Natural Language Processing with Python

Steven Bird, Ewan Klein, and Edward Loper

O'REILLY®

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

Natural Language Processing with Python
by Steven Bird, Ewan Klein, and Edward Loper

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editor: Julie Steele
Production Editor: Loranah Dimant
Copyeditor: Genevieve d'Entremont
Proofreader: Loranah Dimant

Indexer: Ellen Troutman Zaig
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Printing History:
June 2009: First Edition.

ISBN: 978-0-596-51649-9

[M]

1244726609

Table of Contents

Preface . . . . **ix**

1. Language Processing and Python **1**
1.1 Computing with Language: Texts and Words 1
1.2 A Closer Look at Python: Texts as Lists of Words 10
1.3 Computing with Language: Simple Statistics 16
1.4 Back to Python: Making Decisions and Taking Control 22
1.5 Automatic Natural Language Understanding 27
1.6 Summary 33
1.7 Further Reading 34
1.8 Exercises 35

2. Accessing Text Corpora and Lexical Resources **39**
2.1 Accessing Text Corpora 39
2.2 Conditional Frequency Distributions 52
2.3 More Python: Reusing Code 56
2.4 Lexical Resources 59
2.5 WordNet 67
2.6 Summary 73
2.7 Further Reading 73
2.8 Exercises 74

3. Processing Raw Text **79**
3.1 Accessing Text from the Web and from Disk 80
3.2 Strings: Text Processing at the Lowest Level 87
3.3 Text Processing with Unicode 93
3.4 Regular Expressions for Detecting Word Patterns 97
3.5 Useful Applications of Regular Expressions 102
3.6 Normalizing Text 107
3.7 Regular Expressions for Tokenizing Text 109
3.8 Segmentation 112
3.9 Formatting: From Lists to Strings 116

3.10 Summary 121
3.11 Further Reading 122
3.12 Exercises 123

4. Writing Structured Programs 129
4.1 Back to the Basics 130
4.2 Sequences 133
4.3 Questions of Style 138
4.4 Functions: The Foundation of Structured Programming 142
4.5 Doing More with Functions 149
4.6 Program Development 154
4.7 Algorithm Design 160
4.8 A Sample of Python Libraries 167
4.9 Summary 172
4.10 Further Reading 173
4.11 Exercises 173

5. Categorizing and Tagging Words 179
5.1 Using a Tagger 179
5.2 Tagged Corpora 181
5.3 Mapping Words to Properties Using Python Dictionaries 189
5.4 Automatic Tagging 198
5.5 N-Gram Tagging 202
5.6 Transformation-Based Tagging 208
5.7 How to Determine the Category of a Word 210
5.8 Summary 213
5.9 Further Reading 214
5.10 Exercises 215

6. Learning to Classify Text 221
6.1 Supervised Classification 221
6.2 Further Examples of Supervised Classification 233
6.3 Evaluation 237
6.4 Decision Trees 242
6.5 Naive Bayes Classifiers 245
6.6 Maximum Entropy Classifiers 250
6.7 Modeling Linguistic Patterns 254
6.8 Summary 256
6.9 Further Reading 256
6.10 Exercises 257

7. Extracting Information from Text 261
7.1 Information Extraction 261

7.2 Chunking 264
7.3 Developing and Evaluating Chunkers 270
7.4 Recursion in Linguistic Structure 277
7.5 Named Entity Recognition 281
7.6 Relation Extraction 284
7.7 Summary 285
7.8 Further Reading 286
7.9 Exercises 286

8. Analyzing Sentence Structure 291
8.1 Some Grammatical Dilemmas 292
8.2 What's the Use of Syntax? 295
8.3 Context-Free Grammar 298
8.4 Parsing with Context-Free Grammar 302
8.5 Dependencies and Dependency Grammar 310
8.6 Grammar Development 315
8.7 Summary 321
8.8 Further Reading 322
8.9 Exercises 322

9. Building Feature-Based Grammars 327
9.1 Grammatical Features 327
9.2 Processing Feature Structures 337
9.3 Extending a Feature-Based Grammar 344
9.4 Summary 356
9.5 Further Reading 357
9.6 Exercises 358

10. Analyzing the Meaning of Sentences 361
10.1 Natural Language Understanding 361
10.2 Propositional Logic 368
10.3 First-Order Logic 372
10.4 The Semantics of English Sentences 385
10.5 Discourse Semantics 397
10.6 Summary 402
10.7 Further Reading 403
10.8 Exercises 404

11. Managing Linguistic Data 407
11.1 Corpus Structure: A Case Study 407
11.2 The Life Cycle of a Corpus 412
11.3 Acquiring Data 416
11.4 Working with XML 425

11.5 Working with Toolbox Data 431
11.6 Describing Language Resources Using OLAC Metadata 435
11.7 Summary 437
11.8 Further Reading 437
11.9 Exercises 438

Afterword: The Language Challenge 441

Bibliography 449

NLTK Index 459

General Index 463

Preface

This is a book about Natural Language Processing. By "natural language" we mean a language that is used for everyday communication by humans; languages such as English, Hindi, or Portuguese. In contrast to artificial languages such as programming languages and mathematical notations, natural languages have evolved as they pass from generation to generation, and are hard to pin down with explicit rules. We will take Natural Language Processing—or NLP for short—in a wide sense to cover any kind of computer manipulation of natural language. At one extreme, it could be as simple as counting word frequencies to compare different writing styles. At the other extreme, NLP involves "understanding" complete human utterances, at least to the extent of being able to give useful responses to them.

Technologies based on NLP are becoming increasingly widespread. For example, phones and handheld computers support predictive text and handwriting recognition; web search engines give access to information locked up in unstructured text; machine translation allows us to retrieve texts written in Chinese and read them in Spanish. By providing more natural human-machine interfaces, and more sophisticated access to stored information, language processing has come to play a central role in the multilingual information society.

This book provides a highly accessible introduction to the field of NLP. It can be used for individual study or as the textbook for a course on natural language processing or computational linguistics, or as a supplement to courses in artificial intelligence, text mining, or corpus linguistics. The book is intensely practical, containing hundreds of fully worked examples and graded exercises.

The book is based on the Python programming language together with an open source library called the *Natural Language Toolkit* (NLTK). NLTK includes extensive software, data, and documentation, all freely downloadable from *http://www.nltk.org/*. Distributions are provided for Windows, Macintosh, and Unix platforms. We strongly encourage you to download Python and NLTK, and try out the examples and exercises along the way.

Audience

NLP is important for scientific, economic, social, and cultural reasons. NLP is experiencing rapid growth as its theories and methods are deployed in a variety of new language technologies. For this reason it is important for a wide range of people to have a working knowledge of NLP. Within industry, this includes people in human-computer interaction, business information analysis, and web software development. Within academia, it includes people in areas from humanities computing and corpus linguistics through to computer science and artificial intelligence. (To many people in academia, NLP is known by the name of "Computational Linguistics.")

This book is intended for a diverse range of people who want to learn how to write programs that analyze written language, regardless of previous programming experience:

New to programming?
: The early chapters of the book are suitable for readers with no prior knowledge of programming, so long as you aren't afraid to tackle new concepts and develop new computing skills. The book is full of examples that you can copy and try for yourself, together with hundreds of graded exercises. If you need a more general introduction to Python, see the list of Python resources at *http://docs.python.org/*.

New to Python?
: Experienced programmers can quickly learn enough Python using this book to get immersed in natural language processing. All relevant Python features are carefully explained and exemplified, and you will quickly come to appreciate Python's suitability for this application area. The language index will help you locate relevant discussions in the book.

Already dreaming in Python?
: Skim the Python examples and dig into the interesting language analysis material that starts in Chapter 1. You'll soon be applying your skills to this fascinating domain.

Emphasis

This book is a **practical** introduction to NLP. You will learn by example, write real programs, and grasp the value of being able to test an idea through implementation. If you haven't learned already, this book will teach you **programming**. Unlike other programming books, we provide extensive illustrations and exercises from NLP. The approach we have taken is also **principled**, in that we cover the theoretical underpinnings and don't shy away from careful linguistic and computational analysis. We have tried to be **pragmatic** in striking a balance between theory and application, identifying the connections and the tensions. Finally, we recognize that you won't get through this unless it is also **pleasurable**, so we have tried to include many applications and examples that are interesting and entertaining, and sometimes whimsical.

Note that this book is not a reference work. Its coverage of Python and NLP is selective, and presented in a tutorial style. For reference material, please consult the substantial quantity of searchable resources available at *http://python.org/* and *http://www.nltk.org/*.

This book is not an advanced computer science text. The content ranges from introductory to intermediate, and is directed at readers who want to learn how to analyze text using Python and the Natural Language Toolkit. To learn about advanced algorithms implemented in NLTK, you can examine the Python code linked from *http://www.nltk.org/*, and consult the other materials cited in this book.

What You Will Learn

By digging into the material presented here, you will learn:

- How simple programs can help you manipulate and analyze language data, and how to write these programs
- How key concepts from NLP and linguistics are used to describe and analyze language
- How data structures and algorithms are used in NLP
- How language data is stored in standard formats, and how data can be used to evaluate the performance of NLP techniques

Depending on your background, and your motivation for being interested in NLP, you will gain different kinds of skills and knowledge from this book, as set out in Table P-1.

Table P-1. Skills and knowledge to be gained from reading this book, depending on readers' goals and background

Goals	Background in arts and humanities	Background in science and engineering
Language analysis	Manipulating large corpora, exploring linguistic models, and testing empirical claims.	Using techniques in data modeling, data mining, and knowledge discovery to analyze natural language.
Language technology	Building robust systems to perform linguistic tasks with technological applications.	Using linguistic algorithms and data structures in robust language processing software.

Organization

The early chapters are organized in order of conceptual difficulty, starting with a practical introduction to language processing that shows how to explore interesting bodies of text using tiny Python programs (Chapters 1–3). This is followed by a chapter on structured programming (Chapter 4) that consolidates the programming topics scattered across the preceding chapters. After this, the pace picks up, and we move on to a series of chapters covering fundamental topics in language processing: tagging, classification, and information extraction (Chapters 5–7). The next three chapters look at ways to parse a sentence, recognize its syntactic structure, and construct representations of meaning (Chapters 8–10). The final chapter is devoted to linguistic data and how it can be managed effectively (Chapter 11). The book concludes with an Afterword, briefly discussing the past and future of the field.

Within each chapter, we switch between different styles of presentation. In one style, natural language is the driver. We analyze language, explore linguistic concepts, and use programming examples to support the discussion. We often employ Python constructs that have not been introduced systematically, so you can see their purpose before delving into the details of how and why they work. This is just like learning idiomatic expressions in a foreign language: you're able to buy a nice pastry without first having learned the intricacies of question formation. In the other style of presentation, the programming language will be the driver. We'll analyze programs, explore algorithms, and the linguistic examples will play a supporting role.

Each chapter ends with a series of graded exercises, which are useful for consolidating the material. The exercises are graded according to the following scheme: ○ is for easy exercises that involve minor modifications to supplied code samples or other simple activities; ◑ is for intermediate exercises that explore an aspect of the material in more depth, requiring careful analysis and design; ● is for difficult, open-ended tasks that will challenge your understanding of the material and force you to think independently (readers new to programming should skip these).

Each chapter has a further reading section and an online "extras" section at *http://www.nltk.org/*, with pointers to more advanced materials and online resources. Online versions of all the code examples are also available there.

Why Python?

Python is a simple yet powerful programming language with excellent functionality for processing linguistic data. Python can be downloaded for free from *http://www.python.org/*. Installers are available for all platforms.

Here is a five-line Python program that processes *file.txt* and prints all the words ending in `ing`:

```
>>> for line in open("file.txt"):
...     for word in line.split():
...         if word.endswith('ing'):
...             print word
```

This program illustrates some of the main features of Python. First, whitespace is used to *nest* lines of code; thus the line starting with `if` falls inside the scope of the previous line starting with `for`; this ensures that the `ing` test is performed for each word. Second, Python is *object-oriented*; each variable is an entity that has certain defined attributes and methods. For example, the value of the variable `line` is more than a sequence of characters. It is a string object that has a "method" (or operation) called `split()` that