Weighted Tree Automata and Transducers for Syntactic Natural Language Processing

Jonathan May
Thesis Defense
April 20, 2010
How do we view natural language?

As a string?
How do we view natural language?

As a string?
How do we view natural language?

As a string?

context window
How do we view natural language?

As a string?

i gave

context window
How do we view natural language?

As a string?

i gave my
context window
How do we view natural language?

As a string?

\[ i \_gave\_my\_son \]

context
window
How do we view natural language?

As a string?

\[ i \text{ gave } \underline{my \text{ son}} \]

\[ \underline{context} \]

\[ \underline{window} \]
How do we view natural language?

As a string?

i gave my son a
context window
How do we view natural language?

As a string?

i gave my son a baseball bat

context window
How do we view natural language?

As a string?

i gave my son a baseball bat

context window

is
How do we view natural language?

As a string?

i gave my son?

a baseball bat

is three years old

context window
How do we view natural language?

As a string?

Language is more hierarchical than this!
How do we view natural language?

Or as a tree?

$S$
How do we view natural language?

Or as a tree?

S

NP  VP
How do we view natural language?

Or as a tree?

\[
\begin{array}{c}
S \\
NP & VP \\
i
\end{array}
\]
How do we view natural language?

Or as a tree?
How do we view natural language?

Or as a tree?

\[ S \rightarrow NP \quad NP \rightarrow VP \quad VP \rightarrow \text{gave} \quad NP \rightarrow \text{my son} \]
How do we view natural language?

Or as a tree?

i gave my son a bat
How do we view natural language?

Or as a tree?

Trees provide syntactic context!
String World vs Tree World

string
String World vs Tree World

- string
  - great formalisms
String World vs Tree World

- string
  - great formalisms
  - useful algorithms
String World vs Tree World

<table>
<thead>
<tr>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
</tr>
<tr>
<td>toolkits</td>
</tr>
</tbody>
</table>
String World vs Tree World

- string
  - great formalisms
  - useful algorithms
  - toolkits
  - rapid progress
String World vs Tree World

<table>
<thead>
<tr>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
</tr>
<tr>
<td>toolkits</td>
</tr>
<tr>
<td>rapid progress</td>
</tr>
<tr>
<td>limited expressiveness</td>
</tr>
<tr>
<td>string</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
</tr>
<tr>
<td>toolkits</td>
</tr>
<tr>
<td>rapid progress</td>
</tr>
</tbody>
</table>

String World vs Tree World
String World vs Tree World

<table>
<thead>
<tr>
<th>string</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
<td></td>
</tr>
<tr>
<td>toolkits</td>
<td></td>
</tr>
<tr>
<td>rapid progress</td>
<td></td>
</tr>
<tr>
<td>limited expressiveness</td>
<td></td>
</tr>
</tbody>
</table>
String World vs Tree World

<table>
<thead>
<tr>
<th>string</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
<td></td>
</tr>
<tr>
<td>toolkits</td>
<td></td>
</tr>
<tr>
<td>rapid progress</td>
<td></td>
</tr>
<tr>
<td>limited expressiveness</td>
<td>powerful expressiveness</td>
</tr>
</tbody>
</table>
## String World vs Tree World

<table>
<thead>
<tr>
<th>string</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
<td>few algorithms</td>
</tr>
<tr>
<td>toolkits</td>
<td></td>
</tr>
<tr>
<td>rapid progress</td>
<td>powerful expressiveness</td>
</tr>
<tr>
<td>limited expressiveness</td>
<td></td>
</tr>
</tbody>
</table>
## String World vs Tree World

<table>
<thead>
<tr>
<th></th>
<th>string</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
<td>great formalisms</td>
<td></td>
</tr>
<tr>
<td>useful algorithms</td>
<td>few algorithms</td>
<td></td>
</tr>
<tr>
<td>toolkits</td>
<td>no toolkits</td>
<td></td>
</tr>
<tr>
<td>rapid progress</td>
<td></td>
<td>powerful expressiveness</td>
</tr>
<tr>
<td>limited expressiveness</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# String World vs Tree World

<table>
<thead>
<tr>
<th></th>
<th>String</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Great formalisms</td>
<td>Great formalisms</td>
<td>Great formalisms</td>
</tr>
<tr>
<td>Useful algorithms</td>
<td>Few algorithms</td>
<td>No toolkits</td>
</tr>
<tr>
<td>Toolkits</td>
<td>No toolkits</td>
<td>Powerful expressiveness</td>
</tr>
<tr>
<td>Rapid progress</td>
<td>Slow progress</td>
<td>Limited expressiveness</td>
</tr>
</tbody>
</table>

Monday, April 19, 2010
# Contributions

<table>
<thead>
<tr>
<th></th>
<th>string</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
<td>great formalisms</td>
<td></td>
</tr>
<tr>
<td>useful algorithms</td>
<td>few algorithms</td>
<td></td>
</tr>
<tr>
<td>toolkits</td>
<td>no toolkits</td>
<td></td>
</tr>
<tr>
<td>rapid progress</td>
<td>slow progress</td>
<td></td>
</tr>
<tr>
<td>limited expressiveness</td>
<td>powerful expressiveness</td>
<td></td>
</tr>
</tbody>
</table>
## Contributions

<table>
<thead>
<tr>
<th></th>
<th>String</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Great formalisms</td>
<td>great formalisms</td>
<td>great formalisms</td>
</tr>
<tr>
<td>Useful algorithms</td>
<td>new algorithms!</td>
<td>no toolkits</td>
</tr>
<tr>
<td>Toolkits</td>
<td>no toolkits</td>
<td></td>
</tr>
<tr>
<td>Rapid progress</td>
<td>slow progress</td>
<td></td>
</tr>
<tr>
<td>Limited expressiveness</td>
<td>powerful expressiveness</td>
<td></td>
</tr>
</tbody>
</table>

22 Monday, April 19, 2010
# Contributions

<table>
<thead>
<tr>
<th>string</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
<td><strong>new algorithms!</strong></td>
</tr>
<tr>
<td>toolkits</td>
<td><strong>new toolkit!</strong></td>
</tr>
<tr>
<td>rapid progress</td>
<td>slow progress</td>
</tr>
<tr>
<td>limited expressiveness</td>
<td>powerful expressiveness</td>
</tr>
</tbody>
</table>
## Contributions

<table>
<thead>
<tr>
<th>string</th>
<th>tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>great formalisms</td>
<td>great formalisms</td>
</tr>
<tr>
<td>useful algorithms</td>
<td><strong>new algorithms!</strong></td>
</tr>
<tr>
<td>toolkits</td>
<td><strong>new toolkit!</strong></td>
</tr>
<tr>
<td>rapid progress</td>
<td><strong>rapid progress!</strong></td>
</tr>
<tr>
<td>limited expressiveness</td>
<td>powerful expressiveness</td>
</tr>
</tbody>
</table>

Monday, April 19, 2010
Weighted finite-state string machines

Acceptor

Transducer
Weighted finite-state string machines

Acceptor

Transducer

the blue dwarf/.048
green hairy elf/.0144
the red hairy hairy elf/.000432
...
Weighted finite-state string machines

Acceptor

Transducer

the blue dwarf/.048
green hairy elf/.0144
the red hairy hairy elf/.000432
...

the blue elf : el duende azúl/.0576
the blue man : el duende triste/.048
...
Using WFSTs for NLP

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer
Using WFSTs for NLP

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

the blue dwarf
Using WFSTs for NLP

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

the blue dwarf

Machine Translation
Using WFSTs for NLP

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer

the blue dwarf -> Machine Translation

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
el azúl duende / .01
...

