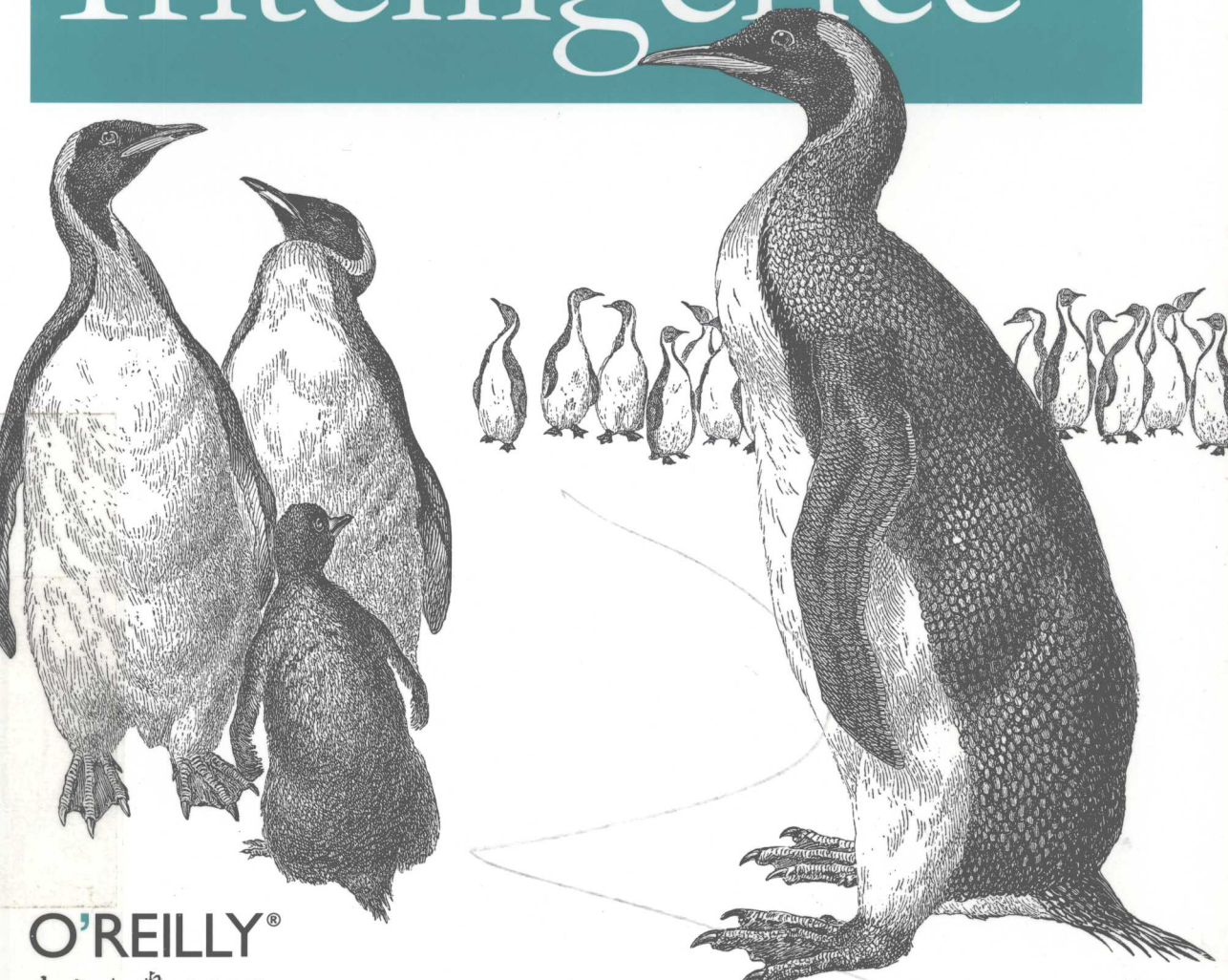
集体智慧编程（影印版）

Programming

Collective Intelligence



O'REILLY®

東南大學出版社

Toby Segaran 著

集体智慧编程(影印版)

