

Содержание

Об авторе	8
О рецензенте	9
Предисловие	10
Глава 1. Социальные медиа, социальные данные и Python	21
Начало	21
Социальные медиа – проблемы и возможности	22
Возможности	23
Проблемы	25
Технология анализа социальных данных	28
Инструменты Python для науки о данных	31
Настройка среды разработки Python	32
Эффективный анализ данных	35
Машинное обучение	39
Обработка естественного языка	43
Анализ социальных сетей	48
Визуализация данных	49
Обработка данных в Python	51
Создание составных конвейеров данных	53
Резюме	54
Глава 2. #ОбработкаТвиттера – хештеги, темы и временные ряды	55
Начало работы	55
Twitter API	56
Ограничение частоты запросов	56
Поиск и потоковая обработка	57
Выборка данных из Twitter	58
Получение твитов из ленты	60
Структура твита	62
Применение потокового интерфейса Streaming API	66
Анализ твитов – сущности	69
Анализ твитов – текст	73
Анализ твитов – временные ряды	79
Резюме	82
Глава 3. Пользователи, читатели и сообщества в Twitter	83
Пользователи, друзья и читатели	83
Возвращаясь к интерфейсу Twitter API	83
Структура профиля пользователя	85
Загрузка профилей друзей и читателей	87
Анализ связей	89
Измерение степени влияния и вовлеченности	94
Анализ читателей	98
Анализ диалога	104

Привязка твитов к географической карте	107
От твитов к GeoJSON	108
Простота создания карт с Folium.....	110
Резюме.....	116
Глава 4. Сообщения, страницы и взаимодействие пользователей	
в Facebook	117
Интерфейс Facebook Graph API	117
Регистрация приложения	118
Аутентификация и безопасность	119
Доступ к Facebook Graph API из Python	121
Анализ сообщений	124
Структура сообщения.....	127
Частотно-временной анализ	127
Анализ страниц Facebook.....	129
Получение сообщений со страницы	131
Измерение степени вовлеченности	135
Визуализация сообщений в виде облака слов.....	141
Резюме.....	142
Глава 5. Тематический анализ в Google+	144
Начало работы с Google+ API	144
Поиск в Google+	147
Вывод результатов поиска в веб-интерфейсе	149
Декораторы в Python.....	150
Маршруты и шаблоны Flask.....	151
Заметки и действия со страницы Google+	154
Анализ текстов и статистическая мера TF-IDF для заметок	157
Получение словосочетаний при помощи n-грамм	163
Резюме.....	163
Глава 6. Вопросы и ответы в сети Stack Exchange	165
Вопросы и ответы	165
Начало работы с Stack Exchange API.....	168
Поиск вопросов с тегами	170
Поиск пользователя	172
Обработка дампов данных из Stack Exchange	175
Классификация текстов по тегам в вопросах	180
Обучение с учителем и классификация текстов	180
Алгоритмы классификации	184
Оценка.....	187
Классификация текстов на данных из сети Stack Exchange	189
Встраивание классификатора в приложение реального времени.....	193
Резюме.....	198
Глава 7. Блоги, RSS, Википедия и обработка естественного языка	199
Блоги и обработка естественного языка	199
Получение данных из блогов и веб-сайтов	200
WordPress.com API	200

Blogger API	203
Каналы RSS и Atom	206
Получение данных из Википедии	207
Несколько слов о выборке данных из Веб	210
Основы обработки естественного языка	210
Предварительная обработка текста	211
Извлечение информации	220
Резюме	225
Глава 8. Анализ других данных	226
Большое количество социальных API	226
Анализ видео на YouTube	226
Анализ открытого программного обеспечения на GitHub	231
Анализ сведений о местных предприятиях в Yelp	238
Создание собственного клиента на Python	243
Простой интерфейс для вызовов по протоколу HTTP	243
Резюме	245
Глава 9. Связанные данные и Семантическая паутина	247
Паутина данных	247
Словарь Семантической паутины	249
Микроформаты	252
Связанные данные и открытые данные	254
Среда описания ресурса RDF	255
Формат данных JSON-LD	256
Инициатива Schema.org	257
Анализ связей из DBpedia	258
Анализ географических координат	260
Извлечение геоданных из Википедии	260
Нанесение геоданных на карты Google Maps	263
Резюме	267
Приложение А. Анализ данных из социальной сети «Вконтакте»	269
Анализ сообщества и определение его типичного участника	270
Определение центральных узлов социального графа	276
Отображение центральностей на графике	277
Прочие операции	279
Предметный указатель	281

Об авторе

Марко Бонцанини – исследователь-аналитик из Лондона (Соединенное Королевство). Имеет докторскую степень в области информационного поиска Лондонского университета королевы Марии. Специализируется на анализе текстовой информации и поисковых приложениях, и многие годы с удовольствием занимался решением разнообразных задач управления информацией и науки о данных.

Ведет персональный блог на <http://marcobonzanini.com>, где обсуждает различные технические темы, главным образом, связанные с языком Python, анализом текстовой информации и наукой о данных.

Когда Марко не занят работой над проектами на Python, он с удовольствием принимает участие в жизни сообщества, посещая конференции и неформальные встречи PyData с разработчиками, а также очень любит варить домашнее пиво.

Эта книга является результатом долгого процесса, который выходит далеко за рамки простой подготовки материалов книги. Многие так или иначе принимали участие в формировании конечного результата. В первую очередь я хотел бы благодарить команду издательства Packt Publishing, в особенности Сонали Вернекар (Sonali Vernekar) и Сиддхеш Салви (Siddhesh Salvi) за предоставленную возможность работать над этой книгой и за помощь в течение всего процесса. Доктор Вейай «Уэйи» Сюй (Dr. Wei Ai «Wayne» Xu) выступил в роли научного редактора настоящей книги и внес множество предложений по ее улучшению, за что ему отдельное спасибо. Многие коллеги и друзья в случайных разговорах, глубоких обсуждениях и во время работы над проектами в прошлом немало способствовали повышению качества материала, представленного в этой книге. Особо хотел бы поблагодарить доктора Мигеля Мартинеса-Альвареса (Dr. Miguel Martinez-Alvarez), Марко Кампана (Marco Campana) и Стефано Кампана (Stefano Campana). Я также рад быть частью лондонского сообщества PyData, группы умнейших людей, которые регулярно встречаются, чтобы поговорить о Python и науке о данных в располагающей для этого атмосфере. И наконец, но не в последнюю очередь, хотел бы отметить Даниэлу (Daniela), поощрявшую меня в течение всего процесса, делясь со мной своими мыслями, предлагая улучшения и обеспечивая уютную обстановку по возвращении после работы.

О рецензенте

Вейай Уэйн Сюй – доцент в отделе коммуникаций Массачусетского университета (Амхерст, США) и сотрудничающий с институтом вычислительной социологии этого же университета. Ранее Сюй преподавал теорию сетей в институте сетевых наук Бостонского северо-восточного университета. Результаты его исследований в области онлайн-сообществ, «сарафанного радио» и социального капитала публиковались в различных авторитетных научных журналах. Сюй также оказывал содействие четырем национальным проектам в области стратегической коммуникации и общественного мнения. Кроме основной работы является соучредителем лаборатории данных под названием CuriosityBits Collective (<http://www.curiositybits.org/>).

Предисловие

За прошедшие несколько лет популярность социальных ресурсов существенно выросла благодаря тому, что все больше и больше пользователей стало обмениваться разнообразными видами информации их посредством. Компании используют социально-медийные платформы, для продвижения своих брендов, профессионалы регистрируют общедоступные учетные записи и используют их для налаживания связей, а обычные пользователи занимаются обсуждением любых тем. Увеличение числа пользователей также означает увеличение объемов данных, ожидающих анализа.

Вы, читатель этой книги, вероятнее всего разработчик, инженер, аналитик, исследователь или студент, который хотел бы применить приемы анализа к данным, хранящимся на социальных ресурсах. С этой точки зрения вы, как практик в области анализа данных (или будущий практик), не будете испытывать недостатка в потенциальных возможностях и вызовах.

Книга «Анализ социальных медиа на Python» даст основные инструменты, которые помогут вам использовать это обилие данных в своих интересах. Знакомство с главными инструментами анализа данных на Python начнется с предоставления вводной информации о методах обработки естественного языка, машинного обучения, анализа социальных сетей и визуализации данных. Последующее пошаговое руководство по самым популярным социально-медийным платформам, включая Twitter, Facebook, Google+, Stack Overflow, Blogger, YouTube и другим, расскажет, как получить доступ к данным этих сетей, и как применять различные виды анализа для извлечения полезной информации из необработанных данных.

В настоящей книге затрагиваются три главных аспекта, а именно:

- **Социально-медийные API:** каждая платформа обеспечивает доступ к своим данным по-своему. Понимание, как с ними взаимодействовать, даст возможность ответить на вопросы: *как получить данные?* и *какие данные можно получить?* Это важно потому, что невозможно анализировать данные, не имея к ним доступа. Каждая глава посвящена отдельной социально-медийной платформе и подробно рассказывает, как взаимодействовать с соответствующим API.
- **Приемы анализа данных:** простое извлечение данных из API не имеет особой ценности. Следующий шаг – ответ на вопрос: *что можно делать с данными?* Каждая глава описывает понятия, которые помогут вам понять суть того или иного вида анализа данных и в чем заключается их ценность. В теоретическом плане мы лишь слегка затронем внешние аспекты, особо не вдаваясь в подробности, которые оставим академическим учебникам. Главная цель этой книги: привести практические примеры, которые дадут вам возможность легко приступить к работе.
- **Инструменты Python для анализа данных:** поняв, что можно делать с данными, нам останется ответить на последний вопрос: *как это делается?* Python зарекомендовал себя как один из главных языков для анализа

данных. Простые синтаксис и семантика языка вместе с его богатой экосистемой поддержки научных вычислений обеспечивают пологую кривую обучения для новичков и всевозможные изоощренные инструменты для экспертов. Настоящая книга знакомит с главными библиотеками для Python, используемыми в мире научных вычислений, в том числе NumPy, pandas, NetworkX, scikit-learn, NLTK и многими другими. Практические примеры имеют форму коротких сценариев, которые можно использовать (и возможно расширять) для выполнения различных и интересных видов анализа данных из социальных ресурсов, к которым вы имеете доступ.

