Этот интерфейс не связан с внутренним устройством API и предоставляет простой в использовании интерфейс. Здесь можно экспериментировать и тестирувать рекомендации. Вы можете создать сборку, загрузить каталогные данные и данные об использовании

и протестировать фактическую сборку даже в оффлайновом режиме всего за несколько щелчков мыши. Чтобы получить доступ к интерфейсу пользователя API рекомендаций, перейдите на страницу по адресу <https://recommendations-portal.azurewebsites.net/>. Обратите внимание, что сейчас этот URL-адрес может быть другим. Поэтому я настоятельно рекомендую, когда вам понадобится доступ к этому порталу, перейдите на страницу по адресу www.microsoft.com/cognitive-services/en-us/recommendations-api (рис. 9.9).

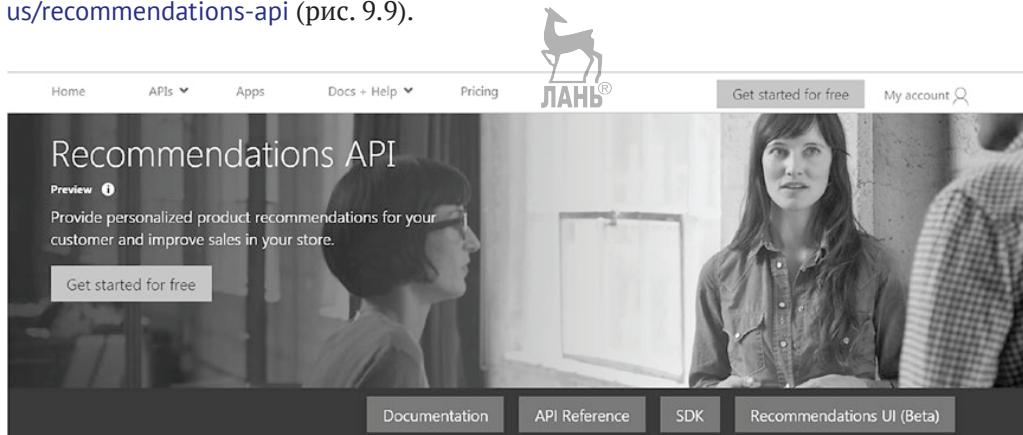


Рис. 9.9 ❖ Домашняя страница API рекомендаций

Когда вы попадете в веб-интерфейс API рекомендаций, то увидите экран, показанный на рис. 9.10, где вы должны предоставить свой ключ подписки на API рекомендаций.

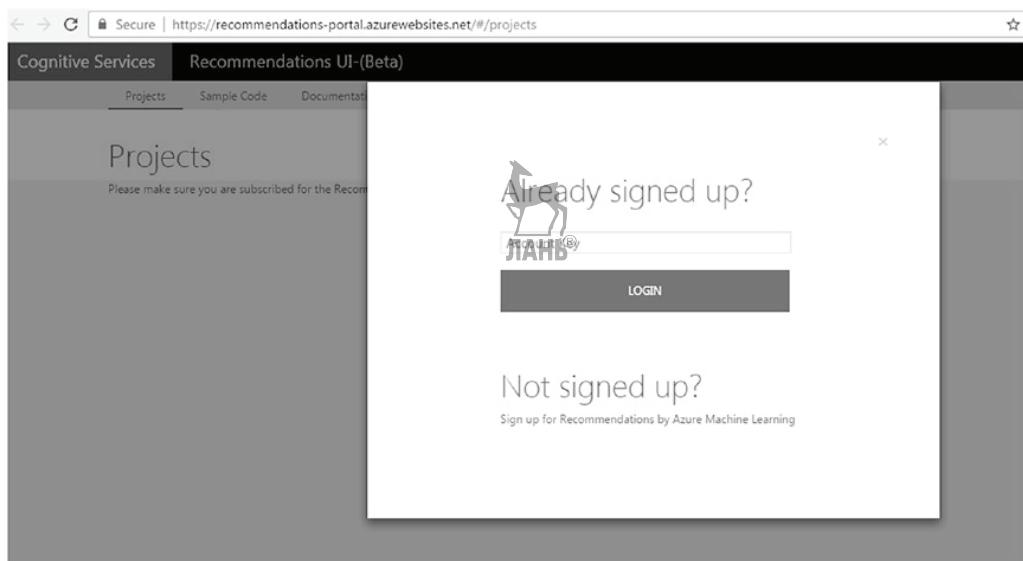


Рис. 9.10 ❖ Введите свой ключ подписки

Введите ключ подписки, после чего вы будете перенаправлены на домашнюю страницу, показанную на рис. 9.11.



Скриншот домашней страницы веб-интерфейса API рекомендаций. В верхней панели навигации есть вкладка "Recommendations UI-(Beta)". В меню слева выбрана вкладка "Projects". Список проектов включает "TestModel" и "BookModel". Кнопка "+ ADD PROJECT" расположена в нижней части списка. В правом верхнем углу видна надпись "Logged in".

Рис. 9.11 ♦ Домашняя страница веб-интерфейса API рекомендаций для авторизованного пользователя

После авторизации будет показан список всех созданных вами проектов вместе с идентификатором модели `modelId`. Для создания нового проекта укажите в текстовом поле его имя и нажмите кнопку **Add Project** (Добавить проект). Проекты в этом веб-интерфейсе являются обычными контейнерами (моделями). Каждый из этих проектов активируется щелчком мыши на проекте. Щелчок мышью на одной из моделей перенаправит вас на страницу сборки, созданной на основе определенной модели. Здесь можно добавить каталогный файл, файл с данными об использовании, увидеть список сборок вместе с их балльными оценками и признаками и создать новую сборку (рис. 9.12).

Скриншот страницы книжной модели веб-интерфейса API рекомендаций. В левой колонке есть разделы "Step 1: Add a catalog file (schema)" (кнопка "BROWSE FOR FILE...") и "Step 2: Add usage files (schema)" (кнопка "BROWSE FOR FILE..."). В правой колонке находится раздел "Step 3: Create a build" с кнопкой "NEW BUILD". Ниже расположена таблица с информацией о сборках:

ACTIVE BUILD	BUILD ID	TYPE	STATUS	DATE CREATED	
1624000	SAR	Success	4/24/2017, 6:27:00 PM	SCORE	
1623950	Rank	Success	4/24/2017, 10:40:17 AM	FEATURES	
1623876	FBT	Error : Some error occurred while executing this build, see appropriate command for further info	4/23/2017, 10:13:37 PM		
1623837	Recommendation	Success	4/23/2017, 10:32:50 AM	SCORE	
1623835	Recommendation	Success	4/23/2017, 10:26:09 AM	SCORE	

Рис. 9.12 ♦ Страница книжной модели веб-интерфейса API рекомендаций

Щелчок мышью на одной из сборок перенаправит вас на страницу с подробной информацией о сборке. На этой странице будут показаны статистика данных, используемые параметры сборки, офлайновые метрические показатели (если они задействованы) и балльная оценка вместе с рекомендацией (рис. 9.13). Чтобы создать клон сборки на основе этих параметров, достаточно нажать кнопку **Make**.

clone (Создать клон), расположенный в нижнем правом углу страницы, в разделе **Build Parameters** (Параметры сборки).



