Finite-State Language Processing

edited by
Emmanuel Roche and Yves Schabes

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London England

This book was set in Computer Modern by the editors and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Finite-State Language Processing / edited by Emmanuel Roche and Yves Schabes.
p. cm. —(Language, speech, and communication)
"A Bradford book."
Includes index.
ISBN 0-262-18182-7 (hc: alk. paper)
1. Natural language processing (Computer science). I. Roche, Emmanuel. II. Schabes, Yves. III. Series.
QA76.9.N38F56 1997
006.3/5/015113—dc2 96-48159
CIP

Finite-State Language Processing

Language, Speech, and Communication

Statistical Language Learning, Eugene Charniak, 1994

The Development of Speech Perception, edited by Judith Goodman and Howard C. Nusbaum, 1994

Construal, Lyn Frazier and Charles Clifton, Jr., 1995

The Generative Lexicon, James Pustejovsky,1996

The Origins of Grammar: Evidence from Early Language Comprehension, Kathy Hirsh-Pasek and Roberta Michnick Golinkoff, 1996

Language and Space, edited by Paul Bloom, Mary A. Peterson, Lynn Nadel, and Merrill F. Garrett, 1996

Corpus Processing for Lexical Acquisition, edited by Branimir Boguraev and James Pustejovsky, 1996

Methods for Assessing Children's Syntax, edited by Dana McDaniel, Cecile McKee, and Helen Smith Cairns, 1996

The Balancing Act: Combining Symbolic and Statistical Approaches to Language, edited by Judith Klavans and Philip Resnik, 1996

The Discovery of Spoken Language, Peter W. Jusczyk, 1996

Lexical Competence, Diego Marconi, 1997

Finite-State Language Processing, edited by Emmanuel Roche and Yves Schabes, 1997

Preface

The theory of finite-state automata is rich, and finite-state automata techniques are used in a wide range of domains, including switching theory, pattern matching, pattern recognition, speech processing, handwriting recognition, optical character recognition, encryption algorithm, data compression, indexing, and operating system analysis (e.g., Petri-net).

Finite-state devices, such as finite-state automata, graphs, and finite-state transducers, have been present since the emergence of computer science and are extensively used in areas as various as program compilation, hardware modeling, and database management. Although finite-state devices have been known for some time in computational linguistics, more powerful formalisms such as context-free grammars or unification grammars have typically been preferred. However, recent mathematical and algorithmic results in the field of finite-state technology have had a great impact on the representation of electronic dictionaries and on natural language processing. As a result, a new technology for language is emerging out of both industrial and academic research. This book, a discussion of fundamental finite-state algorithms, constitutes an approach from the perspective of natural language processing.

The book is organized as follows.

Chapter 1. *Emmanuel Roche and Yves Schabes.* In this introductory chapter, the basic notions of finite-state automata and finite-state transducers are described. The fundamental properties of these machines are described and illustrated with simple formal language examples as well as natural language examples. This chapter also explains the main algorithms used with finite-state automata and transducers.

Chapter 2. *David Clemenceau. Finite-State Morphology: Inflections and Derivations in a single Framework using Dictionaries and Rules.* This chapter describes how finite-state techniques can be used to encode a large scale morphological lexicon where productive derivational rules apply. The productive nature of derivational morphology is a source of complications for morphological analysis. Many words are not listed in any static morphological

dictionary, no matter how big or accurate it is. In other words, the size of the lexicon is infinite. Many unknown words fall into the category of words derived from a stem and inflectional affixes. For example, the word *reanalysizable* is most likely not found in any dictionary. However, this word is part of the language in the sense that it can be used and it is easily understandable. In this chapter the author shows how finite-state transducers can be used naturally to handle this problem. In addition, Clemenceau shows that the use of a large dictionary and derivational rules lead to a homogeneous and efficient representation using finite-state transducers.

Chapter 3. *Kimmo Koskenniemi. Representations and Finite-state Components in Natural Language.* Koskenniemi describes a formal system called "two-level rules" which encodes finite-state transducers. The declarative rules hold in parallel between the phenomenon in question and its analysis. Each two-level rule is then compiled to a finite-state transducer and the set of rules are combined using the intersection operation on finite-state transducers. Koskenniemi illustrates the applicability of two-level rules to morphology and syntax.