Если исследование области, охватывающей эти три главных темы, представляет интерес, тогда эта книга для вас.

О ЧЕМ РАССКАЗЫВАЕТСЯ В КНИГЕ

Глава 1 «*Социальные медиа, социальные данные и Python*» вводит главные понятия анализа данных с использованием языка Python применительно к социальным медиа. Здесь читатель найдет краткий обзор технологий машинного обучения, обработки естественного языка, анализа социальных сетей и визуализации данных, а также обсуждение главных инструментов Python для анализа данных и справочную информацию о настройке среды программирования на Python.

Глава 2 «*#Твиттер – хештеги, темы и временные ряды*» открывает практическое обсуждение анализа данных из социальной сети Twitter. После подготовки приложения, взаимодействующего с программным интерфейсом Twitter API, эта глава объясняет, как получить данные посредством потокового API и как выполнить частотный анализ хештегов и текста. Здесь также обсуждаются некоторые виды анализа временных рядов для поиска закономерностей распределения твитов во времени.

Глава 3 «*Пользователи, читатели и сообщества в Twitter*» продолжает обсуждение анализа данных из социальной сети Twitter, сосредотачив внимание на пользователях и взаимодействиях между ними. Эта глава показывает, как выявлять связи и диалоги между пользователями. Здесь объясняются такие интересные применения, как кластеризация (сегментация) пользователей и измерение степени влияния и вовлеченности пользователей.

Глава 4 «*Сообщения, страницы и взаимодействия пользователей в Facebook*» посвящена социальной сети Facebook и ее программному интерфейсу Facebook Graph API. В данной главе подробно разбираются способы взаимодействия с Graph API, включая аспекты безопасности и конфиденциальности, и приводятся примеры извлечения сообщений из профиля пользователя и страниц в Facebook. Принципы анализа временных рядов и степени вовлеченности пользователей применяются к таким взаимодействиям пользователей, как комментарии, отметки «Нравится» и реакции.

Глава 5 «*Тематический анализ в Google+*», охватывает социальную сеть Google. Здесь подробно разбираются способы доступа к централизованной платформе Google и обсуждаются примеры поиска содержимого и пользователей в Google+. Эта глава также показывает, как построить данные, поступающие из Google API, в собственное веб-приложение, сконструированное на основе микрофреймворка Flask.

Глава 6 «*Вопросы и ответы в сети Stack Exchange*» объясняет тему работы с вопросами и ответами и в качестве главного примера использует сеть Stack Exchange. Здесь читатель узнает, как искать пользователей и содержимое на различных сайтах этой сети, и прежде всего на Stack Overflow. Используя дампы данных этой сети для онлайн-обработки, эта глава знакомит с методами машинного обучения с учителем для классификации текстов, и показывает, как встроить модель машинного обучения в приложение реального времени.

Глава 7 «*Блоги, RSS, Википедия и обработка естественного языка*» знакомит с приемами анализа текстовой информации. Всемирная паутина полна возможностей с точки зрения анализа текстов, и эта глава покажет, как взаимодействовать с различными источниками данных, такими как WordPress.com API, Blogger API, каналы RSS и Wikipedia API. На основе текстовых данных формализуются и расширяются основные понятия обработки естественного языка, кратко упоминаемые на протяжении всей книги. Затем читателю на специальных примерах демонстрируется процедура извлечения ссылок на сущности из произвольного текста.

Глава 8 «*Анализ других данных*» напоминает о многочисленных возможностях анализа данных, доступных за пределами наиболее распространенных социальных сетей. Демонстрирует примеры извлечения данных из YouTube, GitHub и Yelp, а также обсуждает создание собственного программного клиента, если конкретная платформа его не предоставляет.

Глава 9 «*Связанные данные и Семантическая паутина*» дает обзор Семантической паутины и связанных с ней технологий. Здесь рассматриваются темы связанных данных, микроформатов и RDF, и предлагаются примеры анализа семантической информации из DBpedia и Википедии.

Что потребуется для работы с книгой

Прилагаемые к книге примеры программного кода предполагают наличие у читателя последней версии Python в Linux, macOS либо Windows. Программный код был протестирован в Python 3.4.* и Python 3.5.*. Более старые версии (Python 3.3.* или Python 2.*) в явном виде не поддерживаются.

Глава 1 «*Социальные медиа, социальные данные и Python*» содержит инструкции по настройке локальной среды разработки и знакомит с кратким списком инструментов, используемых на протяжении всей книги. Вам также понадобятся некоторые важнейшие библиотеки для научных вычислений (NumPy, pandas и matplotlib), машинного обучения (scikit-learn), обработки естественного языка (NLTK) и анализа социальных сетей (NetworkX).

Кому адресована книга

Эта книга адресована в меру опытному разработчику на языке Python, желающим использовать общедоступные API для сбора данных из социально-медийных платформ и выполнять статистический анализ для извлечения полезной информации из данных. Книга предполагает элементарное знакомство со стандартной библиотекой Python и приводит практические примеры, которые станут вам ориентиром для создания своего собственного аналитического проекта на основе социальных данных.

СОГЛАШЕНИЯ

В этой книге используются разные стили оформления текста, которые разделяют различные виды информации. Ниже приводятся примеры этих стилей и поясняется их значение.

Элементы программного кода в тексте, имена таблиц в базах данных, имена папок и файлов, расширения файлов, пути к каталогам в файловой системе, фиктивные адреса URL, ввод пользователя и учетные записи в Twitter оформляются так: «Кроме того, атрибут `genre` представлен списком с переменным числом значений».

Блоки кода оформляются следующим образом:

```
from timeit import timeit
import numpy as np

if __name__ == '__main__':
    setup_sum = 'data = list(range(10000))'
    setup_np = 'import numpy as np;'
    setup_np += 'data_np = np.array(list(range(10000)))'
```

Когда потребуется привлечь ваше внимание к определенному фрагменту в блоке программного кода, он будет выделяться жирным:

Наберите ваш вопрос либо наберите "exit", чтобы выйти.

> **What's up with Gandalf and Frodo lately? They haven't been in the Shire for a while...**


Вопрос: What's up with Gandalf and Frodo lately? They haven't been in the Shire for a while...

Предсказанные метки: plot-explanation, the-lord-of-the-rings


Ввод или вывод в командной строке будет оформляться так:

```
$ pip install --upgrade [package name]
```

Новые термины и важные слова будут выделены жирным. Текст, отображаемый на экране, например в меню или в диалогах, будет оформляться так: «На странице конфигурации Keys and Access Tokens (Ключи и маркеры доступа) разработчик сможет найти ключ API и пароль, а также маркер доступа и секретный маркер доступа».

 Так оформляются предупреждения и важные примечания.

 Так оформляются советы и рекомендации.

 Так оформляются дополнения от переводчика к тексту оригинальной книги.

ОТЗЫВЫ И ПОЖЕЛАНИЯ

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или может быть не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

СКАЧИВАНИЕ ИСХОДНОГО КОДА ПРИМЕРОВ

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф в разделе «Читателям – Файлы к книгам».

СПИСОК ОПЕЧАТОК

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

СКАЧИВАНИЕ ЦВЕТНЫХ ИЛЛЮСТРАЦИЙ

Мы также подготовили файл PDF с цветными иллюстрациями, диаграммами и скриншотами, которые в этой книге имеют черно-белое оформление. Цветные иллюстрации помогут вам лучше понять обсуждаемые темы. Вы можете загрузить файл с иллюстрациями по адресу: https://www.packtpub.com/sites/default/files/downloads/MasteringSocialMediaMiningWithPython_ColorImages.pdf.

НАРУШЕНИЕ АВТОРСКИХ ПРАВ

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в Интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли принять меры.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, и помогающую нам предоставлять вам качественные материалы.

ВОПРОСЫ

Вы можете присылать любые вопросы, касающиеся данной книги, по адресу dm@dmk-press.ru или questions@packtpub.com. Мы постараемся разрешить возникшие проблемы.

КОММЕНТАРИЙ ПЕРЕВОДЧИКА

Весь материал настоящей книги приведен в соответствии с последними действующими версиями программных библиотек (время перевода книги — июль 2017 г.), дополнен свежей информацией и протестирован в среде Windows 10. Тестирование программного кода осуществлялось в Python 3.6.1.

Прилагаемый к книге адаптированный и скорректированный исходный код примеров лучше всего разместить в домашней папке пользователя (`/home/Ваши_проекты_Python` или `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\Ваши_проекты_Python`). Ниже приведена структура папки с прилагаемыми примерами:

Chap01-Chap09	Исходный код примеров на языке Python
Дополнение	Сценарий на Python для анализа данных из социальной сети «ВКонтакте»

Далее приведены особенности инсталляции некоторых используемых программных библиотек Python.