Parameters

Model ID: 6e02116c-977c-4af0-01d0-e7e94f8a8901

SCORE	BUILD PARAMETERS	DATA STATISTICS	OFFLINE METRICS
Build Type	Recommendation		
Description	Simple recommendations build		
Number of Model Iterations	25		
Number of Model Dimensions	20		
Item Cut Off Lower Bound	1		
Item Cut Off Upper Bound	2147483647		
User Cut Off Lower Bound	0		
User Cut Off Upper Bound	0		
Use Features In Model	false		
Modeling Feature List	<input type="checkbox"/> "Color (1)" <input type="checkbox"/> availability (1) <input type="checkbox"/> Size (1)		
Enable Feature Correlation	false		
Reasoning Feature List	<input type="checkbox"/> "Color (1)" <input type="checkbox"/> availability (1) <input type="checkbox"/> Size (1)		
Allow Cold Item Placement	false		
Enable U2I	true		
Enable Modeling Insights	false		
MAKE CLONE			

Рис. 9.13 ❖ Страница с подробной информацией о модели

У вас также есть возможность задать правила для вашей сборки, перейдя на вкладку **Rules** (Правила). Эта вкладка находится на домашней странице модели (рис. 9.14).

Rules

Model ID: 6e02116c-977c-4af0-01d0-e7e94f8a8901

BUILDS	RULES	
Block List BlockList enables you to provide a list of items that you do not want in recommendation results. Example: block_item_id_1.block_item_id_2.block_item_id_3	Upsale Upsale enables you to always include items to return in recommendation results. Example: upsale_item_id_1.upsale_item_id_2.upsale_item_id_3	White List White List enables you to provide list of items where recommendation results must be chosen. Example: whitelist_item_id_1.whitelist_item_id_2
ADD	ADD	ADD
Seed Block List Per Seed Block List enables you to provide per item a list of items that cannot be returned as recommendation results. Example: seed_item_id_1.seed_item_id_2.exclude_item_id_1.exclude_item_id_2	Feature Block List Feature Block List enables you to provide a feature name and a list of feature values that recommendation results must not satisfy. Example: feature_name!feature_value_1,feature_value_2	Feature White List Feature White List enables you to provide a feature name and a list of feature values that recommendation results must satisfy. Example: feature_name!feature_value_1,feature_value_2
ADD	ADD	ADD

Рис. 9.14 ❖ Страница правил, применяемых к определенной сборке

Резюме

В этой главе вы исследовали API рекомендаций и его применение. Вы также изучили некоторые наиболее успешные практические методы использования API рекомендаций. В конце главы было показано, как работать с веб-интерфейсом API рекомендаций. В **следующей главе** вы узнаете, куда движется область когнитивных приложений в ИИ ближайшего будущего.



Глава 10

ИИ в ближайшем будущем



В предыдущих главах мы рассмотрели несколько самых сильных приемов использования API когнитивных служб (Cognitive Services API). В главе 1 вы начали свою экскурсию по миру искусственного интеллекта. В главах 2 и 3 был создан первый проект с элементами искусственного интеллекта, используя Visual Studio в качестве инструмента разработки, а также исследовали разговорный интерфейс пользователя (РИП). Последующие главы помогли вам понять, каким образом ваше приложение сможет глубоко понимать и интерпретировать содержимое Всемирной паутины или машины пользователя, разумно реагировать на прямое взаимодействие с пользователем через устную речь или текст, предоставлять разумные рекомендации в отношении товаров или услуг с учетом каждого отдельного пользователя. Вы также познакомились с построением приложений, которые с опорой на элементы искусственного интеллекта используют язык и взаимодействуют с пользователем, чтобы успешнее конкурировать на сегодняшнем рынке. Разработчики обычно представляют искусственный интеллект как трудноразрешимую задачу, сопряженную с написанием алгоритмов. Одна из целей этой книги состоит в том, чтобы развеять это заблуждение и продемонстрировать, с какой легкостью и с помощью нескольких строк программного кода можно создавать удивительные приложения. К настоящему времени вы должны быть убеждены, что внедрение API когнитивных служб в ваше приложение не представляет труда. Компания Microsoft, несомненно, проделала большую работу, предоставив разработчикам возможность отвлечься от всех нюансов глубокой нейронной сети и многосложных алгоритмов, предложив простые в использовании прикладные программные интерфейсы REST API. Итак, что же дальше? Куда вам следует двигаться далее?

Я уверен, некоторые ваши вопросы о том, как и когда использовать эти API, решены. Но теперь у вас могут возникнуть новые вопросы. Что будет происходить дальше? Разве искусственный интеллект касается только использования когнитивных служб с применением механизма RESTful? Имеются ли перспективы, когда все эти когнитивные приложения можно будет интегрировать с устройствами, сделав в результате эти устройства умнее? Что насчет рабочих мест? Вы, возможно, слышали, что ИИ собирается отобрать X% всех рабочих мест в течение следующих нескольких лет или около того. Какие проблемы существуют в области когнитивных вычислений? Эта глава попытается развеять некоторые животрепещущие заблуждения и будет выступать в качестве руководства, помогающего вам принимать решения, значительно опережая время. В конце этой главы у вас будет ясное понимание о:

- неправильных представлениях в сравнении с фактической стороной искусственного интеллекта;
- рисках и проблемах, лежащих в основе когнитивных технологий;
- концепции компании Microsoft в отношении ИИ;
- шагах в сторону искусственного интеллекта.

Почему ИИ имеет такую популярность?

В предыдущих главах мы вскользь показали, почему искусственный интеллект имеет прорывной характер. Для нас важно упорядочить свои представления в единую систему, чтобы предвидеть будущие инновации и направления. **Прорыв в области когнитивных вычислений произошел в 2011 г. матчем в игру Jeopardy! между компьютером Watson и человеком.** Для многих этот матч стал потрясением. Большинство из нас впервые увидели силу когнитивных вычислений. С тех пор в этой сфере произошло много инноваций по всему миру и продолжает происходить каждый день и каждый час. Интересно наблюдать за быстрым распространением когнитивных вычислений благодаря их результатам. Некоторые последствия когнитивных вычислений привели к созданию программных продуктов и информационных технологий, которые помогают нам делать то, что мы не могли делать прежде.

- i** Популярность ИИ в наше время всегда напоминает мне одну цитату из романа Эрнеста Хемингуэя «И солнце восходит»:
- Как ты обанкротился? – спросил Билл.
 - Двумя способами, – ответил Майк. – Постепенно и затем внезапно.

Искусственный интеллект, конечно же, оказывает на нас революционное воздействие. Он пересек черту, перейдя от постепенности к внезапности. ИИ не является новой концепцией. Глава 1 была посвящена краткому рассмотрению истории искусственного интеллекта. Прежде чем мы попытаемся разрушить неверные представления о проблемах, связанных с ИИ, и рассмотрим ближайшее будущее этого направления, разберемся, почему ИИ имеет такую бешенную популярность. А эта популярность прежде всего обусловлена следующими причинами:

- увеличение вычислительной мощности;
- усовершенствование алгоритмов ИИ;
- большое количество данных;
- Интернет и облачные вычисления.

Рост вычислительной мощности

Огромный рост вычислительной мощности был одним из критических факторов, который популяризовал ИИ. Увеличение вычислительной мощности помогает быстро обрабатывать большие объемы данных.

- i** Вычисления, требовавшие в конце 1980-х годов месяцев обработки, сейчас выполняются в течение нескольких секунд, даже на дешевых компьютерах. Вычислительная мощность наших мобильных телефонов низкого ценового сектора намного больше суперкомпьютеров 1980-х.

Именно так. Появление графических процессоров (GPU), тензорных процессоров Google (TPU) и программируемых пользователем вентильных матриц (FPGA) показывает нам, что мы прошли эру закона Гордона Мура. В дополнение к этой вычислительной мощности понизилась стоимость хранения данных. По оценочным данным, к 2020-му или чуть ранее 1 Гб хранения будет стоить менее \$0.001. В дополнение к снижающейся ежедневно стоимости хранения инновации в области производства кристаллов микросхем предлагают все больше и больше вычислительной мощности по низкой цене. Кроме этого, происходят огромные инновации в области новой структуры микросхем. Появление нейроморфных чипов феноменально изменило вычислительную мощность. Нейроморфные чипы не только обеспечивают повышенное в миллион раз быстродействие, но и экономят потребление энергии.

