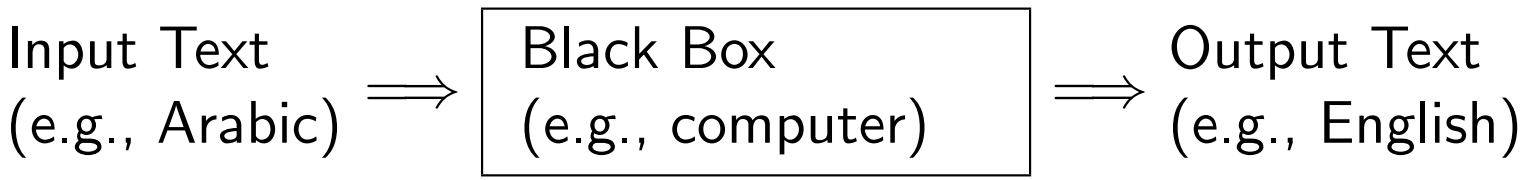
Tree Transducers in Machine Translation

Andreas Maletti

December 6, 2006

Machine Translation

Schema



Applications

- Automatic news wire translations
- Automatic translation of proceedings
- ...
- Powerful in conjunction with speech recognition and synthesis

MT-Evaluation 2006

See for yourself:

http://www.nist.gov/speech/tests/mt/mt06eval_official_results.html

Disclaimer

These results are not to be construed, or represented as endorsements of any participant's system or commercial product, or as official findings on the part of NIST or the U.S. Government.
$\langle \ldots \rangle$

MT-Evaluation 2006 (cont'd)

Conditions:

- Large Data Track (limited training data; publicly available)
- Arabic-to-English and Chinese-to-English
- 1 week processing time
- $\approx$ 40 competitors (industrial and academic)

Results

Site	BLEU-score
google	0.4281
ibm	0.3954
isi	0.3908
rwth	0.3906
apptek	0.3874

MT-Evaluation 2006 (cont'd)

Conditions:

- Large Data Track (limited training data; publicly available)
- Arabic-to-English and Chinese-to-English
- 1 week processing time
- $\approx$ 40 competitors (industrial and academic)

Results

Site	BLEU-score
google	0.4281
ibm	0.3954
isi	0.3908
rwth	0.3906
apptek	0.3874

Site	BLEU-score
isi	0.3393
google	0.3316
lw	0.3278
rwth	0.3022
ict	0.2913

Phrase-based Machine Translation

Features

- Process input and output string
- String-based transformations
- $\Rightarrow$ (Weighted) finite-state automata and transducers

Example

Translation with FSA and FST

Mary did not slap the green witch.

Phrase-based Machine Translation

Features

- Process input and output string
- String-based transformations
- $\Rightarrow$ (Weighted) finite-state automata and transducers

Example

Translation with FSA and FST

Mary did not slap the green witch.

Mary ε not slap slap slap the green witch.

Phrase-based Machine Translation

Features

- Process input and output string
- String-based transformations
- $\Rightarrow$ (Weighted) finite-state automata and transducers

Example

Translation with FSA and FST

Mary ε not slap slap slap the green witch.

Mary not slap slap slap NULL the green witch.

Phrase-based Machine Translation

Features

- Process input and output string
- String-based transformations
- $\Rightarrow$ (Weighted) finite-state automata and transducers

Example

Translation with FSA and FST

Mary not slap slap slap NULL the green witch.

Mary no dió una bofetada a la verde bruja.

Phrase-based Machine Translation

Features

- Process input and output string
- String-based transformations
- $\Rightarrow$ (Weighted) finite-state automata and transducers

Example

Translation with FSA and FST

Mary no dió una bofetada a la verde bruja.

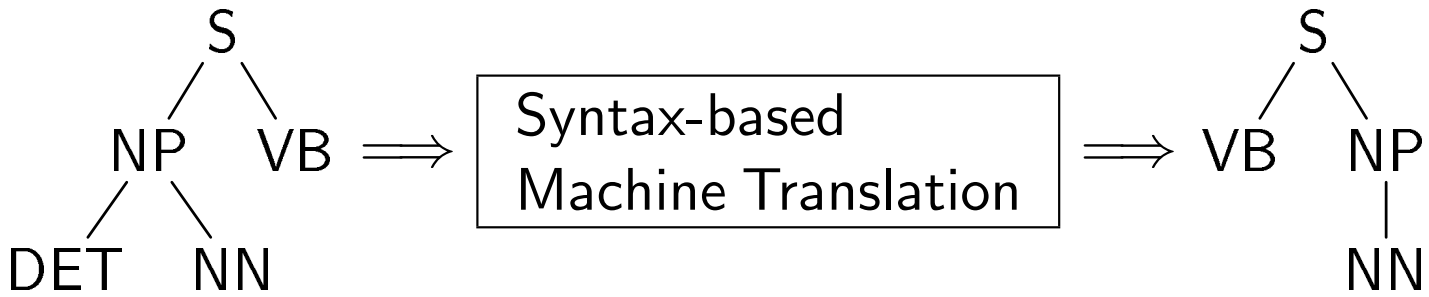
Mary no dió una bofetada a la bruja verde.

