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 like 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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

New to Python?:	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/`. 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 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 learnt 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, 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 analyse 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 I.1.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Table I.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 learnt 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?

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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 we can use to break a line into its words. To apply a method to an object, we write the object name, followed by a period, followed by the method name, i.e. `line.split()`. Third, methods have *arguments* expressed inside parentheses. For instance, in the example, `word.endswith('ing')` had the argument `'ing'` to indicate that we wanted words ending with *ing* and not something else. Finally — and most importantly — Python is highly readable, so much so that it is fairly easy to guess what the program does even if you have never written a program before.

We chose Python because it has a shallow learning curve, its syntax and semantics are transparent, and it has good string-handling functionality. As an interpreted language, Python facilitates interactive exploration. As an object-oriented language, Python permits data and methods to be encapsulated and re-used easily. As a dynamic language, Python permits attributes to be added to objects on the fly, and permits variables to be typed dynamically, facilitating rapid development. Python comes with an extensive standard library, including components for graphical programming, numerical processing, and web connectivity.

Python is heavily used in industry, scientific research, and education around the world. Python is often praised for the way it facilitates productivity, quality, and maintainability of software. A collection of Python success stories is posted at `http://www.python.org/about/success/`.

NLTK defines an infrastructure that can be used to build NLP programs in Python. It provides basic classes for representing data relevant to natural language processing; standard interfaces for performing tasks such as part-of-speech tagging, syntactic parsing, and text classification; and standard implementations for each task which can be combined to solve complex problems.

NLTK comes with extensive documentation. In addition to this book, the website at `http://www.nltk.org/` provides API documentation that covers every module, class and function in the toolkit, specifying parameters and giving examples of usage. The website also provides many HOWTOs with extensive examples and test cases, intended for users, developers and instructors.

Software Requirements

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

To get the most out of this book, you should install several free software packages. Current download pointers and instructions are available at `http://www.nltk.org/`.

Python:	The material presented in this book assumes that you are using Python version 2.4 or 2.5. We are committed to porting NLTK to Python 3.0 once the libraries that NLTK depends on have been ported.
NLTK:	The code examples in this book use NLTK version 2.0. Subsequent releases of NLTK will be backward-compatible.
NLTK-Data:	This contains the linguistic corpora that are analyzed and processed in the book.
NumPy:	(recommended) This is a scientific computing library with support for multidimensional arrays and linear algebra, required for certain probability, tagging, clustering, and classification tasks.
Matplotlib:	(recommended) This is a 2D plotting library for data visualization, and is used in some of the book's code samples that produce line graphs and bar charts.
NetworkX:	(optional) This is a library for storing and manipulating network structures consisting of nodes and edges. For visualizing semantic networks, also install the *Graphviz* library.
Prover9:	(optional) This is an automated theorem prover for first-order and equational logic, used to support inference in language processing.

Natural Language Toolkit (NLTK)

NLTK was originally created in 2001 as part of a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania. Since then it has been developed and expanded with the help of dozens of contributors. It has now been adopted in courses in dozens of universities, and serves as the basis of many research projects. See I.2 for a list of the most important NLTK modules.

Table I.2:

Language processing tasks and corresponding NLTK modules with examples of functionality

Language processing task	NLTK modules	Functionality
Accessing corpora	nltk.corpus	standardized interfaces to corpora and lexicons
String processing	nltk.tokenize, nltk.stem	tokenizers, sentence tokenizers, stemmers
Collocation discovery	nltk.collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	nltk.tag	n-gram, backoff, Brill, HMM, TnT
Classification	nltk.classify, nltk.cluster	decision tree, maximum entropy, naive Bayes, EM, k-means
Chunking	nltk.chunk	regular expression, n-gram, named-entity
Parsing	nltk.parse	chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	nltk.sem, nltk.inference	lambda calculus, first-order logic, model checking
Evaluation metrics	nltk.metrics	precision, recall, agreement coefficients
Probability and estimation	nltk.probability	frequency distributions, smoothed probability distributions

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Language processing task	NLTK modules	Functionality
Applications	nltk.app, nltk.chat	graphical concordancer, parsers, WordNet browser, chatbots
Linguistic fieldwork	nltk.toolbox	manipulate data in SIL Toolbox format

NLTK was designed with four primary goals in mind:

Simplicity:	To provide an intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annotated language data
Consistency:	To provide a uniform framework with consistent interfaces and data structures, and easily-guessable method names
Extensibility:	To provide a structure into which new software modules can be easily accommodated, including alternative implementations and competing approaches to the same task
Modularity:	To provide components that can be used independently without needing to understand the rest of the toolkit

Contrasting with these goals are three non-requirements — potentially useful qualities that we have deliberately avoided. First, while the toolkit provides a wide range of functions, it is not encyclopedic; it is a toolkit, not a system, and it will continue to evolve with the field of NLP. Second, while the toolkit is efficient enough to support meaningful tasks, it is not highly optimized for runtime performance; such optimizations often involve more complex algorithms, or implementations in lower-level programming languages such as C or C++. This would make the software less readable and more difficult to install. Third, we have tried to avoid clever programming tricks, since we believe that clear implementations are preferable to ingenious yet indecipherable ones.

