Random Generation of Nondeterministic Tree Automata

Thomas Hanneforth[1] and Andreas Maletti[2] and Daniel Quernheim[2]

[1] Department of Linguistics
University of Potsdam, Germany

[2] Institute for Natural Language Processing
University of Stuttgart, Germany

maletti@ims.uni-stuttgart.de

Hanoi, Vietnam (TTATT 2013)

Outline

Motivation

Nondeterministic Tree Automata

Random Generation

Analysis

Tree Substitution Grammar with Latent Variables

Experiment [Shindo et al., ACL 2012 best paper]

grammar	F1 score $\|w\| \leq 40$	F1 score full
CFG = LTL		62.7
TSG [Post, Gildea, 2009] = xLTL	82.6	
TSG [Cohn et al., 2010] = xLTL	85.4	84.7
CFGlv [Collins, 1999] = NTA	88.6	88.2
CFGlv [Petrov, Klein, 2007] = NTA	90.6	90.1
CFGlv [Petrov, 2010] = NTA		91.8
TSGlv (single) = RTG	91.6	91.1
TSGlv (multiple) = RTG	92.9	92.4
Discriminative Parsers		
Carreras et al., 2008		91.1
Charniak, Johnson, 2005	92.0	91.4
Huang, 2008	92.3	91.7

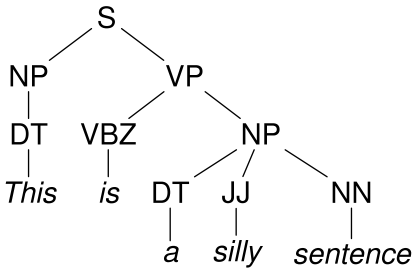
Tree Substitution Grammar with Latent Variables

Experiment [Shindo et al., ACL 2012 best paper]

grammar	F1 score $\lvert w \rvert \leq 40$	F1 score full
CFG = LTL		62.7
TSG [Post, Gildea, 2009] = xLTL	82.6	
TSG [Cohn et al., 2010] = xLTL	85.4	84.7
CFGlv [Collins, 1999] = NTA	88.6	88.2
CFGlv [Petrov, Klein, 2007] = NTA	90.6	90.1
CFGlv [Petrov, 2010] = NTA		91.8
TSGlv (single) = RTG	91.6	91.1
TSGlv (multiple) = RTG	92.9	92.4
Discriminative Parsers		
Carreras et al., 2008		91.1
Charniak, Johnson, 2005	92.0	91.4
Huang, 2008	92.3	91.7

Berkeley Parser

Example parse



from `http://tomato.banatao.berkeley.edu:8080/parser/parser.html`

Berkeley Parser

Example productions

S-1 $\to$ ADJP-2 S-1	$0.0035453455987323125 \cdot 10^{0}$
S-1 $\to$ ADJP-1 S-1	$2.108608433271444 \cdot 10^{-6}$
S-1 $\to$ VP-5 VP-3	$1.6367163259885093 \cdot 10^{-4}$
S-2 $\to$ VP-5 VP-3	$9.724998692152419 \cdot 10^{-8}$
S-1 $\to$ PP-7 VP-0	$1.0686659961009547 \cdot 10^{-5}$
S-9 $\to$ “ NP-3	$0.012551243773149695 \cdot 10^{0}$

Formalism

Berkeley parser = CFG (local tree grammar) + relabeling (+ weights)

Typical NTA

Sizes

- English BERKELEY parser grammar (1,133 states and 4,267,277 transitions) — 153 MB
- English EGRET parser grammar — 107 MB
- Chinese EGRET parser grammar — 98 MB

EGRET = HUI ZHANG's C++ reimplementation of the BERKELEY parser (Java)

Algorithm testing

Observations

- even efficient algorithms run slow on such data
- often require huge amounts of memory
- impossible for inefficient algorithms

Algorithm testing

Observations

- even efficient algorithms run slow on such data
- often require huge amounts of memory
- impossible for inefficient algorithms
- realistic, but difficult to use as test data

Algorithm testing

Observations

- even efficient algorithms run slow on such data
- often require huge amounts of memory
- impossible for inefficient algorithms
- realistic, but difficult to use as test data

Testing on random NTA

- straightforward to implement
- straightforward to scale

Algorithm testing

Observations

- even efficient algorithms run slow on such data
- often require huge amounts of memory
- impossible for inefficient algorithms
- realistic, but difficult to use as test data

Testing on random NTA

- straightforward to implement
- straightforward to scale
- but what is the significance of the results?

Outline

Motivation

Nondeterministic Tree Automata

Random Generation

Analysis

Tree automaton

Definition (Thatcher and Wright, 1965)

A tree automaton is a tuple $A = (Q, \Sigma, I, R)$ with

- alphabet Q — *states*
- ranked alphabet Σ — *terminals*
- $I \subseteq Q$ — *final states*
- finite set $R \subseteq \Sigma(Q) \times Q$ — *rules*

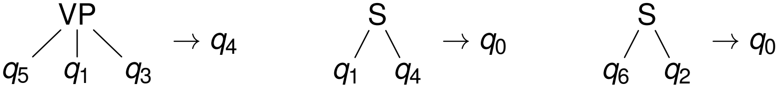
Remark

Instead of (ℓ, q) we write $\ell \to q$

Regular Tree Grammar

Example

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $\Sigma = \{\text{VP}, \text{S}, \dots\}$
- $F = \{q_0\}$
- and the following rules:



Regular Tree Grammar

Definition (Derivation semantics)

Sentential forms: $\xi, \zeta \in T_\Sigma(Q)$

$$\xi \Rightarrow_A \zeta$$

if there exist position $w \in \text{pos}(\xi)$ and rule $\ell \to q \in R$

- $\xi = \xi[\ell]_w$
- $\zeta = \xi[q]_w$

Regular Tree Grammar

Definition (Derivation semantics)

Sentential forms: $\xi, \zeta \in T_\Sigma(Q)$

$$\xi \Rightarrow_A \zeta$$

if there exist position $w \in \text{pos}(\xi)$ and rule $\ell \to q \in R$

- $\xi = \xi[\ell]_w$
- $\zeta = \xi[q]_w$

Definition (Recognized tree language)

$$L(A) = \{t \in T_\Sigma \mid \exists f \in F\colon t \Rightarrow_A^* f\}$$

Outline

Motivation

Nondeterministic Tree Automata

Random Generation

Analysis

Previous Approaches

Héam et al. 2009

- for deterministic tree-walking automata
  (and deterministic top-down tree automata)

Previous Approaches

Héam et al. 2009

- for deterministic tree-walking automata
  (and deterministic top-down tree automata)
- focus on generating automata uniformly at random
  (for estimating average-case complexity)

Previous Approaches

Héam et al. 2009

- for deterministic tree-walking automata
  (and deterministic top-down tree automata)
- focus on generating automata uniformly at random
  (for estimating average-case complexity)
- generator used for evaluation of conversion from det. TWA to NTA

Previous Approaches

HUGOT et al. 2010

- for tree automata with global equality constraints

Previous Approaches

Hugot et al. 2010

- for tree automata with global equality constraints
- focus on avoiding trivial cases
  (removal of unreachable states, minimum height requirement)

Previous Approaches

HUGOT et al. 2010

- for tree automata with global equality constraints
- focus on avoiding trivial cases
  (removal of unreachable states, minimum height requirement)
- generator used for evaluation of emptiness checker

Our Approach

Goals

- randomly generate non-trivial NTA
- generator (potentially) usable for all NTA algorithms

Our Approach

Goals

- randomly generate non-trivial NTA
- generator (potentially) usable for all NTA algorithms

When is an NTA non-trivial?

- large number of states
- large number of rules

Our Approach

Goals

- randomly generate non-trivial NTA
- generator (potentially) usable for all NTA algorithms

When is an NTA non-trivial?

- ~~large number of states~~
- ~~large number of rules~~

Our Approach

Goals

- randomly generate non-trivial NTA
- generator (potentially) usable for all NTA algorithms

When is an NTA non-trivial?

- ~~large number of states~~
- ~~large number of rules~~
- its language contains large trees

Our Approach

Goals

- randomly generate non-trivial NTA
- generator (potentially) usable for all NTA algorithms

When is an NTA non-trivial?

- ~~large number of states~~
- ~~large number of rules~~
- its language contains large trees

Our Approach

Goals

- randomly generate non-trivial NTA
- generator (potentially) usable for all NTA algorithms

When is an NTA non-trivial?

- ~~large number of states~~
- ~~large number of rules~~
- its language contains large trees
- its language has many MYHILL-NERODE congruence classes
  $\rightarrow$ canonical NTA has many states
  (canonical NTA = equivalent minimal deterministic NTA)

Our Approach

Restrictions

- binary trees
  (all RTL can be such encoded with linear overhead)

Our Approach

Restrictions

- binary trees
  (all RTL can be such encoded with linear overhead)
- each state is final with probability .5

Our Approach

Restrictions

- binary trees
  (all RTL can be such encoded with linear overhead)
- each state is final with probability .5
- uniform probability for binary/nullary rules

Our Approach

Restrictions

- binary trees
  (all RTL can be such encoded with linear overhead)
- each state is final with probability .5
- uniform probability for binary/nullary rules
- three parameters
  1. input binary ranked alphabet $\Sigma = \Sigma_2 \cup \Sigma_0$
  2. number n of states of generated NTA — scaling
  3. nullary rule probability d_0 — for all nullary rules
  4. binary rule probability d_2 — for all binary rules

Our Approach

Algorithm

1. Generate n states $[n] = \{1, \ldots, n\}$

Our Approach

Algorithm

1. Generate n states $[n] = \{1, \ldots, n\}$
2. Make q final with probability 0.5 $\forall q \in [n]$

Our Approach

Algorithm

1. Generate n states $[n] = \{1, \ldots, n\}$
2. Make q final with probability 0.5 $\forall q \in [n]$
3. Add rule $\alpha \to q$ with probability d_0 $\forall \alpha \in \Sigma_0, q \in [n]$

Our Approach

Algorithm

1. Generate n states $[n] = \{1, \ldots, n\}$
2. Make q final with probability 0.5 $\qquad \forall q \in [n]$
3. Add rule $\alpha \to q$ with probability d_0 $\qquad \forall \alpha \in \Sigma_0, q \in [n]$
4. Add rule $\sigma(q_1, q_2) \to q$ with probability d_2 $\qquad \forall \sigma \in \Sigma_2, q, q_1, q_2 \in [n]$

Our Approach

Algorithm

1. Generate n states $[n] = \{1, \ldots, n\}$
2. Make q final with probability 0.5 $\quad \forall q \in [n]$
3. Add rule $\alpha \to q$ with probability d_0 $\quad \forall \alpha \in \Sigma_0, q \in [n]$
4. Add rule $\sigma(q_1, q_2) \to q$ with probability d_2 $\quad \forall \sigma \in \Sigma_2, q, q_1, q_2 \in [n]$
5. Reject if it is not trim

Our Approach

Algorithm

1. Generate n states $[n] = \{1, \ldots, n\}$
2. Make q final with probability 0.5 $\forall q \in [n]$
3. Add rule $\alpha \to q$ with probability d_0 $\forall \alpha \in \Sigma_0, q \in [n]$
4. Add rule $\sigma(q_1, q_2) \to q$ with probability d_2 $\forall \sigma \in \Sigma_2, q, q_1, q_2 \in [n]$
5. Reject if it is not trim

Evaluation

1. Determinize
2. Minimize
3. Number of obtained states
   $=$ complexity of the original random NTA

Outline

Motivation

Nondeterministic Tree Automata

Random Generation

Analysis

Determinization

Definition (Power-set construction)

$\mathcal{P}(A) = (\mathcal{P}(Q), \Sigma, F', R')$ with

- $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$
- $\alpha \to \{q \in Q \mid \alpha \to q \in R\} \in R'$ $\quad \forall \alpha \in \Sigma_0$
- $\sigma(S_1, S_2) \to \{q \in Q \mid \sigma(q_1, q_2) \to q \in R, q_1 \in S_1, q_2 \in S_2\} \in R'$
  $\forall \sigma \in \Sigma_2, S_1, S_2 \subseteq Q$

Determinization

Definition (Power-set construction)

$\mathcal{P}(A) = (\mathcal{P}(Q), \Sigma, F', R')$ with

- $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$
- $\alpha \to \{q \in Q \mid \alpha \to q \in R\} \in R'$ $\quad \forall \alpha \in \Sigma_0$
- $\sigma(S_1, S_2) \to \{q \in Q \mid \sigma(q_1, q_2) \to q \in R, q_1 \in S_1, q_2 \in S_2\} \in R'$
  $\forall \sigma \in \Sigma_2, S_1, S_2 \subseteq Q$

Note

$\to$ will be the guiding definition for the analytical analysis

Analytical Analysis

Intuition

- power-set construction should create each state $S \subseteq Q$
- given states S_1, S_2 selected uniformly at random, each state $q \in Q$ should occur in target of $\sigma(S_1, S_2)$ with probability .5
  (the same intuition is also used for string automata)
- this intuition will create large NTA after determinization
  (but that they remain large after minimization is non-trivial)
- $\rightarrow$ we will confirm the intuition experimentally

Analytical Analysis

Theorem

If $d_2 = 4(1 - \sqrt[n^2]{.5})$ and $d_0 = .5$, then the intuition is met.

Proof.

Let $S_1, S_2 \subseteq Q$ be selected uniformly at random $\qquad \sigma \in \Sigma_2, q \in Q$

$$\begin{aligned}
&\pi(q \in \overline{\sigma}(S_1, S_2)) \\
&= 1 - \pi(q \notin \overline{\sigma}(S_1, S_2)) \\
&= 1 - \prod_{q_1, q_2 \in Q} \Big(1 - \pi(q_1 \in S_1) \cdot \pi(q_2 \in S_2) \cdot \pi(\sigma(q_1, q_2) \to q \in R)\Big) \\
&= 1 - \big(1 - \frac{d_2}{4}\big)^{n^2} = 1 - \big(1 - 1 + \sqrt[n^2]{.5}\big)^{n^2} = 1 - (\sqrt[n^2]{.5})^{n^2} = \frac{1}{2} \qquad \square
\end{aligned}$$

Analytical Predictions

n	d_2	d'_2	CI
2	.636		
3	.297		
4	.170		
5	.109		
6	.076		
7	.056		

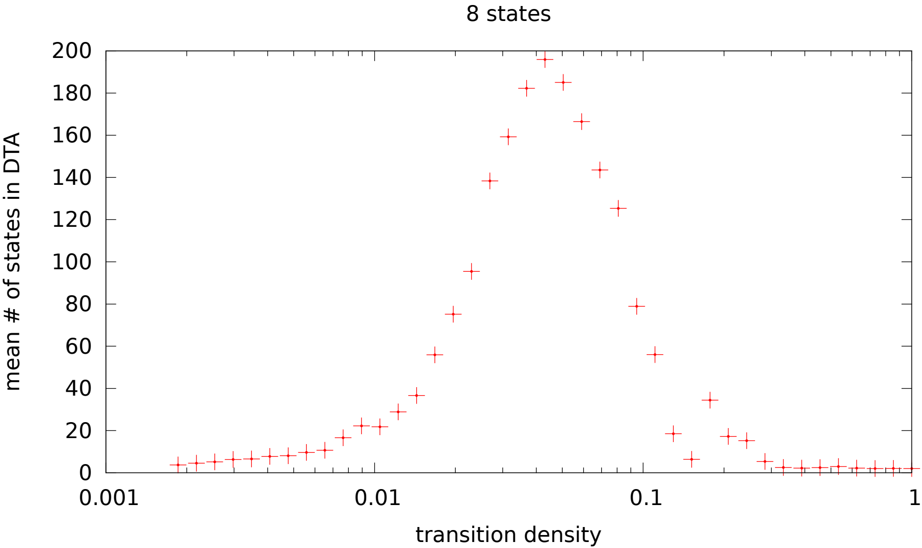
n	d_2	d'_2	CI
8	.043		
9	.034		
10	.028		
11	.023		
12	.019		
13	.016		

Empirical Evaluation

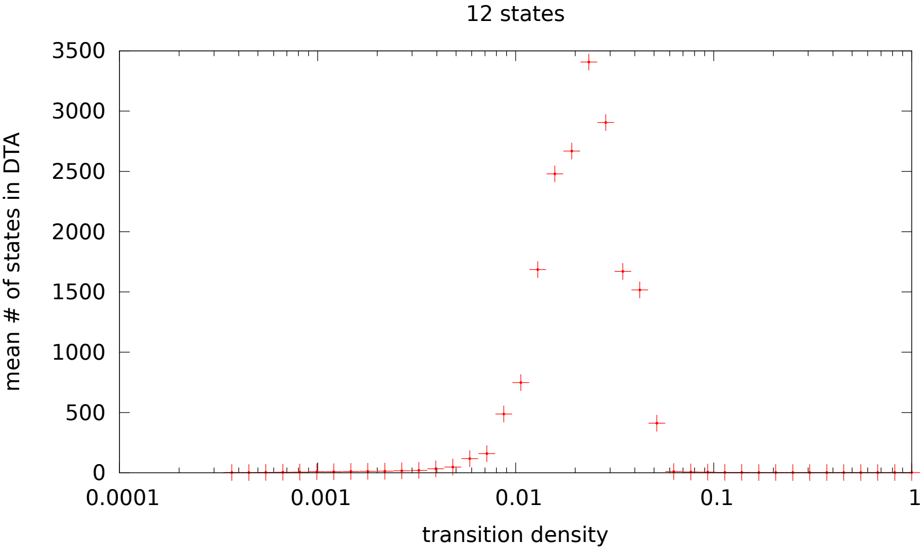
Setup

- $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$
- evaluation for random NTA with various densities d_2
  (at least 40 random NTA per data point d_2)
- logarithmic scale for d_2
  (enough datapoints on both sides of the spike)

Empirical Evaluation



Empirical Evaluation



Empirical Evaluation

Observations

- (almost perfect) log-normal distributions
- we can determine the mean
  (empirical and analytical)

Empirical Evaluation

Observations

- (almost perfect) log-normal distributions
- we can determine the mean
  (empirical and analytical)
- $\rightarrow$ hardest instances

Empirical Evaluation

Observations

- (almost perfect) log-normal distributions
- we can determine the mean
  (empirical and analytical)
- $\rightarrow$ hardest instances
- outside hardest instances: all trivial

Empirical Evaluation

Observations

- (almost perfect) log-normal distributions
- we can determine the mean
  (empirical and analytical)
- $\rightarrow$ hardest instances
- outside hardest instances: all trivial
- only test on random NTA for hardest density

Analytical Predictions

n	d_2	d_2'	CI
2	.636		
3	.297		
4	.170		
5	.109		
6	.076		
7	.056		

n	d_2	d_2'	CI
8	.043		
9	.034		
10	.028		
11	.023		
12	.019		
13	.016		

Analytical Predictions + Empirical Evaluation

n	d_2	d_2'	CI
2	.636	.626	
3	.297	.257	
4	.170	.133	
5	.109	.086	
6	.076	.064	
7	.056	.050	

n	d_2	d_2'	CI
8	.043	.041	
9	.034	.034	
10	.028	.028	
11	.023	.025	
12	.019	.021	
13	.016	.019	

Analytical Predictions + Empirical Evaluation

n	d_2	d_2'	CI
2	.636	.626	[.577,.680]
3	.297	.257	[.209,.316]
4	.170	.133	[.102,.174]
5	.109	.086	[.064,.114]
6	.076	.064	[.048,.085]
7	.056	.050	[.038,.066]

n	d_2	d_2'	CI
8	.043	.041	[.032,.053]
9	.034	.034	[.027,.043]
10	.028	.028	[.023,.034]
11	.023	.025	[.021,.030]
12	.019	.021	[.018,.025]
13	.016	.019	[.016,.022]

CI = confidence interval; 95% confidence level

Conclusion

Use random NTA carefully!