Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Survey: Finite-state technology in natural language processing ☆

Andreas Maletti

Institute for Natural Language Processing, Universität Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany

ARTICLE INFO

Article history:
Received 30 November 2015
Received in revised form 19 April 2016
Accepted 19 May 2016
Available online xxxx

Keywords:
Finite-state automaton
Tree automaton
Context-free grammar
Natural language processing
Tokenization
Part-of-speech tagging
Parsing
Machine translation

ABSTRACT

In this survey, we will discuss current uses of finite-state information in several statistical natural language processing tasks. To this end, we will review standard approaches in tokenization, part-of-speech tagging, and parsing, and illustrate the utility of finite-state information and technology in these areas. The particular problems were chosen to allow a natural progression from simple prediction to structured prediction. We aim for a sufficiently formal presentation suitable for readers with a background in automata theory that allows to appreciate the contribution of finite-state approaches, but we will not discuss practical issues outside the core ideas. We provide instructive examples and pointers into the relevant literature for all constructions. We close with an outlook on finite-state technology in statistical machine translation.

1. Introduction

Finite-state technology has played a major part in the development of several state-of-the-art tools in natural language processing [1,2]. In this survey, we will recall three major problems in the domain of natural language processing, whose state-of-the-art solutions have successfully utilized finite-state technology. Several other areas (e.g., computational morphology) have benefitted from finite-state technology as well, so we necessarily had to select. We chose the mentioned topics because they allow a nice progression from simple prediction to structured prediction, and in particular, require no deep linguistic background. In general, this survey is written for the theoretical computer scientist, so concepts from the area of formal languages are introduced rather tersely. Additional room is provided for the explanation of the application problems, their solutions in terms of automata, and their evaluation. References to the relevant literature are provided in the content sections, which are supposed to be rather self-contained, although we reuse some methodology in the parsing section.

In all three applications, an inital model is extracted from annotated (positive) training data using standard methods in supervised training. This initial model is then transformed into an automaton for further processing. In the area of tokenization, the finite-state technology essentially delivers the operations required to work with the extracted model. We illustrate this on the basic approach, in which we model a scorer for fully annotated data. Using standard finite-state operations we can obtain a scorer for unannotated data from this scorer. In the area of part-of-speech tagging we follow a similar approach and first model a scorer for the fully annotated data. However, in the next step we show how such a model can be optimized to given unannotated training data with the help of the well-known forward–backward algorithm.

☆ Research financially supported by the German Research Foundation (DFG) grant MA 4959 / 1-1.
E-mail address: maletti@ims.uni-stuttgart.de.

http://dx.doi.org/10.1016/j.tcs.2016.05.030
0304-3975/

In this unsupervised optimization step, the structure of the automaton remains the same, but the transition weights are adjusted to better fit the unannotated data. Practical considerations like overfitting the data (i.e., the over-adjustment of the automaton such that it exactly represents only the training data) will not be discussed to allow a clean presentation.

In the last part, we will discuss the classical parsing problem for natural languages. Whereas we discuss weighted finite-state automata in the first two application areas, we will mostly talk about weighted context-free grammars and weighted tree automata in this section. A preliminary section is provided to recall those basic notions. As before, we use supervised training to obtain an initial (local) model from annotated training data. Next we demonstrate a successful approach that combines the weight adjustment procedure, which we already discussed for part-of-speech tagging, with a change of the structure of the automaton. In this case, the process derives truly hidden finite-state information that is not present in the fully annotated training data, so this optimization is again unsupervised. This automatic deduction of hidden finite-state information outperforms all comparable efforts to manually provide additional annotation for parsing models in the training data. We conclude with a short outlook on the use of finite-state information in statistical machine translation. In this area, finite-state models are used to model the translation process, but typically local variants, which do not actually use the power of "hidden" finite-state information (i.e., they only use clues from the part of the input currently under consideration), are actually used. Some minor uses have been demonstrated in the literature, but the majority of the used models remains essentially local.

2. String automata

2.1. Basic notions

We denote the set of nonnegative integers by $\mathbb{N}$. An alphabet is simply a finite set. The set of all sequences (or words) over the alphabet Σ is denoted by Σ^*, of which $\varepsilon \in \Sigma^*$ is the empty word. The length of a word $w \in \Sigma^*$ is denoted by $|w|$, and we write $w(i)$ to denote the i-th letter in w for all $1 \le i \le |w|$. Given $1 \le i \le j \le |w|$, we let $w[i, j] = w(i) \cdots w(j)$ be the substring from position i to j in w. We will use semirings [3,4] as weight structures. A semiring is an algebraic structure $(S, +, \cdot, 0, 1)$ that consists of two monoids $(S, +, 0)$ and $(S, \cdot, 1)$ such that (i) the additive monoid $(S, +, 0)$ is commutative and (ii) the generalized distributivity law holds for both directions; i.e.,

$$\Big(\sum_{i=1}^{k} s_i\Big) \cdot s = \sum_{i=1}^{k} (s_i \cdot s) \qquad \text{and} \qquad s \cdot \Big(\sum_{i=1}^{k} s_i\Big) = \sum_{i=1}^{k} (s \cdot s_i)$$

for all $k \in \mathbb{N}$ and $s, s_1, \ldots, s_k \in S$. The semiring $(S, +, \cdot, 0, 1)$ is commutative if its multiplicative monoid $(S, \cdot, 1)$ is commutative. Two commutative semirings will be particularly relevant for this contribution:

- the semiring $(\mathbb{Q}_{\ge 0}, +, \cdot, 0, 1)$ of nonnegative rational numbers together with the usual operations and
- the semiring $([0, 1], \max, \cdot, 0, 1)$ of the (rational) unit interval together with the maximum (as addition) and the usual multiplication.

In the following, let $(S, +, \cdot, 0, 1)$ be a semiring. Given a mapping $f\colon A \to S$, we let $\operatorname{supp}(f) = \{a \in A \mid f(a) \neq 0\}$.

Finite-state automata [5] have been extended already in the 1960s to handle weights [6]. The recent handbook [7] provides an in-depth discussion of the developed theory and its applications (also NLP applications in chapter 14 [8]). A weighted string automaton [over the semiring $(S, +, \cdot, 0, 1)$] is a tuple $\mathcal{A} = (Q, \Sigma, I, \mathrm{wt}, F)$, where Q is a finite set of states, Σ an alphabet, $I\colon Q \to S$ is an initial weight assignment, $\mathrm{wt}\colon Q \times \Sigma \times Q \to S$ is a transition weight assignment, and $F\colon Q \to S$ is a final weight assignment. The labeled graph underlying the transition weight assignment is $\operatorname{supp}(\mathrm{wt}) = \{\langle q, \sigma, p\rangle \mid \mathrm{wt}(q, \sigma, p) \neq 0\}$. The automaton $\mathcal{A}$ is acyclic if the underlying graph $\operatorname{supp}(\mathrm{wt})$ has no cycle; i.e., there exists no sequence $\langle q_0, \sigma_1, q_1\rangle, \ldots, \langle q_{k-1}, \sigma_k, q_k\rangle$ of elements of $\operatorname{supp}(\mathrm{wt})$ such that $q_0 = q_k$. Moreover, it is deterministic if $I(q) \neq 0$ for at most one $q \in Q$ and for every $q \in Q$ and $\sigma \in \Sigma$, there exists at most one $p \in Q$ such that $\langle q, \sigma, p\rangle \in \operatorname{supp}(\mathrm{wt})$. Finally, the automaton $\mathcal{A}$ is unambiguous if for every $k \in \mathbb{N}$ and $\sigma_1, \ldots, \sigma_k \in \Sigma$ there exists at most one sequence $\langle q_0, \sigma_1, q_1\rangle, \ldots, \langle q_{k-1}, \sigma_k, q_k\rangle$ of elements of $\operatorname{supp}(\mathrm{wt})$ such that $I(q_0) \neq 0$ and $F(q_k) \neq 0$.

For such a weighted string automaton $\mathcal{A}$ and $q, p \in Q$, we define the mapping $\mathcal{A}_{q,p}\colon \Sigma^* \to S$ recursively by

$$\mathcal{A}_{q,p}(\varepsilon) = \begin{cases} 1 & \text{if } q = p \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad \mathcal{A}_{q,p}(\sigma w) = \sum_{r \in Q} \Big(\mathrm{wt}(q, \sigma, r) \cdot \mathcal{A}_{r,p}(w)\Big) \tag{1}$$

for all $\sigma \in \Sigma$ and $w \in \Sigma^*$. The weighted string automaton $\mathcal{A}$ generates the mapping $\mathcal{A}\colon \Sigma^* \to S$ given by

$$\mathcal{A}(w) = \sum_{q,p \in Q} \big(I(q) \cdot \mathcal{A}_{q,p}(w) \cdot F(p)\big)$$

for every $w \in \Sigma^*$. The such generated mappings are called the regular weighted string languages, which are closed under several important operations. For example, when the semiring is commutative, they are closed under the Hadamard

Please cite this article in press as: A. Maletti, *Survey:* Finite-state technology in natural language processing, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.05.030