Особенности программного обеспечения

Библиотеки для Python обычно можно скачать из каталога библиотек Python PyPi (<https://pypi.python.org/>). Но имейте в виду, если предполагается использовать библиотеки SciPy и Scikit-learn в Windows, вам придется также установить библиотеку NumPy+MKL. Библиотека NumPy+MKL привязана к библиотеке Intel® Math Kernel Library и включает необходимые динамические библиотеки (DLL) в каталоге `numpy.core`. Загрузите соответствующие whl-файлы из репозитория (<http://www.lfd.uci.edu/~gohlke/pythonlibs/>) и установите (например, в 64-разрядной версии Windows и Python 3.6: `pip install numpy-1.13.0+mkl-cp36-cp36m-win_amd64.whl`). Соответствующая процедура установки описана ниже. Далее приводятся сведения о базовых библиотеках:

- **NumPy** – основная библиотека для научных вычислений на Python;
- **Matplotlib** – библиотека для работы с двумерными графиками. Требуется наличие `numpy` и некоторых других пакетов;
- **Pandas** – инструмент для анализа структурных данных и временных рядов. Требуется наличие пакета `numpy` и некоторых других;
- **Scikit-learn** – интегратор классических алгоритмов машинного обучения. Требуется наличие `numpy+mkl`;
- **SciPy** – библиотека, используемая в математике, естественных науках и инженерном деле. Требуется наличие `numpy+mkl`.

Факультативно:

- **Jupyter** – интерактивная вычислительная среда;
- **PyQt5** – библиотека инструментов для создания графического интерфейса, требуется для работы инструментальной среды программирования Spyder;
- **Spyder** – инструментальная среда программирования на Python

Протокол установки библиотек

Среди прилагаемых примеров имеется файл `requirements.txt`, выполняющий установку всех используемых в книге библиотек в пакетном режиме одной командой:

```
pip3 install -r requirements.txt
```

Обратите внимание, что прежде чем выполнить эту команду, необходимо перейти в каталог, где находится этот файл, например, командой `cd`. Этот файл также можно найти в репозитории с примерами к книге по адресу: <https://github.com/bonzanini/Book-SocialMediaMiningPython>.

Если вы захотите установить более свежие версии библиотек и контролировать весь процесс установки, ниже предлагается список команд установки библиотек в том порядке, в каком они встречаются в тексте книги. В некоторых случаях библиотеки должны устанавливаться из `whl`-файлов, которые можно загрузить из репозитория (<http://www.lfd.uci.edu/~gohlke/pythonlibs/>).

```
pip3 install --upgrade pip или pip install -U pip
pip3 install virtualenv
pip3 install numpy
```

либо как whl: `pip3 install numpy-1.13.0+mkl-cp36-cp36m-win_amd64.whl`

```
pip3 install pandas
pip3 install scipy
```

либо как whl: `pip3 install scipy-0.19.0-cp36-cp36m-win_amd64.whl`

```
pip3 install Scikit-learn
```

либо как whl: `pip3 install scikit_learn-0.18.1-cp36-cp36m-win_amd64.whl`

```
pip3 install matplotlib
pip3 install nltk
pip3 install Pyro4
pip3 install gensim
pip3 install networkx
pip3 install tweepy
pip3 install folium
pip3 install facebook-sdk
pip3 install wordcloud
```

либо как whl: `pip3 install wordcloud-1.3.1-cp36-cp36m-win_amd64.whl`

```
pip3 install Pillow
pip3 install google-api-python-client
pip3 install flask
pip3 install beautifulsoup4
pip3 install py-stackexchange
pip3 install lxml
pip3 install requests
pip3 install feedparser
pip3 install wikipedia
pip3 install PyGithub
pip3 install yelp
pip3 install rdflib
pip3 install mf2py
pip3 install pykml
```

факультативно:

```
pip3 install jupyter
pip3 install pyqt5
pip3 install spyder
```



В зависимости от типа операционной системы, версии Python и версий библиотек, версии whl-файлов для установки могут отличаться от приведенных выше, где показаны последние версии (по состоянию на август 2017) для 64-разрядной версии Windows и Python 3.6.1.

Перечень использованных примеров

Глава 1

```
python demo_gensim.py lord_of_the_rings.txt
```

Глава 2

```
python twitter_get_user_timeline.py marcobonzanini
python twitter_get_user_timeline.py PacktPub
python twitter_streaming.py \#RWC2015 \#RWCFinal rugby
python twitter_hashtag_frequency.py user_timeline_PacktPub.jsonl
python twitter_hashtag_stats.py user_timeline_PacktPub.jsonl
python twitter_mention_frequency.py user_timeline_PacktPub.jsonl
python twitter_term_frequency.py filename.jsonl
python twitter_time_series.py stream__RWC2015__RWCFinal_Rugby.jsonl
```

Глава 3

```
python twitter_get_home_timeline_toscreen.py
python twitter_get_home_timeline.py
python twitter_get_user_timeline.py marcobonzanini
python twitter_get_user_timeline.py PacktPub
python twitter_streaming.py \#RWC2015 \#RWCFinal rugby
python twitter_hashtag_frequency.py user_timeline_PacktPub.jsonl
python twitter_hashtag_stats.py user_timeline_PacktPub.jsonl
python twitter_mention_frequency.py user_timeline_PacktPub.jsonl
python twitter_term_frequency.py user_timeline_PacktPub.jsonl
python twitter_term_frequency_graph.py user_timeline_PacktPub.jsonl
python twitter_time_series.py stream__RWC2015__RWCFinal_Rugby.jsonl
python twitter_make_geojson.py --tweets stream__RWC2015__RWCFinal_Rugby.jsonl --geojson
rwc2015_final.geo.json
```

Глава 4

```
python twitter_get_user.py PacktPub
python twitter_get_user.py marcobonzanini
python twitter_followers_stats.py PacktPub
python twitter_followers_stats_set.py PacktPub
python twitter_followers_stats_numpy.py PacktPub
python twitter_influence.py marcobonzanini PacktPub
python twitter_cluster_users.py --filename users/marcobonzanini/followers.jsonl --k 5
--max-features 200 --max-ngram 3
python twitter_conversation.py stream__RWC2015__RWCFinal_Rugby.jsonl
python twitter_map_example.py --map example_map.htm
python twitter_map_basic.py --geojson rwc2015_final.geo.json --map rwc2015_final_tweets.html
python twitter_map_clustered.py --geojson rwc2015_final.geo.json --map rwc2015_final_tweets_
clustered.html
python facebook_my_profile.py
python facebook_get_friends.py
python facebook_get_my_posts.py
```

```
python facebook_get_my_posts_more_fields.py
python facebook_post_time_stats.py -f my_posts.jsonl
python facebook_get_page_info.py --page PacktPub
python facebook_get_page_posts.py --page PacktPub --n 500
python facebook_top_posts.py --page PacktPub
python facebook_top_posts_plot.py --page PacktPub
python facebook_posts_wordcloud.py --page PacktPub
```

Дополнение:

```
twitter_get_oldtweets_in_bulk.py
```

Глава 5

```
python gplus_search_example.py --query packt
python gplus_search_web_gui.py
python gplus_get_page_activities.py --page +packtpublishing --max-results 1000
python gplus_activities_keywords.py --file activities_+LarryPage.jsonl --keywords 5
```

Глава 6

```
python stack_search_keyword.py --tags "python;nosql" --n 10
python stack_search_user.py --name joel
python stack_search_user.py --name joel --sort creation --order asc
python stack_xml2json.py --xml movies.stackexchange_5.com --json movies.tags.jsonl
python stack_xml2json.py --xml movies.stackexchange_4.com --json movies.posts.jsonl --clean-post
python stack_classification_prepare_dataset.py --tags-file movies.tags.jsonl --posts-file
movies.posts.jsonl --output movies.questions4classification.jsonl --min-df 10
python stack_classification_predict_tags.py --questions movies.questions4classification.jsonl
python stack_classification_predict_tags.py --questions movies.questions4classification.jsonl
--min-df 5
python stack_classification_save_model.py --questions movies.questions4classification.jsonl
--min-df 5 --output questions-svm-classifier.pickle
python stack_classification_user_input.py --model questions-svm-classifier.pickle
```

Глава 7

```
python blogs_wp_get_posts.py --domain marcobonzanini.com --output posts.marcobonzanini.com.
jsonl --posts 100
python blogs_blogger_get_posts.py --url http://googleresearch.blogspot.co.uk --posts 50
--output posts.googleresearch.jsonl
python blogs_rss_get_posts.py --rss-url http://feeds.bbc.co.uk/news/rss.xml --json rss.bbc.jsonl
python blogs_entities.py --entity London
```

Глава 8

```
python youtube_search_video_pagination.py --query python --n 50 --output videos.jsonl
python github_get_user.py --user bonzanini --get-repos
python github_search_user.py --query [здесь ваш запрос] --sort followers --order desc
python github_search_repos.py --query python --sort stars --order desc
python yelp_search_business.py --location London --search breakfast --limit 5
python yelp_search_business.py --location "San Francisco" --search "craft beer" --limit 5
```

Глава 9

```
python rdf_summarize_entity.py --entity "Python"
python rdf_summarize_entity.py --entity "Python_(programming_language)"
```

```
python micro_geo_wiki.py --url "https://en.wikipedia.org/wiki/London"
python micro_geo_wiki.py --url "https://en.wikipedia.org/wiki/List_of_United_States_cities_by_
population"
python micro_geo2kml.py --url "https://en.wikipedia.org/wiki/List_of_United_States_cities_by_
population" --output us_cities.kml --n 20
```

Установка библиотек Python из whl-файлов

Библиотеки для Python можно разрабатывать не только на чистом Python. Часто библиотеки пишутся на C (динамические библиотеки) и для них пишется обертка Python или же библиотека пишется на Python, но для оптимизации узких мест часть кода пишется на C. Такие библиотеки получаются очень быстрыми, однако программисту на Python тяжелее установить их ввиду банального отсутствия соответствующих знаний или необходимых компонентов и настроек в рабочей среде (в особенности в Windows). Для решения описанных проблем разработан специальный формат (файлы с расширением .whl) для распространения библиотек, который содержит заранее скомпилированную версию библиотеки со всеми ее зависимостями. Формат whl поддерживается всеми основными платформами (Mac OS X, Linux, Windows).