Chapter 4. *Lauri Karttunen. The Replace Operator.* This chapter introduces a replace operator to the calculus of regular expressions and defines a set of replacement expressions that concisely encode several alternate variations of the operation. Replace expressions denote regular relations, defined in terms of other regular-expression operators. The basic case is unconditional obligatory replacement. This chapter develops several versions of conditional replacement that allow the operation to be constrained by context.

Chapter 5. *Fernando C. N. Pereira and Rebecca N. Wright. Finite-State Approximation of Phrase-Structure Grammars.* Phrase-structure grammars are effective models for important syntactic and semantic aspects of natural languages, but can be computationally too demanding for use as language models in real-time speech recognition. Therefore, finite-state models are used instead, even though they lack expressive power. To reconcile those two alternatives, the authors design an algorithm to compute finite-state approximations of context-free grammars and context-free-equivalent augmented phrase-structure grammars. The approximation is exact for certain context-free grammars generating regular languages, including all left-linear and right-linear context-free grammars. The algorithm has been used to build finite-state language models for limited-domain speech recognition tasks.

Chapter 6. *Max D. Silberztein. The Lexical Analysis of Natural Languages.* This chapter shows how finite-state techniques can be used in the lexical analysis of natural language text. The task of lexically analyzing text goes well beyond the recognition of single words. In this approach, complex words such as the compound *hard disk* are analyzed and marked in the lexical analysis process. Max Silberztein shows how simple words, complex words (such as

compound words) and local syntactic rules can be encoded within the same finite-state framework. In addition, the chapter also shows how local finite-state grammars can be used in the task of disambiguating the output of the lexical analysis of text.

Chapter 7. *Emmanuel Roche and Yves Schabes. Deterministic Part-of-Speech Tagging with Finite-State Transducers.* Using the problem of part-of-speech tagging, this chapter illustrates the use of a cascade of finite-state transducers combined with composition to perform part-of-speech tagging. It also illustrates the use of the determinization algorithm for finite-state transducers. In this chapter, a finite-state tagger inspired by Eric Brill's rule-based tagger is presented. It operates in optimal time in the sense that the time to assign tags to a sentence corresponds to the time required to deterministically follow a single path in a deterministic finite-state machine. This result is achieved by encoding the application of the rules found in the tagger as a non-deterministic finite-state transducer and then turning it into a deterministic transducer. The resulting deterministic transducer yields a part-of-speech tagger whose speed is dominated by the access time of mass storage devices. The results presented in this paper are more general than part-of-speech tagging since it is shown that, in general, transformation-based systems can be turned into subsequential finite-state transducers.

Chapter 8. *Emmanuel Roche. Parsing with Finite-State Transducers.* This chapter describes how finite-state transducers can be used for natural language parsing.

Chapter 9. *Atro Voutilainen. Designing a (Finite-State) Parsing Grammar.* The author shows how the intersection of finite-state automata provide an efficient way to implement a parser. In this approach, phrase boundaries (such as markers that specify the beginning and the end of a noun phrase) are encoded in the local rules represented with finite-state machines. This chapter discusses the design of a finite-state parsing grammar from a linguistic point of view. Attention is paid to the specification of the grammatical representation that can be viewed as the linguistic task definition of the parser, and to the design of the parsing grammar and the heuristic data-driven component. Illustrations are given from an English grammar.

Chapter 10. *Pasi Tapanainen. Applying a Finite-State Intersection Grammar.* This chapter deals with the potential quadratic explosion of the intersection algorithm for finite-state automata. Given an input sentence and a syntactic grammar encoded by a set of finite-state automata, the result of applying the grammar to the input string can be seen as the intersection of the input with all of the grammar rules. Although the final result of this intersection is small, intermediate results can explode in size. In addition, a precompilation of the grammar into one single finite-state automaton yields an automaton whose size cannot be practically managed. The author shows how different

ordering schemes can be used to guarantee that the intermediate results of the intersection do not explode in size.