- i** Триумф компьютера Watson над чемпионами игры Jeopardy стал возможным благодаря использованию нативного процессора TrueNorth. Процессор TrueNorth основан на нейроморфных чипах, которые имитируют работу мозга. Чипы TrueNorth имеют один миллион компьютерных «нейронов», которые работают параллельно с миллионами межнейронных связей, что делает обработку быстрее, чем когда-либо.

ОТКРЫТИЯ В АЛГОРИТМАХ ИИ

Прошлое десятилетие стало феноменальным с точки зрения инноваций в алгоритмах ИИ. Вернувшись к главе 1, вы узнаете о распространении алгоритмов ИИ и их успешном внедрении. Ключевые инновации в области машинного обучения, в особенности в областях глубокого обучения, мелкого (неглубокого) обучения и нейронных сетей, уже успешно решили некоторые сложные задачи. Например, благодаря глубокому обучению было отмечено сокращение более чем на 25% коэффициента словарных ошибок в распознавании речи и машинном переводе. Одно из изменений в области ИИ, которое случилось в течение прошедшего десятилетия, состояло в наделении машин способностью обучаться на основе исторических данных в противовес программированию. Этот фундаментальный подход принес целый ряд революционных изменений в ИИ. Вследствие недавних изобретений в области алгоритмов ИИ мы движемся в сторону эпохи разговорного интерфейса пользователя. Распространение персональных помощников приобрело повсеместный характер. В частности, Сири, Кортана и Google Now – вот лишь некоторые из побочных изобретений в сфере алгоритмов ИИ.

- i** Amazon Алекса и Google Home дали начало распространению разговорных интерфейсов пользователя (РИП). Кроме того, компания Cubic является одной из революционных компаний, специализирующейся на разговорной платформе с одноименным названием, которая способна общаться с пользователем и интегрироваться с любым сетевым устройством. Например, Cubic можно использовать для включения света до возвращения домой или для получения совета о том, как быстрее добраться до вашего места назначения, или же использовать для заказа доставки любого товара.

Мы стали свидетелями эры, когда управляемые искусственным интеллектом машины побеждали чемпионов, начиная с чемпионов по шахматам и заканчивая чемпионами по игре в го. Совсем недавно компьютер Libratus, разработанный ко-

мандой из Университета Карнеги-Меллона, победил чемпионов по покеру. Покер, как вы знаете, требует скрывать определенную информацию до определенного момента, и создание модели, которая могла бы справляться с такими ситуациями, представляет большую сложность.

ДАННЫЕ – ЭТО НОВАЯ ВАЛЮТА

Мы не можем использовать вычислительную мощность и только одни алгоритмы ИИ. Как было отмечено ранее, машинное обучение использует исторические данные, чтобы идентифицировать шаблоны (образы). Чем больше объем данных, тем легче идентификация шаблонов.

i Большинство самоуправляемых автомобилей генерирует приблизительно 100 гигабайт данных в секунду. Это ли не показатель огромных объемов данных? На самом деле каждый день создается 2.5 квинтиллиона байт данных. Прошедшие два года произвели 90% данных, произведенных за все время. По оценочным данным, к 2020 году 25 миллиардов таких устройств будут подключены к сети. 80% из этих данных будут неструктурированными, а это означает, что они не смогут обрабатываться обычными компьютерными системами. Рассмотрим еще один пример огромного объема данных: здравоохранение. Каждый человек производит 1 миллион гигабайт медицинских данных. Некоторые компании, в частности IBM и Enlitic, уже используют инструменты глубокого обучения и эти данные для идентификации шаблонов в медицинских данных, повышая эффективность работы врача, а также выполняя предсказания здоровья.

В условиях, когда почти каждый день появляются новые устройства, эти данные должны правильно использоваться всей мощью алгоритмов ИИ и вычислительных возможностей, чтобы выполнять самые разные виды работ. Рисунок 10.1 показывает некоторые устройства, которые могут генерировать данные.

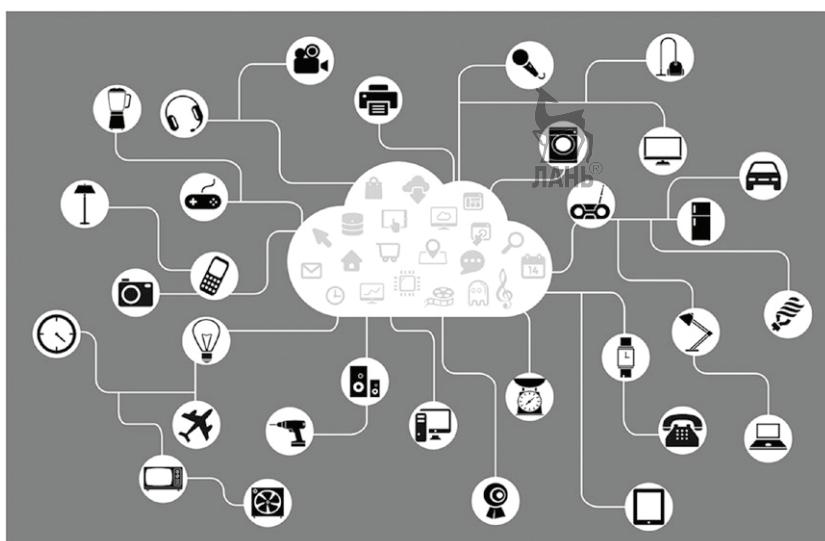


Рис. 10.1 ♦ Некоторые устройства, в которых могут быть собраны и использованы данные для решения сложной бизнес-задачи

Распространение технологии интернета вещей и больших данных также во многом поспособствовало развитию алгоритмов ИИ. По сути дела, когда данных было недостаточно, основную роль в решении задачи играл человек. В последние годы мы стали свидетелями усовершенствований в самых разных областях. Особено там, где много текстовых данных, изображений и речи. Благодаря применению технологий глубокого обучения исследователи в области ИИ заменили человеческий интеллект этими данными, в результате чего обеспечивается повышение точности. Даже сейчас человеческий интеллект все еще используется там, где предметная область не располагает достаточно большим объемом данных.

Некоторые из вас могут подумать, что в будущем когнитивные вычисления заменят человеческий мозг. Надо признать, что существует несколько проектов, таких как Project Nengo, в рамках которого предпринимаются попытки создать и смоделировать человеческий разум путем объединения крупномасштабной нейронной системы. Однако большая часть исследований в области ИИ ориентирована на создание системы, которая способна анализировать и обрабатывать данные, с которыми человеческий разум справиться не способен. Если вы по-настоящему следите за появлением платформ ИИ, таких как Watson, то можете заметить, что они были разработаны для придания системе способностиправляться с огромными объемами генерированных данных. Большая часть исследований была сосредоточена вокруг создания новых алгоритмов, систем и чипов, позволяющих обрабатывать огромный объем массива данных. Эти данные способны создавать новое понимание и разрешать сложные задачи. Возьмем данные из области здравоохранения. Если мы сможем анализировать 1 миллион гигабайт медицинских данных в расчете на одного человека, то мы, по-видимому, сможем получать более глубокую информацию и, как следствие, понимание того, как сделать человека здоровым всегда. Без всякого сомнения, данные становятся столь же ценными, что и нефть, и, по сути, новой валютой организации. **Организация, которая имеет данные и способна генерировать новое понимание решаемой задачи, станет лидером в будущем.** Теперь вы знаете, почему компания Google приобрела компанию YouTube (данные в форме видео). Теперь вы, вероятно, знаете, почему компания Facebook приобрела WhatsApp. Согласно количественным показателям приобретения компаний за прошедшие пять лет, большинство приобретенных компаний связано с когнитивными вычислениями и ИИ либо является компаниями, обладающими данными и способностью генерировать новое понимание решаемой задачи.