Установка производится с помощью диспетчера библиотек pip. В отличие от обычной установки командой `pip install <имя_библиотеки>`, вместо имени библиотеки указывается путь к whl-файлу: `pip install <путь/к/whl_файлу>`. Например,

```
pip install C:\temp\scipy-0.19.0-cp36-cp36m-win_amd64.whl
```

Откройте окно командой строки и выполнив команду `cd` перейдите в каталог, где находится whl-файл. Затем просто скопируйте в команду выше имя whl-файла. В этом случае полный путь указывать не понадобится. Например,

```
pip install scipy-0.19.0-cp36-cp36m-win_amd64.whl
```

При выборе библиотеки важно, чтобы разрядность устанавливаемой библиотеки и интерпретатора совпадали. Пользователи Windows могут загрузить whl-файлы на веб-странице <http://www.lfd.uci.edu/~gohlke/pythonlibs/> Кристофа Голька из Лаборатории динамики флуоресценции Калифорнийского университета в г. Ирвайн. Библиотеки там постоянно обновляются и в архиве содержится все, что только может понадобиться.

Установка и настройка инструментальной среды Spyder

Spyder — это инструментальная среда для научных вычислений на языке Python (Scientific PYthon Development EnviRonment) для Windows, Mac OS X и Linux. Это простая, легковесная и бесплатная интерактивная среда разработки на Python, которая предлагает функционал, аналогичный среде разработки на MATLAB, включая готовые к использованию виджеты PyQt5 и PySide: редактор исходного кода, редактор массивов данных NumPy, редактор словарей, консоли Python и IPython и многое другое.

Чтобы установить среду Spyder в Ubuntu Linux, используя системный диспетчер пакетов, достаточно выполнить всего одну команду:

```
sudo apt-get install spyder
```

Чтобы установить ее с использованием диспетчера библиотек `pip`:

```
sudo apt-get install python-qt5 python-sphinx
sudo pip install spyder
```

А чтобы обновить:

```
sudo pip install -U spyder
```

Установить Spyder в Fedora 25 можно командой:

```
dnf install python-spyder
```

Установка среды Spyder в Windows:

```
pip install spyder
```



Среда Spyder требует обязательной установки библиотеки `PyQt5`.

Глава 1

Социальные медиа, социальные данные и Python

Настоящая книга посвящена технологиям *анализа данных из социальных сетей* с использованием языка *Python*. Три выделенных ключевых термина в предыдущем предложении помогают определить целевую аудиторию настоящей книги: любой разработчик, инженер, аналитик, исследователь или студент, интересующийся исследованием области, где встречаются эти три темы.

Эта глава охватывает следующие темы:

- социальные медиа и социальные данные;
- процесс анализа данных из социальных сетей;
- настройка среды Python для разработки приложений;
- инструменты Python для обработки научных данных;
- обработка данных на Python.

Начало

Во втором квартале 2015 г. компания Facebook объявила, что число активных пользователей этой социальной сети превысило 1,5 миллиарда. В 2013 г. компания Twitter заявила, что ежедневно обрабатывает более 500 миллионов твитов. Читателям также будет интересно узнать, что в 2015 г. веб-сайт Stack Overflow объявил о том, что с момента его открытия было задано больше 10 миллионов вопросов.

Данные числа являются лишь верхушкой айсберга, учитывая экспоненциальный рост популярности социальных медиа по мере увеличения числа пользователей, обменивающихся информацией посредством различных платформ. Это обилие данных дает практикам анализа данных уникальные возможности. Цель настоящей книги состоит в том, чтобы помочь читателю овладеть приемами использования API социальных медиа для сбора данных, которые можно проанализировать при помощи инструментов Python и получить интересные выводы о том, как пользователи взаимодействуют между собой.

Настоящая глава закладывает основы для начального обсуждения проблем и возможностей анализа данных из социальных медиа и знакомит с некоторыми инструментами Python, которые будут использоваться в последующих главах.

СОЦИАЛЬНЫЕ МЕДИА – ПРОБЛЕМЫ И ВОЗМОЖНОСТИ

В традиционных средствах массовой информации пользователи, как правило, являются простыми потребителями. Информация течет в одном направлении: от публикатора к пользователям. Социальные медиа разрушают эту модель, позволяя каждому пользователю одновременно быть потребителем и публикатором. По этой теме написано большое количество академических статей, в которых авторы пытались определить, что на самом деле означает термин «социальные медиа», например Андреас М. Каплан (Andreas M. Kaplan) и Майкл Хэенлейн (Michael Haenlein) в своей статье «*Users of the world, unite! The challenges and opportunities of Social Media*» («Пользователи всех стран, объединяйтесь! Проблемы и возможности социальных медиа»). Разные социально-медийные платформы объединяют следующие аспекты:

- интернет-приложения;
- информация, генерируемая пользователями;
- сетевые взаимодействия.

По своей сути социальные медиа – это интернет-приложения. Ясно, что прогресс в области интернет- и мобильных технологий способствовал распространению социальных медиа. Посредством мобильного телефона можно мгновенно соединиться с социально-медийной платформой, опубликовать свой материал или разузнать последние новости.

Социально-медийные платформы управляются информацией, генерируемой пользователями. В противоположность традиционной модели, в них потенциальным публикатором является каждый пользователь. И, что более важно, любой пользователь может взаимодействовать с любым другим пользователем, делясь сведениями, оставляя комментарии или выражая положительную оценку посредством кнопки Like (Нравится) (то есть, «лайкая», «плюсуя» или щелкая на пиктограмме с большим пальцем вверх).

Социальные медиа основаны на сетевых взаимодействиях. Как описано выше, суть социальных медиа заключается во взаимодействии пользователей между собой. Быть на связи – вот центральная идея большей части социально-медийных платформ, и информация, которую вы получаете посредством своей ленты новостей, обусловлен вашими связями.

С учетом этих ключевых особенностей, характерных для большинства платформ, социальные медиа используются, чтобы:

- оставаться на связи с друзьями и семьей (например, посредством Facebook);
- вести микроблог и узнавать последние новости (например, посредством Twitter);
- оставаться на связи с профессиональной сетью (например, посредством LinkedIn);
- делиться мультимедийным материалом (например, посредством Instagram, YouTube, Vimeo и Flickr);
- находить ответы на вопросы (например, посредством Stack Overflow, Stack Exchange и Quora);
- находить и организовывать интересующую информацию (например, посредством Pinterest).

Цель этой книги – дать ответ на центральный вопрос: как извлекать полезные знания из данных, поступающих из социальных медиа? А теперь отступим на шаг назад и определим, что такое *знание* и что понимать под *полезностью*.

Традиционные определения понятия «знание» пришли из информатики. Обычно «знание» изображается как часть пирамиды, иногда называемой иерархией знаний, в основании которой лежат данные, на среднем уровне – информация, и наверху – знание. Эта пирамида изображена на рис. 1.1.

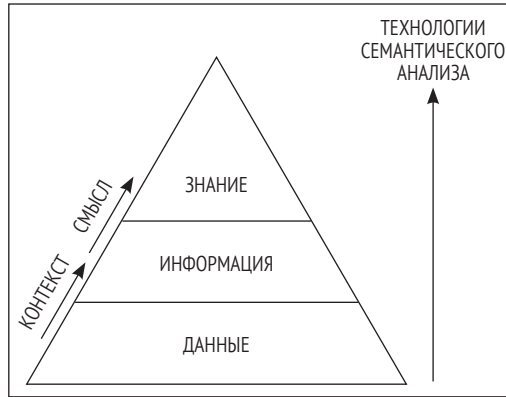


Рис. 1.1 ❖ От необработанных данных к семантическому знанию

Восхождение на пирамиду отражает процесс извлечения знания из исходных данных. Путь от исходных данных к чистому знанию лежит через интеграцию контекста и смысла. По мере восхождения вверх по пирамиде, применение технологий анализа помогает получить все более глубокое понимание исходных данных, и что еще более важно, понимание пользователей, которые производят эти данные. Другими словами, увеличивается полезность знаний.

В данном контексте полезное знание означает *эффективное* знание, то есть, знание, которое позволяет лицу, принимающему решения, реализовать бизнес-стратегию. Прочтя эту книгу, вы поймете ключевые принципы извлечения ценности из социальных данных. Понимание, как пользователи взаимодействуют посредством социально-медийных платформ, является одним из ключевых аспектов в этом процессе.

Следующие ниже разделы формулируют некоторые проблемы и возможности анализа данных из социально-медийных платформ.

Возможности

Ключевой потенциал систем анализа данных кроется в *извлечении полезных выводов* из данных. Цель данного процесса состоит в том, чтобы ответить на интересные (а иногда и трудные) вопросы, используя технологии анализа данных, и достичь полного понимания конкретной предметной области. Например, розничный онлайн-магазин может применить анализ данных, чтобы понять, как их потребители выбирают товары. По результатам анализа они смогут рекомен-


довать клиентам товары, в зависимости от их покупательских привычек (например, пользователи, покупающие изделие А, также покупают изделие В). В общем случае это улучшает качество обслуживания клиентов и их удовлетворенность, что в итоге поможет нарастить продажи.