Chapter 11. *Maurice Gross. The Construction of Local Grammars.* Gross observes that while a systematic categorization of the objects to be studied is an important part of sciences such as biology or astronomy, such categorizations are rare to nonexistent in the field of linguistics. Gross's goal is to account for all the possible sentences within a given corpus and beyond. This chapter gives examples where the finite constraints encoded with finite-state automata can be exhaustively described in a local way, that is, without interferences from the rest of the grammar. These examples demonstrate that a cumulative approach to the construction of a grammar is indeed possible.

Chapter 12. *Mehryar Mohri. On the Use of Sequential Transducers in Natural Language Processing.* This chapter considers the use of a type of transducers that support very efficient programs: deterministic or sequential transducers. It examines several areas of computational linguistics such as morphology, phonology and speech processing. For each, the author briefly describes and discusses the time and space advantages offered by these transducers.

Chapter 13. *Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. FASTUS: A Cascaded Finite-state Transducer for Extracting Information from Natural-Language Text.* This chapter illustrates the technique of combining a cascade of finite-state transducers with composition. The authors show how such a cascade can be used to build a system for extracting information from free text in English, and potentially other languages. The finite-state automata have been built and tested using a corpus of news articles and transcripts of radio broadcasts on Latin American terrorism. The resulting system called **FASTUS** (a slightly permuted acronym for Finite-State Automaton Text Understanding System) is able to fill templates recording, among other things, the perpetrators and victims of each terrorist act, the occupations of the victims, the type of physical entity attacked or destroyed, the date, the location, and the effect on the targets. FASTUS has been very successful in practice. The system is an order of magnitude faster than any comparable system that does not take advantage of the finite-state techniques. Moreover, within a very short development time, state-of-the-art performance can be achieved. FASTUS has been shown to be very competitive with other systems in competitions organized by the Message Understanding Conference.

Chapter 14. *Éric Laporte. Rational Transductions for Phonetic Conversion and Phonology.* Phonetic conversion, and other conversion problems related to phonetics, can be performed by finite-state tools. This chapter presents a finite-state conversion system, BiPho, based on transducers and bimachines. The linguistic data used by this system are described in a readable format and

actual computation is efficient. The system constitutes a spelling-to-phonetics conversion system for French.

Chapter 15. *Fernando C. N. Pereira and Michael D. Riley. Speech Recognition by Composition of Weighted Finite Automata.* This chapter presents a general framework based on weighted finite automata and weighted finite-state transducers for describing and implementing speech recognizers. The framework allows us to represent uniformly the information sources and data structures used in recognition, including context-dependent units, pronunciation dictionaries, language models and lattices. Furthermore, general but efficient algorithms can used for combining information sources in actual recognizers and for optimizing their application. In particular, a single *composition* algorithm is used both to combine in advance information sources such as language models and dictionaries and to combine acoustic observations and information sources dynamically during recognition.

Acknowledgments

The editors would like to thank several people who helped us to bring this project to fruition. We thank MERL - A Mitsubishi Electric Research Laboratory, Information Technology America, Cambridge, USA. We also thank the anonymous referees for their advice regarding revisions of this book. And finally, we thank professor Stuart Shieber and the students at Harvard University who took part to the 1996 seminar on "Engineering Approaches to Natural Language Processing" for valuable comments on previous versions of this volume.

Contents

Preface xi

Acknowledgments xvii

1 Introduction
Emmanuel Roche and Yves Schabes **1**
1.1 Preliminaries 1
1.2 Finite-State Automata 3
1.3 Finite-State Transducers 14
1.4 Bibliographical Notes 63

2 Finite-State Morphology: Inflections and Derivations in a Single Framework Using Dictionaries and Rules
David Clemenceau **67**
2.1 Introduction 67
2.2 Towards a Structured Dictionary 69
2.3 MORPHO: a Morphological Analyzer Based on a Dictionary and a Two-Level System 81
2.4 A Single Framework for Inflections and Derivations Recognition and Generation 91
2.5 Conclusion 96