ВОЗНИКНОВЕНИЕ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

Распространение Интернета и облачных вычислений – это четвертая опора для формирования искусственного интеллекта. После своего появления предложение продукта должно достигнуть широких масс и стать доступным почти на каждом устройстве. Это стало реальностью с распространением Интернета и облачных вычислений. Облачные вычисления обеспечивают не только охват больших масс, но и уникальную возможность обрабатывать сложную вычислительную задачу абстрактным образом. К примеру, возьмем API когнитивных служб Microsoft. Тысячи и тысячи серверов вычислительного облака Microsoft Azure лежат

в основе обработки каждого вашего запроса к когнитивному API. Рисунок 10.2 показывает, как облачные вычисления помогают продуктам достигать каждого устройства.

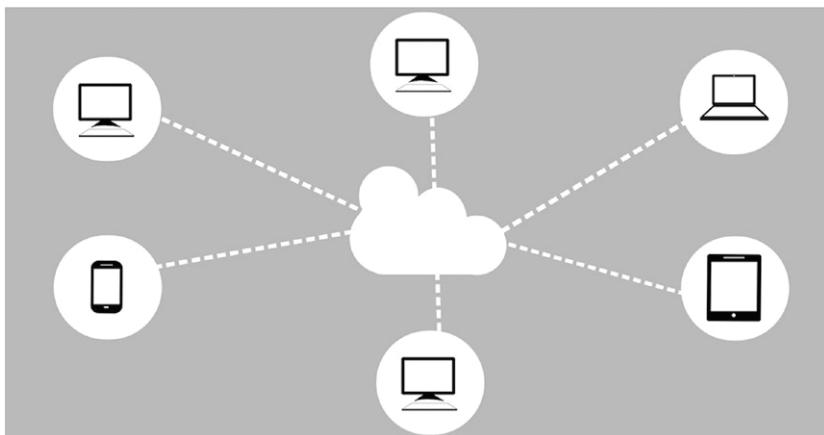


Рис. 10.2 ❖ Простая диаграмма использования облачных вычислений для взаимодействия с устройствами

Службы против решений?

Когнитивное пространство Microsoft продолжает расти устойчивыми темпами. То, что в 2016 г. начиналось с четырех когнитивных API, во время написания этой книги выросло до 29 API. Компания Microsoft предлагает комплекты программных продуктов. Кроме того, она также имеет комплект с облачно-ориентированными возможностями. Вы можете поинтересоваться, зачем в компании Microsoft создали эти службы, вместо того чтобы создать конкретные решения и программные продукты. Ответ прост. Последние несколько лет стратегия Microsoft состояла в концентрации всей своей деятельности не в создании систем, а вокруг создания платформы. Эта платформа и службы впоследствии могут использоваться другими для решения проблемно-ориентированных задач. Данная платформа предназначена, чтобы масштабироваться путем создания со временем все большего количества предложений в когнитивном пространстве и предоставления конечным пользователям опыта их использования с эффектом присутствия. Не удивляйтесь, если меньше чем через год список когнитивных API вырастет до 50.

Когнитивные категории

Давайте теперь попытаемся разобраться, какие проблемы существуют сейчас в когнитивном пространстве. Эти проблемы не только говорят о существующих преобладающих ограничениях, но и указывают на направления исследований в каждой из этих когнитивных областей. Рисунок 10.3 показывает обобщенную классификацию когнитивных API. Почти все выпущенные компанией Microsoft API могут быть отнесены к одной из этих групп.



Рис. 10.3 ♦ Различные категории когнитивных API

Когнитивность – это процесс представления информации машине и использования этого представления для автоматического рассуждения. Наблюдательность – это процесс имитации поведения человека как взаимодействие с помощью речи, текста или зрения, как это делают люди. Активное обучение – это процесс автоматического самосовершенствования с течением времени. Классическим примером активного обучения является интеллектуальная служба понимания языка Microsoft Language Understanding Intelligent Service (LUIS), которую мы подробно рассмотрели в главе 6. Физическое действие требует объединения этих трех категорий и использования устройств для разумного взаимодействия.

ПРОБЛЕМЫ И БУДУЩЕЕ ПОНИМАНИЯ ЕЯ

Мы подробно обсудили тему понимания естественного языка в главе 4. Обработка и понимание естественного языка могут использоваться во множестве сценариев, таких как извлечение, выделение и анализ контекста и смысла предложения. Существует целый ряд сценариев, в частности медицинский, академический или юридический, где каждый день генерируется большой объем текстового содержимого. Эти данные среди прочих могут поступать в результате, например, научных исследований, судебных разбирательств и медицинской диагностики. Механизм понимания естественного языка может помочь вам обрабатывать текст, но одну из основных проблем представляет понимание правильного контекста. Как подробно говорилось ранее, можно понять синтаксис (грамматику) и семантику (значение), но глубокое понимание прагматики (контекста) представляет настоящую проблему. Если вы знакомы с механизмами понимания естественного языка, то вы знаете, что очень большие усилия направляются на тренировку этих механизмов правильно понимать контекст. Смогут ли в будущем алгоритмы понимать контекстную компоненту? В этой области проводятся широкие исследования. Некоторые компании, специализирующиеся на ИИ, в частности Pat.ai, похоже, имеют многообещающие результаты, потому что они сосредоточены на сопоставлении значений, а не на статистическом анализе. В будущем можно ожидать появления различных подключенных к Интернету устройств с использованием технологии понимания ЕЯ, способных понимать контекст и затем интегрироваться с такими компонентами, как речь и поиск с целью обеспечения наилучшего взаимодействия с эффектом присутствия.

ПРОБЛЕМЫ И БУДУЩЕЕ ОБРАБОТКИ РЕЧИ

Как отмечалось в главе 7, инновации в обработке речи проделали долгий путь. Сейчас большой интерес вызывают работы, связанные с естественным взаимодействием с помощью речи. Естественное взаимодействие сегодня в особенности сосредоточено вокруг автомобилей и персональных помощников. Исследованиям в этой области еще предстоит пройти долгий путь, чтобы обеспечить применение естественного взаимодействия со 100%-й надежностью. Возьмем, например, автоответчик. При наборе 1 предоставляется информация об остатке, при наборе 2 производится оплата счета и т. д. Эти автоматизированные агенты временами так расстраивают, что возникает желание обратиться к человеку. Разве было бы не хорошо, если бы автоответчик был более интерактивным и задавал вам вопросы типа «Скажите мне, что вы хотите сегодня сделать». На основе ответа он мог бы предоставить фактическую помощь. Такие речевые взаимодействия оказывали бы пользователю реальную поддержку. Распознавание речи должно характеризоваться незаметным взаимодействием с пользователем. Помимо естественных взаимодействий, обработка речи в настоящее время сталкивается с большим количеством других проблем, таких как:

- точность;
- результативность;
- реагирование на пользователя;
- грамматические противоречия;
- фоновые шумы;
- разнообразные акценты.

Возьмем, например, точность. В автомобилях точность распознавания может падать из-за плохого качества звукового сигнала. Таким же образом ваш агент может не быть готовым слушать, когда пользователь начал говорить, что в итоге приведет к плохому реагированию на запрос пользователя. Распознавание речи также сталкивается с трудностями, связанными с результативностью и быстродействием. Одна из проблем естественного взаимодействия связана со способностью технологии распознавания речи, понимать и работать в темпе человеческой речи. До известной степени в этой области были достигнуты определенные результаты, но тем не менее еще предстоит разработать новые алгоритмы с правильным темпом, которые обеспечат непрерывный опыт взаимодействия в режиме реального времени. Точность также сопряжена с произношением слов, которые звучат одинаково. Например, для механизма распознавания речи слова «bet» и «bed» похожи.