Многие организации в разных сферах деятельности могут применять технологии анализа данных для совершенствования своего бизнеса. Вот только некоторые примеры:

- банковское дело:
 - выявление лояльных клиентов с целью предложения им эксклюзивных программ;
 - распознавание мошеннических схем с целью уменьшения затрат;
- медицина:
 - изучение поведения пациентов с целью предсказания посещений хирурга;
 - помощь врачам в идентификации успешных способов лечения в зависимости от анамнеза пациента;
- розничная торговля:
 - изучение схем поведения во время покупок с целью улучшения качества обслуживания клиентов;
 - увеличение эффективности рекламных кампаний за счет более точной направленности;
 - анализ оперативных данных транспортных потоков для поиска самого быстрого маршрута доставки продуктов питания.

Как все это транслируется в область социальных медиа? По сути дела, вопрос состоит в том, как пользователи делятся своими данными посредством социально-медийных платформ. Организации больше не ограничиваются анализом данных, собираемых непосредственно, поскольку теперь они обладают доступом к гораздо большему объему данных.

Решение задачи сбора таких данных реализуется посредством удобных и универсальных API. Социально-медийные платформы применяют широко распространенную практику, предлагая разработчикам веб-ориентированные прикладные программные интерфейсы (Web API) для встраивания функций платформы в их приложения.

 **Прикладной программный интерфейс.** Прикладной программный интерфейс (Application Programming Interface, API), или просто программный интерфейс – это комплекс определений процедур и протоколов, описывающих поведение программного компонента, такого как библиотека или удаленная служба, в терминах разрешенных действий, входных и выходных данных. Используя сторонние API, разработчикам не нужно беспокоиться о внутреннем устройстве компонентов, а только о том, как его использовать.

Термином Web API мы обозначаем веб-службу, которая предоставляет ряд URI-адресов (возможно после прохождения аутентификации) для доступа к данным. Для разработки таких API широко используется архитектурный подход, который называется **передачей состояния представления** (Representational State Transfer, REST). Программный интерфейс, реализующий архитектуру REST, называется **RESTful API**. Однако, мы предпочитаем использовать более общий термин «Web API», потому что многие из существующих API не всегда строго соответствуют

принципами архитектуры REST. Для целей настоящей книги глубокое понимание архитектуры REST не требуется.

ПРОБЛЕМЫ

Некоторые проблемы анализа социальных медиа обусловлены проблемами анализа данных вообще.

Занимаясь обработкой социальных данных, мы часто имеем дело с **большими данными**. Чтобы понять значение больших данных и проблем, связанных с ними, можно вернуться к традиционному их определению (которое в 2001 г. дал Дуг Лэни (Doug Laney) в своем исследовании «*3D Data Management: Controlling Data Volume, Velocity and Variety, Doug Laney*»), известному также, как *три V больших данных*: объем (volume), разнообразие (variety) и скорость (velocity). За прошедшие годы это определение было дополнено новыми V, прежде всего проблемой *ценности (value)*, поскольку получение добавленной стоимости является для компаний одной из главных целей использования больших данных. Первое из трех исходных V – *объем (volume)* – означает, что данные могут храниться на нескольких разных машинах. Это, конечно, требует другой инфраструктуры, чем для обработки малых данных (например, уместающихся в оперативной памяти). Более того, объем также связан со *скоростью (velocity)* в том смысле, что объем данных увеличивается настолько быстро, что понятие *большой* превращается в подвижную цель. Наконец, проблема *разнообразия (variety)* обусловлена тем, что данные могут иметь разные форматы и структуры, часто несовместимые между собой и с различной семантикой. Данные из социальных медиа соответствуют всем трем V.

Появление больших данных подтолкнуло процесс разработки новых подходов к хранению данных в системах под общим названием NoSQL. Это обобщенный (зонтичный) термин для многочисленных парадигм баз данных, имеющих одну общую черту, которая заключается в уходе от традиционных реляционных хем хранения данных и продвижении взамен нее конструкции с динамической схемой. Данная книга посвящена вовсе не технологиям баз данных, но из этой области для нас могут оказаться ценными приемы работы со смесью структурированных, неструктурированных и полуструктурированных данных. Выражение «*структурированные данные*» относятся к информации, имеющей определенную организацию и обычно представленной в табличной форме. В этом случае связь с реляционными базами данных очевидна. Следующая таблица демонстрирует пример структурированных данных, которые описывают книги, проданные книжным магазином:

Заголовок	Жанр	Цена
1984	Политическая фантастика	12
Война и мир	Военный роман	10

Такого рода данные структурированы – каждый элемент имеет ясную организацию, а именно, три атрибута: «заголовок», «жанр» и «цена».

Полной противоположностью структурированным данным являются данные неструктурированные, то есть информация, не организованная в соответствии с некоторой предопределенной моделью данных. Неструктурированные данные

обычно представлены в простом текстовом виде, например, в виде электронных писем, документов, сообщений в социальных сетях и т. д. Представленные в этой книге методы могут использоваться для извлечения из неструктурированных данных шаблонов с тем, чтобы обеспечить некоторую структуру.

Между структурированными и неструктурированными данными находятся полуструктурированные данные, то есть данные с гибкой или не полностью предопределенной структурой. Подобные структуры иногда называют самоописательными. Типичным примером полуструктурированного формата данных может служить формат JSON (JavaScript Object Notation, то есть, формат записи объектов на JavaScript). Как следует из названия, формат JSON заимствует правила определения данных из языка программирования JavaScript. Этот формат приобрел чрезвычайную популярность, благодаря широкому применению для обмена данными между клиентом и сервером в веб-приложениях. Ниже приводится пример данных в формате JSON – расширенной версии предыдущих данных о книгах:

```
[
  {
    "заголовок": "1984",
    "цена": 12,
    "автор": "Джордж Оруэлл",
    "жанр": ["Политическая фантастика", "Социальная фантастика"]
  },
  {
    "заголовок": "Война и мир",
    "цена": 10,
    "жанр": ["Исторический роман", "Мелодрама", "Военный роман"]
  }
]
```

Обратите внимание, что первая книга имеет атрибут `автор`, который отсутствует во второй книге. Кроме того, атрибут `жанр` здесь имеет вид списка с переменным числом значений. Оба этих аспекта обычно не допускаются в полностью структурированном (реляционном) формате данных, но прекрасно уживаются в JSON и с полуструктурированными данными в целом.

Обсуждение структурированных и неструктурированных данных сводится к обработке разных форматов и принятию ряда подходов к решению проблемы целостности. Выражение **«целостность данных»** используется, чтобы очертить круг проблем, возникающих из-за присутствия искаженных, непоследовательных или неполных данных.

Непоследовательные и неполные данные часто встречаются при анализе сведений, генерируемых пользователем, и такие данные требуют особенно пристального внимания. Редко можно наблюдать пользователей, которые делятся своими данными систематически, почти в формальной манере. Наоборот, социальные медиа часто имеют неформальную и порой противоречивую организацию. Например, если пользователь захочет оставить жалобу на продукт на странице производителя в Facebook, он сначала должен выставить отметку *«Нравится»* самой странице, что никак не соответствует неудовлетворенности производителем из-за низкокачественного продукта. Понимание, как пользователи взаимодействуют в социально-медийных платформах, играет важнейшую роль при разработке анализа.

Для разработки приложений анализа данных также требуется рассмотреть вопросы, связанные с **доступом к данным**, в частности, когда политика компании в итоге приводит к нехватке данных для проведения анализа. Говоря проще, к ситуации, когда данные не всегда открыто доступны. В предыдущем абзаце мы уже обсудили, что проблема нехватки данных при анализе социальных медиа не настолько важна, по сравнению с другими корпоративными средами, поскольку большинство социально-медийных платформ предлагает хорошо спроектированные универсальные API, открывающие доступ к данным, в которых мы нуждаемся. Доступность данных, разумеется, по-прежнему зависит от того, как пользователи обмениваются ими и как они предоставляют доступ. Например, пользователи Facebook могут задавать уровень детализации сведений, отображаемых в их общедоступном профиле, и ограничивать доступность своей информации только кругом своих друзей. Информация о профиле, в частности, дне рождения, текущем месте проживания и трудовой биографии (и многом другом) может быть отмечена как частная или как общедоступная. Аналогично, когда мы пытаемся получить доступ к такого рода данным через Facebook API, пользователи, подписавшиеся на наше приложение, имеют возможность открывать нам доступ только к ограниченному подмножеству данных.

Еще одна и последняя общая проблема анализа данных кроется в понимании самого процесса анализа и способности объяснить его результаты. Другими словами, не всегда получается правильно сформулировать вопрос перед началом анализа данных. В большинстве случаев **научно-исследовательские и опытно-конструкторские (R&D)** процессы управляются результатами первичного анализа, то есть, чтобы понять, как подступиться к проблеме, прежде нужно начать ее анализировать. Связанное с этим понятие в статистике описывается фразой *«корреляция не подразумевает причинную обусловленность»*. Можно применить большое количество статистических исследований и выявить корреляцию между двумя переменными, то есть, что два события происходят вместе, но этого недостаточно, чтобы установить причинно-следственную связь в любом направлении. Забавные примеры причудливых корреляций можно найти по всей Всемирной паутине. Популярный случай был опубликован в одном из самых уважаемых медицинских журналов «New England Journal of Medicine», который показал интересную корреляцию между количеством шоколада, потребленного на душу населения в расчете на страну, и числом Нобелевских лауреатов (*«Chocolate Consumption, Cognitive Function, and Nobel Laureates»*, Франц Х. Мессерли (Franz H. Messerli), 2012).