3 Representations and Finite-State Components in Natural Language
Kimmo Koskenniemi **99**
3.1 A Framework 99
3.2 Two-Level Morphology 101
3.3 Finite-State Syntactic Grammar 108
3.4 Experiences 114

4 The Replace Operator
Lauri Karttunen **117**
4.1 Introduction 117

4.2 Unconditional Replacement 121
4.3 Conditional Replacement 128
4.4 Comparisons 144
4.5 Conclusion 146

5 Finite-State Approximation of Phrase-Structure Grammars
Fernando C. N. Pereira and Rebecca N. Wright **149**
5.1 Motivation 149
5.2 The Approximation Method 150
5.3 Formal Properties 155
5.4 Implementation and Example 163
5.5 Informal Analysis 166
5.6 Related Work and Conclusions 168

6 The Lexical Analysis of Natural Languages
Max D. Silberztein **175**
6.1 The Lexical Analysis of Programming Languages and of Natural Languages 175
6.2 The Units of Analysis 178
6.3 The Representation of Simple Words 180
6.4 Representation of Compound Words 189
6.5 Representation of the Results of the Analysis, Elimination of Ambiguities by Local Grammars 193
6.6 Tagging Programs and Lexical Analysis 198
6.7 Conclusion 201

7 Deterministic Part-of-Speech Tagging with Finite-State Transducers
Emmanuel Roche and Yves Schabes **205**
7.1 Introduction 205
7.2 Overview of Brill's Tagger 207
7.3 Complexity of Brill's Tagger 209
7.4 Construction of the Finite-State Tagger 210
7.5 Lexical Tagger 215
7.6 Tagging Unknown Words 216
7.7 Empirical Evaluation 217
7.8 Finite-State Transducers 218
7.9 Determinization 223
7.10 Subsequentiality of Transformation-Based Systems 233
7.11 Implementation of Finite-State Transducers 236

7.12 Acknowledgments 237
7.13 Conclusion 237

8 Parsing with Finite-State Transducers
Emmanuel Roche **241**
8.1 Introduction 241
8.2 Background 243
8.3 A Top-Down Parser for Context-Free Grammars 244
8.4 Morphology 247
8.5 A Parser for Transformation Grammars 254
8.6 Finite-State Acceleration 268
8.7 A Transducer Parser for Tree-Adjoining Grammars 271
8.8 Conclusion 278

9 Designing a (Finite-State) Parsing Grammar
Atro Voutilainen **283**
9.1 Introduction 283
9.2 Framework 284
9.3 Grammatical Representation 288
9.4 Sample Rule 293
9.5 Heuristic Techniques 298
9.6 Final Remarks 303

10 Applying a Finite-State Intersection Grammar
Pasi Tapanainen **311**
10.1 Introduction 311
10.2 Straight Intersection 313
10.3 Sequential Methods 314
10.4 Parallel Intersection 320
10.5 Hybrid Intersection-Search Method 321
10.6 A Small Comparison 322
10.7 Theoretical Worst-Case Study 323
10.8 Conclusion 326

11 The Construction of Local Grammars
Maurice Gross **329**
11.1 Introduction 329
11.2 Linguistic Modules 334
11.3 Transformations 349
11.4 Conclusion 352

12 On the Use of Sequential Transducers in Natural Language Processing
Mehryar Mohri **355**
12.1 Introduction 355
12.2 Definitions 356
12.3 Characterization and Extensions 359
12.4 Phonology and Morphology 364
12.5 Representation of Large Dictionaries 365
12.6 Syntax 372
12.7 Speech Processing 375
12.8 Conclusion 378
12.9 Acknowledgments 378

13 FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text
Jerry R. Hobbs et al. **383**
13.1 Introduction 383
13.2 The Information Extraction Task 384
13.3 The Finite-State Approach 388
13.4 Overview of the FASTUS Architecture 389
13.5 Complex Words 391
13.6 Basic Phrases 391
13.7 Complex Phrases 393
13.8 Domain Events 395
13.9 Merging Structures 399
13.10 History of the FASTUS System 400
13.11 Conclusions 402