For Instructors

Natural Language Processing is often taught within the confines of a single-semester course at advanced undergraduate level or postgraduate level. Many instructors have found that it is difficult to cover both the theoretical and practical sides of the subject in such a short span of time. Some courses focus on theory to the exclusion of practical exercises, and deprive students of the challenge and excitement of writing programs to automatically process language. Other courses are simply designed to teach programming for linguists, and do not manage to cover any significant NLP content. NLTK was originally developed to address this problem, making it feasible to cover a substantial amount of theory and practice within a single-semester course, even if students have no prior programming experience.

A significant fraction of any NLP syllabus deals with algorithms and data structures. On their own these can be rather dry, but NLTK brings them to life with the help of interactive graphical user interfaces that make it possible to view algorithms step-by-step. Most NLTK components include a demonstration that performs an interesting task without requiring any special input from the user. An effective way to deliver the materials is through interactive presentation of the examples in this book, entering them in a Python session, observing what they do, and modifying them to explore some empirical or theoretical issue.

This book contains hundreds of exercises that can be used as the basis for student assignments. The simplest exercises involve modifying a supplied program fragment in a specified way in order to answer a concrete question. At the other end of the spectrum, NLTK provides a flexible framework for graduate-level research projects, with standard implementations of all the basic data structures and algorithms, interfaces to dozens of widely used datasets (corpora), and a flexible and extensible architecture. Additional support for teaching using NLTK is available on the NLTK website.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

We believe this book is unique in providing a comprehensive framework for students to learn about NLP in the context of learning to program. What sets these materials apart is the tight coupling of the chapters and exercises with NLTK, giving students — even those with no prior programming experience — a practical introduction to NLP. After completing these materials, students will be ready to attempt one of the more advanced textbooks, such as *Speech and Language Processing*, by Jurafsky and Martin (Prentice Hall, 2008).

This book presents programming concepts in an unusual order, beginning with a non-trivial data type — lists of strings — then introducing non-trivial control structures such as comprehensions and conditionals. These idioms permit us to do useful language processing from the start. Once this motivation is in place, we return to a systematic presentation of fundamental concepts such as strings, loops, files, and so forth. In this way, we cover the same ground as more conventional approaches, without expecting readers to be interested in the programming language for its own sake.

Two possible course plans are illustrated in I.3. The first one presumes an arts/humanities audience, whereas the second one presumes a science/engineering audience. Other course plans could cover the first five chapters, then devote the remaining time of time to a single area, such as text classification (Chapters 6-7), syntax (Chapters 8-9), semantics (Chapter 10), or linguistic data management (Chapter 11).

Table I.3:

Suggested course plans; approximate number of lectures per chapter

Chapter	Arts and Humanities	Science and Engineering
1 Language Processing and Python	2-4	2
2 Accessing Text Corpora and Lexical Resources	2-4	2
3 Processing Raw Text	2-4	2
4 Writing Structured Programs	2-4	1-2
5 Categorizing and Tagging Words	2-4	2-4
6 Learning to Classify Text	0-2	2-4
7 Extracting Information from Text	2	2-4
8 Analyzing Sentence Structure	2-4	2-4
9 Building Feature Based Grammars	2-4	1-4
10 Analyzing the Meaning of Sentences	1-2	1-4
11 Managing Linguistic Data	1-2	1-4
Total	18-36	18-36

Conventions Used in This Book

The following typographical conventions are used in this book:

Bold -- Indicates new terms.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Italic -- Used within paragraphs to refer to linguistic examples, the names of texts, and URLs; also used for filenames and file extensions.

`Constant width` -- Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, statements, and keywords; also used for program names.

`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 or by values determined by context; also used for metavariables within program code examples.

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

Caution!

This icon indicates a warning or caution.

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 youÕre 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 OÕReilly 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 productÕs documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: ÒNatural Language Processing with Python, by Steven Bird, Ewan Klein, and Edward Loper. O'Reilly Media, 978-0-596-51649-9.Ó 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*.

Acknowledgments

The authors are indebted to the following people for feedback on earlier drafts of this book: Doug Arnold, Michaela Atterer, Greg Aumann, Kenneth Beesley, Steven Bethard, Ondrej Bojar, Chris Cieri, Robin Cooper, Grev Corbett, James Curran, Dan Garrette, Jean Mark Gawron, Doug Hellmann, Nitin Indurkhya,

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Mark Liberman, Peter Ljunglöf, Stefan Müller, Robin Munn, Joel Nothman, Adam Przepiorkowski, Brandon Rhodes, Stuart Robinson, Jussi Salmela, Kyle Schlansker, Rob Speer, and Richard Sproat. We are thankful to many students and colleagues for their comments on the class materials that evolved into these chapters, including participants at NLP and linguistics summer schools in Brazil, India, and the USA. This book would not exist without the members of the `nltk-dev` developer community, named on the NLTK website, who have given so freely of their time and expertise in building and extending NLTK.