Выполняя первичный анализ, важно помнить, что корреляция (два события, происходящие вместе) является двунаправленной, тогда как причинная обусловленность (событие А вызвало событие В) – однонаправленной. Это шоколад делает вас более умным? Или же умные люди любят шоколад больше, чем средний человек? Случайно ли совпадение этих двух событий? Или же существует третья, пока невидимая, переменная, которая играет некую роль в корреляции? Просто наблюдать корреляцию недостаточно, чтобы описать причинную обусловленность, но это часто интересная отправная точка, чтобы задать важные вопросы о данных, которые мы наблюдаем.

Следующий раздел обобщает способы взаимодействий наших приложений с социально-медийным API и выполнения требуемого анализа.

ТЕХНОЛОГИЯ АНАЛИЗА СОЦИАЛЬНЫХ ДАННЫХ

Прежде чем перейти к обсуждению подробностей в последующих главах, в данном разделе мы в общих чертах обсудим процесс создания приложения анализа социальных данных.

В общем случае процесс можно разбить на следующие этапы:

- аутентификация;
- сбор данных;
- очистка и предварительная обработка данных;
- моделирование и анализ;
- представление результатов.

На рис. 1.2 изображена схема данного процесса:

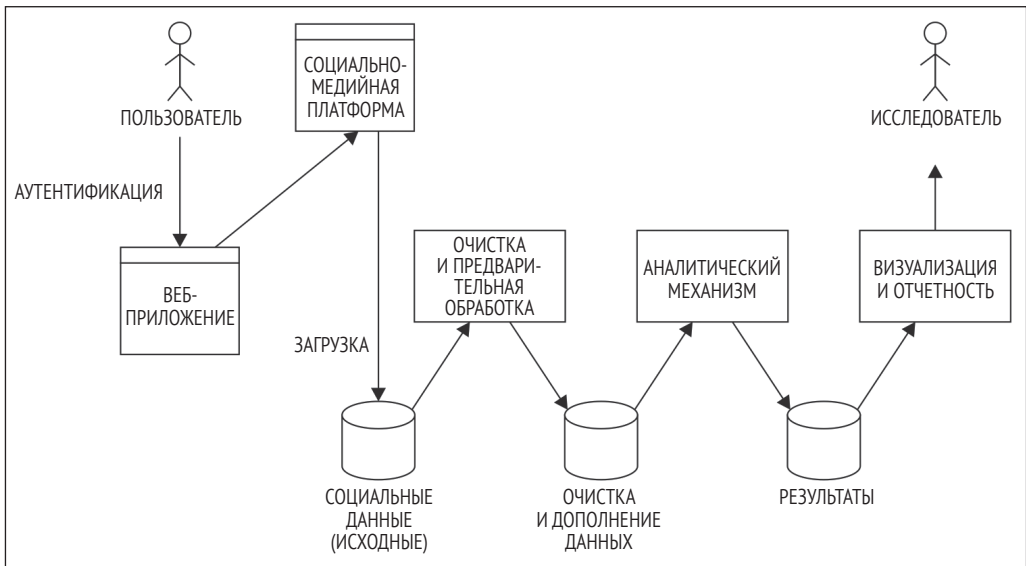


Рис. 1.2 ❖ Процесс анализа социальных данных

Этап аутентификации обычно выполняется с использованием промышленного стандарта под названием **открытый протокол авторизации (OAuth)**. Это трехсторонний процесс, в том смысле, что вовлекает трех агентов: пользователя, потребителя (наше приложение) и провайдера ресурса (социально-медийной платформы). Он состоит из следующих этапов:

- 1) пользователь соглашается дать потребителю доступ к социально-медийной платформе;
- 2) поскольку пользователь не передает свой пароль непосредственно потребителю, потребитель предварительно обращается к провайдеру ресурса, чтобы получить маркер и секретный ключ для приложения. Они используются для подписи каждого запроса и предотвращения взлома;
- 3) затем пользователь перенаправляется с маркером к провайдеру ресурса, который просит подтвердить разрешение на доступ потребителя к данным пользователя;

- 4) в зависимости от особенностей социально-медийной платформы, провайдер может также попросить подтвердить, возможность выполнения потребителем каких-либо действий от имени пользователя, например, отправку обновлений, публикацию ссылок и т. д.;
- 5) провайдер ресурса передает потребителю действительный маркер;
- 6) далее маркер может вернуться к пользователю, который подтверждает доступ.

На рис. 1.3 изображен процесс авторизации со всеми описанными этапами. Важно помнить, что обмен учетными данными (имя пользователя/пароль) происходит только между пользователем и провайдером ресурса на этапах 3 и 4. Весь обмен на других этапах опирается на маркеры:

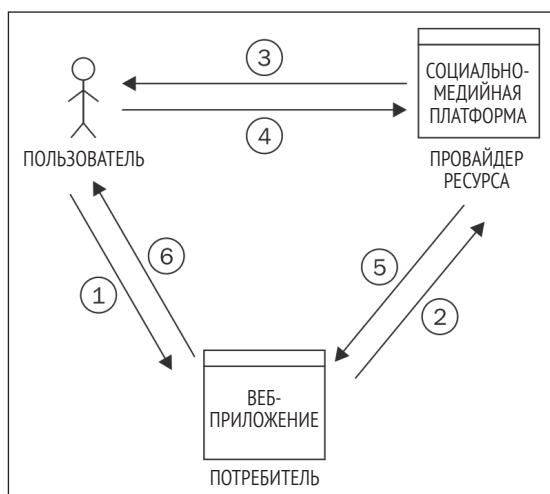


Рис. 1.3 ❖ Процесс авторизации согласно стандарту OAuth

С точки зрения пользователя этот сложный процесс происходит, когда пользователь вызывает наше веб-приложение и нажимает кнопку «**Вход с учетными данными Facebook**» (или Twitter, Google+ и т. д.). Затем пользователь должен подтвердить, что дает нашему приложению полномочия, и все остальное происходит уже без его участия.

С точки зрения разработчика самое замечательное, что экосистема Python уже имеет проверенные библиотеки для большинства социально-медийных платформ, которые включают реализацию процесса аутентификации. Как только вы регистрируете свое приложение в целевой службе, соответствующая платформа предоставит вашему приложению все необходимые маркеры авторизации. На рис. 1.4 изображен скриншот Twitter-приложения с названием Intro to Text Mining. На вкладке с настройками Keys and Access Tokens (Ключи и маркеры доступа) разработчик может найти ключ API и секретный ключ, а также маркер доступа и секретный маркера доступа. Более подробно детали авторизации для каждой социально-медийной платформы мы обсудим в соответствующих главах.

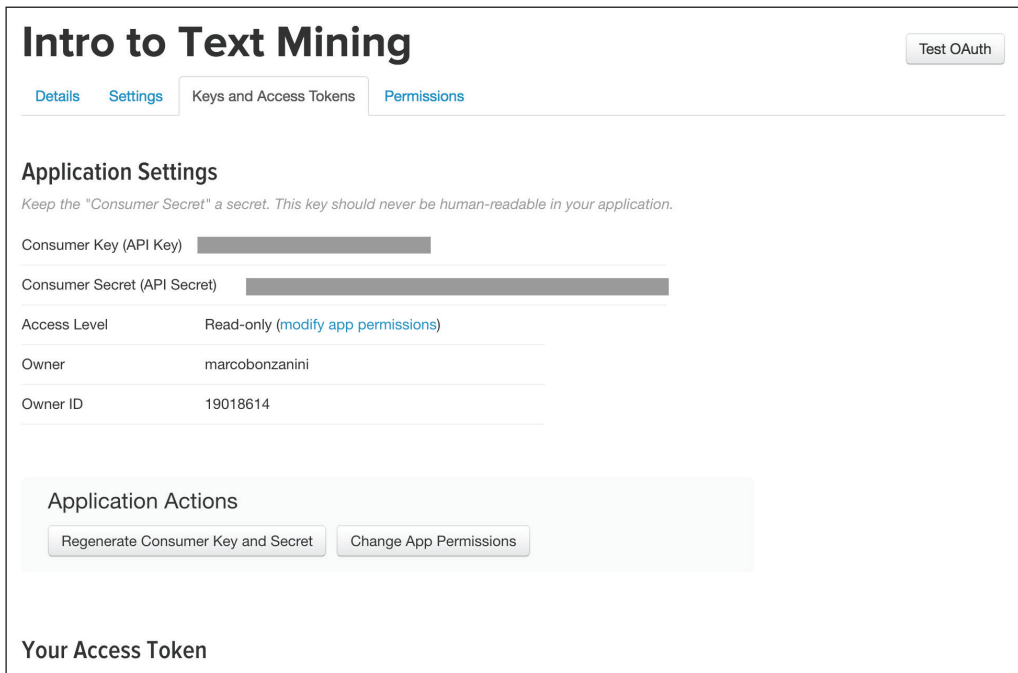


Рис. 1.4 ❖ Настройки для Twitter-приложения с названием Intro to Text Mining. Страница содержит все маркеры авторизации для использования разработчиком в своем приложении.

Этапы сбора, очистки и предварительной обработки данных так же зависят от особенностей социально-медийной платформы. В частности, этап сбора данных требует предварительной авторизации, поскольку для загрузки доступны только те данные, на получение которых дано разрешение. Очистка и предварительная обработка, с другой стороны, функционально зависят от вида моделирования и анализа данных, которые мы используем для получения желаемых выводов.

Как показано на рис. 1.2, моделирование и анализ выполняется компонентом «**Аналитический механизм**». Типичные задачи обработки данных, с которыми мы будем встречаться на протяжении всей книги, связаны с анализом текстов и графов.

Анализ текстов (или аналитическая обработка текстов) – это процесс получения структурированной информации из неструктурированных текстовых данных. Анализ текстов применим к большинству социально-медийных платформ, где пользователям разрешено публиковать информацию в форме сообщений или комментариев.

Приведем некоторые примеры применения анализа текстов:

- **классификация документов:** задача определения принадлежности документа к одной или нескольким категориям;
- **кластеризация документов:** задача группировки документов в когерентные и отличные друг от друга подмножества (например, по теме или подтеме), называемые кластерами;

- **реферирование документов:** задача создания сокращенной версии документа с целью уменьшения информационной нагрузки на пользователя, при сохранении самых важных аспектов, описанных в первоисточнике;
- **выделение сущностей:** задача поиска и классификации сущностей, упоминаемых в тексте, на категории, такие как люди, организации или географические местоположения;
- **анализ мнений:** задача идентификации и классификации чувств и мнений, выраженных в тексте, с целью понимания отношения к конкретному продукту, теме, службе и т. д.

Не все эти применения рассчитаны на социальные медиа, но растущий объем текстовых данных, доступных благодаря этим платформам, делает социальные медиа естественной площадкой для применения анализа текстов.

Анализ графов также сосредоточен на структуре данных. Графы просты для понимания, и одновременно являются мощной структурой данных, которая достаточно универсальна, чтобы применяться к многообразным представлениям данных. В графах есть два главных компонента для рассмотрения: узлы, представляющие сущности, или объекты, и ребра, представляющие связи, или соединения между узлами. В контексте социальных медиа очевидное использование графа – представление социальных связей пользователей. В более общем плане, в общественных науках графовая структура, используемая для представления социальных связей, также именуется социальной сетью.

Использование такой структуры данных внутри социальной сети позволяет естественным образом представить пользователей в виде узлов, а их связи (такие как *друзья* или *читатели*) – как ребра. Тем самым такую информацию, как *друзья друзей, которым нравится Python*, легко получить путем простого обхода графа (то есть, обход графа по ребрам от одного узла к другому). Теория и анализ графов предлагают гораздо больше возможностей для получения более глубоких выводов, которые не столь очевидны, как в предыдущем примере.

После высокоуровневого обсуждения анализа социальных данных, в следующем разделе мы рассмотрим несколько инструментов Python, которые обычно используются в проектах анализа данных.

ИНСТРУМЕНТЫ PYTHON ДЛЯ НАУКИ О ДАННЫХ

До сих пор, для обозначения задач и технических методов, которые собираемся применять на протяжении всей книги, мы использовали термин «анализ данных». В заголовке этого раздела упоминается термин «наука о данных». Использование этого термина имело в последние годы взрывной эффект, в особенности в деловой среде. В то же время многие ученые и журналисты подвергали его использованию критике, называя его новомодным словечком. В то же время другие академические институты начали предлагать курсы по науке о данных, и появилось множество книг и статей по данному предмету. Вместо того, чтобы выразить твердое мнение о том, где необходимо провести черту между разными дисциплинами, мы ограничимся тем, что посмотрим, как в наше время растет общий интерес к многочисленным областям, включая науку о данных, анализ данных, статистику, машинное обучение, искусственный интеллект, визуализацию данных и др. Обсуждаемые темы по своему характеру междисциплинарные, и все

они время от времени что-то заимствуют друг у друга. Мы живем в удивительное время, когда можно работать в любой из этих областей, испытывая большой интерес со стороны общественности и наблюдая постоянный ажиотаж с появлением новых достижений.

Цель этого раздела – познакомить вас с языком Python, как инструментом науки о данных, и описать часть экосистемы Python, которую мы собираемся использовать в последующих главах.

Python – один из самых интересных языков для проектов аналитической обработки данных. Ниже перечислены некоторые причины, объясняющие, почему он соответствует данному назначению:

- декларативный и интуитивный синтаксис;
- богатая экосистема для обработки данных;
- эффективность.

Python имеет плоскую кривую обучения, благодаря превосходному синтаксису. Будучи динамическим и интерпретируемым языком, он способствует быстрой разработке и интерактивному исследованию. Экосистема для обработки данных частично описывается в последующих разделах, которые познакомят с главными библиотеками, которые мы будем использовать в этой книге.

С точки зрения эффективности, интерпретируемые и высокоуровневые языки не славятся особым быстродействием. Такие инструменты, как NumPy, достигают эффективности, внутренне подключаясь к низкоуровневым динамическим библиотекам и предоставляя дружелюбный интерфейс Python. Кроме того, большое число проектов используют **Cython** – надмножество Python – обогащающее язык и позволяющее, кроме всего прочего, определять типы переменных и компилировать программный код. Многие другие проекты в мире Python находятся в процессе решения вопросов эффективности с глобальной целью – ускорить выполнение реализаций на чистом Python. В настоящей книге мы не будем углубляться в Cython и другие перспективные проекта, а воспользуемся библиотекой NumPy (в том числе посредством других библиотек, которые используют NumPy) для аналитической обработки данных.

Настройка среды разработки Python

Когда я начал работу над этой книгой, только-только вышла версия Python 3.5, привлекавшая внимание некоторыми новейшими возможностями, такими как улучшенная поддержка асинхронного программирования и семантическое определение подсказок о типах в исходном программном коде. С точки зрения использования, версия Python 3.5 пока еще не получил широкого распространения, но она представляет текущее направление развития языка.



Примеры, прилагаемые к книге, совместимы с версиями Python 3.4+ и 3.5+.

Пускаясь в дискуссии о выборе между Python 2 и Python 3, следует помнить один важный аргумент: через несколько лет поддержка Python 2 будет прекращена (во время написания книги разработчики Python предполагали прекратить поддержку Python 2 в 2020). В Python 2 не добавляется новых возможностей, потому что для этой ветви языка поддерживается только исправление ошибок. С другой стороны, многие библиотеки по-прежнему сначала разрабатываются для Python 2,

и только потом в них добавляется поддержка Python 3. По этой причине иногда могут возникать проблемы совместимости с той или иной библиотекой, которые, как правило, быстро разрешаются сообществом разработчиков. В целом, если нет веских причин, предпочтение должно отдаваться Python 3, в особенности для новых «целинных» проектов.

Инструменты *pip* и *virtualenv*

Чтобы содержать среду разработки в чистоте и облегчить переход от разработки к эксплуатации, предлагается для управления виртуальной средой и установки зависимостей использовать библиотеку *virtualenv*. Библиотека *virtualenv* позволяет создавать и управлять изолированными окружениями Python. Такие окружения помогут избежать загрязнения глобального окружения Python библиотеками, которые могут быть несовместимыми друг с другом. Этот инструмент позволяет поддерживать сразу несколько проектов, требующих разных конфигураций, и легко переключаться между ними. Более того, виртуальное окружение можно установить в локальной папке, доступной пользователям без административных привилегий.

Установить библиотеку *virtualenv* в глобальном окружении Python и сделать ее доступной всем пользователям можно с помощью диспетчера пакетов *pip*, выполнив команду в терминале (Linux/Unix) или в командной строке (Windows):

```
$ [sudo] pip install virtualenv
```

Команда *sudo* может потребоваться в Linux/Unix или macOS, если текущий пользователь не имеет административных привилегий.

Если библиотека уже установлена, ее можно обновить до последней версии:

```
$ pip install --upgrade [имя-библиотеки]
```



Начиная с Python 3.4, диспетчер *pip* поставляется вместе с Python. В предыдущих версиях его требуется устанавливать отдельно, как описывается на странице проекта (<https://github.com/pypa/pip>). Данный инструмент можно также использовать для обновления его самого:

```
$ pip install --upgrade pip
```

После установки *virtualenv* в глобальном окружении, появляется возможность определить для каждого проекта свое окружение Python со своими зависимостями, не загрязняя глобальное окружение. Благодаря этому установка зависимостей для каждого отдельного проекта становится чрезвычайно простым делом.

Чтобы установить виртуальную среду, выполните следующие шаги:

```
$ mkdir my_new_project # создать папку для нового проекта
$ cd my_new_project    # перейти в папку проекта
$ virtualenv my_env     # настроить индивидуальную виртуальную среду
```

В результате будет создана подпапка *my_env*, которая является также именем виртуальной среды, созданной в текущем каталоге. В этой подпапке хранятся все инструменты, необходимые для создания изолированной среды Python, включая двоичные файлы Python и стандартную библиотеку. Активировать среду можно следующей командой:

```
$ source my_env/bin/activate
```

После активации появится следующее приглашение к вводу:

```
(my_env) $
```

Теперь диспетчер библиотек `pip` для Python будет устанавливать их в эту конкретную среду:

```
(my_env)$ pip install [имя-библиотеки]
```

Все новые библиотеки Python, устанавливаемые в активированном окружении с помощью `pip`, помещаются в папку `my_env/lib/python{Версия}/site-packages`. Отметим, что для выполнения этой команды не требуются административные привилегии, потому что папка является локальной.

Чтобы деактивировать виртуальное окружение, достаточно выполнить команду:

```
$ deactivate
```

Описанный процесс должен работать с официальными дистрибутивами Python, которые поставляются (или доступны для загрузки) с вашей операционной системой.

Conda, Anaconda u Miniconda

Существует еще один вариант управления зависимостями, под названием **conda** (<http://conda.pydata.org/>), набирающий популярность в научном сообществе. Conda – это диспетчер библиотек с открытым исходным кодом и среды, позволяющий устанавливать несколько версий библиотек (и их зависимостей) и упрощающий переключение между ними. Он поддерживает Linux, macOS и Windows, и, хотя изначально создавался для Python, может использоваться для упаковки и распространения любого программного обеспечения.

Диспетчер `conda` распространяется в виде двух дистрибутивов: полная версия `Anaconda`, включающая около 100 пакетов для научных вычислений, и облегченная версия `Miniconda`, включающая Python и установщик `conda` без внешних библиотек.

Если вы только начинаете изучать Python, имеете немного свободного времени, чтобы загрузить большой дистрибутив, и достаточный объем свободного дискового пространства, и не хотите устанавливать все библиотеки вручную, начните с дистрибутива `Anaconda`. Для Windows и macOS, дистрибутив `Anaconda` доступен с графическим установщиком или с установщиком командной строки. Скриншот рис. 1.5 показывает процесс установки в macOS. Для Linux имеется только установщик командной строки. Во всех случаях есть выбор между Python 2 и Python 3. Если вы предпочтете иметь полный контроль над своей системой, тогда `Miniconda`, скорее всего, будет для вас лучшим выбором.

После установки выбранной версии `conda` можно создать новое окружение `conda`, выполнив команду:

```
$ conda create --name my_env python=3.4 # или предпочитаемая версия
```

Среда активируется командой:

```
$ conda activate my_env
```

Подобно тому, как это происходит в `virtualenv`, в строке приглашения появится имя окружения:

```
(my_env) $
```

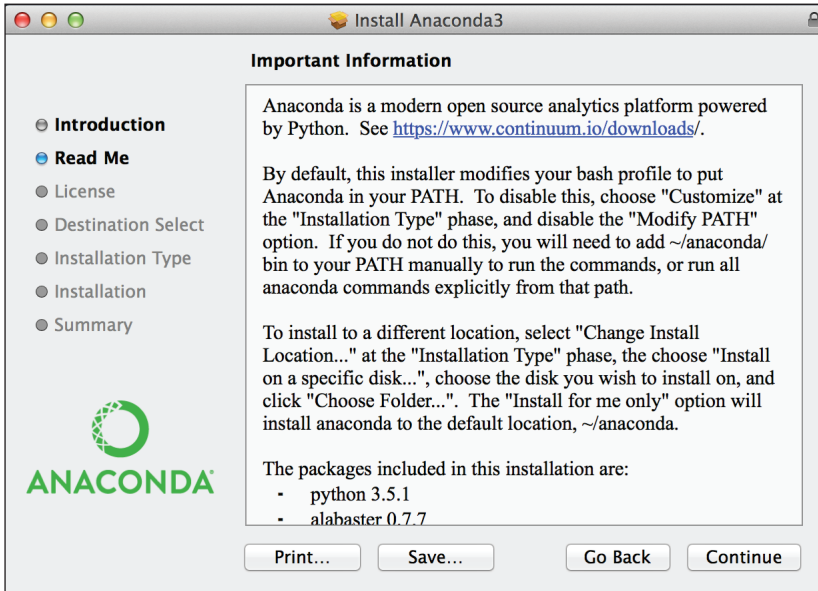


Рис. 1.5 ❖ Процесс установки дистрибутива Anaconda

Новые библиотеки в это окружение можно устанавливать командой:

```
$ conda install [имя-библиотеки]
```

Наконец, деактивировать окружение можно следующей командой:

```
$ conda deactivate
```

Еще одной замечательной чертой `conda` является возможность устанавливать библиотеки с помощью диспетчера `pip`, то есть, если конкретную библиотеку или какую-то ее версию не получается установить командой `conda install`, всегда можно вернуться к традиционному диспетчеру пакетов для Python.

Если не указано обратное, по умолчанию `conda` ищет библиотеки на веб-сайте <https://anaconda.org>, тогда как `pip` использует каталог пакетов Python (Python Package Index, **PyPI**) по адресу: <https://pypi.python.org/pypi>. Неофициально последний иногда называют «сырной лавкой» **CheeseShop**. Оба диспетчера можно настроить на установку библиотек из локальной файловой системы или частного хранилища.

В следующем разделе для установки необходимых библиотек будет использоваться диспетчер `pip`, но вы можете переключиться на `conda`, если предпочитаете использовать данную альтернативу.

Эффективный анализ данных

Этот раздел знакомит с двумя основополагающими библиотеками для научных вычислений на Python: **NumPy** и **pandas**.

Библиотека NumPy (Numerical Python) предлагает инструменты быстрой и эффективной обработки массивоподобных структур данных. Хранение и манипуляция числовыми данными с использованием встроенных в Python структур данных

(к примеру, списков или словарей) оказывается намного медленнее, чем использование массива из NumPy. Кроме того, массивы NumPy часто используются другими библиотеками в качестве контейнеров данных для различных алгоритмов, выполняющих операции с векторами и матрицами.

Чтобы установить NumPy с помощью `pip/virtualenv`, используйте следующую команду:

```
$ pip install numpy
```

Библиотеки NumPy и `pandas` уже входят в состав полной версии Anaconda, поэтому тем, кто пользуется ею, предыдущий шаг установки не потребуется.

Основной структурой данных в NumPy является многомерный массив `ndarray`.

Следующий пример сеанса в интерактивной оболочке Python демонстрирует создание простого массива при помощи NumPy:

```
>>> import numpy as np
>>> data = [1, 2, 3] # список целых чисел
>>> my_arr = np.array(data)
>>> my_arr
array([1, 2, 3])
>>> my_arr.shape
(3,)
>>> my_arr.dtype
dtype('int64')
>>> my_arr.ndim
1
```

Этот пример наглядно показывает, что данные представлены одномерным массивом (атрибут `ndim`) с тремя элементами, как и ожидалось. Массив имеет тип данных `int64`, поскольку на входе все элементы целочисленные.

Быстродействие массива NumPy можно оценить, выполнив профилирование простой операции, например, вычисления суммы, с помощью модуля `timeit`:

```
# Chap01/demo_numpy.py
from timeit import timeit
import numpy as np

if __name__ == '__main__':
    setup_sum = 'data = list(range(10000))'
    setup_np = 'import numpy as np;'
    setup_np += 'data_np = np.array(list(range(10000)))'
    run_sum = 'result = sum(data)'
    run_np = 'result = np.sum(data_np)'

    time_sum = timeit(run_sum, setup=setup_sum, number=10000)
    time_np = timeit(run_np, setup=setup_np, number=10000)

    print("Время выполнения встроенной функции sum(): {}".format(time_sum))
    print("Время выполнения функции np.sum(): {}".format(time_np))
```


Модуль `timeit` принимает в первом параметре фрагмент кода и многократно выполняет его, после чего выводит время, потребовавшееся для этого. Чтобы сосредоточиться на определенной части анализируемого программного кода, настройка исходных данных и необходимые операторы импорта перемещены

в параметр `setup`, который будет выполнен всего один раз и не будет включен в профилирование. Последний параметр – `number` – ограничивает число итераций 10 000 вместо значения по умолчанию, равного 1 000 000. Мы на своем компьютере получили такой результат:

Время выполнения встроенной функции `sum()`: 0.9970562970265746

Время выполнения функции `np.sum()`: 0.07551316602621228

Встроенная функция `sum()` более чем в 10 раз медленнее функции `sum()` из NumPy. Более сложные фрагменты кода могут показывать еще большую разницу.

 **Соглашения об именах.** Сообщество разработчиков на Python пришло к соглашениям, касающимся импортирования некоторых популярных библиотек. NumPy и pandas – два самых известных примера. Они обычно импортируются под псевдонимом, например:

```
import numpy as np
```

При таком подходе, как можно заметить в предыдущих примерах, к функциям из NumPy можно обращаться, как `np.имя_функции()`. Аналогично библиотека pandas импортируется под псевдонимом `pd`. В общем случае импортирование всего пространства имен библиотеки оператором `from numpy import *` считается плохой практикой, потому что в результате загрязняется текущее пространство имен.

Массивы NumPy имеют следующие особенности, о которых следует помнить:

- размер массива NumPy фиксируется при создании, в отличие, к примеру, от списков Python, размер которого может изменяться динамически, поэтому операции, изменяющие размер массива, в действительности создают новый массив и удаляют исходный;
- все элементы массивов должны быть одного типа (кроме массивов объектов, которые, как следствие, могут иметь разные размеры);
- библиотека NumPy помогает использовать операции с векторами и писать более компактный и читаемый код.

Библиотека pandas – вторая из представленных в этом разделе. Она реализована поверх NumPy, поэтому тоже обеспечивает высокую скорость вычислений, и главное, предлагает удобные структуры данных **Series** и **DataFrame** (то есть, числовые ряды и таблицы данных), которые позволяют гибко и сжато манипулировать данными.

Перечислим лишь некоторые из замечательных особенностей библиотеки pandas:

- быстрые и эффективные объекты для манипулирования данными;
- инструменты для чтения и записи данных в разных форматах, таких как CSV, текст, электронные таблицы MS Excel или структуры данных SQL;
- интеллектуальная обработка пропущенных данных и связанное с ней согласование данных;
- создание срезов больших наборов данных на основе меток;
- SQL-подобное агрегирование и преобразование данных;
- поддержка временных рядов;
- интегрированная визуализация данных в виде графиков;
- библиотеку pandas можно установить из «сырной лавки» CheeseShop, как обычно:

```
$ pip install pandas
```