Monday, April 19, 2010
MT as weighted transducers

Imagine an English sentence

Re-order the words

Translate into Spanish

the green ball = .098
a green green horse = .0036
horse green the ❌
Imagine an English sentence

Re-order the words

Translate into Spanish
MT as weighted transducers

Imagine an English sentence → Re-order the words → Translate into Spanish

- Start state
  - the: the/0.1
  - ball: ball the/1
  - green: green the/1
  - the: the the/1

- Ball state
  - ball: ball ball/1
  - green: green ball/1
  - the: the ball/1

- Green state
  - ball: ball green/1
  - green: green green/1
  - the: the green/1

- Translation states:
  - the green ball 0.216
  - the ball green 0.63
  - green the ball 0.03
MT as weighted transducers

Imagine an English sentence

Re-order the words

Translate into Spanish

ball: pelota/.8
ball: bola/.2

the: el/.35
the: la/.35
the: los/.15
the: las/.15

green: verde/.7
green: campo/.3

horse: caballo/.9
horse: bayo/.1

the ball green

la pelota verde .196
el bola campo .021
MT as weighted transducers

Generative story: we corrupt good English into (possibly bad) Spanish
Decoding story: given some good Spanish, determine the best good English that could produce it
Secret weapons

- WFST toolkits do this calculation for us:
  - AT&T FSM\(^1\) / Google OpenFst\(^2\)
  - USC/ISI Carmel\(^3\)
- Generic operations for manipulation, combination, inference, training

<table>
<thead>
<tr>
<th>WFST toolkit operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-best</td>
</tr>
<tr>
<td>em training</td>
</tr>
<tr>
<td>determinization</td>
</tr>
<tr>
<td>composition</td>
</tr>
<tr>
<td>pipeline inference</td>
</tr>
<tr>
<td>on-the-fly inference</td>
</tr>
</tbody>
</table>

1: Mohri, Pereira, Riley, ’98  
2: Allauzen et al., ’07  
3: Graehl, ’97
Widely applicable!

the blue elf

Machine Translation
(Kumar & Byrne '03)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

Monday, April 19, 2010
Widely applicable!

the blue elf

Machine Translation
(Kumar & Byrne '03)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

Decipherment
(Ravi & Knight '09)

“i love killing people” / .2
“eggs, milk, flour, tin foil” / .002
...

Monday, April 19, 2010
Widely applicable!

Machine Translation (Kumar & Byrne '03)

the blue elf

Decipherment (Ravi & Knight '09)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

Speech Recognition (Pereira et al. '94)

“i love killing people” / .2
“eggs, milk, flour, tin foil” / .002
...

“this is the bbc” / .5
“this is the bee I see” / .1
“the sister beyoncé” / .1
...
Widely applicable!

Machine Translation (Kumar & Byrne '03)

Widely applicable!

the blue elf

Decipherment (Ravi & Knight '09)

Speech Recognition (Pereira et al. '94)

Poetry Generation (Greene & Knight '10)

el enano azúl / .3
el enano triste / .1
el duende azúl / .05
...

“i love killing people” / .2
“eggs, milk, flour, tin foil” / .002
...

“this is the bbc” / .5
“this is the bee I see” / .1
“the sister beyoncé” / .1
...

“there was a computer from apple that wore a red rose in its lapel” / .1
...

Monday, April 19, 2010
NLP work using WFSTs

**Translation**  
(Kumar & Byrne '03)

**Decipherment**  
(Ravi & Knight '09)

**Speech Recognition**  
(Pereira et al. '94)

**Poetry Generation**  
(Greene & Knight '10)

**OCR**  
(Kolak et al. '03)

**Morphology**  
(Karttunen et al. '92)

**POS Tagging**  
(Church '88)

**Spelling Correction**  
(Boyd '09)

**Transliteration**  
(Knight & Graehl '98)

Also see summary: book chapter of *Handbook of Weighted Automata* (Knight & May '08)
Limitations of strings

• Can’t do arbitrary long-distance reordering
• Can’t maintain arbitrary long-distance dependencies
• Can’t naturally integrate syntax information
Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information
Limitations of strings

• Can’t do arbitrary long-distance reordering
• Can’t maintain arbitrary long-distance dependencies
• Can’t naturally integrate syntax information
Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information

\[
\begin{array}{c}
S \\
V P & N P \\
\text{ 33}\,
\end{array}
\]

(1st-person-singular)
Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information

But that’s what we want!
Limitations of strings

• Can’t do arbitrary long-distance reordering
• Can’t maintain arbitrary long-distance dependencies
• Can’t naturally integrate syntax information

But that’s what we want!

Parsing
(Collins ’97)
Limitations of strings

• Can’t do arbitrary long-distance reordering
• Can’t maintain arbitrary long-distance dependencies
• Can’t naturally integrate syntax information

But that’s what we want!

Parsing (Collins ’97)  Question Answering (Echihabi & Marcu ’03)
Limitations of strings

• Can’t do arbitrary long-distance reordering
• Can’t maintain arbitrary long-distance dependencies
• Can’t naturally integrate syntax information

But that’s what we want!

Parsing  Question Answering
(Collins ’97)  (Echihabi & Marcu ’03)

Language Modeling
(Charniak ’01)
Limitations of strings

- Can’t do arbitrary long-distance reordering
- Can’t maintain arbitrary long-distance dependencies
- Can’t naturally integrate syntax information

But that’s what we want!

Parsing (Collins ’97)  Question Answering (Echihabi & Marcu ’03)
Language Modeling  Summarization (Charniak ’01) (Knight & Marcu ’03)
Limitations of strings

• Can’t do arbitrary long-distance reordering
• Can’t maintain arbitrary long-distance dependencies
• Can’t naturally integrate syntax information

But that’s what we want!

Parsing (Collins ‘97)  Question Answering (Echihabi & Marcu ’03)
Language Modeling (Charniak ’01)  Summarization (Knight & Marcu ’03)

Machine Translation (Yamada & Knight ’01)  (Galley et al. ’04)
  (Mi et al. ’08)  (Zhang et al. ’08)
Lots of work with tree models, but NO tree toolkit!