We are grateful to the U.S. National Science Foundation, the Linguistic Data Consortium, an Edward Clarence Dyason Fellowship, and the Universities of Pennsylvania, Edinburgh, and Melbourne for supporting our work on this book.

We thank Julie Steele, Abby Fox, Loranah Dimant, and the rest of the O'Reilly team, for organizing comprehensive reviews of our drafts from people across the NLP and Python communities, for cheerfully customizing O'Reilly's production tools, and for meticulous copy-editing work.

Finally, we owe a huge debt of gratitude to our partners, Kay, Mimo, and Jee, for their love, patience, and support over the many years that we worked on this book. We hope that our children — Andrew, Alison, Kirsten, Leonie, and Maaike — catch our enthusiasm for language and computation from these pages.

About the Authors

Steven Bird is Associate Professor in the Department of Computer Science and Software Engineering at the University of Melbourne, and Senior Research Associate in the Linguistic Data Consortium at the University of Pennsylvania. He completed a PhD on computational phonology at the University of Edinburgh in 1990, supervised by Ewan Klein. He later moved to Cameroon to conduct linguistic fieldwork on the Grassfields Bantu languages under the auspices of the Summer Institute of Linguistics. More recently, he spent several years as Associate Director of the Linguistic Data Consortium where he led an R&D team to create models and tools for large databases of annotated text. At Melbourne University, he established a language technology research group and has taught at all levels of the undergraduate computer science curriculum. In 2009, Steven is President of the Association for Computational Linguistics.

Ewan Klein is Professor of Language Technology in the School of Informatics at the University of Edinburgh. He completed a PhD on formal semantics at the University of Cambridge in 1978. After some years working at the Universities of Sussex and Newcastle upon Tyne, Ewan took up a teaching position at Edinburgh. He was involved in the establishment of Edinburgh's Language Technology Group in 1993, and has been closely associated with it ever since. From 2000–2002, he took leave from the University to act as Research Manager for the Edinburgh-based Natural Language Research Group of Edify Corporation, Santa Clara, and was responsible for spoken dialogue processing. Ewan is a past President of the European Chapter of the Association for Computational Linguistics and was a founding member and Coordinator of the European Network of Excellence in Human Language Technologies (ELSNET).

Edward Loper has recently completed a PhD on machine learning for natural language processing at the the University of Pennsylvania. Edward was a student in Steven's graduate course on computational linguistics in the fall of 2000, and went on to be a TA and share in the development of NLTK. In addition to NLTK, he has helped develop two packages for documenting and testing Python software, `epydoc` and `doctest`.

Royalties

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Royalties from the sale of this book are being used to support the development of the Natural Language Toolkit.

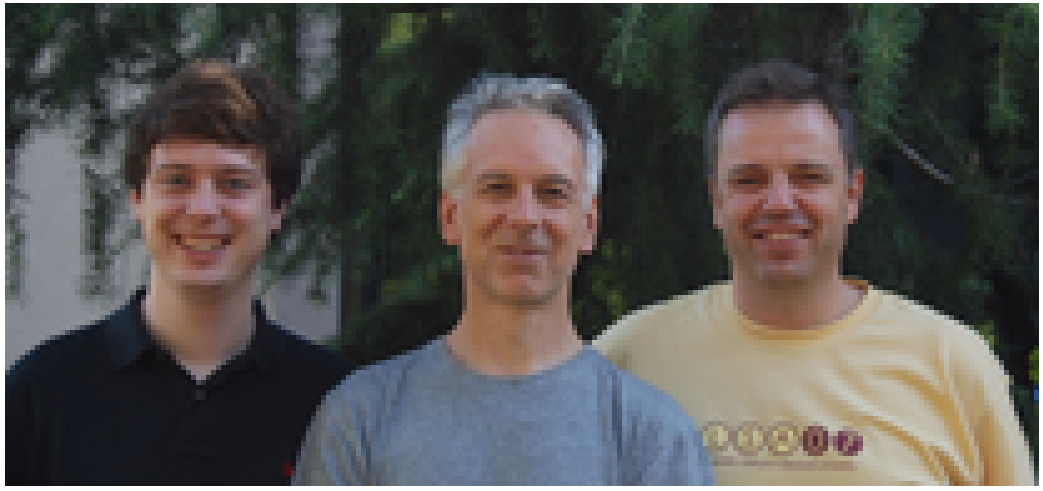

Figure I.1: Edward Loper, Ewan Klein, and Steven Bird, Stanford, July 2007

About this document...

This is a chapter from *Natural Language Processing with Python*, by Steven Bird, Ewan Klein and Edward Loper, Copyright © 2009 the authors. It is distributed with the *Natural Language Toolkit* [`http://www.nltk.org/`], Version 2.0.1rc1, under the terms of the *Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States License* [http://creativecommons.org/licenses/by-nc-nd/3.0/us/].

This document was built on Mon 15 Oct 2012 16:46:09 EST

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD