Finite-State Technology in Natural Language Processing

Andreas Maletti

Institute for Natural Language Processing
Universität Stuttgart, Germany

maletti@ims.uni-stuttgart.de

Umeå — August 18, 2015

Roadmap

1. Linguistic Basics and Weighted Automata
2. Part-of-Speech Tagging
3. Parsing
4. Machine Translation

Always ask questions right away!

Linguistic Basics

Units

- Sentence (syntactic unit expressing complete thought)
  - Alla sätt är bra utom de dåliga.
  - “Are you serious?” she asked.

Linguistic Basics

Units

- Sentence (syntactic unit expressing complete thought)
- Clause (grammatically complete syntactic unit)
  - Vännens örfil är ärligt menad, fiendens kyssar vill bedra.
    (2 main clauses)
  - People who live in glass houses should not throw stones.
    (main clause + relative clause)

Linguistic Basics

Units

- Sentence (syntactic unit expressing complete thought)
- Clause (grammatically complete syntactic unit)
- Phrases (smaller syntactic units)
  - the green car (noun phrase = noun and its modifiers)
  - killed the snake (verb phrase = verb and its objects)

Linguistic Basics

Units

- Sentence (syntactic unit expressing complete thought)
- Clause (grammatically complete syntactic unit)
- Phrases (smaller syntactic units)
- Token (smallest unit = "word"; often derived from lexicon entry)
  - house, car, lived, smallest, 45th, STACS, Knuth
  - but tricky: Knuth's vs. Knuth 's; well-known vs. well - known

Linguistic Basics

Tokenization

Splitting text into sentences and tokens

- relatively simple for English, Swedish, German, etc.
  (actually often via complicated regular expression)

Linguistic Basics

Tokenization

Splitting text into sentences and tokens

- relatively simple for English, Swedish, German, etc.
  (actually often via complicated regular expression)
- hard for other languages:
  - Chinese: 小洞不补，大洞吃苦。
    (A small hole not plugged will make you suffer a big hole.)

Tokenization

Splitting text into sentences and tokens

- relatively simple for English, Swedish, German, etc.
  (actually often via complicated regular expression)
- hard for other languages:
  - Chinese: 小洞不补，大洞吃苦。
    (A small hole not plugged will make you suffer a big hole.)
  - Turkish: Çekoslovakyalılaştıramadıklarımızdanmışsınız
    (You are said to be one of those
    that we couldn't manage to convert to a Czechoslovak)

Linguistic Basics

Tokenization

Splitting text into sentences and tokens

- relatively simple for English, Swedish, German, etc.
  (actually often via complicated regular expression)
- hard for other languages:
  - Chinese: 小洞不补，大洞吃苦。
    (A small hole not plugged will make you suffer a big hole.)
  - Turkish: Çekoslovakyalılaştıramadıklarımızdanmışsınız
    (You are said to be one of those
    that we couldn't manage to convert to a Czechoslovak)
  - Hungarian: legeslegmegszentségteleníttethetetlenebbjeitekként
    (like the most of most undesecratable ones of you)

Linguistic Basics

Tokenization

Splitting text into sentences and tokens

- relatively simple for English, Swedish, German, etc.
  (actually often via complicated regular expression)
- hard for other languages:
  - Chinese: 小洞不补，大洞吃苦。
    (A small hole not plugged will make you suffer a big hole.)
  - Turkish: Çekoslovakyalılaştıramadıklarımızdanmışsınız
    (You are said to be one of those
    that we couldn't manage to convert to a Czechoslovak)
  - Hungarian: legeslegmegszentségteleníttethetetlenebbjeitekként
    (like the most of most undesecratable ones of you)

Example (English sentence-ending full stop)

RE . ⟨WhiteSpace⟩$^+$ [A–Z] covers most cases (in English)

Linguistic Basics

Example (English)

- tokens usually separated by whitespace
- sentence end marker "." highly ambiguous:
  - common abbreviations
  - dates, ordinals, and phone numbers

Linguistic Basics

Example (English)

- tokens usually separated by whitespace
- sentence end marker "." highly ambiguous:
  - common abbreviations
  - dates, ordinals, and phone numbers
- STANFORD tokenizer
  - implemented in JAVA (based on JFlex)
  - compiles RegEx into DFA and runs DFA
  - can process 1,000,000 tokens per second

Weighted Automata

Definition

A weighted automaton is a system $(Q, \Sigma, I, \Delta, F, \mathrm{wt})$

- finite set Q of states
- input alphabet Σ
- initial states $I \subseteq Q$
- transitions $\Delta \subseteq Q \times \Sigma \times Q$
- final states $F \subseteq Q$

Example

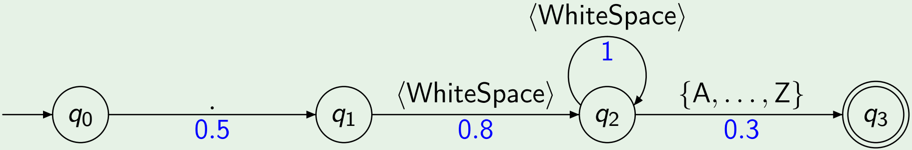
Weighted Automata

Definition

A weighted automaton is a system $(Q, \Sigma, I, \Delta, F, \mathrm{wt})$

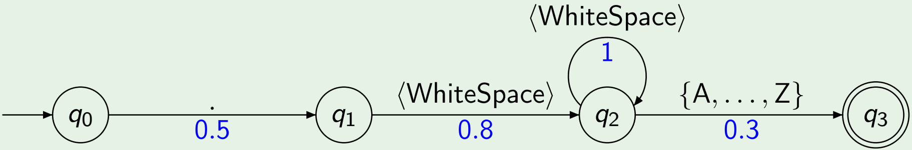
- finite set Q of states
- input alphabet Σ
- initial states $I \subseteq Q$
- transitions $\Delta \subseteq Q \times \Sigma \times Q$
- final states $F \subseteq Q$
- transition weights $\mathrm{wt} \colon \Delta \to [0, 1]$

Example



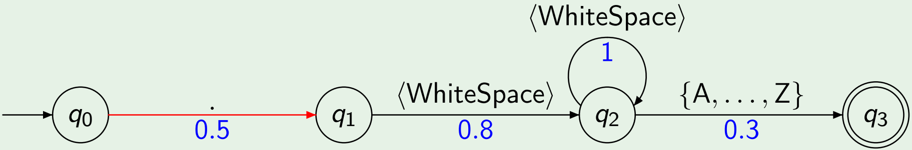
Weighted Automata

Example



Weighted Automata

Example

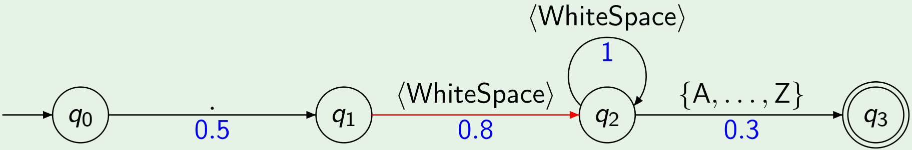


Definition (Semantics)

- Weight of a run: product of the transition weights
  (run $(q_0, ., q_1)(q_1, \_, q_2)(q_2, \text{F}, q_3)$ has weight $0.5 \cdot 0.8 \cdot 0.3 = 0.12$)

Weighted Automata

Example

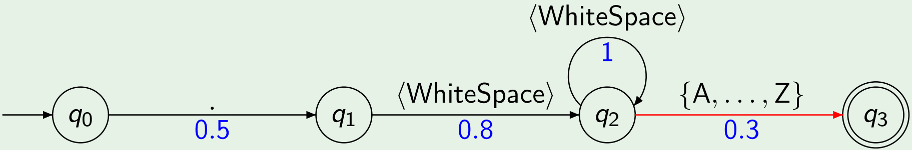


Definition (Semantics)

- Weight of a run: product of the transition weights
  (run $(q_0, ., q_1)(q_1, \_, q_2)(q_2, \mathrm{F}, q_3)$ has weight $0.5 \cdot 0.8 \cdot 0.3 = 0.12$)

Weighted Automata

Example

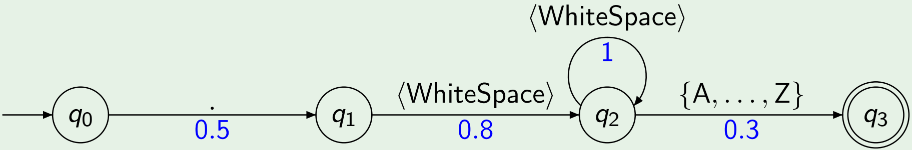


Definition (Semantics)

- Weight of a run: product of the transition weights
  (run $(q_0, ., q_1)(q_1, \_, q_2)(q_2, \mathrm{F}, q_3)$ has weight $0.5 \cdot 0.8 \cdot 0.3 = 0.12$)

Weighted Automata

Example



Definition (Semantics)

- Weight of a run: product of the transition weights
  (run $(q_0, ., q_1)(q_1, \_, q_2)(q_2, \mathrm{F}, q_3)$ has weight $0.5 \cdot 0.8 \cdot 0.3 = 0.12$)
- Weight of an input: sum of the weights of all successful runs
  (input "._F" has weight 0.12)

Part-of-Speech Tagging

Part-of-Speech Tagging

Motivation

- Indexing (GOOGLE): Which (meaning-carrying) tokens to index in

  Alla sätt är bra utom de dåliga.