Большинство титанов информационной индустрии в области обработки речи, таких как IBM, Google и Microsoft, сосредоточено на усовершенствовании точности речи и уменьшении коэффициента словарных ошибок (word error rate). Совсем недавно компания Google сообщила, что им удалось свести свой коэффициент словарных ошибок к 4,9%, что означает, что они способны правильно предсказывать 19 слов из 20. Это большой успех, но, чтобы сделать распознавание речи надежнее и точнее, в будущем коэффициент словарных ошибок должен свестись менее чем к 1%. Очень важно также иметь систему, в которой обработка естественного языка и распознавание речи работают совместно. Модуль обработки ЕЯ может брать на себя работу по извлечению смысла из сказанного и на основе контекста сообщать

модулю распознавания речи правильное слово. Помимо этих сценариев, есть другие области, где обработка речи по-прежнему сталкивается со многими проблемами. Технология распознавания речи должна быть достаточно рациональной, чтобы справляться с условиями внешней среды, имеющей большое количество фонового шума. Microsoft предложила службу персонализированной обработки речи (подробно рассмотренную в главе 7), которая до некоторой степени пытается решить эту проблему. Эта область является совершенно новой и, конечно, должна быть тщательно протестирована в реальной производственной среде.

Большое количество компаний также пытается использовать речь в области автоматического распознавания речи (APP). Возьмем, к примеру, Google, использующий YouTube для выполнения APP вместе с переводом. Можно возлагать вину на различные акценты или на то, что веб является многоязычной средой, но так или иначе нам еще предстоит увидеть APP, обладающий 100%-й точностью. Однако можно с определенностью сказать, что уже довольно скоро мы станем свидетелями появления надежного машинного перевода, работающего в реальном времени.

ПРОБЛЕМЫ И БУДУЩЕЕ ПОИСКА

API Bing поиска, разработанный компанией Microsoft, привнес в комплект API большое количество функциональных возможностей. Глава 8 подробно раскрыла особенности API Bing поиска и предлагаемых им возможностей. Вы не только можете выполнять поиск в Интернете, но и отыскивать новости, видео и т. д. Теперь поиск также может поставлять локализованные результаты. В будущем поиск станет более контекстным. В будущем вы сможете отыскивать домашние устройства так же, как вы отыскиваете любое изделие в вебе. Это будет феноминальным опытом. Представьте, что вы ищете свой кошелек по мобильному телефону, как показано на рис. 10.4. Чтобы достигнуть этого, поиск должен быть интегрирован с другими когнитивными технологиями.

ПРОБЛЕМЫ И БУДУЩЕЕ РЕКОМЕНДАЦИЙ

Microsoft и другие компании уже вкладывают большие средства и усилия в рекомендательные механизмы и системы. Глава 9 продемонстрировала некоторые преимущества и результаты использования рекомендаций. Для некоторых лидирующих компаний рекомендации играют ключевую роль. Сегодня большинство рекомендаций опирается на методы, которые учитывают товары, часто совместно приобретаемые (FBT), навигацию пользователя по сайту или предыдущую историю. А теперь обратимся к данным этого же клиента из социальных медиа. Порядка 500 миллионов твитов и 55 миллионов обновлений ленты Facebook происходят каждый день. В среднем человек проводит в Интернете пять или более часов каждый день, просматривая кинофильмы и телешоу по телеви-

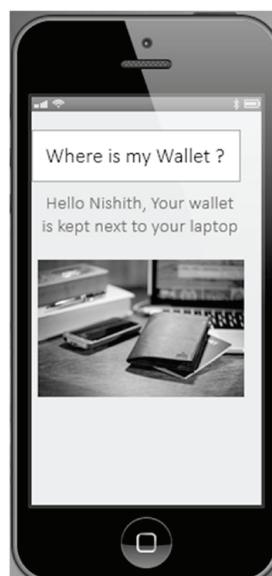


Рис. 10.4 ♦ Поиск кошелька по телефону

дению или телефону. В будущем рекомендации будут опираться на информацию, извлеченную из социальных сетей, каналов, которые вы смотрите по телевизору, и разговоров, которые вы ведете по телефону.

ПРЕЖДЕ ВСЕГО ИИ

Вам наверняка знакомы термины «*прежде всего облако*» и «*прежде всего мобильность*». В начале внедрения эта технология получает приоритетное внимание. В прошлом ИТ-индустрия претерпевала множество трансформаций, как, например, клиент-серверная архитектура, веб, распределенные вычисления, облачные вычисления, и вот теперь ИИ. Вместе с тем ни одно из этих направлений не получило столь же интенсивного развития, как в настоящее время происходит с ИИ. Не имеет значения, какую роль вы играете в своей организации или есть ли у вас свое собственное дело или стартап, очевидно одно: ИИ не только будет влиять на выполняемую нами работу, но и окажет сильное воздействие на нашу повседневную жизнь в целом. Если оглядеться вокруг, то можно увидеть, что любая ИТ-компания сосредоточивается на ИИ. Такие вещи, как программы-роботы, персональные помощники, разговорные интерфейсы и машинное обучение, – это некоторые из тех слов, находящихся у всех на слуху, которые вы встретите на домашних страницах большинства компаний. Для этих компаний важно не только сконцентрировать свои усилия на предложении новых услуг в области ИИ, но и на своих людях и своих существующих продуктах. Причина, почему важно сосредоточиться на людях, – прежде всего в том, что человек без машины или машина без человека не в состоянии произвести радикальные изменения. Организации должны подумать, чем занять людей, если их задачи выполняют машины. **Компании, которые сосредоточивают свои усилия на инновациях в области ИИ, на своих людях и продуктах, неизбежно будут процветать в будущем.** Это позволит компаниям использовать свое конкурентное преимущество, которое даст возможность двигаться дальше. Вы не только должны думать, как с помощью искусственного интеллекта изменить свой бизнес, но и думать об использовании ИИ и машинного обучения людьми как основы своей бизнес-стратегии.

ИНТЕЛЛЕКТУАЛЬНАЯ ПЕРИФЕРИЯ

До сих пор мы пытались использовать API-службы при помощи механизма REST. Эти API, безусловно, помогли нам решить некоторые сложные задачи, которые невозможно было решить прежде. Еще одна причина этой API-ориентированной модели состоит в том, что она отвлекается от тысячи процессоров и виртуальных машин, работающих на фоне API. С продвижением технологии интернета вещей и сенсорной техники появляются невероятные возможности для автоматизации и анализа данных. Вспомните о персональных помощниках типа Google Now или Кортана. Перед тем как произойдет фактическая обработка, все запросы направляются в облако. Причина передачи этих запросов в облако – в том, что мобильное приложение не имеет достаточной вычислительной мощности по обработке сложных вычислений. Передача вызовов API в обе стороны, безусловно, требует времени. Эта модель в большинстве ситуаций работает, но случается, когда не удается довести поставленную задачу до конца. Поэтому существует необхо-

димость в развертывании этих моделей на периферии, а не на облаке. Сегодня в этом направлении проводится много исследований. Помимо нейроморфного чипа, о котором в книге упоминалось ранее, многие компании проектируют чипы, с которыми вычислительная мощность устройства будет приближена к вычислительной мощности облака. Недавно компания Intel выпустила чип под названием Joule platform (платформа Джоуля). Этот чип позволяет разворачивать сложные модели и может применяться в большинстве устройств, таких как устройства с подключением к Интернету, роботы и дроны. Модернизация процессора для мобильных устройств идет почти ежедневно. И совершенно очевидно, что недалек тот день, когда машинно обучаемая модель будет непосредственно развернута на мобильном телефоне. Генеральный директор Google Сундар Пичаи во время Саммита Google I/O в мае 2017-го показал новый чип Google, который в конечном счете может использоваться для тренировки и работы глубоких нейронных сетей. Генеральный директор Microsoft Сатья Наделла также объявил об интеллектуальном решении для периферии на основе различных проектов, таких как проект Рим (Project Rome) и другие. Компания Apple также объявила о конструировании вторичного мобильного процессора для приведения в действие искусственного интеллекта. Очень скоро компания Microsoft, по-видимому, объявит о новой центральной модели для использования API на периферийных устройствах. В будущем большое внимание может привлечь Microsoft Graph, поскольку под эгидой ее конечной точки планируется объединить все устройства и приложения путем создания так называемой текстурной модели (fabric model) надежных веб-служб. Эта модель откроет широкий набор возможностей для решения задач, и мы сможем использовать эти технологии там, где интернет-соединение остается медленным, в частности в густых лесах Африки, или в самой отдаленной деревне в Индии, или под водой либо в самой глубокой части океана.