Programming Collective Intelligence

Toby Segaran

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

集体智慧编程：英文 /（美）西格兰（Segaran, T.）著. —影印本. — 南京：东南大学出版社，2008.3
书名原文：Programming Collective Intelligence
ISBN 978-7-5641-1139-7

Ⅰ.集… Ⅱ.西… Ⅲ.主页制作－程序设计－英文
Ⅳ.TP393.092

中国版本图书馆 CIP 数据核字（2008）第 024161 号

江苏省版权局著作权合同登记
图字：10-2007-222 号

集体智慧编程

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号　　邮编：210096
出 版 人：江　汉
网　　址：http://press.seu.edu.cn
电子邮件：press@seu.edu.cn
印　　刷：扬中市印刷有限公司
开　　本：787 毫米 × 980 毫米　16 开本
印　　张：22.75 印张
字　　数：378 千字
版　　次：2008 年 3 月第 1 版
印　　次：2008 年 3 月第 1 次印刷
书　　号：ISBN 978-7-5641-1139-7/TP · 183
印　　数：1~2500 册
定　　价：58.00 元（册）

本社图书若有印装质量问题，请直接与读者服务部联系。电话（传真）：025-83792328

O'Reilly Media, Inc. 介绍

为了满足读者对网络和软件技术知识的迫切需求，世界著名计算机图书出版机构O'Reilly Media, Inc. 授权东南大学出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly Media, Inc. 是世界上在 Unix、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时也是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为二十世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc. 一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc. 是最稳定的计算机图书出版商——每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc. 具有深厚的计算机专业背景，这使得O'Reilly Media, Inc. 形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc. 所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc. 还有许多固定的作者群体——他们本身是相关领域的技术专家、咨询专家；而现在编写著作，O'Reilly Media, Inc. 依靠他们及时地推出图书。因为O'Reilly Media, Inc. 紧密地与计算机业界联系着，所以O'Reilly Media, Inc. 知道市场上真正需要什么图书。

出版说明

随着计算机技术的成熟和广泛应用，人类正在步入一个技术迅猛发展的新时期。计算机技术的发展给人们的工业生产、商业活动和日常生活都带来了巨大的影响。然而，计算机领域的技术更新速度之快也是众所周知的，为了帮助国内技术人员在第一时间了解国外最新的技术，东南大学出版社和美国 O'Reilly Meida, Inc.达成协议，将陆续引进该公司的代表前沿技术或者在某专项领域享有盛名的著作，以影印版或者简体中文版的形式呈献给读者。其中，影印版书籍力求与国外图书“同步”出版，并且“原汁原味”展现给读者。

我们真诚地希望，所引进的书籍能对国内相关行业的技术人员、科研机构的研究人员和高校师生的学习和工作有所帮助，对国内计算机技术的发展有所促进。也衷心期望读者提出宝贵的意见和建议。

最新出版的影印版图书，包括：

- 深入浅出 SQL（影印版）
- JavaScript & DHTML Cookbook 第二版（影印版）
- 集体智慧编程（影印版）
- SOA 实践（影印版）
- 学习 PHP & MySQL 第二版（影印版）
- Linux 系统编程（影印版）
- Beautiful Code（影印版）
- Mac OS X: The Missing Manual, Leopard Editon（影印版）
- 高性能网站（影印版）
- WPF 编程 第二版（影印版）
- 敏捷开发艺术（影印版）
- 学习 Python 第三版（影印版）
- Ruby 程序设计语言（影印版）

Foreword

When *Time* magazine picked "You" as the Person of the Year for 2006, it cemented the idea that Web 2.0 is about "user-generated content"—and that Wikipedia, YouTube, and MySpace are the heart of the Web 2.0 revolution. The true story is far more complex than that. The content that users contribute explicitly to Web 2.0 sites is the small fraction that is visible above the surface. Eighty percent of what matters is below, in the dark matter of implicitly contributed data.

In many ways, the defining moment of the Web 2.0 revolution was Google's invention of PageRank, the realization that every link on the World Wide Web was freighted with hidden meaning: a link is a vote about the importance of a site. Understanding those votes, and the relative importance of the sites that were voting, gave better search results than merely studying the web pages themselves. It was the breakthrough that launched Google on its path to becoming the most important tech company of the new century. PageRank is now one of hundreds of implicit factors that Google uses in deciding which search results to feature.

No one would characterize Google as a "user-generated content" company, yet it is clearly at the very heart of Web 2.0. That's why I prefer the phrase "harnessing collective intelligence" as the touchstone of the revolution. A link is user-generated content, but PageRank is a technique for extracting intelligence from that content. So are Flickr's "interestingness" algorithm, Amazon's "people who bought this product also bought..." feature, Last.fm's algorithms for "similar artist radio," eBay's reputation system, and Google's AdSense.

I defined Web 2.0 as "the design of systems that harness network effects to get better the more people use them." Getting users to participate is the first step. Learning from those users and shaping your site based on what they do and pay attention to is the second step.

In *Programming Collective Intelligence*, Toby Segaran teaches algorithms and techniques for extracting meaning from data, including user data. This is the programmer's toolbox for Web 2.0. It's no longer enough to know how to build a

database-backed web site. If you want to succeed, you need to know how to mine the data that users are adding, both explicitly and as a side effect of their activity on your site.

There's been a lot written about Web 2.0 since we first coined the term in 2004, but in many ways, Toby's book is the first practical guide to programming Web 2.0 applications.

—Tim O'Reilly

Preface

The increasing number of people contributing to the Internet, either deliberately or incidentally, has created a huge set of data that gives us millions of potential insights into user experience, marketing, personal tastes, and human behavior in general. This book provides an introduction to the emerging field of *collective intelligence*. It covers ways to get hold of interesting datasets from many web sites you've probably heard of, ideas on how to collect data from users of your own applications, and many different ways to analyze and understand the data once you've found it.

This book's goal is to take you beyond simple database-backed applications and teach you how to write smarter programs to take advantage of the information you and others collect every day.

Prerequisites

The code examples in this book are written in Python, and familiarity with Python programming will help, but I provide explanations of all the algorithms so that programmers of other languages can follow. The Python code will be particularly easy to follow for those who know high-level languages like Ruby or Perl. This book is not intended as a guide for learning programming, so it's important that you've done enough coding to be familiar with the basic concepts. If you have a good understanding of recursion and some basic functional programming, you'll find the material even easier.

This book does not assume you have any prior knowledge of data analysis, machine learning, or statistics. I've tried to explain mathematical concepts in as simple a manner as possible, but having some knowledge of trigonometry and basic statistics will be help you understand the algorithms.

Style of Examples

The code examples in each section are written in a tutorial style, which encourages you to build the applications in stages and get a good appreciation for how the algorithms work. In most cases, after creating a new function or method, you'll use it in an interactive session to understand how it works. The algorithms are mostly simple variants that can be extended in many ways. By working through the examples and testing them interactively, you'll get insights into ways that you might improve them for your own applications.

Why Python?

Although the algorithms are described in words with explanations of the formulae involved, it's much more useful (and probably easier to follow) to have actual code for the algorithms and example problems. All the example code in this book is written in Python, an excellent, high-level language. I chose Python because it is:

Concise
: Code written in dynamically typed languages such as Python tends to be shorter than code written in other mainstream languages. This means there's less typing for you when working through the examples, but it also means that it's easier to fit the algorithm in your head and really understand what it's doing.

Easy to read
: Python has at times been referred to as "executable pseudocode." While this is clearly an exaggeration, it makes the point that most experienced programmers can read Python code and understand what it is supposed to do. Some of the less obvious constructs in Python are explained in the "Python Tips" section below.

Easily extensible
: Python comes standard with many libraries, including those for mathematical functions, XML (Extensible Markup Language) parsing, and downloading web pages. The nonstandard libraries used in the book, such as the RSS (Really Simple Syndication) parser and the SQLite interface, are free and easy to download, install, and use.

Interactive
: When working through an example, it's useful to try out the functions as you write them without writing another program just for testing. Python can run programs directly from the command line, and it also has an interactive prompt that lets you type in function calls, create objects, and test packages interactively.

Multiparadigm
: Python supports object-oriented, procedural, and functional styles of programming. Machine-learning algorithms vary greatly, and the clearest way to

implement one may use a different paradigm than another. Sometimes it's useful to pass around functions as parameters and other times to capture state in an object. Python supports both approaches.

Multiplatform and free
: Python has a single reference implementation for all the major platforms and is free for all of them. The code described in this book will work on Windows, Linux, and Macintosh.

Python Tips

For beginners interested in learning about programming in Python, I recommend reading *Learning Python* by Mark Lutz and David Ascher (O'Reilly), which gives an excellent overview. Programmers of other languages should find the Python code relatively easy to follow, although be aware that throughout this book I use some of Python's idiosyncratic syntax because it lets me more directly express the algorithm or fundamental concepts. Here's a quick overview for those of you who aren't Python programmers:

List and dictionary constructors

Python has a good set of primitive types and two that are used heavily throughout this book are *list* and *dictionary*. A list is an ordered list of any type of value, and it is constructed with square brackets:

```
number_list=[1,2,3,4]
string_list=['a', 'b', 'c', 'd']
mixed_list=['a', 3, 'c', 8]
```

A dictionary is an unordered set of key/value pairs, similar to a hash map in other languages. It is constructed with curly braces:

```
ages={'John':24,'Sarah':28,'Mike':31}
```

The elements of lists and dictionaries can be accessed using square brackets after the list name:

```
string_list[2]   # returns 'b'
ages['Sarah']    # returns 28
```

Significant Whitespace

Unlike most languages, Python actually uses the indentation of the code to define code blocks. Consider this snippet:

```
if x==1:
  print 'x is 1'
  print 'Still in if block'
print 'outside if block'
```

The interpreter knows that the first two print statements are executed when x is 1 because the code is indented. Indentation can be any number of spaces, as long as it is consistent. This book uses two spaces for indentation. When entering the code you'll need to be careful to copy the indentation correctly.

List comprehensions

A *list comprehension* is a convenient way of converting one list to another by filtering and applying functions to it. A list comprehension is written as:

```
[expression for variable in list]
```

or:

```
[expression for variable in list if condition]
```

For example, the following code:

```
l1=[1,2,3,4,5,6,7,8,9]
print [v*10 for v in l1 if v1>4]
```

would print this list:

```
[50,60,70,80,90]
```

List comprehensions are used frequently in this book because they are an extremely concise way to apply a function to an entire list or to remove bad items. The other manner in which they are often used is with the `dict` constructor:

```
l1=[1,2,3,4,5,6,7,8,9]
timesten=dict([(v,v*10) for v in l1])
```

This code will create a dictionary with the original list being the keys and each item multiplied by 10 as the value:

```
{1:10,2:20,3:30,4:40,5:50,6:60,7:70,8:80,9:90}
```

Open APIs

The algorithms for synthesizing collective intelligence require data from many users. In addition to machine-learning algorithms, this book discusses a number of Open Web APIs (application programming interfaces). These are ways that companies allow you to freely access data from their web sites by means of a specified protocol; you can then write programs that download and process the data. This data usually consists of contributions from the site's users, which can be mined for new insights. In some cases, there is a Python library available to access these APIs; if not, it's pretty straightforward to create your own interface to access the data using Python's built-in libraries for downloading data and parsing XML.

Here are some of the web sites with open APIs that you'll see in this book:

del.icio.us
: A social bookmarking application whose open API lets you download links by tag or from a specific user.

Kayak
: A travel site with an API for conducting searches for flights and hotels from within your own programs.

eBay
: An online auction site with an API that allows you to query items that are currently for sale.

Hot or Not
: A rating and dating site with an API to search for people and get their ratings and demographic information.

Akismet
: An API for collaborative spam filtering.

A huge number of potential applications can be built by processing data from a single source, by combining data from multiple sources, and even by combining external information with input from your own users. The ability to harness data created by people in a variety of ways on different sites is a principle element of creating collective intelligence. A good starting point for finding more web sites with open APIs is ProgrammableWeb (*http://www.programmableweb.com*).

Overview of the Chapters

Every algorithm in the book is motivated by a realistic problem that can, I hope, be easily understood by all readers. I have tried to avoid problems that require a great deal of domain knowledge, and I have focused on problems that, while complex, are easy for most people to relate to.

Chapter 1, *Introduction to Collective Intelligence*
: Explains the concepts behind machine learning, how it is applied in many different fields, and how it can be used to draw new conclusions from data gathered from many different people.

Chapter 2, *Making Recommendations*
: Introduces the *collaborative filtering* techniques used by many online retailers to recommend products or media. The chapter includes a section on recommending links to people from a social bookmarking site, and building a move recommendation system from the MovieLens dataset.

Chapter 3, *Discovering Groups*
: Builds on some of the ideas in Chapter 2 and introduces two different methods of *clustering*, which automatically detect groups of similar items in a large dataset. This chapter demonstrates the use of clustering to find groups on a set of popular weblogs and on people's desires from a social networking web site.

Chapter 4, *Searching and Ranking*

Describes the various parts of a search engine including the crawler, indexer, and query engine. It covers the *PageRank* algorithm for scoring pages based on inbound links and shows you how to create a *neural network* that learns which keywords are associated with different results.

Chapter 5, *Optimization*

Introduces algorithms for *optimization*, which are designed to search millions of possible solutions to a problem and choose the best one. The wide variety of uses for these algorithms is demonstrated with examples that find the best flights for a group of people traveling to the same location, find the best way of matching students to dorms, and lay out a network with the minimum number of crossed lines.

Chapter 6, *Document Filtering*

Demonstrates *Bayesian filtering*, which is used in many free and commercial spam filters for automatically classifying documents based on the type of words and other features that appear in the document. This is applied to a set of RSS search results to demonstrate automatic classification of the entries.

Chapter 7, *Modeling with Decision Trees*

Introduces *decision trees* as a method not only of making predictions, but also of modeling the way the decisions are made. The first decision tree is built with hypothetical data from server logs and is used to predict whether or not a user is likely to become a premium subscriber. The other examples use data from real web sites to model real estate prices and "hotness."

Chapter 8, *Building Price Models*

Approaches the problem of predicting numerical values rather than classifications using *k-nearest neighbors* techniques, and applies the optimization algorithms from Chapter 5. These methods are used in conjunction with the eBay API to build a system for predicting eventual auction prices for items based on a set of properties.

Chapter 9, *Advanced Classification: Kernel Methods and SVMs*

Shows how *support-vector machines* can be used to match people in online dating sites or when searching for professional contacts. Support-vector machines are a fairly advanced technique and this chapter compares them to other methods.

Chapter 10, *Finding Independent Features*

Introduces a relatively new technique called *non-negative matrix factorization*, which is used to find the independent features in a dataset. In many datasets the items are constructed as a composite of different features that we don't know in advance; the idea here is to detect these features. This technique is demonstrated on a set of news articles, where the stories themselves are used to detect themes, one or more of which may apply to a given story.

Chapter 11, *Evolving Intelligence*
: Introduces *genetic programming*, a very sophisticated set of techniques that goes beyond optimization and actually builds algorithms using evolutionary ideas to solve a particular problem. This is demonstrated by a simple game in which the computer is initially a poor player that improves its skill by improving its own code the more the game is played.

Chapter 12, *Algorithm Summary*
: Reviews all the machine-learning and statistical algorithms described in the book and compares them to a set of artificial problems. This will help you understand how they work and visualize the way that each of them divides data.

Appendix A, *Third-Party Libraries*
: Gives information on third-party libraries used in the book, such as where to find them and how to install them.

Appendix B, *Mathematical Formulas*
: Contains formulae, descriptions, and code for many of the mathematical concepts introduced throughout the book.

Exercises at the end of each chapter give ideas of ways to extend the algorithms and make them more powerful.

Conventions

The following typographical conventions are used in this book:

Plain text
: Indicates menu titles, menu options, menu buttons, and keyboard accelerators (such as Alt and Ctrl).

Italic
: Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, directories, and Unix utilities.

`Constant width`
: Indicates commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, objects, events, event handlers, XML tags, HTML tags, macros, the contents of files, or the output from commands.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values.

This icon signifies a tip, suggestion, or general note.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Programming Collective Intelligence* by Toby Segaran. Copyright 2007 Toby Segaran, 978-0-596-52932-1."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596529321

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:

http://www.oreilly.com

Safari® Books Online

When you see a Safari® Books Online icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

Acknowledgments

I'd like to express my gratitude to everyone at O'Reilly involved in the development and production of this book. First, I'd like to thank Nat Torkington for telling me that the idea had merit and was worth pitching, Mike Hendrickson and Brian Jepson for listening to my pitch and getting me excited to write the book, and especially Mary O'Brien who took over as editor from Brian and could always assuage my fears that the project was too much for me.

On the production team, I want to thank Marlowe Shaeffer, Rob Romano, Jessamyn Read, Amy Thomson, and Sarah Schneider for turning my illustrations and writing into something you might actually want to look at.

Thanks to everyone who took part in the review of the book, specifically Paul Tyma, Matthew Russell, Jeff Hammerbacher, Terry Camerlengo, Andreas Weigend, Daniel Russell, and Tim Wolters.

Thanks to my parents.

Finally, I owe so much gratitude to several of my friends who helped me brainstorm ideas for the book and who were always understanding when I had no time for them: Andrea Matthews, Jeff Beene, Laura Miyakawa, Neil Stroup, and Brooke Blumenstein. Writing this book would have been much harder without your support and I certainly would have missed out on some of the more entertaining examples.

About the Author

Toby Segaran is a director of software development at Genstruct, a computational biology company, where he designs algorithms and applies data-mining techniques to help understand drug mechanisms. He also works with other companies and open source projects to help them analyze and find value in their collected datasets. In addition, he has built several free web applications including the popular tasktoy and Lazybase. He enjoys snowboarding and wine tasting. His blog is located at *blog.kiwitobes.com*. He lives in San Francisco.

Colophon

The animals on the cover of *Programming Collective Intelligence* are King penguins (*Aptenodytes patagonicus*). Although named for the Patagonia region, King Penguins no longer breed in South America; the last colony there was wiped out by 19th-century sealers. Today, these penguins are found on sub-Antarctic islands such as Prince Edward, Crozet, Macquarie, and Falkland Islands. They live on beaches and flat glacial lands near the sea. King penguins are extremely social birds; they breed in colonies of as many as 10,000 and raise their young in crèches.

Standing 30 inches tall and weighing up to 30 pounds, the King is one of the largest types of penguin—second only to its close relative the Emperor penguin. Apart from size, the major identifying feature of the King penguin is the bright orange patches on its head that extend down to its silvery breast plumage. These penguins have a sleek body frame and can run on land, instead of hopping like Emperor penguins. They are well adapted to the sea, eating a diet of fish and squid, and can dive down 700 feet, far deeper than most other penguins go. Because males and females are similar in size and appearance, they are distinguished by behavioral clues such as mating rituals.

King penguins do not build nests; instead, they tuck their single egg under their bellies and rest it on their feet. No other bird has a longer breeding cycle than these penguins, who breed twice every three years and fledge a single chick. The chicks are round, brown, and so fluffy that early explorers thought they were an entirely different species of penguin, calling them "woolly penguins." With a world population of two million breeding pairs, King penguins are not a threatened species, and the World Conservation Union has assigned them to the Least Concern category.

The cover image is from J. G. Wood's *Animate Creation*. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed.