Book Review

Finite-State Text Processing

Kyle Gorman and Richard Sproat
(Graduate Center, City University of New York & Google LLC)

Morgan & Claypool (Synthesis Lectures on Synthesis Lectures on Human Language Technologies, edited by Graeme Hirst, volume 50), 2021, xvii+140 pp; paperback, ISBN: 9781636391137; ebook, ISBN: 9781636391144; hardcover, ISBN: 9781636391151, doi: 10.2200/S01086ED1V01Y202104HLT050

Reviewed by
Aniello De Santo
Department of Linguistics, University of Utah

The rise in popularity of neural network methods in computational linguistics has led to a richness of valuable books on the topic. On the other hand, there is arguably a shortage of recent materials on foundational computational linguistics methods like finite-state technologies. This is unfortunate, as finite-state approaches not only still find much use in applications for speech and text processing (the core focus of this book), but seem also to be valuable in interpreting and improving neural methods. Moreover, the study of finite-state machines (and their corresponding formal languages) is still proving insightful in theoretical linguistics analyses. In this sense, this book by Gorman and Sproat is a welcome, refreshing contribution aimed at a variety of readers.

The book is organized in eight main chapters, and can be conceptually divided into two parts. The first half of the book serves as an introduction to core concepts in formal language and automata theory (Chapter 1), the basic design principles of the Python library used through the book (Chapter 2), and a variety of finite-state algorithms (Chapters 3 and 4). The rest of the book exemplifies the formal notions of the previous chapters with practical applications of finite-state technologies to linguistic problems like morphophonological analysis and text normalization (Chapter 5, 6, 7). The last chapter (Chapter 8) presents an interesting discussion of future steps from a variety of perspectives, connecting finite-state methods to current trends in the field. In what follows, I discuss the contents of each chapter in more detail.

Chapter 1 provides an accessible introduction to formal languages and automata theory, starting with a concise but helpful historical review of the development of finite-state technologies and their ties to linguistics. The chapter balances intuitive explanations of technical concepts without sacrificing formal rigor, in particular in the presentation of finite-state automata/transducers and their formal definitions. Weighted finite-state automata play a prominent role in the book and the authors introduce them algebraically through the notion of semiring (Pin 1997). This is a welcome choice that makes the book somewhat unique among introductory/application-oriented materials on these topics. Unsurprisingly, this section is the densest part of the chapter and it could have benefitted from an explanation of the intuition behind the connection between wellformedness of a string, recognition by an automaton, and paths over semirings—especially considering how relevant such concepts become when the book transitions

https://doi.org/10.1162/coli_r_00466

into its algorithmic parts. However, the chapter lays down these theoretical notions clearly, and its stands as an intriguing starting point for any reader interested in learning more about the mathematics of monoids and formal languages.

Chapter 2 is essentially a short manual for Pynini (Gorman 2016), the Python library used in the rest of the book. The reader will appreciate how the design choices behind Pynini are grounded in the formal notions introduced in the previous chapter. In this sense, the chapter can surely be used as a quick and dry reference list of Python commands, but it goes beyond that and it would serve as a nice pedagogical tool to highlight connections between implementational choices and mathematical definitions—a perk that carries on consistently throughout the book.

Chapters 3 and 4 are a walkthrough of key algorithms for finite-state machines. Chapter 3 focuses on essential operations such as concatenation, closure, union, composition, and so forth. The generalization of each algorithm from finite-state machines to weighted finite-state machines is also highlighted. Chapter 4 presents slightly more advanced algorithms, focusing, for example, on the problem of optimization. The optimization section comes with a nice practical example from speech recognition, and speech serves as a good motivation to lead into algorithms to compute shortest distance and shortest paths. One possible complaint is that the discussion of a few of these algorithms could have been improved with a step-by-step example of the procedure, although examples of these can be easily found in other resources. Overall, it is in these chapters that the uniqueness of the book begins to shine, as the algorithms are presented starting from the theoretical automaton operation at their core, then illustrated by constructing the relevant automata, and finally exemplified with code snippets in Pynini.

Chapter 5 presents a discussion of phonological rewrite rules in the *Sound Pattern of English* (SPE) tradition (Chomsky and Halle 1968). The chapter is grounded in the history of rewrite rules in linguistics broadly, and then focuses on the characteristics of SPE rules and their connection to rational relations and finite-state transducers. The authors introduce Pynini's approach to the implementation of operations like rule compilation, application, and mechanisms for rule interaction. Then, the chapter presents three practical use cases to exemplify the usefulness of rewrite rules: Spanish grapheme-to-phoneme conversion, vowel harmony Finnish case suffixed, and text normalization within the context of currency expression tagging. All three examples lend themselves to be used as practice exercises, should the reader be inclined to, but more importantly well illustrate the advantages of finite-state technology when approaching problems of this type.

Chapter 6 continues on the line of reasoning started in the previous chapter. Here, the authors focus on motivating the use of finite-state approaches to morphology, once again adding a richness of insightful historical references. Readers not familiar with morphological analysis will still find the chapter easy to follow, as standard terminology is introduced together with theoretical and practical concerns. As in the previous chapter, this one ends with a few examples of Pynini scripts for specific linguistic problems, such as Russian noun morphology, Tagalog infixation, and Yowlume verbal morphology. The Yowlume data are particularly interesting, and succeed in displaying both the expressivity of finite-state transducers and the flexibility with which Pynini can handle them.

Chapter 7 concludes the series of application-oriented chapters, with an overview of use cases including, among others, fuzzy string matching and text generation. In particular, the characterization of noise in terms of transducers across a few different applications is going to be informative for readers new to these technologies. The examples presented in this chapter are also more "practical" than the ones in chapters 5

and 6, concerned less with theoretical analyses and more with solving problems that might arise for existing applications (e.g., T9 disambiguation). In this sense, the three chapters balance each other quite nicely, and illustrate a broad range of possible domains of use for the concepts introduced in the first half of the book.

Finally, Chapter 8 discusses the future of finite-state approaches in the landscape of current trends in computational linguistics (i.e., deep learning methods). The authors pragmatically acknowledge the question of the relevance of finite-state technologies today, and clearly map connections between the concepts in the book and neural methods. In line with the approach taken in the rest of the book, the chapter alternates between practical and broader conceptual questions, including a relevant survey of ongoing work on hybridization and possible open venues of research in the theory of formal languages.

Several appendixes make up the coda of the book, with pointers to Pynini extensions, additional implemented examples, and nice troubleshooting tidbits. Additionally, each chapter comes with its own list of suggested further readings, expanding on a generally impressive list of references.

In sum, the book provides a valuable introduction to finite-state technologies, with an eye to practical software implementation. Alternating linguistic examples with formal definitions and code samples makes both the formal notation and Pynini's syntactic conventions accessible to a casual reader, and in doing so the authors highlight the importance of understanding theoretical foundations when approaching implementational problems. The well-constructed synthesis of these different perspectives makes the book appropriate for a variety of readers and goals. I can imagine it used, for instance, as a pedagogical tool for introductory courses, a go-to manual for practitioners needing a quick software reference, and a practical guide for theoretically minded researchers dipping their toes in software implementation.

References

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. Harper and Row.

Gorman, Kyle. 2016. Pynini: A Python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 75–80.

Pin, Jean-Eric. 1997. Syntactic semigroups. In *Handbook of Formal Languages*, pages 679–746. Springer.

Aniello De Santo is an Assistant Professor in the Linguistics Department at the University of Utah. His research interests lie at the intersection of computational, theoretical, and experimental linguistics. His recent work mainly focuses on symbolic computational models of human sentence processing, and on formal language theoretical approaches to the study of linguistics dependencies. His e-mail address is aniello.desanto@utah.edu.