Parsing (Collins ’97)
Question Answering (Echihabi & Marcu ’03)
Language Modeling (Charniak ’01)
Summarization (Knight & Marcu ’03)

Machine Translation
(Yamada & Knight ’01)
(Galley et al. ’04)
(Mi et al. ’08)
(Zhang et al. ’08)
Weighted finite-state tree machines

Grammar

Transducer

34
Weighted finite-state tree machines

Grammar

Transducer

np
the
red
elf

/ .0576

np
the blue elf
red
red

/ .0017

the
el
duende

blue azúl

triste

np
the
jj
elf
Weighted finite-state tree machines

Grammar

Transducer

Monday, April 19, 2010
Weighted regular tree grammars

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

Tree

\[ \text{NP} \]

Weight

\[ \text{NP} \]

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

Tree

Weight

(Berstel & Reutenauer, 1982)
Weighted regular tree grammars

(Berstel & Reutenauer, 1982)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)

Monday, April 19, 2010
Weighted tree transducers

Tree

Weight

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree transducers

(Kuich, 1998)
Weighted tree-string transducers

(Kuich, 1998)
Weighted tree-string transducers

(Kuich, 1998)
Imagine an English sentence
Re-order the words
Translate into Spanish
Imagine an English sentence

Re-order the words

Translate into Spanish

S

NP

DT JJ NN

a blue dog

MT as weighted tree transducers
Imagine an English sentence

Re-order the words

Translate into Spanish
Imagine an English sentence
Re-order the words
Translate into Spanish

MT as weighted tree transducers

S
NP
DT JJ JJ NN

...
MT as weighted tree transducers

Imagine an English sentence

Re-order the words

Translate into Spanish

S

NP

DT NN JJ JJ

59
Imagine an English sentence
Re-order the words
Translate into Spanish
MT as weighted tree transducers

Imagine an English sentence

Re-order the words

Translate into Spanish

S

... NP

NN JJ JJ

59
Imagine an English sentence

Re-order the words

Translate into Spanish
Great, so now we can solve harder problems!
Great, so now we can solve harder problems!

the dog ran home

Syntax-Based Machine Translation

Parsing

NP
the
JJ
elf
blue and red and yellow

NP
el enano JJ
azul y rojo y amarillo

S
NP VP
the dog ran home
...

Monday, April 19, 2010
Great, so now we can solve harder problems!
Great, so now we can solve harder problems!

Not so fast!

the dog ran home

Syntax-Based Machine Translation

Parsing

Summarization

the elf

blue and red and yellow

el enano

azul y rojo y amarillo

the dog ran home

...
String world has many more available operations than tree world!

<table>
<thead>
<tr>
<th>Operation</th>
<th>String</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-best</td>
<td>yes</td>
<td>alg(^1)</td>
</tr>
<tr>
<td>em training</td>
<td>yes</td>
<td>alg(^2)</td>
</tr>
<tr>
<td>determinization</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>composition</td>
<td>yes</td>
<td>proof of concept(^3)</td>
</tr>
<tr>
<td>pipeline inference</td>
<td>yes</td>
<td>proof of concept(^4)</td>
</tr>
<tr>
<td>on-the-fly inference</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

1: Huang & Chiang, 2005
2: Graehl & Knight, 2004
3: Maletti, 2006
4: Fülöp, Maletti, Vogler, 2010
**Algorithmic contribution I:** weighted determinization

<table>
<thead>
<tr>
<th>Operation</th>
<th>String</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-best</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>em training</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>determinization</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>composition</td>
<td>yes</td>
<td>proof of concept$^3$</td>
</tr>
<tr>
<td>pipeline inference</td>
<td>yes</td>
<td>proof of concept$^4$</td>
</tr>
<tr>
<td>on-the-fly inference</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Monday, April 19, 2010
Algorithmic contribution II: efficient inference

<table>
<thead>
<tr>
<th>Operation</th>
<th>String</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-best</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>em training</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>determinization</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>composition</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>pipeline inference</td>
<td>yes</td>
<td>alg</td>
</tr>
<tr>
<td>on-the-fly inference</td>
<td>yes</td>
<td>alg</td>
</tr>
</tbody>
</table>
## Practical contribution I: weighted tree transducer toolkit

<table>
<thead>
<tr>
<th>Operation</th>
<th>String</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-best</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>em training</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>determinization</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>composition</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>pipeline inference</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>on-the-fly inference</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

---

Monday, April 19, 2010
Practical contribution II: syntactic re-alignment

<table>
<thead>
<tr>
<th>Operation</th>
<th>String</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-best</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>em training</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>determinization</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>composition</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>pipeline inference</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>on-the-fly inference</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Determinization of weighted tree automata

(May & Knight, HLT-NAACL ’06)
(Büchse, May, Vogler, FSMNLP ’09)

Elevated Mohri algorithm (’97) to tree automata
Demonstrated empirical gains in parsing and MT
Algorithmic Contribution I: WTA Determinization

**Before**

- $t$ with transition to $D$ with probability 0.2
- $q$ with transition to $r$ with probability 0.3

- $t$ with transition to $D$ with probability 0.3
- $q$ with transition to $s$ with probability 0.6

**After**

- $q$ with transition to $A$ with probability 0.3
- $r$ with transition to $B$ with probability 0.2
- $s$ with transition to $B$ with probability 0.4
Algorithmic Contribution I: WTA Determinization

Before

A \rightarrow t \rightarrow D \rightarrow q \rightarrow A

r \rightarrow D \rightarrow q \rightarrow B

s \rightarrow B

s \rightarrow C

Non-deterministic rules
(treating grammar as bottom-up acceptor)
Algorithmic Contribution I: WTA Determinization

**Before**

```
( t .2 \rightarrow D )
   \quad ( q \rightarrow r )
```

```
( t .3 \rightarrow D )
   \quad ( q \rightarrow s )
```

**After**

```
( q .3 \rightarrow A )
   \quad ( r .2 \rightarrow B )
   \quad ( s .6 \rightarrow B )
   \quad ( s .4 \rightarrow C )
```

Merge terminal rules with same right sides
Algorithmic Contribution I: WTA Determinization

**BEFORE**

\[
\begin{align*}
&t \quad \rightarrow \quad D \\
&q \quad \rightarrow \quad q \\
&r \quad \rightarrow \quad r
\end{align*}
\]

\[
\begin{align*}
&t \quad \rightarrow \quad D \\
&q \quad \rightarrow \quad q \\
&s \quad \rightarrow \quad s
\end{align*}
\]

\[
\begin{align*}
&\circ \quad \rightarrow \quad A \\
&\circ \quad \rightarrow \quad B \\
&\circ \quad \rightarrow \quad B
\end{align*}
\]

\[
\begin{align*}
&\circ \quad \rightarrow \quad C
\end{align*}
\]

**AFTER**

\[
\begin{align*}
&r/25 \quad \rightarrow \quad B \\
&s/75 \quad \rightarrow \quad B
\end{align*}
\]

Merge terminal rules with same right sides
Algorithmic Contribution I: WTA Determinization

**Before**

- \( t \) with weight 0.2 pointing to \( D \) with \( q \) and \( r \)
- \( t \) with weight 0.3 pointing to \( D \) with \( q \) and \( s \)

**After**

- \( q \) with weight 0.3 pointing to \( A \)
- \( r \) with weight 0.2 pointing to \( B \)
- \( s \) with weight 0.6 pointing to \( B \)
- \( s \) with weight 0.4 pointing to \( C \)

**Portion attributed to each state**

**Sum of weights**
Algorithmic Contribution I: WTA Determinization

Before:

```
 B  .2
  D  
     q
     r

 B  .3
  D  
     q
     s
```

Process the other terminal rules

After:

```
 q  .3
  A

 r  .2
  B

 s  .6
  B

 s  .4
  C
```
Algorithmic Contribution I: WTA Determinization

Before

\[
\begin{align*}
& t \xrightarrow{.2} D \\
& q \quad r \\
& t \xrightarrow{.3} D \\
& q \quad s
\end{align*}
\]

\[
\begin{align*}
& q \xrightarrow{.3} A \\
& r \xrightarrow{.2} B \\
& s \xrightarrow{.6} B \\
& s \xrightarrow{.4} C
\end{align*}
\]

Process the other terminal rules

After

\[
\begin{align*}
& q \xrightarrow{.3} A \\
& r \xrightarrow{.8}{/25\text{s}/.75} B \\
& s \xrightarrow{.4} C
\end{align*}
\]
Algorithmic Contribution I: WTA Determinization

**BEFORE**

```
( t .2 → D )
  |    |
  q    r

( t .3 → D )
  |    |
  q    s
```

```
( q .3 → A )

( r .2 → B )

( s .6 → B )

( s .4 → C )
```

---

**AFTER**

```
( q/1 .3 → A )

( r/25 s/.75 .8 → B )

( s/1 .4 → C )
```

Process the other terminal rules
Algorithmic Contribution I: WTA Determinization

**BEFORE**

```
  t → D
  q
  r
```

```
  t → D
  q
  s
```

**AFTER**

```
  q \rightarrow A
  r \rightarrow B
  s \rightarrow B
  s \rightarrow C
```

Process the other terminal rules
Algorithmic Contribution I: WTA Determinization

Choose rules from the input rtg and new state sequences that match.
Algorithmic Contribution I: WTA Determinization

**Before**

- **t** \(\rightarrow\) D
  - \(q\) \(\rightarrow\) D
  - \(r\) \(\rightarrow\) D
- **t** \(\rightarrow\) D
  - \(q\) \(\rightarrow\) s

**After**

- **t** \(\rightarrow\) D
  - \(q\) \(\rightarrow\) A
  - \(r\) \(\rightarrow\) B
  - \(s\) \(\rightarrow\) C

Choose rules from the input rtg and new state sequences that match.
Algorithmic Contribution I: WTA Determinization

**BEFORE**

- $t \rightarrow D$
  - $q \rightarrow r$
  - $t \rightarrow D$
  - $q \rightarrow s$

**AFTER**

- $q \rightarrow A$
- $r \rightarrow B$
- $s \rightarrow B$
- $s \rightarrow C$

Form new rules from these components
Algorithmic Contribution I: WTA Determinization

**BEFORE**

- $t \rightarrow D$ with weight of 0.2
- $q \rightarrow A$ with weight of 0.3
- $r \rightarrow B$ with weight of 0.2
- $s \rightarrow B$ with weight of 0.6
- $s \rightarrow C$ with weight of 0.4

**AFTER**

- $t/1 \rightarrow D$ with weight of 0.05
- $q/1 \rightarrow A$ with weight of 0.3
- $r/25 \rightarrow (s/75)$ with weight of 0.8
- $s/1 \rightarrow C$ with weight of 0.4

Rule weight of 0.2 times residual of 0.25
Algorithmic Contribution I: WTA Determinization

**BEFORE**

```
q \rightarrow A
r \rightarrow B
s \rightarrow B

Q \rightarrow \text{D}
```

**AFTER**

```
q/1 \rightarrow A
r/25 s/75 \rightarrow B
s/1 \rightarrow C
```
Algorithmic Contribution I: WTA Determinization

**BEFORE**

```
  t --> D
     |  .2
     q --> r

  t --> D
     |  .3
     q --> s
```

```
  q --> A
     .3
  r --> B
     .2

  s --> B
     .6
```

```
  s --> C
     .4
```

**AFTER**

```
  t/1 --> D
     .05
     q/1 --> r/25
     s/75

  t --> D
     .3
     q/1 --> q/1
     r/25
     s/75

  r/25
  s/75

  q/1 --> A
     .3

  s/1 --> C
     .4
```

Monday, April 19, 2010
Algorithmic Contribution I: WTA Determinization

**Before**

- **t** → 0.2 → **D**
  - **q**
  - **r**

- **t** → 0.3 → **D**
  - **q**
  - **s**

**After**

- **q** → 0.3 → **A**
- **r** → 0.2 → **B**
- **s** → 0.6 → **B**
- **s** → 0.4 → **C**

**Rule weight of 0.3 times residual of 0.75**

Monday, April 19, 2010
These rules are identical except for their weight, so we'll sum them.

Before

\[
\begin{align*}
\text{t} \rightarrow & \quad \text{D} \\
\text{q} \quad & \rightarrow \quad \text{r} \\
\text{t} \rightarrow & \quad \text{D} \\
\text{q} \quad & \rightarrow \quad \text{s}
\end{align*}
\]

After

\[
\begin{align*}
\text{t/1} \rightarrow & \quad \text{D} \\
\text{q/1} \quad & \rightarrow \quad \text{r/25} \\
\text{s/1} \rightarrow & \quad \text{C}
\end{align*}
\]
These rules are identical except for their weight, so we’ll sum them.
Algorithmic Contribution I: WTA Determinization

**BEFORE**

- ![Diagram Before](image)

**AFTER**

- ![Diagram After](image)
Algorithmic Contribution I: WTA Determinization

**BEFORE**

```
  t -> D
  q  r
```

```
  t -> D
  q  s
```

```
  q -> A
  r -> B
```

```
  s -> B
```

```
  s -> C
```

**AFTER**

```
  t/l -> D
  q/l
```

```
  r/0.25 s/0.75
```

```
  t -> D
  q/l
```

```
  q/l
```

```
  q/l
```

```
  s/l
```

```
  s/l -> C
```

Monday, April 19, 2010
Algorithmic Contribution I: WTA Determinization

**BEFORE**

```
  t → D
  q → r
  t → D
  q → s
```

**AFTER**

```
  q → A
  r → B
  s → B
  s → C
```

```
  t/1 → D
  q/1 → r/25 s/75
  t/1 → D
  q/1 → s/1
```

```
  q/1 → A
  r/25 s/75 → B
  s/1 → C
```
Algorithmic Contribution I: WTA Determinization

**BEFORE**

```
D
 t → D
 q r

q .3
 A

r .2
 B

s .6
 B

s .4
 C
```

**AFTER**

```
D
 t/1 → D
 q/1

r/25 s/.75
 B

s/1

q/1 .3
 A

r/25 s/.75 .8
 B

s/1 .4
 C
```

Monday, April 19, 2010
Empirical experiments

Machine translation (Galley et al. ’04, ’06)

Determinization removes duplicates and re-ranks n-best lists

<table>
<thead>
<tr>
<th>Method</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undeterminized</td>
<td>21.87</td>
</tr>
<tr>
<td>Top-500 “crunching”</td>
<td>23.33</td>
</tr>
<tr>
<td>Determinized</td>
<td>24.17</td>
</tr>
</tbody>
</table>
Empirical experiments

DOP parsing (Bod ’92)

Determinization removes duplicates and re-ranks n-best lists

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undeterminized</td>
<td>80.23</td>
<td>80.18</td>
<td>80.20</td>
</tr>
<tr>
<td>Top-500 “crunching”</td>
<td>80.48</td>
<td>80.29</td>
<td>80.39</td>
</tr>
<tr>
<td>Determinized</td>
<td>81.09</td>
<td>79.72</td>
<td>80.40</td>
</tr>
</tbody>
</table>
Efficient inference through cascades of weighted tree transducers
(May, Knight, Vogler, Submitted)

- First presentation of algorithms for inference through weighted extended tree transducer cascades
- On-the-fly approach significantly outperforms "classic" approach
Inference through string transducers

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer.
Inference through string transducers

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer
Inference through string transducers

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer
Inference through string transducers

Given a string and a transducer, calculate the highest weighted transformation of the string by the transducer.

- the blue dwarf
- Machine Translation

- el enano azul / .3
- el enano triste / .1
- el duende azul / .05
- el azul duende / .01
- ...

93
Inference through string cascades

Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

(Pereira & Riley, 1997)
Inference through string cascades

Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

(Pereira & Riley, 1997)
Inference through string cascades

Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

(Pereira & Riley, 1997)
Inference through string cascades

Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

(Pereira & Riley, 1997)
Inference through string cascades

Given a string and a cascade, calculate the highest weighted transformation of the string by the cascade

\[ \text{I-BEST}( \begin{array}{cc} \text{A} & \text{B} \\ \text{B}:\text{.2} & \text{A}:\text{.4} \end{array} ) = ? \]

(Pereira & Riley, 1997)
Pipeline approach

I-BEST(A B) = ?

Embed the string

(Pereira & Riley, 1997)
Pipeline approach

I-BEST( a \rightarrow b \rightarrow c ) = ?

Embed the string

(Pereira & Riley, 1997)
Pipeline approach


Compose the cascade

(Pereira & Riley, 1997)
1-BEST(

Compose the cascade

(Pereira & Riley, 1997)
Pipeline approach

Compose the cascade

(Pereira & Riley, 1997)
Algorithmic Contribution II: Efficient Inference

Pipeline approach

I-BEST(\(\ldots\)) = ?

Project the range

(Pereira & Riley, 1997)
Pipeline approach

Find the 1-best path of the result

\(1\text{-BEST}(\text{ADF}) = ?\)

(Dijkstra, 1959)
Algorithmic Contribution II: Efficient Inference

Pipeline approach

Find the 1-best path of the result

(Dijkstra, 1959)
Algorithmic Contribution II: Efficient Inference

Pipeline approach

I-BEST( ) = ?

Find the 1-best path of the result

(Dijkstra, 1959)
Problems with pipeline

- Extra work done to create unused arcs
- Building done without input of all cascade members
On-the-fly approach

\[ \text{I-BEST( } \begin{array}{c} a \\ b \\ c \\ d \\ e \\ f \end{array} \text{ ) } = ? \]

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)
On-the-fly approach

Algorithmic Contribution II: Efficient Inference

1-BEST(  ) = ?

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)
Algorithmic Contribution II: Efficient Inference

On-the-fly approach

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

Monday, April 19, 2010
On-the-fly approach

Algorithmic Contribution II: Efficient Inference

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)
On-the-fly approach

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)
On-the-fly approach

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)
On-the-fly approach

Algorithmic Contribution II: Efficient Inference

Build arcs in result graph as needed
All members of cascade “vote” simultaneously
Less total construction cost

(Mohri, Pereira, Riley, 1999)
On-the-fly approach

I-BEST(  

• Build arcs in result graph as needed
• All members of cascade “vote” simultaneously
• Less total construction cost

(Mohri, Pereira, Riley, 1999)
On-the-fly approach

- Build arcs in result graph as needed
- All members of cascade “vote” simultaneously
- Less total construction cost

(Mohri, Pereira, Riley, 1999)

Algorithmic Contribution II: Efficient Inference
Inference through *tree* cascades?

- In general, tree transducers are not closed under composition

- However, some classes are closed, and by adding additional steps to the process, we can conduct inference

- We provide pipeline and on-the-fly algorithms for applicable classes of weighted tree transducers
Inference through tree cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade
Inference through *tree cascades*

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade.
Inference through tree cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade.

S

U

U

S

U

U

\[ d \rightarrow S \rightarrow T \]

\[ d \rightarrow S \rightarrow H \]

\[ d \rightarrow U \rightarrow V \]

\[ e \rightarrow U \rightarrow V \]

\[ f \rightarrow U \rightarrow W \]

Monday, April 19, 2010
Inference through *tree* cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade.

\[
\begin{align*}
S 
\quad + 
\end{align*}
\]
Inference through tree cascades

Given a tree and a cascade, calculate the highest weighted transformation of the tree by the cascade.
Pipeline approach

\[ \text{1-BEST(} \quad \begin{array}{c} S \\
\text{U} \\
\text{U} \end{array} \quad \) = ? \]

\[ \begin{array}{c}
\text{d} \\
\text{S} \\
\text{V} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{d} \\
\text{S} \\
\text{H} \\
\text{U} \\
\text{V} \\
\text{W} \\
\end{array} \quad + \quad \begin{array}{c}
\text{e} \\
\text{U} \\
\text{V} \\
\text{f} \\
\text{U} \\
\text{W} \\
\end{array} \]

\[ + \quad \begin{array}{c}
\text{g} \\
\text{T} \\
\text{Y} \\
\text{Z} \\
\end{array} \quad + \quad \begin{array}{c}
\text{g} \\
\text{V} \\
\text{Z} \\
\end{array} \]
Pipeline approach

I-BEST (tree) = ?

Embed the tree
Pipeline approach

\[ \text{I-BEST}(\ldots) = ? \]

Compose adjacent transducers
Algorithmic Contribution II: Efficient Inference

Pipeline approach

1-BEST(

```
> ad .3 ➔ T
  bd ce

> ad .7 ➔ H
  cf be

> bd .2 ➔ V
  cf be

> be .8 ➔ V

> cf .6 ➔ W

> ce .8 ➔ V
```

+ 

```
> g T .4 ➔ Y
  g g

> g V .9 ➔ Z
```

) = ?

New step!

Project the range

Monday, April 19, 2010
Algorithmic Contribution II: Efficient Inference

Pipeline approach

Embed the grammar

New step!
Pipeline approach

I-BEST(1)

Identity transducer has more composition cases
Pipeline approach

Compose adjacent transducers

\[ \text{I-BEST}(\text{adg} \xrightarrow{0.12} \text{Y} \xrightarrow{0.18} \text{Z} \xrightarrow{0.72} \text{Z}) = ? \]
Pipeline approach

I-BEST( adg → Y ceg, bdg ) = ?

Project the range
Pipeline approach

Find 1-best path of the result

(Knuth ’77)
On-the-fly approach

I-BEST( ) = ?
On-the-fly approach

I-BEST(\(\text{adg} \to \text{Y}
\)) = ?
On-the-fly approach

I-BEST(

$\begin{array}{c}
\text{x} \\
\text{a} \\
\text{b} \\
\text{c} \\
\text{d} \\
\text{e} \\
\text{f} \\
\text{g} \\
\text{h} \\
\text{i} \\
\text{j} \\
\text{k} \\
\text{l} \\
\text{m} \\
\text{n} \\
\text{o} \\
\text{p} \\
\text{q} \\
\text{r} \\
\text{s} \\
\text{t} \\
\text{u} \\
\text{v} \\
\text{w} \\
\text{x} \\
\text{y} \\
\text{z}
\end{array}$

) = ?

127
On-the-fly approach

\[ \text{I-BEST} = ? \]
Algorithmic Contribution II: Efficient Inference

On-the-fly approach

I-BEST( ) = ?

never paired together

Monday, April 19, 2010
On-the-fly vs. pipeline

- We recovered 1-best English tree through this cascade
- We calculated time to complete for several language models and both pipeline and on-the-fly methods
- On-the-fly was much faster and in some cases the only method that worked in the memory allotted

(Yamada & Knight, 2001)
## On-the-fly vs. pipeline

<table>
<thead>
<tr>
<th>language model</th>
<th>method</th>
<th>time/sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>weak</td>
<td>pipeline</td>
<td>28s</td>
</tr>
<tr>
<td>strong &amp; large</td>
<td>on-the-fly</td>
<td>17s</td>
</tr>
<tr>
<td>strong &amp; small</td>
<td>pipeline</td>
<td>&gt;60s*</td>
</tr>
<tr>
<td></td>
<td>on-the-fly</td>
<td>24s</td>
</tr>
<tr>
<td></td>
<td>pipeline</td>
<td>2.5s</td>
</tr>
<tr>
<td></td>
<td>on-the-fly</td>
<td>.06s</td>
</tr>
</tbody>
</table>

*Ran out of memory before completing*
Extension for tree-string transducers

What if the cascade ends in a tree-string transducer, and we want to pass a string through the cascade?

```
PRP VB NN
i hate snakes
```

- Re-order English words
- Insert function words
- Translate to Japanese

ヘビが大嫌いだ
Extension for tree-string transducers

What if the cascade ends in a tree-string transducer, and we want to pass a string through the cascade?
Extension for tree-string transducers

What if the cascade ends in a tree-string transducer, and we want to pass a string through the cascade?

- Re-order English words
- Insert function words
- Translate to Japanese

- i hate snakes

- on-the-fly or pipeline
- cfg parsing
A weighted tree automata and transducer toolkit

(May & Knight, CIAA ’06)

• Operations for inference, manipulation, and training of tree transducers and automata

• Very easy to experiment quickly, without coding

• http://www.isi.edu/licensed-sw/tiburon
Tiburon example 1: syntax MT cascade

Simplified English trees to Japanese strings

\[
\text{TOP} \quad \text{VB} \quad \text{PRP} \quad \text{VB} \quad \text{NN} \\
i \quad \text{hate} \quad \text{snakes}
\]

\[\text{へビが大嫌いだ}\]

(Yamada & Knight, 2001)
Tiburon example 1: syntax MT cascade

1) Rotate children

(Yamada & Knight, 2001)
Tiburon example 1: syntax MT cascade

2) Insert function words

(Yamada & Knight, 2001)
Tiburon example 1: syntax MT cascade

3) Translate leaves

\(\text{hate} \rightarrow \text{大嫌い}\)

(Yamada & Knight, 2001)
Tiburon example 1: syntax MT cascade

- Task: Decode candidate sentence, get top 5 answers
- Algorithms used: inference through cascade, k-best, determinization

Candidate: 彼らは偽善が大嫌いだ

Correct answer:
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))))
Tiburon example 1: syntax MT cascade

Let’s try it!

% tiburon  -k 5  -m tropical  -e euc-jp  rot ins trans  ej.1.f
Tiburon example 1: syntax MT cascade

Let’s try it!

% tiburon -k 5 -m tropical -e euc-jp rot ins trans ej.1.f

program
Let’s try it!

% tiburon -k 5 -m tropical -e euc-jp rot ins trans ej.1.f

5 best
Tiburon example 1: syntax MT cascade

Let's try it!

% tiburon -k 5 -m tropical -e euc-jp rot ins trans ej.1.f

semiring
Practical Contribution I: Tiburon

Tiburon example 1: syntax MT cascade

Let's try it!

% tiburon  -k 5  -m tropical  -e euc-jp  rot ins trans  ej.1.f
Tiburon example 1: syntax MT cascade

Let’s try it!

% tiburon -k 5 -m tropical -e euc-jp rot ins trans ej.1.f
Tiburon example 1: syntax MT cascade

Let's try it!

% tiburon -k 5 -m tropical -e euc-jp rot ins trans ej.1.f
Tiburon example 1: syntax MT cascade

First try is not so good!
Tiburon example 1: syntax MT cascade

Add in a simple PCFG-based language model
Tiburon example 1: syntax MT cascade

Add in a simple PCFG-based language model
Tiburon example 1: syntax MT cascade

Add in a simple PCFG-based language model

% tiburon -k 5 -m tropical -e euc-jp pcfg-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))#) # 33.024
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))#) # 33.718
TOP(VB(PRP("him") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))#) # 33.718
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("him"))))#) # 33.718
TOP(VB(PRP("them") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("i"))))#) # 33.718

Monday, April 19, 2010
Tiburon example 1: syntax MT cascade

Try a grandparent language model
Tiburon example 1: syntax MT cascade

Try a grandparent language model
Tiburon example 1: syntax MT cascade

Try a grandparent language model

% tiburon -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.603
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them")))))) # 27.297
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.033
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.071
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.726
Practical Contribution I: Tiburon

Tiburon example 1: syntax MT cascade

Try a grandparent language model

% tiburon -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.603
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.297
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.033
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.071
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.726

Correct sentence is 5th
Tiburon example 1: syntax MT cascade

Try a grandparent language model

Correct sentence is 5th
Tiburon example 1: syntax MT cascade

• Combine duplicate derivations in entire search space using *weighted determinization*
Tiburon example 1: syntax MT cascade

- Combine duplicate derivations in entire search space using *weighted determinization*

% tiburon -d 5 -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRPI"i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them")))) # 26.329
TOP(VB(PRPI"i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them")))) # 27.023
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them")))))) # 27.759
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them")))))) # 28.452
TOP(VB(NN(DT("a") NN("clouds") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them")))))) # 31.250
Tiburon example 1: syntax MT cascade

• Combine duplicate derivations in entire search space using **weighted determinization**

% tiburon -d 5 -k 5 -m tropical -e euc-jp gp-lm rot ins trans ej.1.f
TOP(VB(PRP("i") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 26.329
TOP(VB(PRP("i") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.023
TOP(VB(NN("hypocrisy") VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 27.759
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 28.452
TOP(VB(NN(DT("a") NN("clouds")) VB("abominate") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 31.250
Tiburon example 2: training a syntax LM

- The LMs we used before had no hidden states
- Let’s introduce hidden states and learn weights with EM

(Petrov & Klein, ’07)
Tiburon example 2: training a syntax LM

- The LMs we used before had no hidden states
- Let’s introduce hidden states and learn weights with EM

(Petrov & Klein, ’07)
Tiburon example 2: training a syntax LM

- The LMs we used before had no hidden states
- Let's introduce hidden states and learn weights with EM

(Petrov & Klein, ’07)
Tiburon example 2: training a syntax LM

% tiburon -t 50 --randomize trees rtg.4split > 4split-lm
Tiburon example 2: training a syntax LM

% tiburon -t 50 --randomize trees rtg.4split > 4split-lm

50 iterations
Tiburon example 2: training a syntax LM

% tiburon -t 50 --randomize trees rtg.4split > 4split-lm

random initial weights avoids saddles
Tiburon example 2: training a syntax LM

% tiburon -t 50 --randomize trees rtg.4split > 4split-lm

training data
Tiburon example 2: training a syntax LM

% tiburon -t 50 --randomize trees rtg.4split > 4split-lm
Tiburon example 2: training a syntax LM

% tiburon -t 50 --randomize trees rtg.4split > 4split-lm
Cross entropy with normalized initial weights is 1.868; corpus prob is $e^{-269.025}$
Cross entropy after 1 iterations is 1.190; corpus prob is $e^{-171.383}$
Cross entropy after 2 iterations is 1.138; corpus prob is $e^{-163.866}$
Cross entropy after 3 iterations is 1.036; corpus prob is $e^{-149.229}$
...
Cross entropy after 47 iterations is 0.581; corpus prob is $e^{-83.665}$
Cross entropy after 48 iterations is 0.581; corpus prob is $e^{-83.665}$
Cross entropy after 49 iterations is 0.581; corpus prob is $e^{-83.665}$
Tiburon example 2: training a syntax LM

% tiburon -t 50 --randomize trees rtg.4split > 4split-lm
Cross entropy with normalized initial weights is 1.868; corpus prob is e^-269.025
Cross entropy after 1 iterations is 1.190; corpus prob is e^-171.383
Cross entropy after 2 iterations is 1.138; corpus prob is e^-163.866
Cross entropy after 3 iterations is 1.036; corpus prob is e^-149.229
...
Cross entropy after 47 iterations is 0.581; corpus prob is e^-83.665
Cross entropy after 48 iterations is 0.581; corpus prob is e^-83.665
Cross entropy after 49 iterations is 0.581; corpus prob is e^-83.665

Compare with GP-PCFG

% tiburon -t 3 --randomize trees rtg.gp.pcfg > lm
Cross entropy with normalized initial weights is 0.827; corpus prob is e^-119.022
Cross entropy after 1 iterations is 0.732; corpus prob is e^-105.448
Cross entropy after 2 iterations is 0.732; corpus prob is e^-105.448

Monday, April 19, 2010
We can subjectively see state specialization.
Tiburon example 2: training a syntax LM

% tiburon -k 5 -m tropical -e euc-jp 4split-lm rot ins trans ej.1.f
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.556
TOP(VB(NN("fanatic") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.556
TOP(VB(NN("clouds") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.556
TOP(VB(NN("fanatic") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.717
TOP(VB(NN("hypocrisy") VB("is") JJ(JJ("abhorrent") TO(TO("to") PRP("them"))))) # 29.717

Tied for first!
Using tree transducers to improve machine translation

(May & Knight, EMNLP ’07)

• We will now shift focus to improving state-of-the-art syntax MT results

• At core, we’re using the power of training tree transducers to achieve gains
Extracting syntactic rules

1) Obtain alignments
Extracting syntactic rules

1) Obtain alignments

(Galley et al. ’04, ’06)
Extracting syntactic rules

2) Add parse tree

(Taiwan's surplus in trade between the two shores)

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(TAIWAN IN TWO-SHORES TRADE MIDDLE SURPLUS)

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. '04, '06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. '04, '06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
3) Extract rules

(Galley et al. '04, '06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

Taiwan

Surplus

Trade

(Taiwan et al. '04, '06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Taiwan's surplus can explain it; the first set is obtained from bootstrap alignments, the second procedure, and the third is a viable, if poor quality, alternative that is not le.

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Extracting syntactic rules

3) Extract rules

(Galley et al. '04, '06)
Extracting syntactic rules

3) Extract rules

(Galley et al. '04, '06)
Extracting syntactic rules

3) Extract rules

(Galley et al. ’04, ’06)
Bad alignments make bad rules

One bad link makes a totally unusable syntax rule!
Bad alignments make bad rules

One bad link makes a totally unusable syntax rule!
Where do the alignments come from?

- (e-string, f-string)
- sentence pairs
- generative model (IBM model 4) (Brown et al., '93)
- seed data
- unsupervised learning (GIZA++) (Och and Ney, '03)
- Viterbi alignments

Notice, nothing about syntax!
Let's add syntax!

(sentence pairs)

(seed data)

(mult-tree, f-string)

(generative model)

(unsupervised learning)

(Viterbi alignments)

(tree-string syntax rules)

(Training Tree Transducers) (Graehl, Knight, May '08)
Let's add syntax!

(sentence pairs)

(seed data)

(unsupervised learning)

(Viterbi alignments)

(tree-string syntax rules)

(Training Tree Transducers) (Graehl, Knight, May '08)

(tree-to-string transducer)
Let's add syntax!

(sentence pairs)

seed data

unsupervised learning

Viterbi alignments

(tree-string syntax rules)

(Training Tree Transducers) (Graehl, Knight, May '08)

(tree-to-string transducer)

(e-tree, f-string)

generative model

(syntax model)

tiburon -t ...
How we learn

- For each training sentence, build *derivation forest* containing each possible tree of rules that satisfies the sentence pair
- EM iterations set highest probability to most useful rules
- Viterbi derivation has syntax-aware alignments and bad rules are not extracted

(Graehl, Knight, May, ’08)
Experiments

- Build a bootstrap alignment with GIZA
- Obtain new alignments with syntactic re-alignment
- Compare syntax MT system performance on rules extracted from each alignment
## Results

<table>
<thead>
<tr>
<th>source language</th>
<th>original alignments</th>
<th>type</th>
<th>MT system rules (millions)</th>
<th>NIST 2003 BLEU</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>weak</td>
<td>baseline</td>
<td>2.3</td>
<td>47.3</td>
<td>+.6</td>
</tr>
<tr>
<td></td>
<td>re-alignment</td>
<td></td>
<td>2.5</td>
<td>47.9</td>
<td></td>
</tr>
<tr>
<td>Arabic</td>
<td>strong</td>
<td>baseline</td>
<td>3.2</td>
<td>49.6</td>
<td>+.4</td>
</tr>
<tr>
<td></td>
<td>re-alignment</td>
<td></td>
<td>3.6</td>
<td>50.0</td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td>weak</td>
<td>baseline</td>
<td>19.1</td>
<td>37.8</td>
<td>+.9</td>
</tr>
<tr>
<td></td>
<td>re-alignment</td>
<td></td>
<td>26.0</td>
<td>38.7</td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td>strong</td>
<td>baseline</td>
<td>23.4</td>
<td>38.9</td>
<td>+1.1</td>
</tr>
<tr>
<td></td>
<td>re-alignment</td>
<td></td>
<td>33.4</td>
<td>40.0</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions and future work

• Algorithmic contributions
  • Determinization of weighted tree automata
  • Efficient inference through weighted tree transducer cascades

• Practical contributions
  • Weighted tree automata and transducer toolkit
  • Improvements in SMT using tree transducer EM
Future work

• More algorithms!
  • approximate linear k-best
  • on-the-fly tree-to-string inference
• More applications!
  • financial systems
  • gene sequencing
• More formalisms!
  • unranked automata
  • tree-adjoining grammars
Conclusions

• Tiburon makes it easy to use tree transducers in NLP

• (known) Theses using Tiburon:
  • Alexander Radzievskiy -- Masters on parsing with semantic role labels
  • Joseph Tepperman -- PhD on pronunciation evaluation
  • Victoria Fossum -- PhD on machine translation and parsing

• July 2010: ATANLP in Uppsala!
Thanks!

Algorithmic Contribution I: WTA Determinization

Non-deterministic and nonterminal?

```
(t) .2 → D
  q
  r

(u) .3 → D
  q
  r

+ q/1

= .125

(t/.4 u/.6) → D
  q/1
  r/25
  s/.75
```

Monday, April 19, 2010
MT Details

- Decoded 116 short Chinese sentences using the string-to-tree MT model based on (Galley et al. 2004)
  - No language model
  - No reranking
- Counted number of trees in each forest before and after determinization
- 86.3% trees in forest are duplicates on average
  - $1.4 \times 10^{12}$ median per forest pre-determ
  - $2.0 \times 10^{11}$ median per forest post-determ
- Determinization changes top tree 77.6% of the time
- Crunching matches determinization 50.6% of the time
xLNT not closed!

could be arbitrarily long!

(Maletti, Graehl, Hopkins, Knight, ’09)
Closure Under Composition and Recognizability Preservation

<table>
<thead>
<tr>
<th></th>
<th>forward recog</th>
<th>backward recog</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wLNT</td>
<td>wxLNT</td>
<td>xT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>wxLT</td>
</tr>
</tbody>
</table>
Where do the rules come from?

- Ideally we would add all possible rules
- To avoid overflow, we bootstrap with a previous (syntax-free) alignment model
- This follows a rich history in MT (Och & Ney ’00, Fraser & Marcu ‘06)

103 possible rules
Other approaches to this problem

- Cherry and Lin ’06: Discriminatively train ITG-based alignment model influenced by dependency graph
- DeNero and Klein ’07: HMM model modified to incorporate syntax penalty into distortion
- Fossum et al. ‘08: Identify troublesome links and remove them
Where do the rules come from?

- (e-string, f-string)
- IBM
- GIZA
- word pairs
- bootstrap alignments
Where do the rules come from?

(e-string, f-string) → IBM → GIZA → word pairs → bootstrap alignments

rule extraction → rules

(Galley et al. '04)
Where do the rules come from?

- (e-string, f-string)
- IBM
- (e-tree, f-string)
- syntax
- word pairs
- GIZA
- bootstrap alignments
- rule extraction
- rules
- TTT
- final alignments

(Galley et al. '04)
EM size bias

- EM attempts to learn derivations with highest probability.
- Shorter derivations have fewer chances to take a probability “hit” and are thus biased to be favored.
- This, then, tends to favor larger rules, generally the opposite of what we want.
Correcting size bias

- When using a rule with \( n \) non-leaf nodes, prepend \( n-1 \) copies of a special size rule \( S_n \)
- Each competing derivation now has the same number of rules
- Size rules are built into the derivation forests and weights are learned by the same EM procedure
## Complexity Analysis

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>k-best (H&amp;C)</td>
<td>( O(P + D_{\text{max}}k \log k) )</td>
<td>( P = \text{rtg rules} ) ( D_{\text{max}} = \text{max deriv} )</td>
</tr>
<tr>
<td>determinization</td>
<td>( O(Ak^zL) )</td>
<td>( A = \text{alph size} ) ( k = \text{max rank} ) ( z = \text{max tree size} ) ( L = \text{lang size} )</td>
</tr>
<tr>
<td>rtg+xLNT</td>
<td>( O(RP^l) )</td>
<td>( R = \text{trans rules} ) ( P = \text{rtg rules} ) ( l = \text{max trans lhs} )</td>
</tr>
<tr>
<td>xT+LNT</td>
<td>( O(R_AR_B^r) )</td>
<td>( R_A = \text{xT rules} ) ( R_B = \text{LNT rules} ) ( r = \text{max } R_A \text{ rhs} )</td>
</tr>
</tbody>
</table>
Dramatic use of size rules

\[ R_{\text{bad}}: \]

\[ \rightarrow x0 \text{ 对外 开放} \]

\[ S_{15} \]

\[ \rightarrow 14 \text{ times} \]
Approximate Algorithms

• linear-time approximate $k$-best
• polynomial time determinization that fails to recognize some trees in the input
• weighted domain projection with relative ordering, but not exact weights, preserved
• mildly incorrect fast composition
• on-the-fly tree-to-string backward application
Engineering

• Battle-test Tiburon implementations and bring it up to production level

• Make greater use of system on biological sequencing and financial systems analysis -- leads to more interesting algorithmic questions, different types of transducers
Explore the limits of Tree Transducers

• Weighting scheme of Collins’ parsing model\(^1\) doesn’t fit well

• Very large tree transducers needed in syntax MT\(^2\)

• Can these models be simplified and still retain their power? Or should different formalisms be used?

1: Collins, 1997
2: DeNeefe and Knight, 2009