УСТРАНЕНИЕ ЗАДАЧ, А НЕ РАБОЧИХ МЕСТ

Прошли те времена, когда машины использовались исключительно для повторяющейся, монотонной и итеративной работы. С каждым днем машины становятся все умнее. Вы уже видели некоторые примеры, где машины с поддержкой ИИ побеждали людей-чемпионов по разным играм. Разумеется, ИИ не ограничивается только играми; на самом деле ИИ приносит революционные изменения во все предметные области. Одной из любопытных и животрепещущих дискуссий в Сети является вопрос, как ИИ повлияет на рабочие места. Взгляните на рис. 10.5. Некоторые эксперты в области ИИ полагают, что очень скоро половина рабочих мест будет замещена искусственным интеллектом. Вы обеспокоены?

Подумайте вот о чем: люди консервативны. Любая новая вещь всегда встречается в штыки. Вспомните, как человек впервые планировал путешествие на самолете. Был страх. Возьмем более простой пример. Что было, когда появились автомобили и мобильные телефоны? Конечно, до определенной степени существовал страх, но мы должны признать, что эти технологии принесли большую пользу. Первоначально эти технологии принимались с подозрительностью, но сегодня они стали частью нашей жизни. Точно так же мы должны приветствовать и те изменения, которые несет с собой ИИ. Этих изменений не стоит бояться. Они коснутся не только нашей работы, но и нашей повседневной жизни. Некоторые зада-

чи, вне всякого сомнения, будут автоматизированы, и, конечно же, преимущества проявятся в краткосрочной и среднесрочной перспективе. Машины всегда помогали людям выполнять однообразную работу. На рис. 10.6 показан робот, везущий тележку.



Рис. 10.5 ♦ Заменит ли когда-нибудь робот человеческого ди-джея?



Рис. 10.6 ♦ Робот, толкающий тележку

Машинное обучение является одним из ключевых компонентов интеллектуально-ориентированного приложения. То, как приложение с элементами искусственного интеллекта отвечает, зависит от того, как хорошо оно натренировано. Подумайте о самоуправляемом автомобиле. Мы должны натренировать автомо-

биль ездить по дороге, но мы также должны обучить систему понятию, что такое плохое и хорошее вождение. Как хорошо будет ездить самоуправляемый автомобиль, зависит от того, как человек сумеет натренировать машину. В некоторой степени это один из классических примеров соединения машины и человеческих ценностей. В мире ИИ это также носит название «усиленного интеллекта» (УИ, augmented intelligence).

i Возьмем еще один пример – обслуживание клиентов. Скажем, клиент звонит в службу поддержки клиентов, чтобы решить свой вопрос. Большинство вопросов элементарно, но некоторые сложны по своей природе. Компания наняла сотни людей в качестве персонала службы поддержки, чтобы отвечать на эти элементарные и сложные вопросы. Сколько времени, по вашему мнению, сотрудник персонала службы поддержки будет мотивированно, энергично и с интересом отвечать на одни и те же вопросы? Без сомнения, никто не любит делать монотонную работу продолжительное время. Различные статистические исследования показали, что производительность сотрудника с течением времени постепенно снижается, если он выполняет однообразную работу. Кроме того, появляется высокая вероятность, что персонал службы поддержки начнет совершать ошибки при даче правильного ответа. Возьмем тот же самый сценарий, где теперь элементарные вопросы направляются к интеллектуальным ботам. Если боты не в состоянии ответить на вопрос, то этот вопрос будет перенаправлен человеку. Сотни людей из службы поддержки, которые отвечали на элементарные вопросы, смогут заниматься тренировкой бота, чтобы сделать его эффективнее. А сами люди будут задействованы на других, совершенно новых направлениях, которые появятся благодаря внедрению этих ИИ-ориентированных ботов.

Вышеупомянутый пример представляет собой классический пример, применяемый в компаниях. Как видно из приведенного выше сценария, боты взяли на себя задачи людей. Боты занимаются ответами на элементарные вопросы службы поддержки клиентов, а персонал занят на новых рабочих местах, таких как тренировка, развертывание и т. д. Внедряя ИИ надлежащим образом, вне всякого сомнения, можно извлекать выгоду в любой предметной области. Это также предоставляет компании уникальную возможность идентифицировать новые перспективы, где могут использоваться сотрудники.

i Возьмем еще один пример. Он связан с Университетом штата Аризона, который сотрудничал с компанией Knewton and Cengage Learning для предоставления студентам персонализированного учебного плана. При помощи предсказательной аналитики они привлекали преподавателей, когда студентам они действительно были необходимы. Начальные экспериментальные результаты были феноменальными – процент успешно сдававших экзамены студентов повысился с 64 до 75.

Если выделить главное, введение ИИ в технологический процесс принесло большую выгоду. При этом, конечно, не стало многих задач, но появились новые задачи. Конечно, разрыв между долей старых задач, которые исчезли, и вновь созданными задачами может быть большим. Но обязанность, чтобы не потерять конкурентные преимущества, стремительно развиваться и создавать новые рабочие места лежит на организациях.

i Рассмотрим случай с компанией Uber, которая полностью изменила рынок такси. Внедрение новой технологии с использованием ИИ и персонального опыта с эффектом присутствия привело к тому, что Uber, работая более чем в 600 городах, является крупнейшей службой такси, даже не владея автотранспортными средствами. Так же как и Uber, компания Airbnb

предоставляет самый широкий набор услуг по предоставлению жилья по всему миру и при этом не владеет ни одним помещением. Она применяет ИИ для динамического формирования цен на комнаты на основе спроса в режиме реального времени. В командах поддержки клиентов работают сотни людей, которые помогают управлять процессами Uber и Airbnb в непрерывном режиме.

Технологии прошлого незаметно устранили многие рабочие места. Как отмечалось ранее, рабочие места, в которых требуется от работника интеллект, останутся и впредь и никогда не будут заменены искусственным интеллектом. Например, можно ли представить, что в будущем робот сможет заменить знаменитого Роэна Аткинсона в «Мистере Бине» даже случайно? Едва ли. Если проследить за развитием процесса, то все началось с автоматизации задач производственных рабочих (синих воротничков), и теперь настала очередь для некоторых задач офисного персонала (белых воротничков). Для всех нас очень важно поддерживать свой образовательный уровень и быть готовыми к этим изменениям. Регулярная практика и повышение квалификации являются сегодня насущной необходимостью. Компании и люди будут обязаны приспосабливаться, если события в эру цифровой революции потребуют перестройки.

 Цифровая революция идет ~~повсюду~~. С 2000 г. более половины из списка 500 крупнейших промышленных компаний США перестали существовать. Вместе с тем еще 10–20 лет назад многих лучших глобальных компаний, которые сегодня являются лидерами в своих отраслях, не было и в помине. В ближайшие годы успешными будут только те компании, которые сосредоточены на машинах и людях.

ТАК КУДА ЖЕ МЫ КАТИМСЯ?

В последующие несколько лет мы станем свидетелями того, как ИИ начнет применяться в самых разных сферах жизни. ИИ стал движущей силой четвертой промышленной революции. Влияние ИИ началось, и в ближайшие годы ИИ, безусловно, начнет преобразовывать нашу повседневную жизнь в массовом порядке. От нас, разработчиков, требуется найти правильные варианты применения и, конечно же, приблизить мир к усиленному интеллекту (УИ) в противоположность искусственному интеллекту (ИИ). Вообразите мир, где большинство автомобилей управляет машинами, где на большинстве фабрик роботов больше, чем людей. Представьте розничный магазин, где вас встречает робот, который понимает ваш вопрос и предлагает клиентоцентричный покупательский опыт, или больницу, где первичный анализ выполняется роботами. Представьте холодильник, который достаточно умен, чтобы заказывать овощи, когда корзина будет пуста, и подумайте о мире, в котором ваши посылки доставляются беспилотниками. Добро пожаловать в антиутопический мир ИИ, который наступит не позднее 2020 г. Возможно, в последующие 15 лет вы увидим машины, способные самовоспроизводиться на основе ИИ и 3D-печати. Возможно, мы также увидим, как роботы будут играть в теннис и побеждать чемпионов мира. И даже увидим, как ИИ-ориентированные роботы будут проводить медицинские операции. Однако мы пока еще не знаем, сможем ли мы когда-либо произвести машину, которая будет обладать силой интуиции и моральных ценностей, как мы.

Люди постоянно спрашивают, что они должны предпринять и как ориентироваться в революционную эру искусственного интеллекта. Искусственный интел-

лект уже начал эту эру в производственной сфере (см. рис. 10.7). И он находится в процессе революционного преобразования офисных рабочих мест. Нам пока неизвестно, насколько революционным он будет, но совершенно очевидно, что совсем скоро станет востребованным самоорганизующийся усиленный интеллект. В этой главе мы попытались ответить на некоторые из этих вопросов, и ее содержимое напомнило мне об известной цитате Франциска Ассизского:

i «Начните делать необходимое, затем возможное, и внезапно вы обнаружите, что совершаете невозможное».

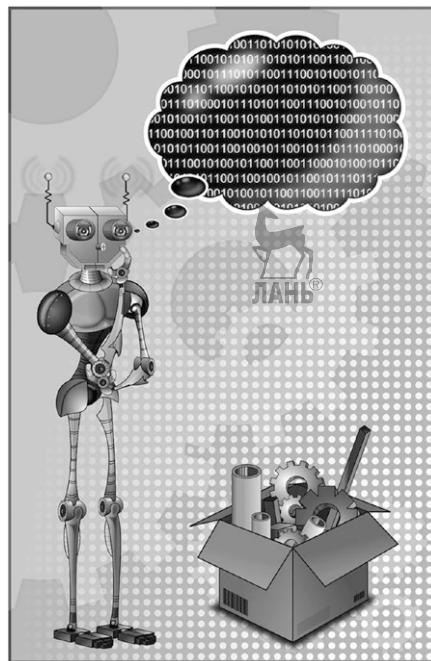


Рис. 10.7 ❖ Робот подводит итог, прежде чем выполнить ручную работу



Тематический указатель

A

Accuracy Testing, флагок, 207
Allow-Control-Allow-Origin, расширение, 118
Amazon, 26, 224, 251
API Bing поиска
 поиск видео, 242
 поиск новостей, 239
API Bing проверки орфографии, 120
API лингвистического анализа, 148
 разбиение на предложения и лексемизация, 149
 синтаксический анализ на основе грамматики составляющих, 148, 154
 частеречная разметка, 148, 152
API обработки речи
 API Bing обработки речи, 192
 API когнитивного поиска, 191
 getValidToken, метод, 193
 Microsoft Speech, 198
 REST API, 191
 RestSharp, 196
 SSML, 198
 Visual Studio 2017, 190
 акустическое моделирование, 198
 грамматика, 198
 когнитивные службы, 191
 конечная точка обработки речи, 194
 модель языка, 198
 облачный сервер Azure, 192
 пакет Bing NuGet, 190
 пакет NuGet, 191
 распознавание речи, 191, 195
 синтез речи, 195
 словарь, 198
 фонемы, 198
Apple Macintosh, компьютер, 73

C

convertTextToSpeech, метод, 197
CreateReply, метод, 87

D

DescribeImage, метод, 59

DoctorType, сущность, 168

DrCheckup, проект, 179

E

ELIZA, компьютерная программа, 99
ExtractText, метод, 64, 66

F

Facebook, 26

G

GetCondition, намерение, 169, 184
GetDefinition, намерение, 108
getValidToken, метод, 193
Google, 23, 223
Google AdSense, 134

H

HealthCheckupDialog, 180

J

JWT-токен (JSON Web Token), 192

M

macOS, 72
MessagesController, класс, 85, 93
Microsoft, 27, 141
Microsoft Azure, облако, 26, 282
Microsoft Bot Framework, инфраструктура, 82
Microsoft Cognitive Client Service, когнитивная клиенская служба Microsoft, 214
Microsoft Cognitive Services, когнитивные службы Microsoft, 45
Microsoft Cognitive Speech API, API когнитивной обработки речи, 212
Microsoft Conversational Model, разговорная модель Microsoft, 205, 213
Microsoft Custom Speech Services, персонализированные службы Microsoft обработки речи, 199
Microsoft Search и Dictation Model, 205, 213
MS-DOS, 72
MSN Поиск, 226

N

Netflix, 251
Nokia S60, мобильный телефон, 74
NuGet, пакет, 84, 190

O

Oxford speech, проект, 191

P

ParseQueryString, функция, 61

R

REST API, механизм, 39
RestSharp, библиотека, 196

S

Smart Adaptive Recommendation (SAR), 260, 269
Speech Synthesis Markup Language (SSML), язык разметки синтеза речи, 195

T

Twitter, 76, 79

U

UNIX/Linux, 72

V

Visual Studio 2017, 57
извлечение текста, 63
ИИ-ориентированное приложение, 57
ключ (ключи) подписки, 52
модель отпечатка голоса, 216
тестирование API, 53, 57
учетная запись когнитивных служб, 50
Visual Studio (VS)
когнитивные службы, 46
среда разработки, 46
учетная запись Azure, 47

W

WhatsApp, 76
WIMP (windows, icons, menus, pointer), графический интерфейс, 73

A

Акустическое моделирование, 198

Б

Большие данные, 26
Бот планирования визита к врачу, 158

В

Веб-модель языка (WebLM)
веб-страница, 141
предсказание следующего слова, 147
разбиение на слова, 143
системы машинного перевода, 141
совместная вероятность, 144
страница результатов поиска в Bing, 142
условная вероятность, 145

Г

Грамматика, 198
Графический интерфейс пользователя (ГИП, GUI), 73

Д

Данные JSON, 109
Данные произношения, 211

З

Закон Гордона Мура, 280

И

Интеллектуальная служба понимания языка (Language Understanding Intelligent Service, LUIS), 41, 107, 284

API для автоматизации, 178

Bot Framework, 157, 160, 178

HealthCheckupDialog, 179

Visual Studio, 179

активное обучение, 177

веб-интерфейс, 109

веб-приложение, 111

данные JSON, 109

диаграмма последовательной архитектуры, 110

добавление намерений, 162

добавление/разметка

высказываний, 162

добавление списка фраз, 176

иерархические сущности, 173

когнитивные службы, 107

масштабная тренировка, 112

намерение None, 185

намерение без сущностей, 180

намерение с полным комплектом сущностей, 181

намерения и сущности, 159

подключение бота к Skype, 186

подписка Azure, 114

предварительно подготовленные сущности, 175

приложение, 108

приложение DefinitionApp, 109

приложение с определением терминов, 116

публикация приложения, 164

разговор на основе текста, 108

разговорные приложения, 157

распознавание именованных сущностей, 112

словарь данных, 159

создание, 161

составные сущности, 170, 183

сущности, 166, 168

эмулятор Bot Framework, 181

языковые культуры, 107

Интервейс пользователя (UI), 71

Интернет вещей (IoT), 282

Интерфейс командной строки (CLI), 71

Искусственный интеллект (AI)

API-управляемая модель, 287

Facebook, 26

Google, 26

Google Переводчик, 28

Microsoft, 27

Microsoft Azure, облачный сервер, 26

алгоритмы, 280

большие данные, 26

буквальный перевод на английский, 29

веб-службы Amazon AWS, 26

вычислительные мощности, 279

данные, 281

знания, 42

ИИ-ориентированное приложение, 289

история и эволюция, 23

когнитивные категории, 283

когнитивные службы Microsoft, 39

компьютерное зрение, 36

машинное обучение против глубокого обучения, 30

машины, 288

область когнитивных вычислений, 278

облачные вычисления, 282

обнаружение объектов, 36

обработка ЕЯ, 34

оптическое распознавание символов (OCR), 38

поиск, 43, 286

понимание ЕЯ, 33, 284

правда против вымысла, 22

предложение на французском языке, 29

преобразование текста в речь (TTS)

и речи в текст (STT), 35

распознавание лиц, 38

распознавание образов, 37

распознавания речи, 35

рекомендации, 286

речь, 35, 41, 285

робот подводит итог, 292

робот толкает тележку, 289

система машинного перевода, 29

службы компьютерного зрения, 40

службы против решений, 283

текстурная модель (fabric model), 288

технологии, 291

традиционные методы обработки данных, 26

усиленный интеллект (УИ), 290

язык, 33, 41

К

Когнитивная модель языка

API Bing проверки орфографии, 120

API анализа текста, 129

анализ тональности текста, 135

выделение ключевых фраз, 133, 134

запрос API Bing проверки

орфографии, 124

запрос идентификации языка, 130

идеи использования, 140

идентификация темы, 137

идентификация языка, 129

интеграция LUIS, 127

обработка естественного языка, 120

орфографические корректоры, 123

отклик API Bing проверки

орфографии, 126

отклик идентификации языка, 132

Л

Лексикон, 198

Логистическая регрессия (LR), 111

Логистические регрессионные

классификаторы (LRCs), 111

М

Массачусетский технологический институт (MIT), 99

Машинное обучение (ML), 28, 30

жесткое кодирование, 29
 машинный перевод, 30
 натренированная модель и данные, 31
 нейронные сети, 32
 не-МО алгоритм машинного перевода, 30
 обучение без учителя, 32
 обучение с подкреплением, 32
 обучение с учителем, 32
 против глубокого обучения, 30
 реальные задачи, 30
 статические программные инструкции, 30
 человекоподобные машины, 31

О

Облачные вычисления, 282
 Обработка естественного языка (NLP), 34
 когнитивные службы, 95
 методы машинного обучения, 96
 намерение, 97
 определение, 95
 правила грамматики, 97
 системы понимания ЕЯ, 95, 106
 сложности, 102
 статистические модели, 104
 сущности, 98
 теория Patom, 106
 Обучение без учителя, 32
 Обучение с подкреплением, 32
 Обучение с учителем, 32
 Оптическое распознавание символов (OPC, OCR), 38

П

Персонализированная акустическая модель
 аудиофайлы и файлы транскрипции, 200, 204
 варианты подписки, 202
 веб-портал Custom Speech Service, 199, 206
 дополнительная информация, 208
 механизм службы обработки речи, 203
 наборы акустических данных, 203
 служба Microsoft персонализированной обработки речи, 199



служба персонализированной обработки речи (Custom Speech Service), 201, 206
 создание, 199, 205
 страница импорта акустических данных, 203
 страница наборов данных, 203
 страница наборов данных адаптации, 201
 файл транскрипции, 200
 флагок Accuracy Testing, 207
 Персонализированная конечная точка
 Преобразования речи в текст, 212
 Персонализированная модель языка, 209
 Поиск изображений, 233
 Поисковые предложения API, 228
 API Bing поиска видео, 242
 API Bing поиска новостей, 239
 Bing Поиск, 223, 228
 MSN Поиск, 226
 повсеместность, 224
 поиск в Google, 223
 предсказательность, 224
 проактивность, 224
 Понимание естественного языка (NLU), 33, 41, 77, 99, 157, 284
 Правда против вымысла, 22
 Прагматика, 103
 Предложений в области поиска корпоративные и социальномедийные приложения, 224
 поиск изображений, 233
 явный поиск, 224
 Преобразование речи в текст (STT), 35
 Преобразование текста в речь (TTS), 35, 195
 Программное обеспечение как служба (SaaS), 23
 Протоколы собраний (MOM), 141

Р

Разговорный интерфейс пользователя (РИП, CUI), 157, 278
 Facebook Messenger, 79
 Microsoft Bot Framework, 82
 Twitter, 79
 Visual Studio, 87
 графические элементы, 73



- графический интерфейс пользователя (ГИП, GUI), 73
 естественный разговор, 75
 ИИ-полные задачи, 78, 96
 интерфейс командной строки, 71
 интерфейс пользователя, 70
 инфраструктура Bot Framework, 82
 класс MessagesController, 85
 первое сообщение и сообщение по умолчанию, 85
 планирование визита к врачу, 90
 преимущества, 76
 приложения для обмена сообщениями, 76
 принципы разработки, 81
 проект приложения с использованием бота, 84
 разговор на основе команд, 74
 разговор на основе текста, 69, 81
 система заказа пиццы на основе чата, 69
 системное сообщение, 93
 сообщения WhatsApp, 68
 среда разработки, 84
 страница по умолчанию, 88
 технологический пейзаж, 73
 технология искусственного интеллекта, 77
 типы, 80
 электронная коммерция, 76
 эмулятор, 89
 эмулятор Bot Framework, 87
 Распознавание говорящего, 215
 verificationProfileID, 217
 аутентификация, 215
 верификация против
 идентификации, 215, 221
 идентификация, 221
 когнитивная модель Microsoft, 215
 распознавание говорящего –
 идентификация, 221
 регистрация для верификации
 говорящего, 216
 состояние операции, 221
 Распознавание именованных сущностей (NEM), 111
 Рекомендации
 FBT-метод, 253, 264
- HTTP-запрос, 263
 Microsoft, 252
 Netflix, 251
 асинхронная операция, 262
 данные, связанные с товарной позицией, 256
 интерфейс пользователя, 273
 каталогный файл, 257
 когнитивные службы, 255
 машинное обучение, 255
 модели и типы, 255, 260
 от клиента к предмету, 254
 от предмета к предмету, 254
 офлайновое оценивание, 272
 параметры, 261
 ранжирование, 267
 статистические модели, 257
 установление правил, 272

 файл с данными
 об использовании, 259
 Рекомендации, 254
 Рекомендация товаров, часто
 приобретаемых вместе (FBT), 253, 264
 Речеориентированные модели, 195
- С**
 Связывание именованных сущностей (NEM), 111
 Семантика, 101, 102
 Синтаксис, 102
 Словарь, 198
 Словосочетания, 108
 Создатель вопросов и ответов, 42
- Ф**
 Фонемы, 198
 Фразы, 108
- Ч**
 Частеречная разметка (POS), 112
- Э**
 Электронная коммерция, 76
 Эмулятор инфраструктуры Bot Framework, 88
- Я**
 Явный поиск, 224

Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «Планета Альянс» наложенным платежом,
выслав открытку или письмо по почтовому адресу:

115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.

Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: www.alians-kniga.ru.
Оптовые закупки: тел. (499) 782-38-89.
Электронный адрес: books@alians-kniga.ru.



Искусственный интеллект для .NET: речь, язык и поиск

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com

Перевод *Логунов А. В.*

Корректор *Синяева Г. И.*

Верстка *Чаннова А. А.*

Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.

Гарнитура «PT Serif». Печать офсетная.

Усл. печ. л. 27,9375. Тираж 200 экз.

Веб-сайт издательства: www.dmkpress.com