Minimization of Deterministic Weighted Tree Automata

Andreas Maletti

March 6, 2009

Syntax-based MT

Overview

$$
\begin{array}{ccccccccc}
\text{Parse Trees} & \Rightarrow & \boxed{\text{Translation (e.g. TST)}} & \Rightarrow & \text{Parse Trees} & \Rightarrow & \boxed{\text{Reranking (e.g. WTA)}} & \Rightarrow & \text{Parse Trees} \\
\Uparrow & & & & & & & & \Downarrow \\
\text{Input Text} & & & & & & & & \text{Final Outputs}
\end{array}
$$

Abbreviations

- TST = Tree Series Transducer
- WTA = Weighted Tree Automaton

Table of Contents

Table of Contents

Overview

Tree Series

- Assigns weight (e.g. a probability) to each tree
- Weight drawn from semiring; e.g. $(\mathbb{R}, +, \cdot, 0, 1)$

Overview

Tree Series

- Assigns weight (e.g. a probability) to each tree
- Weight drawn from semiring; e.g. $(\mathbb{R}, +, \cdot, 0, 1)$

Weighted Tree Automaton

- Finitely represents tree series
- Defines the recognizable tree series

Overview

Tree Series

- Assigns weight (e.g. a probability) to each tree
- Weight drawn from semiring; e.g. $(\mathbb{R}, +, \cdot, 0, 1)$

Weighted Tree Automaton

- Finitely represents tree series
- Defines the recognizable tree series

Application

- Re-ranker for parse trees
- Representation of parses

Syntax

Definition

Weighted tree automaton (wta) is tuple $(Q, \Sigma, \mathcal{A}, F, T)$ where

- Q: finite set of *states*
- Σ: ranked alphabet of *input symbols*
- $\mathcal{A} = (A, +, \cdot, 0, 1)$: semiring of *weights*
- $F \subseteq Q$: *final states*

Syntax

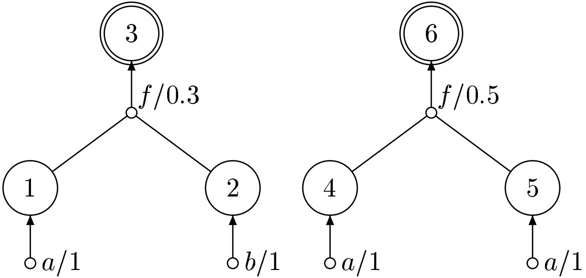
Definition

Weighted tree automaton (wta) is tuple $(Q, \Sigma, \mathcal{A}, F, T)$ where

- Q: finite set of *states*
- Σ: ranked alphabet of *input symbols*
- $\mathcal{A} = (A, +, \cdot, 0, 1)$: semiring of *weights*
- $F \subseteq Q$: *final states*
- T: finite set of transitions of the form $\sigma(q_1, \ldots, q_k) \xrightarrow{a} q$

Syntax — Illustration

Sample Automaton



Semantics

Definition

Left-most derivations are defined as usual.

- Weight of a derivation:
  product of the weights of the employed transitions
  (each transition counted as often as it occurs)

Semantics

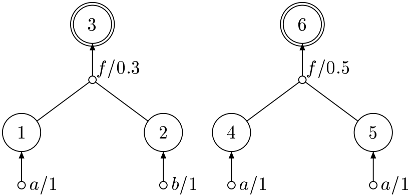
Definition

Left-most derivations are defined as usual.

- Weight of a derivation:
  product of the weights of the employed transitions
  (each transition counted as often as it occurs)
- Weight $\mathrm{wt}(t)$ of a tree t:
  sum of the weights of all left-most derivations that start with t and end in a final state

Semantics — Illustration

Sample Automaton

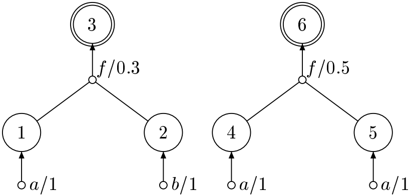


Sample Derivations

Input tree: $f(a, b)$ Derivation: $f(a, b)$

Semantics — Illustration

Sample Automaton

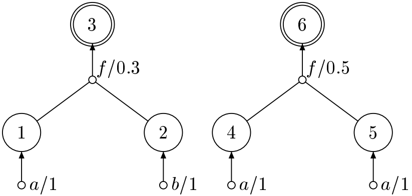


Sample Derivations

Input tree: $f(a, b)$ Derivation: $f(1, b)$ with weight 1

Semantics — Illustration

Sample Automaton

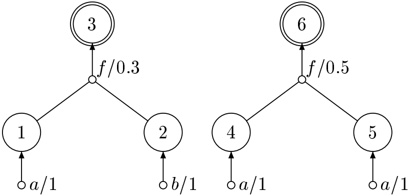


Sample Derivations

Input tree: $f(a, b)$ Derivation: $f(1, 2)$ with weight 1

Semantics — Illustration

Sample Automaton

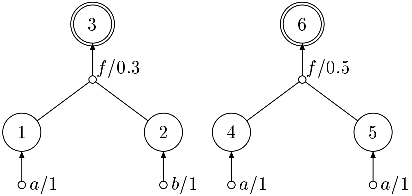


Sample Derivations

Input tree: $f(a, b)$ Derivation: 3 with weight 0.3

Semantics — Illustration

Sample Automaton



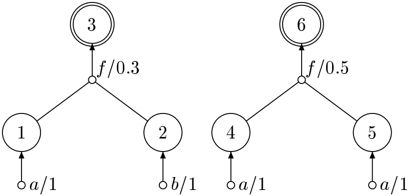
Sample Derivations

Input tree: $f(a, b)$ Derivation: 3 with weight 0.3

$f(4, b)$ with weight 1

Semantics — Illustration

Sample Automaton



Sample Derivations

Input tree: $f(a, b)$ Derivation: 3 with weight 0.3

$f(4, 2)$ with weight 1

Determinism

Definition

Deterministic wta: for every σ and $q_1, \ldots, q_k$ there exists exactly one transition $\sigma(q_1, \ldots, q_k) \xrightarrow{a} q \in T$

Determinism

Definition

Deterministic wta: for every σ and $q_1, \ldots, q_k$ there exists exactly one transition $\sigma(q_1, \ldots, q_k) \xrightarrow{a} q \in T$

Advantage

One derivation for each tree

Determinism

Definition

Deterministic wta: for every σ and $q_1, \ldots, q_k$ there exists exactly one transition $\sigma(q_1, \ldots, q_k) \xrightarrow{a} q \in T$

Advantage

One derivation for each tree

Notes

- Deterministic wta do not use addition
- Recognizable $\neq$ deterministically recognizable
- Determinization sometimes possible [Borchardt & Vogler '03]
- Partial determinization [May & Knight '06]

Table of Contents

Known Minimization Results

	deterministic		nondeterministic	
	string	tree	string	tree
unweighted	$O(lm \log n)$	$O(lm \log n)$	PSPACE	PSPACE
weighted*	$O(lm \log n)$	?	P	P

- l: maximal rank of symbols
- m: number of transitions
- n: number of states

Table of Contents

Overview

Applicability

- Deterministic wta
- Commutative semifield (i.e. multiplicative inverses)

Roadmap

- Myhill-Nerode congruence relation [Borchardt '03]
- Determine signs of life
- Scale maps
- Refinement

Myhill-Nerode congruence

Definition

$p \equiv q$: there exists nonzero a such that for every context C

$$\mathrm{wt}(C[p]) = a \cdot \mathrm{wt}(C[q])$$

Myhill-Nerode congruence

Definition

$p \equiv q$: there exists nonzero a such that for every context C

$$\mathrm{wt}(C[p]) = a \cdot \mathrm{wt}(C[q])$$

Notes

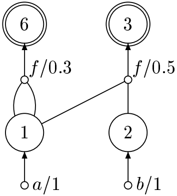
- Semifields are zero-divisor free
- Element a is unique if p is not dead

Signs of Life

Definition

Sign of life of $q \in Q$: context C such that $\mathrm{wt}(C[q]) \neq 0$

Example



State	Sign of life	State	Sign of life
1	$f(\square, b)$	2	$f(1, \square)$
3	$\square$	6	$\square$

Compute Signs of Life

How?

- start with final states
- apply transition as in a grammar
- record reached states and their signs of life
- rinse and repeat with those states

Compute Signs of Life

How?

- start with final states
- apply transition as in a grammar
- record reached states and their signs of life
- rinse and repeat with those states

Theorem

We can determine signs of life in $O(lm)$.

Scaling Map

Definition

$f : Q \to A$ is scaling map for partition Π of Q if

(i) $f(q) = 1$ for every dead state q

(ii) for every block $B \in \Pi$ there exists context C such that

$$\mathrm{wt}(C[q]) = f(q) \cdot \mathrm{wt}(C[r(B)])$$

and C is a sign of life for every live $q \in B$

Note

Scaling maps are nonzero everywhere.

Scaling Map

Definition

$f : Q \to A$ is scaling map for partition Π of Q if

(i) $f(q) = 1$ for every dead state q

(ii) for every block $B \in \Pi$ there exists context C such that

$$\mathrm{wt}(C[q]) = f(q) \cdot \mathrm{wt}(C[r(B)])$$

and C is a sign of life for every live $q \in B$

Note

Scaling maps are nonzero everywhere.

Theorem

A scaling map for Π can be computed in time $O(n^2)$.

Refinement

Definition

Refinement of Π: Split each block $B \in \Pi$ into two blocks, where one block contains all q such that

(ii) $\sigma(C[q]) \xrightarrow{a} q' \equiv_\Pi p' \xleftarrow{b} \sigma(C[r(B)])$

(iii) if q' is live, then

$$f(q)^{-1} \cdot a \cdot f(q') = b \cdot f(p')$$

for all symbols σ and contexts C.

Refinement

Definition

Refinement of Π: Split each block $B \in \Pi$ into two blocks, where one block contains all q such that

(ii) $\sigma(C[q]) \xrightarrow{a} q' \equiv_{\Pi} p' \xleftarrow{b} \sigma(C[r(B)])$

(iii) if q' is live, then

$$f(q)^{-1} \cdot a \cdot f(q') = b \cdot f(p')$$

for all symbols σ and contexts C.

Theorem

Each refinement can be implemented to run in time $O(lm)$.

Complete Algorithm

Algorithm

$(\Pi, \mathsf{sol}, D) \leftarrow \textsc{ComputeSol}(M)$ // in $O(lm)$

$\Pi \leftarrow \textsc{RefineCong}(\Pi)$ // classical minimization in $O(lm \log n)$

$f \leftarrow \textsc{ComputeSM}(\Pi)$ // in $O(n^2)$

repeat

  $\Pi' \leftarrow \Pi$ // store old partition

  $\Pi \leftarrow \textsc{Refine}(\Pi, f)$ // in $O(lm)$

  $f \leftarrow \textsc{UpdateSM}(\Pi, f)$ // in $O(n)$

until $\Pi' = \Pi$

return minimized wta // in $O(lm)$

Complete Algorithm

Algorithm

$(\Pi, \mathsf{sol}, D) \leftarrow \textsc{ComputeSol}(M)$ // in $O(lm)$

$\Pi \leftarrow \textsc{RefineCong}(\Pi)$ // classical minimization in $O(lm \log n)$

$f \leftarrow \textsc{ComputeSM}(\Pi)$ // in $O(n^2)$

repeat

  $\Pi' \leftarrow \Pi$ // store old partition

  $\Pi \leftarrow \textsc{Refine}(\Pi, f)$ // in $O(lm)$

  $f \leftarrow \textsc{UpdateSM}(\Pi, f)$ // in $O(n)$

until $\Pi' = \Pi$

return minimized wta // in $O(lm)$

Notes

- Algorithm runs in $O(lmn)$
- Returns equivalent minimal deterministic wta

Table of Contents

Experiments

State Count

Original	Minimal	Reduction to
98	68	69%
394	308	78%
497	381	77%
727	515	71%
2701	1993	74%
3686	1766	48%

Experiments

State Count

Original	Minimal	Reduction to
98	68	69%
394	308	78%
497	381	77%
727	515	71%
2701	1993	74%
3686	1766	48%

State & Transition Count

Error	Original	Minimal	Reduction to
10^{-4}	$(727, 6485)$	$(629, 6131)$	$(87\%, 95\%)$
10^{-2}	$(727, 6485)$	$(525, 3425)$	$(72\%, 53\%)$

References

- B. Borchardt and H. Vogler
  Determinization of Finite State Weighted Tree Automata.
  J. Autom. Lang. Combin. 8(3), 2003
- B. Borchardt
  The Myhill-Nerode Theorem for Recognizable Tree Series.
  Proc. DLT, LNCS 2710, Springer, 2003
- J. May and K. Knight
  A Better *N*-Best List: Practical Determinization of Weighted Finite Tree Automata.
  Proc. HLT-NAACL, ACL, 2006
- M. Droste, W. Kuich and H. Vogler (eds.)
  Handbook of Weighted Automata.
  Springer, 2009