14 Rational Transductions for Phonetic Conversion and Phonology
Éric Laporte **407**
14.1 Introduction 407
14.2 An Introductory Example 408
14.3 Transductions Related to Phonetics 410
14.4 Construction of the Transductions 413
14.5 Mathematical Properties 418
14.6 Implementation 422
14.7 Conclusion 428

15 Speech Recognition by Composition of Weighted Finite Automata
Fernando C. N. Pereira and Michael D. Riley **431**
15.1 Introduction 431
15.2 Theory 434

15.3 Speech Recognition 442
15.4 Implementation 445
15.5 Applications 446
15.6 Further Work 447

Contributors **455**

Index **457**

1 Introduction

Emmanuel Roche and Yves Schabes

The theory of finite-state automata is rich, and finite-state automata techniques are used in a wide range of domains, including switching theory, pattern matching, pattern recognition, speech processing, handwriting recognition, optical character recognition, encryption algorithm, data compression, indexing, and operating system analysis (e.g., Petri-net).

Finite-state devices, such as finite-state automata, graphs, and finite-state transducers, have been present since the emergence of computer science and are extensively used in areas as various as program compilation, hardware modeling, and database management. Although finite-state devices have been known for some time in computational linguistics, more powerful formalisms such as context-free grammars or unification grammars have typically been preferred. However, recent mathematical and algorithmic results in the field of finite-state technology have had a great impact on the representation of electronic dictionaries and on natural language processing. As a result, a new technology for language is emerging out of both industrial and academic research. This book, a discussion of fundamental finite-state algorithms, constitutes an approach from the perspective of natural language processing.

In this chapter, we describe the fundamental properties of finite-state automata and finite-state transducers, and we illustrate the use of these machines through simple formal language examples as well as natural language examples. We also illustrate some of the main algorithms used in connection with finite-state automata and transducers.

1.1 Preliminaries

Finite-state automata (FSAs) and finite-state transducers (FSTs) are the two main concepts used in this book. Both kinds of machines operate on strings or, in other words, on sequences of symbols. Since the notion of string is so prevalent, in this section we define this concept as well the notations that are used throughout this book.

Strings are built out of an alphabet. An alphabet is simply a set of symbols or characters, and can be finite (as is the English alphabet) or infinite (as is the set of the real numbers). A string is a finite sequence of symbols. The set of strings built on an alphabet Σ is also called the *free monoid* Σ^*. Several notations facilitate the manipulations of strings. For example, depending on the context, either *word* or $w \cdot o \cdot r \cdot d$ denotes the following sequence:

$$(a_i)_{i=1,4} = (w, o, r, d) \tag{1.1}$$

In addition, $\cdot$ denotes the concatenation of strings defined as follows:

$$(a_i)_{i=1,n} \cdot (b_j)_{j=1,m} = (c_i)_{i=1,n+m} \tag{1.2}$$

with

$$c_i = \begin{cases} a_i & \text{if } i \leq n \\ b_{i-n} & n+1 \leq i \leq n+m \end{cases}$$

However, this notation is rarely used in practice. Instead, $wo \cdot rd$ or simply $word$ denotes the concatenation of "*wo*" and "*rd*". The empty string, that is the string containing no character, is denoted by ϵ. The empty string is the neutral element for the concatenation operation. Expressed formally, for a string $w \in \Sigma^*$:

$$w \cdot \epsilon = \epsilon \cdot w = w \tag{1.3}$$

Given two strings u and v, $u \wedge v$ denotes the string that is the longest common prefix of u and v.

Another notion important for the understanding of FSAs and FSTs is the notion of sets of strings. Concatenation, union, intersection, subtraction, and complementation are operations commonly used on sets of strings.

If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ are two sets of strings, the concatenation of L_1 and L_2 is defined as follows:

$$L_1 \cdot L_2 = \{u \cdot v | u \in L_1 \text{and } v \in L_2\} \tag{1.4}$$

The following notations are often used, for any string $u \in \Sigma^*$ and any set $L \subseteq \Sigma^*$: