Extracted from:

Text Processing with Ruby

Extract Value from the Data That Surrounds You

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Text Processing with Ruby

Extract Value from the Data That Surrounds You

Rob Miller

edited by Jacquelyn Carter

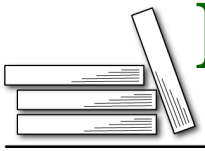
Text Processing with Ruby

Extract Value from the Data That Surrounds You

Rob Miller

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Pragmatic Bookshelf

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *https://pragprog.com*.

The team that produced this book includes:

Jacquelyn Carter (editor)
Potomac Indexing, LLC (index)
Cathleen Small; Liz Welch (copyedit)
Dave Thomas (layout)
Janet Furlow (producer)
Ellie Callahan (support)

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-68050-070-7
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—September 2015

Introduction

Text is everywhere. Newspaper articles, database dumps, spreadsheets, the output of shell commands, keyboard input; it's all text, and it can all be processed in the same fundamental way. Text has been called "the universal interface," and since the early days of Unix in the 1960s this universal interface has survived and flourished—and with good reason.

Unlike binary formats, text has the pleasing quality of being readable by humans as well as computers, making it easy to debug and requiring no distinction between output that's for human consumption and output that's to be used as the input for another step in a process.

Processing text, then, is a valuable skill for any programmer today—just as it was fifty years ago, and just as it's likely to be fifty years hence. In this book I hope to provide a practical guide to all the major aspects of working with text, viewed through the lens of the Ruby programming language—a language that I think is ideally suited to this task.

About This Book

Processing text is generally concerned with three things. The first concern is acquiring the text to be processed and getting it into your program. This is the subject of Part I of this book, which deals with reading from plain text files, standard input, delimited files, and binary files such as PDFs and Word documents.

This first part is fundamentally an exploration of Ruby's core and standard library, and what's possible with `IO` and its derived classes like `File`. Ruby's history and design, and the high-level nature of these tasks, mean that we don't need to dip into third-party libraries much, but we'll use one in particular—Nokogiri—when looking at scraping data from web pages.

The second concern is with actually processing the text once we've got it into the program. This usually means either extracting data from within the text, parsing it into a Ruby data structure, or transforming it into another format.

The most important subject in this second stage is, without a doubt, regular expressions. We'll look at regular expression syntax, how Ruby uses regular expressions in particular, and, importantly, when *not* to use them and instead reach for solutions such as parsers.

We'll also look at the subject of natural language processing in this part of the book, and how we can use tools from computational linguistics to make our programs smarter and to process data that we otherwise couldn't.

The final step is outputting the transformed text or the extracted data somewhere—to a file, to a network service, or just to the screen. Part of this process is concerned with the actual writing process, and part of it is concerned with the form of the written data. We'll look at both of these aspects in the third part of the book.

Together, these three steps are often described as "extract, transform, and load" (ETL). It's a term especially popular with the "big data" folks. Many text processing tasks, even ones that seem on the surface to be very different from one another, fall into this pattern of three steps, so I've tried to mirror that structure in the book.

In general, we're going to explore why Ruby is an excellent tool to reach for when working with text. I also hope to persuade you that you might reach for Ruby sooner than you think—not necessarily just for more complex tasks, but also for quick one-liners.

Most of all, I hope this book offers you some useful techniques that help you in your day-to-day programming tasks. Where possible, I've erred toward the practical rather than the theoretical: if it does anything, I'd like this book to point you in the direction of practical solutions to real-world problems. If your day job is anything like mine, you probably find yourself trawling through text files, CSVs, and command-line output more often than you might like. Helping to make that process quick and—dare I say it?—fun would be fantastic.

Who This Book Is For

Throughout the book, I try not to assume an advanced understanding of Ruby. If you're familiar with Ruby's syntax—perhaps after having dabbled with Rails a little—then that should be enough to get by. Likewise, if Ruby is your first programming language and you're looking to learn about data processing, you should be able to pick things up as you go along—though naturally this book is about teaching text processing more than it is about teaching Ruby.

While the book starts with material likely to be familiar to anyone who's written a command-line application in Ruby, there's still something here for the more advanced user. Even people who've worked with Ruby a lot aren't necessarily aware of the material covered in Chapter 3, *Shell One-Liners*, on page ?, for example, and I see far too many developers reaching for regular expressions to parse HTML rather than using the techniques outlined in Chapter 6, *Scraping HTML*, on page ?.

Even experienced developers might not have written parsers before (covered in Chapter 10, *Writing Parsers*, on page ?), or dabbled in natural language processing (as we do in Chapter 11, *Natural Language Processing*, on page ?)—so hopefully those subjects will be interesting regardless of your level of experience.

How to Read This Book

Although the book follows a structure of extractions first, transformations second, and loading third, the chapters are relatively self-contained and can be read in any order you wish—so feel free to dive into a later chapter if you're particularly interested in the material it covers.

I've tried to include in each of the chapters material of interest even to more advanced Rubyists, so there aren't any chapters that are obvious candidates to skip if you're at that end of the skill spectrum.

If you're not familiar with how to use the command line, there's a beginner's tutorial in Appendix 1, *A Shell Primer*, on page ?, and a guide to various commands in Appendix 2, *Useful Shell Commands*, on page ?. These appendixes will give you more than enough command-line knowledge to follow all of the examples in the book.

About the Code

All of the code samples in the book can be downloaded from the book's website.[1] They've been tested in Ruby 2.2 running on OS X, Linux, and Cygwin on Microsoft Windows, but they should run just fine on any version of Ruby after 2.0 (released in February 2013).

The book assumes that you're running in a Unix-like environment. Users of Mac OS X, Linux, and BSD will be right at home. Microsoft Windows users, though, will only be able to get the most out of some sections of the book by

1. https://pragprog.com/book/rmtpruby/text-processing-with-ruby

installing Cygwin.[2] Cygwin provides a Unix-like environment on Windows, including a full command-line environment and Unix shell. This gives Windows users access to the core text processing utilities referenced in this book. This is particularly true of the chapters on shell one-liners, writing flexible filters with ARGF, and writing to other processes.

Online Resources

The page for this book on the Pragmatic Bookshelf website[3] contains a discussion forum, where you can post any comments or questions you might have about the book and make suggestions for any changes or expansions you'd like to see in future editions. If you discover any errors in the book, you can submit them there, too.

Rob Miller
August 2015

2. https://www.cygwin.com/
3. https://pragprog.com/book/rmtpruby/text-processing-with-ruby