Syntax-based Machine Translation

Features

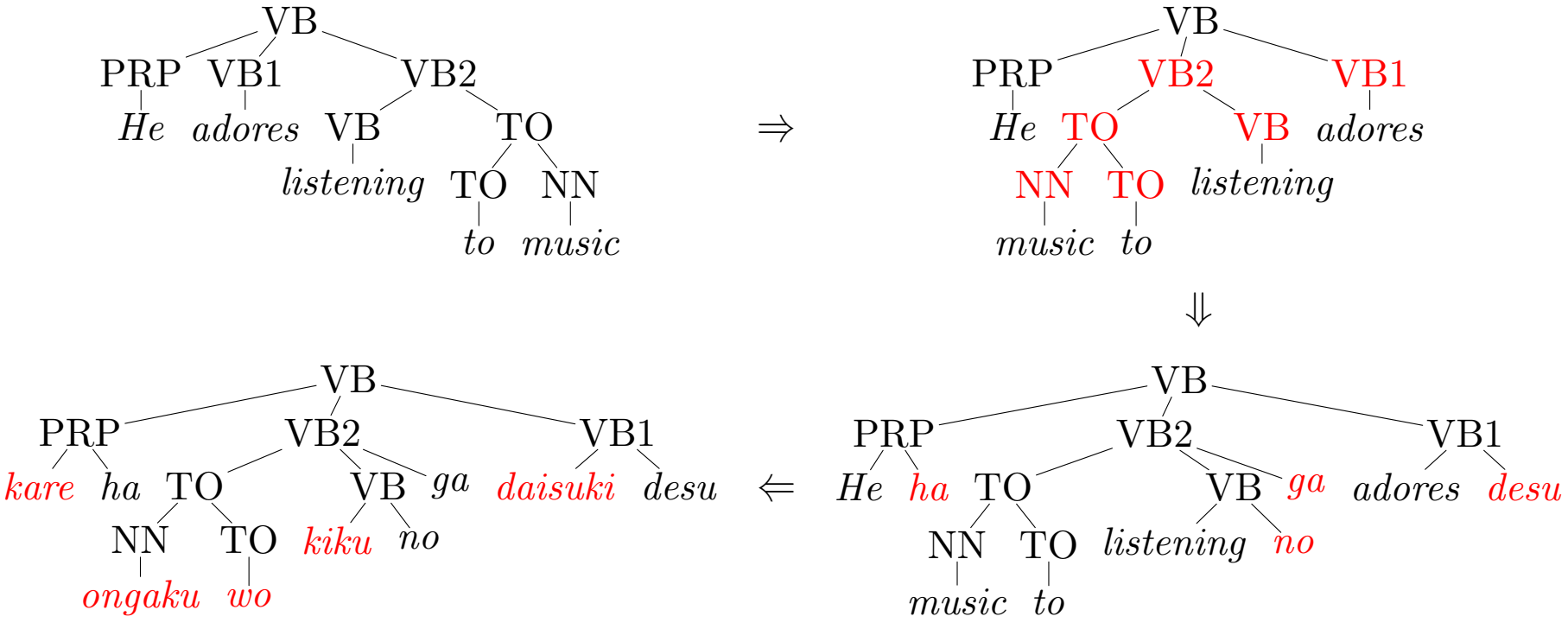
- Process parse trees of sentence instead of sentence
- Tree-based transformations
- $\Rightarrow$ (Weighted) tree automata and tree transducers

Schema



Syntax-based Machine Translation (cont'd)

Example



kare ha ongaku wo kiku no ga daisuki desu

Translation Patterns

Extended Top-down Tree Transducer

Definition

An extended top-down tree transducer is a tuple $(Q, \Sigma, \Delta, I, R)$ where

- Q is a finite set of states;
- Σ and Δ are input and output ranked alphabet;
- $I \subseteq Q$ is a set of initial states; and
- R is a finite set of rewrite rules of the form

$$q(t) \to r$$

with $q \in Q$, $t \in T_\Sigma(X)$ linear, and $r \in T_\Delta(Q(\mathrm{var}(t)))$.

Semantics of Extended Top-down Tree Transducers

$M = (Q, \Sigma, \Delta, I, R)$ an extended tdtt

Definition

Define $\Rightarrow_M \subseteq T_\Delta(Q(T_\Sigma))^2$ by $\xi \Rightarrow_M \xi'$ iff

- there exists a position $w \in \mathrm{pos}(\xi)$;
- there exists a rule $(l \to r) \in R$; and
- there exists a substitution $\theta \colon X \to T_\Sigma$

such that

$$l\theta = \xi|_w \quad \text{and} \quad \xi' = \xi[r\theta]_w \ .$$

Definition

The tree transformation computed by M is defined by

$$\|M\| = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in I \colon q(t) \Rightarrow_M^* u\} \ .$$

A Hierarchy

Open Problems (according to [Knight, Graehl 2005])

- What is the most efficient algorithm for selecting the k-best trees from a probabilistic regular tree grammar?

- How can efficient integrated search be carried out, so that all tree acceptors and transducers in a cascade can simultaneously participate in the best-tree search?

Open Problems (according to [Knight, Graehl 2005])

- What is the most efficient algorithm for selecting the k-best trees from a probabilistic regular tree grammar?

  Better k-best Parsing by L. Huang and D. Chiang
  $O(|E| + |D_{\max}| k \log k)$ for k-best derivations

- How can efficient integrated search be carried out, so that all tree acceptors and transducers in a cascade can simultaneously participate in the best-tree search?

Open Problems (according to [Knight, Graehl 2005])

- What is the most efficient algorithm for selecting the k-best trees from a probabilistic regular tree grammar?

  Better k-best Parsing by L. Huang and D. Chiang
  $O(|E| + |D_{\max}| k \log k)$ for k-best derivations

- How can efficient integrated search be carried out, so that all tree acceptors and transducers in a cascade can simultaneously participate in the best-tree search?

  open

Open Problems (according to [Knight, Graehl 2005])

- What search heuristics (beaming, thresholding, etc.) are necessary for efficient application of tree transducers to large-scale natural language problems?

- What is the most efficient algorithm for composing probabilistic linear and nondeleting tree transducers?

Open Problems (according to [Knight, Graehl 2005])

- What search heuristics (beaming, thresholding, etc.) are necessary for efficient application of tree transducers to large-scale natural language problems?

  mostly open (some ideas in Tiburon)

- What is the most efficient algorithm for composing probabilistic linear and nondeleting tree transducers?

Open Problems (according to [Knight, Graehl 2005])

- What search heuristics (beaming, thresholding, etc.) are necessary for efficient application of tree transducers to large-scale natural language problems?

  mostly open (some ideas in Tiburon)

- What is the most efficient algorithm for composing probabilistic linear and nondeleting tree transducers?

  Bottom-up and Top-down Tree Series Transformations by J. Engelfriet, Z. Fülöp, and H. Vogler
  complexity not analysed; not specific for $\mathbb{R}$

Open Problems (according to [Knight, Graehl 2005])

- What is the most efficient algorithm for intersecting probabilistic regular tree grammars?

- What are the most efficient algorithms for forward and backward application of tree/tree and tree/string transducers?

Open Problems (according to [Knight, Graehl 2005])

- What is the most efficient algorithm for intersecting probabilistic regular tree grammars?

  The Theory of Recognizable Tree Series by B. Borchardt
  no complexity analysis; not specific for $\mathbb{R}$

- What are the most efficient algorithms for forward and backward application of tree/tree and tree/string transducers?

Open Problems (according to [Knight, Graehl 2005])

- What is the most efficient algorithm for intersecting probabilistic regular tree grammars?

  The Theory of Recognizable Tree Series by B. Borchardt
  no complexity analysis; not specific for $\mathbb{R}$

- What are the most efficient algorithms for forward and backward application of tree/tree and tree/string transducers?

  custom implementation in Tiburon

Open Problems (according to [Knight, Graehl 2005])

- For large tree transducers, what data structures, indexing strategies, and caching techniques will support efficient algorithms?

- What is the linguistically most appropriate tree transducer class for machine translation? For text summarization? Which classes best handle the most common linguistic constructions, and which classes best handle the most difficult ones?

Open Problems (according to [Knight, Graehl 2005])

- For large tree transducers, what data structures, indexing strategies, and caching techniques will support efficient algorithms?

  Prototype implementation in Tiburon

- What is the linguistically most appropriate tree transducer class for machine translation? For text summarization? Which classes best handle the most common linguistic constructions, and which classes best handle the most difficult ones?

Open Problems (according to [Knight, Graehl 2005])

- For large tree transducers, what data structures, indexing strategies, and caching techniques will support efficient algorithms?

  Prototype implementation in Tiburon

- What is the linguistically most appropriate tree transducer class for machine translation? For text summarization? Which classes best handle the most common linguistic constructions, and which classes best handle the most difficult ones?

  nondeleting and linear extended top-down for machine translation; open for summarization

Open Problems (according to [Knight, Graehl 2005])

- Can compact regular tree grammars encode high-performing tree-based language models with appropriate back-off strategies, in the same way that FSA tools can implement *n*-gram models?

- What are the theoretical and computational properties of extended top-down tree transducers?

Open Problems (according to [Knight, Graehl 2005])

- Can compact regular tree grammars encode high-performing tree-based language models with appropriate back-off strategies, in the same way that FSA tools can implement *n*-gram models?

  open

- What are the theoretical and computational properties of extended top-down tree transducers?

Open Problems (according to [Knight, Graehl 2005])

- Can compact regular tree grammars encode high-performing tree-based language models with appropriate back-off strategies, in the same way that FSA tools can implement *n*-gram models?

  open

- What are the theoretical and computational properties of extended top-down tree transducers?

  class of nondeleting and linear extended top-down tree transformations not closed under composition

Open Problems (according to [Knight, Graehl 2005])

- Where do synchronous grammars and tree cloning fit into the tree transducer hierarchy?

- As many syntactic and semantic theories generate acyclic graphs rather than trees, can graph transducers adequately capture the desired transformations?

Open Problems (according to [Knight, Graehl 2005])

- Where do synchronous grammars and tree cloning fit into the tree transducer hierarchy?

  syn. grammars as powerful as extended top-down

- As many syntactic and semantic theories generate acyclic graphs rather than trees, can graph transducers adequately capture the desired transformations?

Open Problems (according to [Knight, Graehl 2005])

- Where do synchronous grammars and tree cloning fit into the tree transducer hierarchy?

  syn. grammars as powerful as extended top-down

- As many syntactic and semantic theories generate acyclic graphs rather than trees, can graph transducers adequately capture the desired transformations?

  open

Open Problems (according to [Knight, Graehl 2005])

- Are there tree transducers that can move unbounded material over unbounded distances, while maintaining efficient computational properties?

- In analogy with extended context-free grammars, are there types of tree transducers that can process tree sets which are not limited to a finite set of rewrites (e.g., $S \rightarrow NP\ VP\ PP^*$)?

Open Problems (according to [Knight, Graehl 2005])

- Are there tree transducers that can move unbounded material over unbounded distances, while maintaining efficient computational properties?

  open

- In analogy with extended context-free grammars, are there types of tree transducers that can process tree sets which are not limited to a finite set of rewrites (e.g., $S \rightarrow NP\ VP\ PP^*$)?

Open Problems (according to [Knight, Graehl 2005])

- Are there tree transducers that can move unbounded material over unbounded distances, while maintaining efficient computational properties?

  open

- In analogy with extended context-free grammars, are there types of tree transducers that can process tree sets which are not limited to a finite set of rewrites (e.g., $S \rightarrow NP\ VP\ PP^*$)?

  mostly open

Open Problems (according to [Knight, Graehl 2005])

- Can we build tree transducer models for machine translation that:
  - efficiently train on large amounts of data,
  - accurately model that data by assigning it higher probability than other models, and
  - when combined with search algorithms, yield grammatical and accurate translations?

- Can we build useful, generic tree-transducer toolkits, and what sorts of programming interfaces will be most effective?

Open Problems (according to [Knight, Graehl 2005])

- Can we build tree transducer models for machine translation that:
  - efficiently train on large amounts of data,
  - accurately model that data by assigning it higher probability than other models, and
  - when combined with search algorithms, yield grammatical and accurate translations?

  wide open

- Can we build useful, generic tree-transducer toolkits, and what sorts of programming interfaces will be most effective?

Open Problems (according to [Knight, Graehl 2005])

- Can we build tree transducer models for machine translation that:
  - efficiently train on large amounts of data,
  - accurately model that data by assigning it higher probability than other models, and
  - when combined with search algorithms, yield grammatical and accurate translations?

  wide open

- Can we build useful, generic tree-transducer toolkits, and what sorts of programming interfaces will be most effective?

  Tiburon