Part-of-Speech Tagging

Motivation

- Indexing (GOOGLE): Which (meaning-carrying) tokens to index in

  Alla sätt är bra utom de dåliga.

- Usually nouns, adjectives, and verbs are meaning-carrying
- Part-of-speech tagging
  = task of assigning a grammatical function to each token
  = task of determining the word class for each token (of a sentence)

Part-of-Speech Tagging

Motivation

- Indexing (GOOGLE): Which (meaning-carrying) tokens to index in

  Alla sätt är bra utom de dåliga.

- Usually nouns, adjectives, and verbs are meaning-carrying
- Part-of-speech tagging
  = task of assigning a grammatical function to each token
  = task of determining the word class for each token (of a sentence)

Example

Alla	sätt	är	bra	utom	de	dåliga
det.	noun	verb	adj.	subord. conj.	det.	adj.

Part-of-Speech Tagging

Motivation

- Indexing (GOOGLE): Which (meaning-carrying) tokens to index in

  Alla sätt är bra utom de dåliga.
- Usually nouns, adjectives, and verbs are meaning-carrying
- Part-of-speech tagging
  = task of assigning a grammatical function to each token
  = task of determining the word class for each token (of a sentence)

Example

Alla	sätt	är	bra	utom	de	dåliga
det.	noun	verb	adj.	subord. conj.	det.	adj.

Part-of-Speech Tagging

Tags (from the PENN tree bank — English)

- DT = determiner
- NN = noun (singular or mass)
- JJ = adjective
- MD = modal
- VB = verb (base form)
- VBD = verb (past tense)
- VBG = verb (gerund or present participle)

Part-of-Speech Tagging

Tags (from the PENN tree bank — English)

- DT = determiner
- NN = noun (singular or mass)
- JJ = adjective
- MD = modal
- VB = verb (base form)
- VBD = verb (past tense)
- VBG = verb (gerund or present participle)

Example (Tagging exercise)

show

Part-of-Speech Tagging

Tags (from the PENN tree bank — English)

- DT = determiner
- NN = noun (singular or mass)
- JJ = adjective
- MD = modal
- VB = verb (base form)
- VBD = verb (past tense)
- VBG = verb (gerund or present participle)

Example (Tagging exercise)

show VB

Part-of-Speech Tagging

Tags (from the PENN tree bank — English)

- DT = determiner
- NN = noun (singular or mass)
- JJ = adjective
- MD = modal
- VB = verb (base form)
- VBD = verb (past tense)
- VBG = verb (gerund or present participle)

Example (Tagging exercise)

the show

Part-of-Speech Tagging

Tags (from the PENN tree bank — English)

- DT = determiner
- NN = noun (singular or mass)
- JJ = adjective
- MD = modal
- VB = verb (base form)
- VBD = verb (past tense)
- VBG = verb (gerund or present participle)

Example (Tagging exercise)

the show DT NN

Part-of-Speech Tagging

History

- 1960s
  - manually tagged BROWN corpus (1,000,000 words)
  - tag lists with frequency for each token
    e.g., {VB, MD, NN} for can
  - excluding ling.-implausible sequences (e.g. DT VB)
  - "most common tag" yields 90% accuracy

[CHARNIAK, 97]

Part-of-Speech Tagging

History

- 1960s
  - manually tagged BROWN corpus (1,000,000 words)
  - tag lists with frequency for each token
    e.g., {VB, MD, NN} for can
  - excluding ling.-implausible sequences (e.g. DT VB)
  - "most common tag" yields 90% accuracy

[CHARNIAK, 97]

- 1980s
  - hidden MARKOV models (HMM)
  - dynamic programming and VITERBI algorithms

(wA algorithms)

Part-of-Speech Tagging

History

- 1960s
  - manually tagged BROWN corpus (1,000,000 words)
  - tag lists with frequency for each token
    e.g., {VB, MD, NN} for can
  - excluding ling.-implausible sequences (e.g. DT VB)
  - "most common tag" yields 90% accuracy

[CHARNIAK, 97]

- 1980s
  - hidden MARKOV models (HMM)
  - dynamic programming and VITERBI algorithms

(wA algorithms)

- 2000s
  - British national corpus (100,000,000 words)
  - parsers are better taggers (wTA algorithms)

Statistical approach

Given a sequence $w = w_1 \cdots w_k$ of tokens,
determine the most likely sequence $t_1 \cdots t_k$ of part-of-speech tags

(t_i is the tag of w_i)

$$
\begin{aligned}
(\hat{t}_1, \ldots, \hat{t}_k) &= \arg\max_{(t_1,\ldots,t_k)} \; p(t_1, \ldots, t_k \mid w) \\
&= \arg\max_{(t_1,\ldots,t_k)} \; \frac{p(t_1, \ldots, t_k, w)}{p(w)} \\
&= \arg\max_{(t_1,\ldots,t_k)} \; p(t_1, \ldots, t_k, w_1, \ldots, w_k) \\
&= \arg\max_{(t_1,\ldots,t_k)} \; p(t_1, w_1) \cdot \prod_{i=2}^{k} p(t_i, w_i \mid t_1, \ldots, t_{i-1}, w_1, \ldots, w_{i-1})
\end{aligned}
$$

Markov Model

Modelling as stochastic process

- introduce event $E_i = w_i \cap t_i = (w_i, t_i)$

$$p(t_1, w_1) \cdot \prod_{i=2}^{k} p(t_i, w_i \mid t_1, \ldots, t_{i-1}, w_1, \ldots, w_{i-1})$$

$$= p(E_1) \cdot \prod_{i=2}^{k} p(E_i \mid E_1, \ldots, E_{i-1})$$

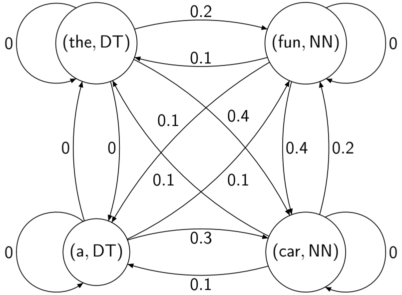
- assume MARKOV property

$$p(E_i \mid E_1, \ldots, E_{i-1}) = p(E_i \mid E_{i-1}) = p(E_2 \mid E_1)$$

Markov Model

Summary

- initial weights $p(E)$ (not indicated below)
- transition weights $p(E \mid E')$



Markov Model

Maximum likelihood estimation (MLE)

Assume that likelihood = relative frequency in corpus

- initial weights $p(E)$
  How often does E start a tagged sentence?
- transition weights $p(E \mid E')$
  How often does E follow E'?

Markov Model

Maximum likelihood estimation (MLE)

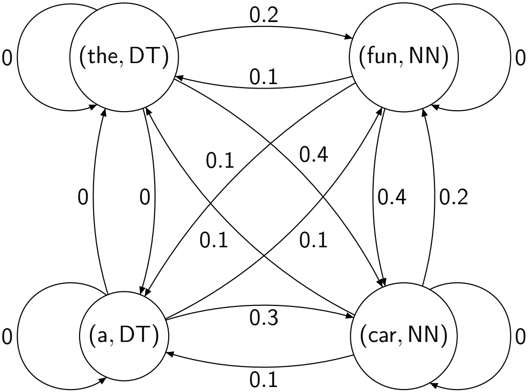
Assume that likelihood = relative frequency in corpus

- initial weights $p(E)$
  How often does E start a tagged sentence?
- transition weights $p(E \mid E')$
  How often does E follow E'?

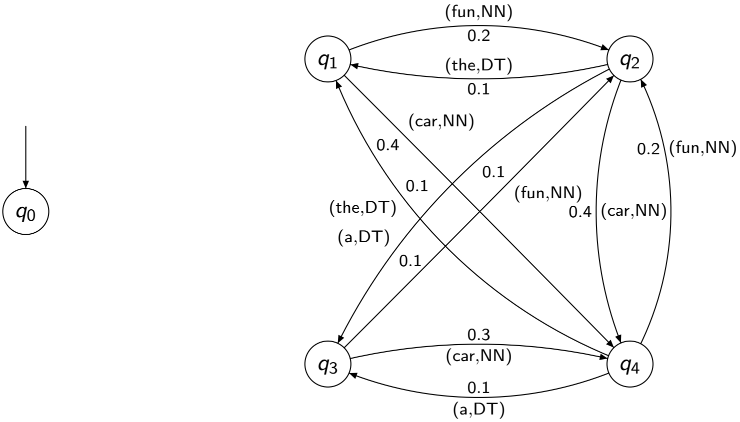
Problems

- Vocabulary: $\approx$ 350,000 English tokens,
  but only 50,000 tokens (14%) in BROWN corpus
- Sparsity: (car, NN) (fun, NN) not attested in corpus, but plausible
  (frequency estimates might be wrong)

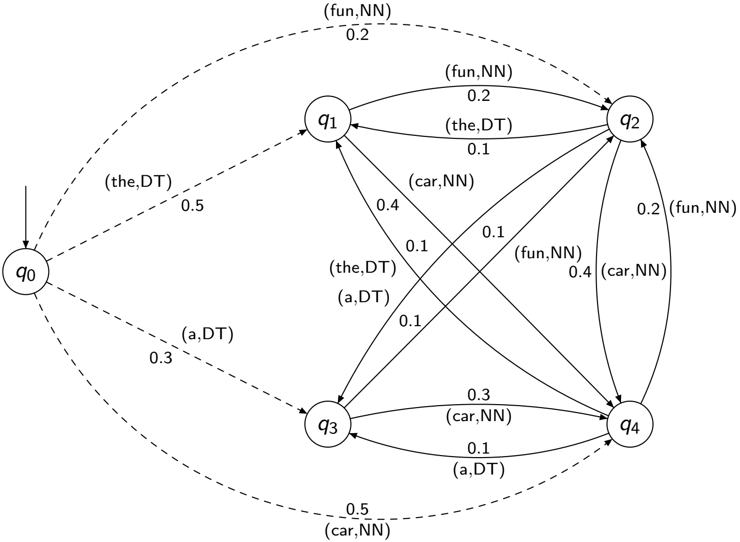
Transformation into Weighted Automaton



Transformation into Weighted Automaton



Transformation into Weighted Automaton



Part-of-Speech Tagging

Typical questions

- Decoding: (or language model evaluation)
  Given model M and sentence w, determine probability $M_1(w)$
  - project labels to first components
  - evaluate w in the obtained wA M_1
  - efficient: initial-algebra semantics (forward algorithm)

Part-of-Speech Tagging

Typical questions

- Decoding: (or language model evaluation)
  Given model M and sentence w, determine probability $M_1(w)$
  - project labels to first components
  - evaluate w in the obtained wA M_1
  - efficient: initial-algebra semantics (forward algorithm)
- Tagging:
  Given model M and sentence w,
  determine the best tag sequence $t_1 \cdots t_k$
  - intersect M with the DFA for w and any tag sequence
  - determine best run in the obtained wA
  - efficient: VITERBI algorithm

Part-of-Speech Tagging

Typical questions

- (Weight) Induction: (or MLE training)
  Given NFA $(Q, \Sigma, I, \Delta, F)$ and sequence $\overline{w}_1, \dots, \overline{w}_k$ of tagged sentences $\overline{w}_i \in \Sigma^*$, determine transition weights $\mathrm{wt} \colon \Delta \to [0, 1]$ such that $\prod_{i=1}^{k} M_{\mathrm{wt}}(\overline{w}_i)$ is maximal with $M_{\mathrm{wt}} = (Q, \Sigma, I, \Delta, F, \mathrm{wt})$
  - no closed solution (in general), but many approximations
  - efficient: hill-climbing methods (EM, simulated annealing, etc.)

Part-of-Speech Tagging

Typical questions

- (Weight) Induction: (or MLE training)
  Given NFA $(Q, \Sigma, I, \Delta, F)$ and sequence $\overline{w}_1, \ldots, \overline{w}_k$ of tagged sentences $\overline{w}_i \in \Sigma^*$, determine transition weights $\mathrm{wt} \colon \Delta \to [0, 1]$ such that $\prod_{i=1}^{k} M_{\mathrm{wt}}(\overline{w}_i)$ is maximal with $M_{\mathrm{wt}} = (Q, \Sigma, I, \Delta, F, \mathrm{wt})$
  - no closed solution (in general), but many approximations
  - efficient: hill-climbing methods (EM, simulated annealing, etc.)
- Learning: (or HMM induction)
  Given NFA $(Q, \Sigma, I, \Delta, F)$ and sequence $w_1, \ldots, w_k$ of untagged sentences w_i, determine transition weights $\mathrm{wt} \colon \Delta \to [0, 1]$ such that $\prod_{i=1}^{k} (M_{\mathrm{wt}})_1(w_i)$ is maximal with $M_{\mathrm{wt}} = (Q, \Sigma, I, \Delta, F, \mathrm{wt})$
  - no exact solution (in general), but many approximations
  - efficient: hill-climbing methods (EM, simulated annealing, etc.)

Part-of-Speech Tagging

Issues

- WA too big (in comparison to training data)
  - cannot reliably estimate that many probabilities $p(E \mid E')$
  - simplify model
    e.g., assume transition probability only depends on tags

$$p\big((w, t) \mid (w', t')\big) = p(t \mid t')$$

Part-of-Speech Tagging

Issues

- WA too big (in comparison to training data)
  - cannot reliably estimate that many probabilities $p(E \mid E')$
  - simplify model
    e.g., assume transition probability only depends on tags

$$p\big((w, t) \mid (w', t')\big) = p(t \mid t')$$

- unknown words
  - no statistics on words that do not occur in corpus
  - allow only assignment of open tags
    (open tag = potentially unbounded number of elements, e.g. NNP)
    (closed tag = fixed finite number of elements, e.g. DT or PRP)
  - use morphological clues (capitalization, affixes, etc.)
  - use context to disambiguate
  - use "global" statistics

Part-of-Speech Tagging

TCS contributions

- efficient evaluation and complexity considerations
  (initial-algebra semantics, best runs, best strings, etc.)
- model simplifications
  (trimming, determinization, minimization, etc.)
- model transformations
  (projection, intersection, RegEx-to-DFA, etc.)
- model induction
  (grammar induction, weight training, etc.)

Parsing

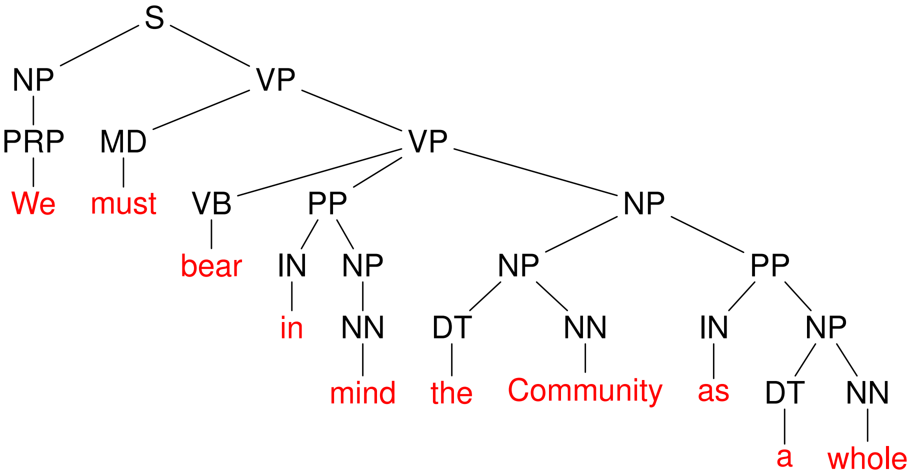
Parsing

Motivation

- (syntactic) parsing
  = determining the syntactic structure of a sentence
- important in several applications:
  - co-reference resolution
    (determining which noun phrases refer to the same object/concept)
  - comprehension
    (determining the meaning)
  - speech repair and sentence-like unit detection in speech
    (speech offers no punctuation; needs to be predicted)

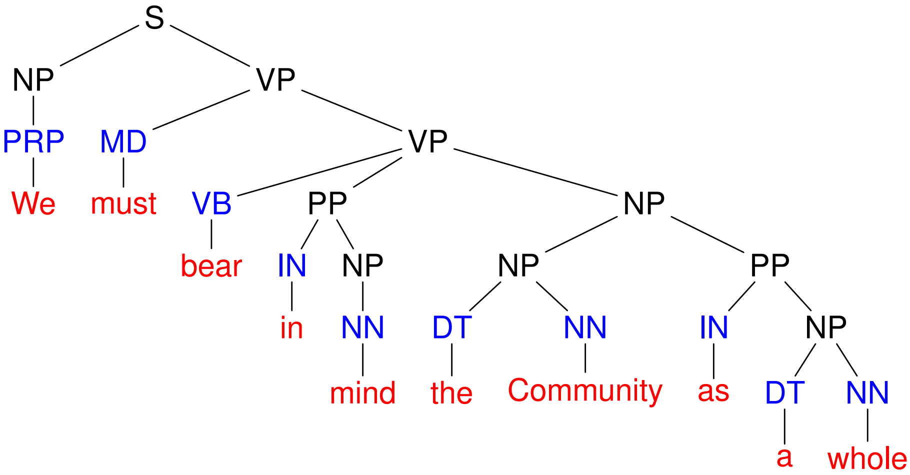
Parsing

We must bear in mind the Community as a whole



Parsing

We must bear in mind the Community as a whole



Trees

Finite sets Σ and W

Definition

Set $T_\Sigma(W)$ of Σ-trees indexed by W is smallest T

- $w \in T$ for all $w \in W$
- $\sigma(t_1, \ldots, t_k) \in T$ for all $k \in \mathbb{N}$, $\sigma \in \Sigma$, and $t_1, \ldots, t_k \in T$

Trees

Finite sets Σ and W

Definition

Set $T_\Sigma(W)$ of Σ-trees indexed by W is smallest T

- $w \in T$ for all $w \in W$
- $\sigma(t_1, \ldots, t_k) \in T$ for all $k \in \mathbb{N}$, $\sigma \in \Sigma$, and $t_1, \ldots, t_k \in T$

Notes

- obvious recursion & induction principle

Parsing

Problem

- assume a hidden $g\colon W^* \to T_\Sigma(W)$ (reference parser)
- given a finite set $T \subseteq T_\Sigma(W)$ generated by g (training set)
- develop a system representing $f\colon W^* \to T_\Sigma(W)$ approximating g (parser)

Parsing

Problem

- assume a hidden $g\colon W^* \to T_\Sigma(W)$ (reference parser)
- given a finite set $T \subseteq T_\Sigma(W)$ (training set)
  generated by g
- develop a system representing $f\colon W^* \to T_\Sigma(W)$ (parser)
  approximating g

Parsing

Problem

- assume a hidden $g\colon W^* \to T_\Sigma(W)$ (reference parser)
- given a finite set $T \subseteq T_\Sigma(W)$ (training set)
  generated by g
- develop a system representing $f\colon W^* \to T_\Sigma(W)$ (parser)
  approximating g

Clarification

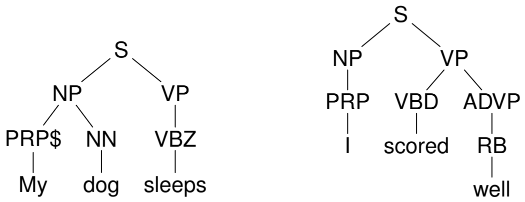
- T generated by $g \iff T = g(L)$ for some finite $L \subseteq W^*$
- for approximation we could use $|\{w \in W^* \mid f(w) = g(w)\}|$

Parsing

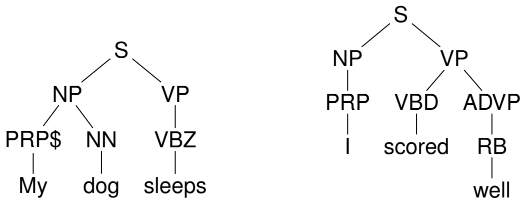
Short history

- before 1990
  - hand-crafted rules based on POS tags (unlexicalized parsing)
  - corrections and selection by human annotators
- 1990s
  - PENN tree bank (1,000,000 words)
  - weighted local tree grammars (weighted CFG) as parsers (often still unlexicalized)
  - WALL STREET JOURNAL tree bank (30,000,000 words)
- since 2000
  - weighted tree automata (weighted CFG with latent variables)
  - lexicalized parsers

Weighted Local Tree Grammars



Weighted Local Tree Grammars



LTG production extraction

simply read off CFG productions:

S ⟶ NP VP	NP ⟶ PRP$ NN
PRP$ ⟶ My	NN ⟶ dog
VP ⟶ VBZ	VBZ ⟶ sleeps
NP ⟶ PRP	PRP ⟶ I
VP ⟶ VBD ADVP	VBD ⟶ scored
ADVP ⟶ RB	RB ⟶ well

Weighted Local Tree Grammars

Observations

- LTG offer unique explanation on tree level
  (rules observable in training data; as for POS tagging)
- but ambiguity on the string level
  (i.e., on unannotated data; as for POS tagging)

→ weighted productions

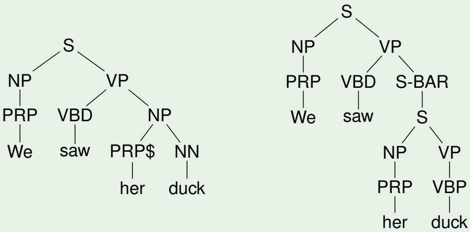
Weighted Local Tree Grammars

Observations

- LTG offer unique explanation on tree level
  (rules observable in training data; as for POS tagging)
- but ambiguity on the string level
  (i.e., on unannotated data; as for POS tagging)

→ weighted productions

Illustration



Weighted Local Tree Grammars

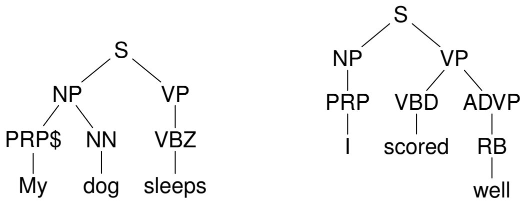
Definition

A weighted local tree grammar (wLTG) is a weighted CFG $G = (N, W, S, P, \mathrm{wt})$

- finite set N (nonterminals)
- finite set W (terminals)
- $S \subseteq N$ (start nonterminals)
- finite set $P \subseteq N \times (N \cup W)^*$ (productions)
- mapping $\mathrm{wt} \colon P \to [0, 1]$ (weight assignment)

It computes the weighted derivation trees of the wCFG

Weighted Local Tree Grammars



wLTG production extraction

simply read of CFG productions and keep counts:

S ⟶ NP VP (2)	NP ⟶ PRP$ NN (1)
PRP$ ⟶ My (1)	NN ⟶ dog (1)
VP ⟶ VBZ (1)	VBZ ⟶ sleeps (1)
NP ⟶ PRP (1)	PRP ⟶ I (1)
VP ⟶ VBD ADVP (1)	VBD ⟶ scored (1)
ADVP ⟶ RB (1)	RB ⟶ well (1)

Weighted Local Tree Grammars

wLTG production extraction

normalize counts: (here by left-hand side)

$$\text{S} \longrightarrow \text{NP VP} \quad (2)$$

$$\text{NP} \longrightarrow \text{PRP\$ NN} \quad (1) \qquad \text{NP} \longrightarrow \text{PRP} \quad (1)$$

$$\text{PRP\$} \longrightarrow \text{My} \quad (1)$$

$$\text{NN} \longrightarrow \text{dog} \quad (1)$$

$$\text{VP} \longrightarrow \text{VBZ} \quad (1) \qquad \text{VP} \longrightarrow \text{VBD ADVP} \quad (1)$$

$$\text{VBZ} \longrightarrow \text{sleeps} \quad (1)$$

$$\text{PRP} \longrightarrow \text{I} \quad (1)$$

$$\text{VBD} \longrightarrow \text{scored} \quad (1)$$

$$\text{ADVP} \longrightarrow \text{RB} \quad (1)$$

$$\text{RB} \longrightarrow \text{well} \quad (1)$$

Weighted Local Tree Grammars

wLTG production extraction

normalize counts: (here by left-hand side)

$$\mathrm{S} \xrightarrow{1} \mathrm{NP\ VP}$$

$$\mathrm{NP} \xrightarrow{0.5} \mathrm{PRP\$\ NN} \qquad \mathrm{NP} \xrightarrow{0.5} \mathrm{PRP}$$

$$\mathrm{PRP\$} \xrightarrow{1} \mathrm{My}$$

$$\mathrm{NN} \xrightarrow{1} \mathrm{dog}$$

$$\mathrm{VP} \xrightarrow{0.5} \mathrm{VBZ} \qquad \mathrm{VP} \xrightarrow{0.5} \mathrm{VBD\ ADVP}$$

$$\mathrm{VBZ} \xrightarrow{1} \mathrm{sleeps}$$

$$\mathrm{PRP} \xrightarrow{1} \mathrm{I}$$

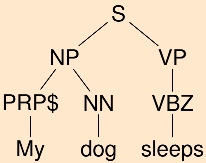
$$\mathrm{VBD} \xrightarrow{1} \mathrm{scored}$$

$$\mathrm{ADVP} \xrightarrow{1} \mathrm{RB}$$

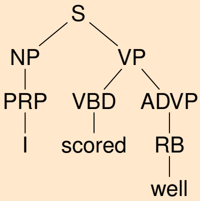
$$\mathrm{RB} \xrightarrow{1} \mathrm{well}$$

Weighted Local Tree Grammars

Weighted parses



weight: 0.25



weight: 0.25

Weighted LTG productions

(only productions with weight $\neq 1$)

$\text{NP} \xrightarrow{0.5} \text{PRP\$ NN}$

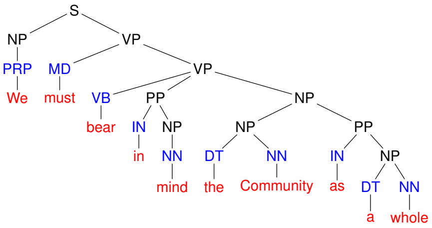
$\text{VP} \xrightarrow{0.5} \text{VBZ}$

$\text{NP} \xrightarrow{0.5} \text{PRP}$

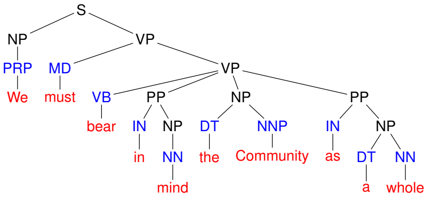
$\text{VP} \xrightarrow{0.5} \text{VBD ADVP}$

Parser Evaluation

BERKELEY parser [Reference]:



CHARNIAK-JOHNSON parser:



Parser Evaluation

Definition (ParseEval measure)

- precision = number of correct constituents
  (heading the same phrase as in reference)
  divided by number of all constituents in parse

Parser Evaluation

Definition (ParseEval measure)

- precision = number of correct constituents
  (heading the same phrase as in reference)
  divided by number of all constituents in parse
- recall = number of correct constituents
  divided by number of all constituents in reference

Parser Evaluation

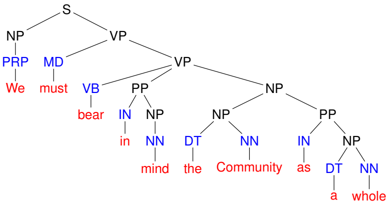
Definition (ParseEval measure)

- precision = number of correct constituents (heading the same phrase as in reference) divided by number of all constituents in parse
- recall = number of correct constituents divided by number of all constituents in reference
- (weighted) harmonic mean

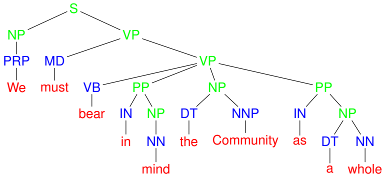
$$F_\alpha = (1 + \alpha^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\alpha^2 \cdot \text{precision} + \text{recall}}$$

Parser Evaluation

Reference



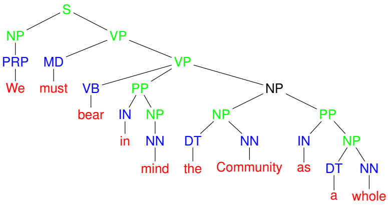
Parser output



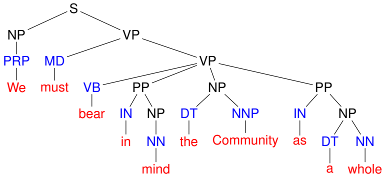
- precision $= \frac{9}{9} = 100\%$

Parser Evaluation

Reference



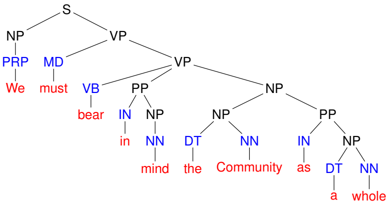
Parser output



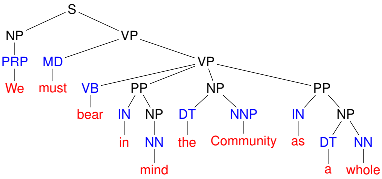
- precision $= \frac{9}{9} = 100\%$
- recall $= \frac{9}{10} = 90\%$

Parser Evaluation

Reference



Parser output



- precision $= \frac{9}{9} = 100\%$
- recall $= \frac{9}{10} = 90\%$
- $F_1 = 2 \cdot \frac{1 \cdot 0.9}{1 + 0.9} = 95\%$

Parser Evaluation

Standardized Setup

- training data: PENN treebank Sections 2–21
  (articles from the WALL STREET JOURNAL)
- development test data: PENN treebank Section 22
- evaluation data: PENN treebank Section 23

Experiment [POST, GILDEA, '09]

grammar model	precision	recall	F_1
wLTG	75.37	70.05	72.61

Parser Evaluation

Standardized Setup

- training data: PENN treebank Sections 2–21
  (articles from the WALL STREET JOURNAL)
- development test data: PENN treebank Section 22
- evaluation data: PENN treebank Section 23

Experiment [POST, GILDEA, '09]

grammar model	precision	recall	F_1
wLTG	75.37	70.05	72.61

These are bad compared to the state-of-the-art!

Parser Evaluation

State-of-the-art models

- context-free grammars with latent variables (CFG_{lv})
  [Collins, '99], [Klein, Manning, '03], [Petrov, Klein, '07]
- tree substitution grammars with latent variables (TSG_{lv})
  [Shindo et al., '12]
- (both as expressive as weighted tree automata)
- other models

Parser Evaluation

State-of-the-art models

- context-free grammars with latent variables (CFG_{lv}) [COLLINS, '99], [KLEIN, MANNING, '03], [PETROV, KLEIN, '07]
- tree substitution grammars with latent variables (TSG_{lv}) [SHINDO et al., '12]
- (both as expressive as weighted tree automata)
- other models

Experiment [SHINDO et al., '12]

grammar model	F_1
wLTG = wCFG	72.6
wTSG [COHN et al., 2010]	84.7
$wCFG_{lv}$ [PETROV, 2010]	91.8
$wTSG_{lv}$ [SHINDO et al., 2012]	92.4

Grammars with Latent Variables

Definition

A grammar with latent variables is (grammar with relabeling)

- a grammar G generating $L(G) \subseteq T_\Sigma(W)$
- a (total) mapping $\rho \colon \Sigma \to \Delta$ functional relabeling

Grammars with Latent Variables

Definition

A grammar with latent variables is (grammar with relabeling)

- a grammar G generating $L(G) \subseteq T_\Sigma(W)$
- a (total) mapping $\rho \colon \Sigma \to \Delta$ functional relabeling

Definition (Semantics)

$$L(G, \rho) = \rho(L(G)) = \{\rho(t) \mid t \in L(G)\}$$

Language class: $\mathrm{REL}(\mathcal{L})$ for language class $\mathcal{L}$

Weighted Tree Automata

Definition

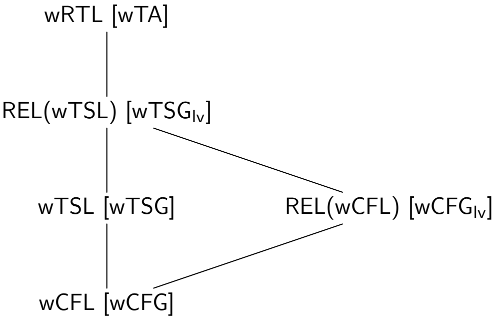
A weighted tree automaton (wTA) is a system $G = (Q, N, W, S, P, \mathrm{wt})$

- finite set Q (states)
- finite set N (nonterminals)
- finite set W (terminals)
- $S \subseteq Q$ (start states)
- finite set $P \subseteq \big(Q \times N \times (Q \cup W)^+\big) \cup \big(Q \times W\big)$ (productions)
- mapping $\mathrm{wt} \colon P \to [0, 1]$ (weight assignment)

production $(q, n, w_1, \ldots, w_k)$ is often written $q \to n(w_1, \ldots, w_k)$

Theorem

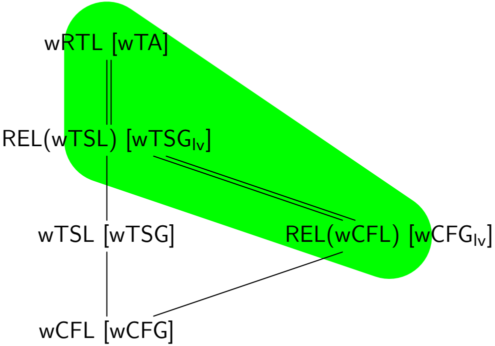
$\text{REL}(\text{wLTL}) = \text{REL}(\text{wTSL}) = \text{wRTL}$



Grammars with Latent Variables

Theorem

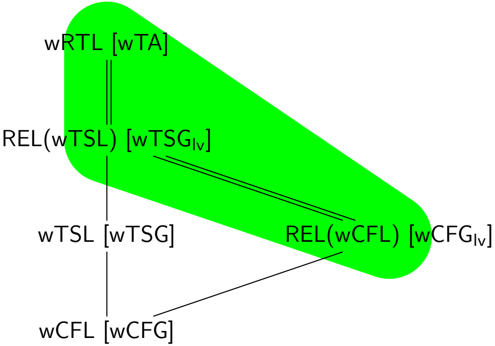
REL(wLTL) = REL(wTSL) = wRTL



Grammars with Latent Variables

Theorem

REL(wLTL) = REL(wTSL) = wRTL



here: latent variables ≈ finite-state

Parsing

Typical questions

- Decoding: (or language model evaluation)
  Given model M and sentence w, determine probability $M(w)$
  - intersect M with the DTA for w and any parse
  - evaluate w in the obtained WTA
  - efficient: initial-algebra semantics (forward algorithm)

Parsing

Typical questions

- **Decoding:** (or language model evaluation)
  Given model M and sentence w, determine probability $M(w)$
  - intersect M with the DTA for w and any parse
  - evaluate w in the obtained WTA
  - efficient: initial-algebra semantics (forward algorithm)
- **Parsing:**
  Given model M and sentence w,
  determine the best parse t for w
  - intersect M with the DTA for w and any parse
  - determine best tree in the obtained WTA
  - efficient: none (NP-hard even for wLTG)

Parsing

Statistical parsing approach

Given wLTG *M* and sentence *w*, return highest-scoring parse for *w*

Parsing

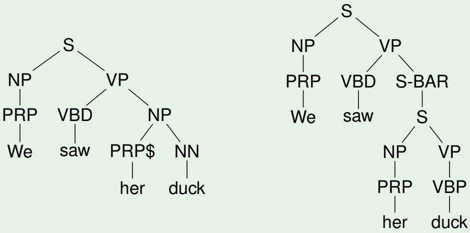
Statistical parsing approach

Given wLTG *M* and sentence *w*, return highest-scoring parse for *w*

Consequence

The first parse should be prefered

(“duck” more frequently a noun, etc.)



Parsing

TCS contributions

- efficient evaluation and complexity considerations
  (initial-algebra semantics, best runs, best trees, etc.)
- model simplifications
  (trimming, determinization, minimization, etc.)
- model transformations
  (intersection, normalization, lexicalization, etc.)
- model induction
  (grammar induction, weight training, spectral learning, etc.)

Parsing

NLP contribution to TCS

- good source of (relevant) problems
- good source for practical techniques
  (e.g., fine-to-coarse decoding)
- good source of (relevant) large wTA

language	**states**	**non-lexical productions**
English	1,132	1,842,218
Chinese	994	1,109,500
German	981	616,776

Machine Translation

Machine Translation

Applications

- Technical manuals

Example (An mp3 player)

The synchronous manifestation of lyrics is a procedure for can broadcasting the music, waiting the mp3 file at the same time showing the lyrics.

Machine Translation

Applications

- Technical manuals

Example (An mp3 player)

The synchronous manifestation of lyrics is a procedure for can broadcasting the music, waiting the mp3 file at the same time showing the lyrics. With the this kind method that the equipments that synchronous function of support up broadcast to make use of document create setup, you can pass the LCD window way the check at the document contents that broadcast.

Machine Translation

Applications

- Technical manuals

Example (An mp3 player)

The synchronous manifestation of lyrics is a procedure for can broadcasting the music, waiting the mp3 file at the same time showing the lyrics. With the this kind method that the equipments that synchronous function of support up broadcast to make use of document create setup, you can pass the LCD window way the check at the document contents that broadcast. That procedure returns offerings to have to modify, and delete, and stick top , keep etc. edit function.

Machine Translation

Applications

- Technical manuals
- US military

Example (Speech-to-text [JONES et al., '09])

E: Okay, what is your name?
A: Abdul.

E: And your last name?
A: Al Farran.

Machine Translation

Applications

- Technical manuals
- US military

Example (Speech-to-text [JONES et al., '09])

E:	Okay, what is your name?	*E:*	Okay, what's your name?
A:	Abdul.	*A:*	milk a mechanic and I am here I mean yes
E:	And your last name?		
A:	Al Farran.		

Machine Translation

Applications

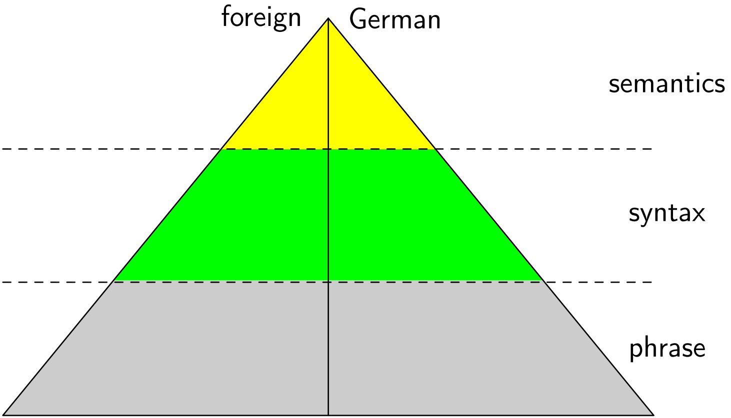
- Technical manuals
- US military

Example (Speech-to-text [JONES et al., '09])

E:	Okay, what is your name?	*E:*	Okay, what's your name?
A:	Abdul.	*A:*	milk a mechanic and I am here I mean yes
E:	And your last name?	*E:*	What is your last name?
A:	Al Farran.	*A:*	every two weeks my son's name is ismail

Machine Translation

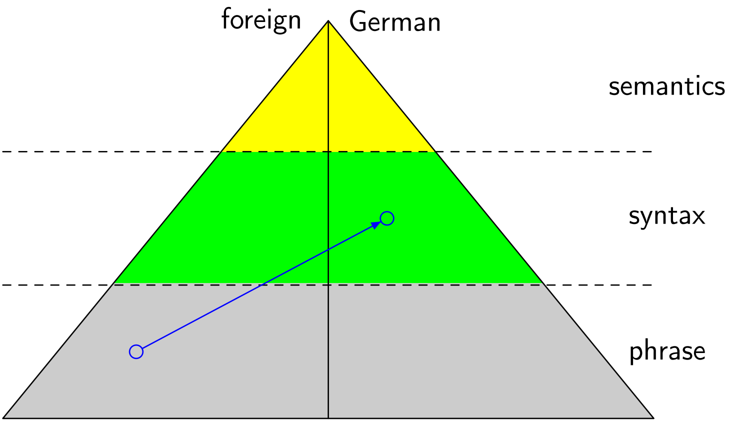
VAUQUOIS triangle:



Translation model:

Machine Translation

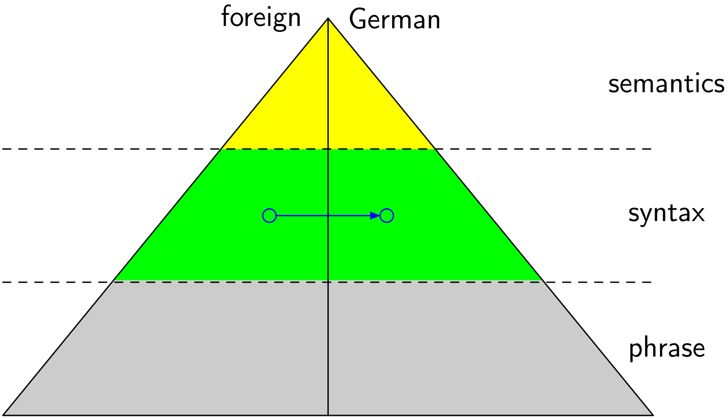
VAUQUOIS triangle:



Translation model: string-to-tree

Machine Translation

VAUQUOIS triangle:



Translation model: tree-to-tree

Machine Translation

Training data

- parallel corpus
- word alignments
- parse trees for the target sentences

Machine Translation

Training data

- parallel corpus
- word alignments
- parse trees for the target sentences

Parallel Corpus

linguistic resource containing example translations

(sentence level)

Machine Translation

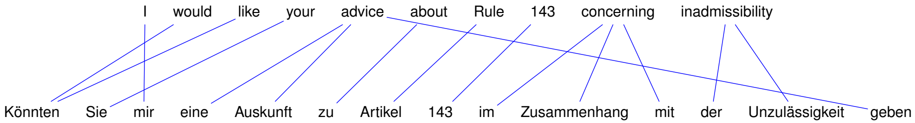
parallel corpus, word alignments, parse tree

I would like your advice about Rule 143 concerning inadmissibility

Könnten Sie mir eine Auskunft zu Artikel 143 im Zusammenhang mit der Unzulässigkeit geben

Machine Translation

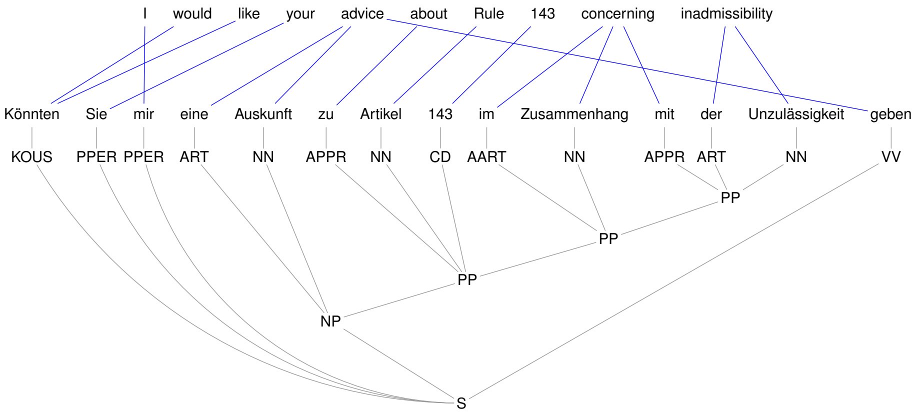
parallel corpus, word alignments, parse tree



via GIZA++ [OCH, NEY, '03]

Machine Translation

parallel corpus, word alignments, parse tree



via BERKELEY parser [PETROV et al., '06]

Extended Tree Transducer

Extended top-down tree transducer (STSG)

- variant of [M., GRAEHL, HOPKINS, KNIGHT, '09]
- rules of the form NT $\to (r, r_1)$ for nonterminal NT
  - right-hand side r of context-free grammar rule
  - right-hand side r_1 of regular tree grammar rule

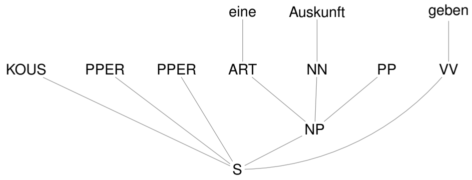
Extended Tree Transducer

Extended top-down tree transducer (STSG)

- variant of [M., GRAEHL, HOPKINS, KNIGHT, '09]
- rules of the form NT $\to (r, r_1)$ for nonterminal NT
  - right-hand side r of context-free grammar rule
  - right-hand side r_1 of regular tree grammar rule

PPER KOUS PPER advice PP

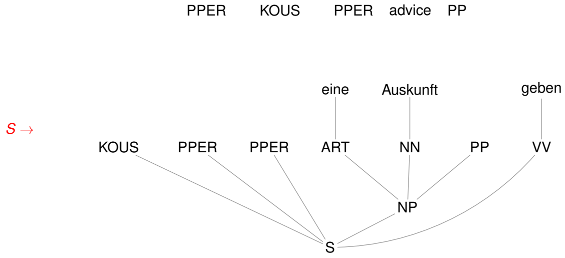
$S \to$



Extended Tree Transducer

Extended top-down tree transducer (STSG)

- variant of [M., GRAEHL, HOPKINS, KNIGHT, '09]
- rules of the form NT $\rightarrow (r, r_1)$ for nonterminal NT
  - right-hand side r of context-free grammar rule
  - right-hand side r_1 of regular tree grammar rule

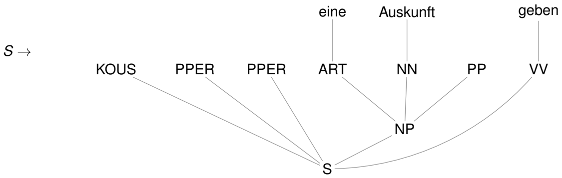


Extended Tree Transducer

Extended top-down tree transducer (STSG)

- variant of [M., GRAEHL, HOPKINS, KNIGHT, '09]
- rules of the form NT $\rightarrow (r, r_1)$ for nonterminal NT
  - right-hand side r of context-free grammar rule
  - right-hand side r_1 of regular tree grammar rule

PPER KOUS PPER advice PP



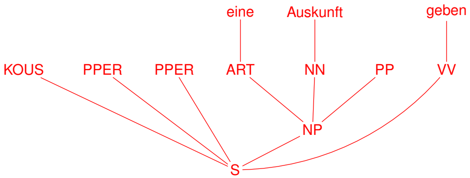
Extended Tree Transducer

Extended top-down tree transducer (STSG)

- variant of [M., GRAEHL, HOPKINS, KNIGHT, '09]
- rules of the form NT $\to (r, r_1)$ for nonterminal NT
  - right-hand side r of context-free grammar rule
  - right-hand side r_1 of regular tree grammar rule

PPER KOUS PPER advice PP

$S \to$

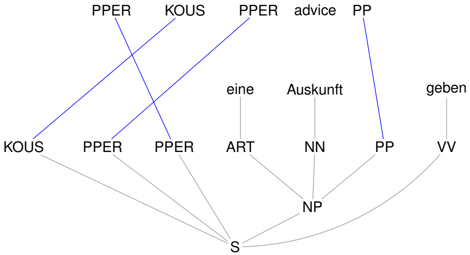


Extended Tree Transducer

Extended top-down tree transducer (STSG)

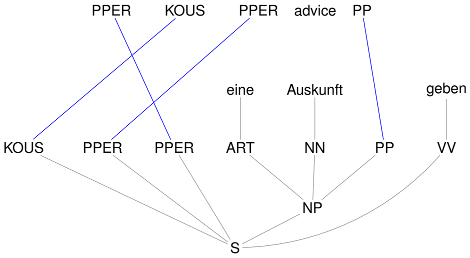
- variant of [M., GRAEHL, HOPKINS, KNIGHT, '09]
- rules of the form NT $\rightarrow (r, r_1)$ for nonterminal NT
  - right-hand side r of context-free grammar rule
  - right-hand side r_1 of regular tree grammar rule
- (bijective) synchronization of nonterminals

$S \rightarrow$



Extended Tree Transducer

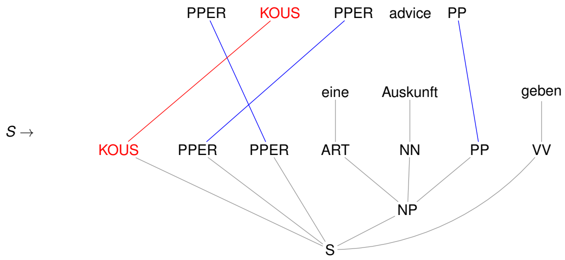
$S \rightarrow$



Rule application

1. Selection of synchronous nonterminals

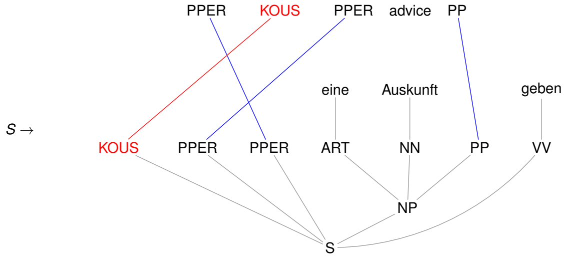
Extended Tree Transducer



Rule application

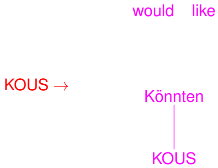
1. Selection of synchronous nonterminals

Extended Tree Transducer

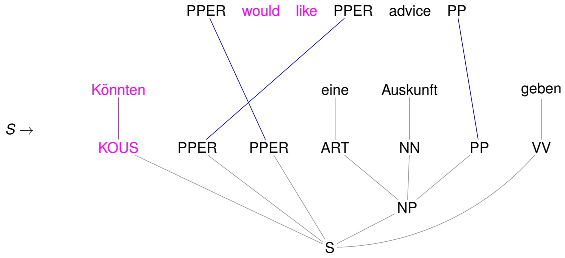


Rule application

1. Selection of synchronous nonterminals
2. Selection of suitable rule

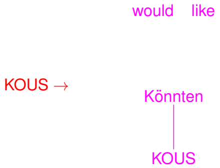


Extended Tree Transducer

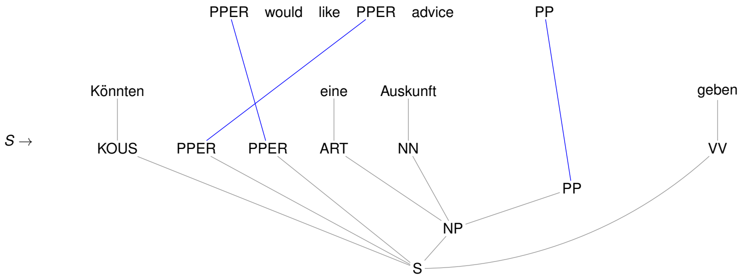


Rule application

1. Selection of synchronous nonterminals
2. Selection of suitable rule
3. Replacement on both sides



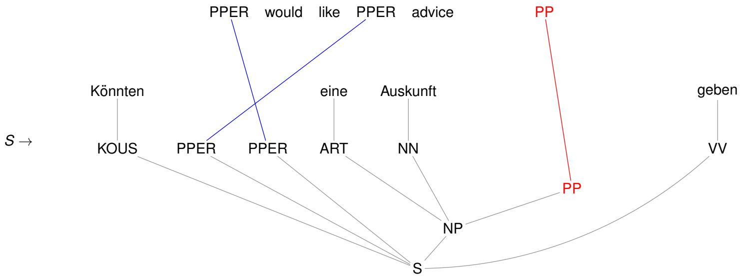
Extended Tree Transducer



Rule application

1. synchronous nonterminals

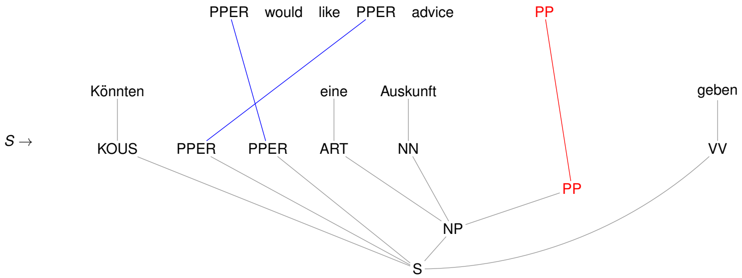
Extended Tree Transducer



Rule application

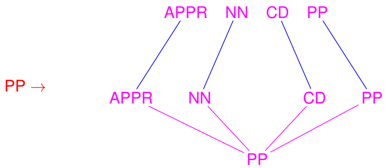
1. synchronous nonterminals

Extended Tree Transducer

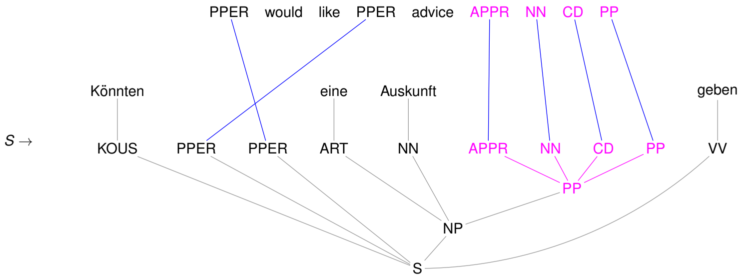


Rule application

1. synchronous nonterminals
2. suitable rule

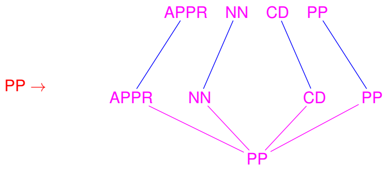


Extended Tree Transducer



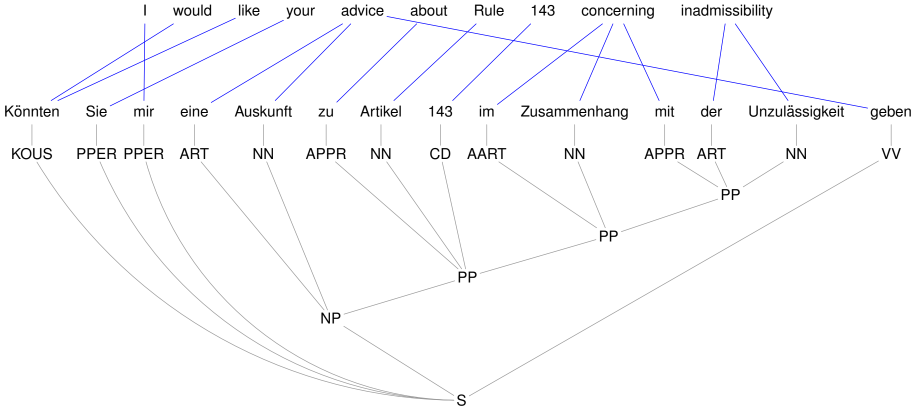
Rule application

1. synchronous nonterminals
2. suitable rule
3. replacement



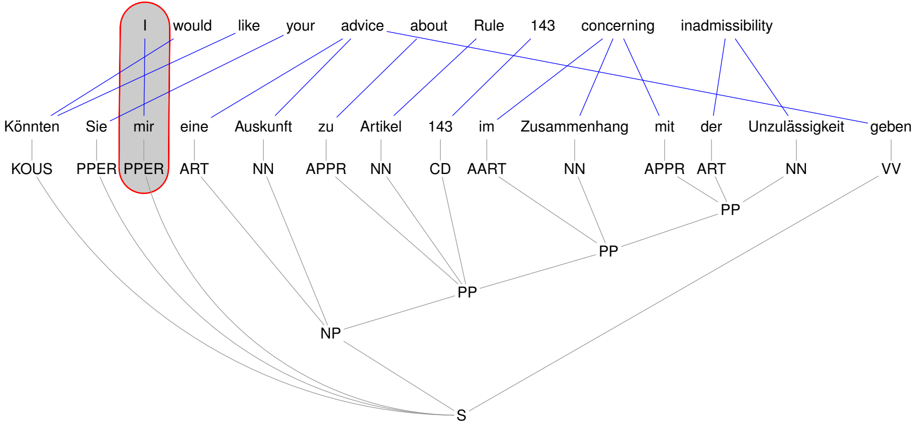
Rule extraction

following [GALLEY, HOPKINS, KNIGHT, MARCU, '04]



Rule extraction

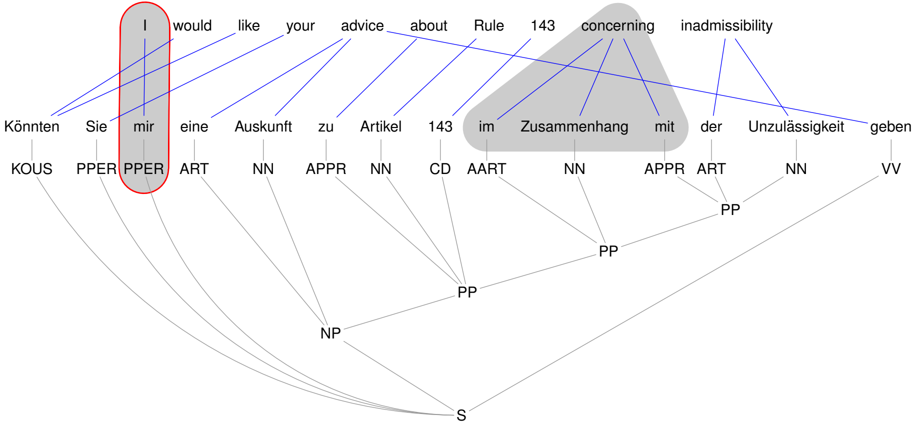
following [Galley, Hopkins, Knight, Marcu, '04]



extractable rules marked in red

Rule extraction

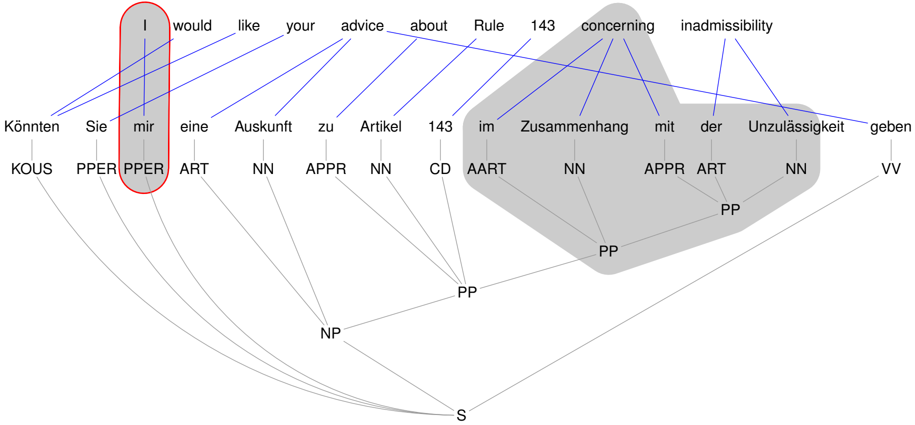
following [Galley, Hopkins, Knight, Marcu, ’04]



extractable rules marked in red

Rule extraction

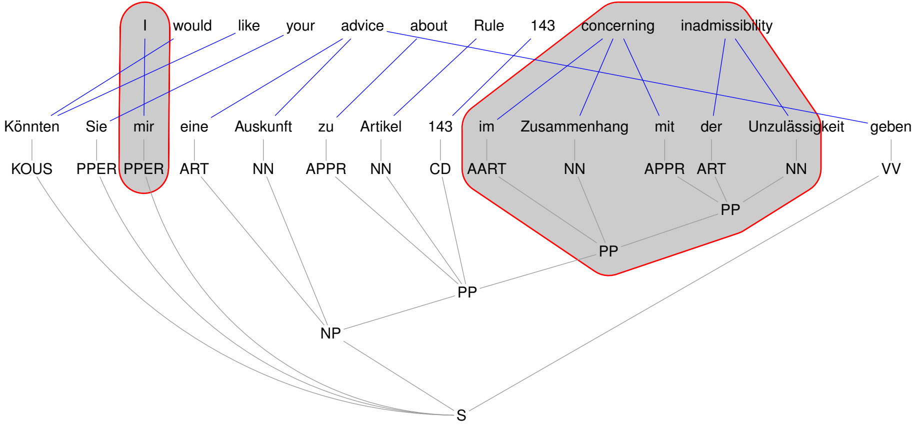
following [GALLEY, HOPKINS, KNIGHT, MARCU, '04]



extractable rules marked in red

Rule extraction

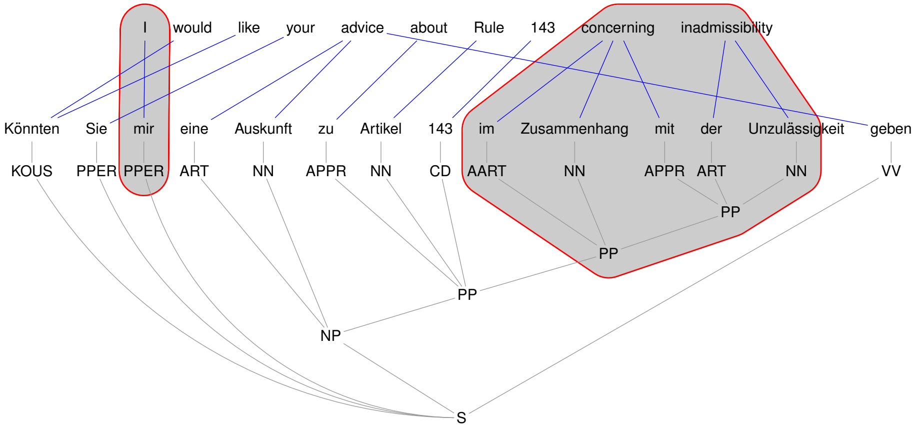
following [GALLEY, HOPKINS, KNIGHT, MARCU, '04]



extractable rules marked in red

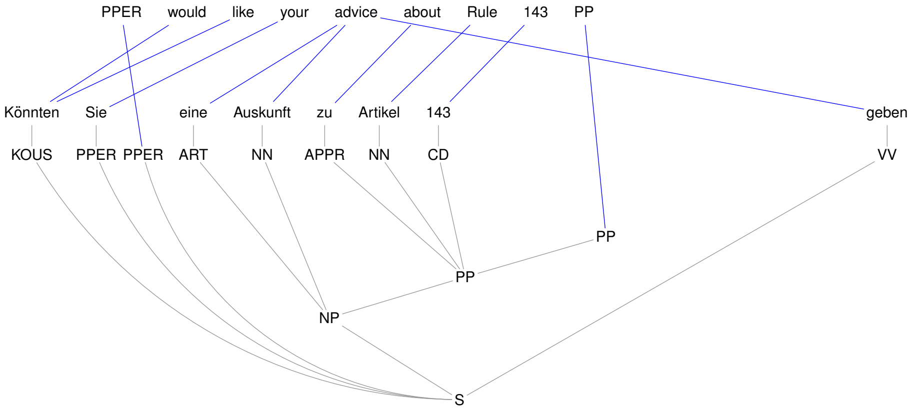
Rule extraction

Removal of extractable rule:



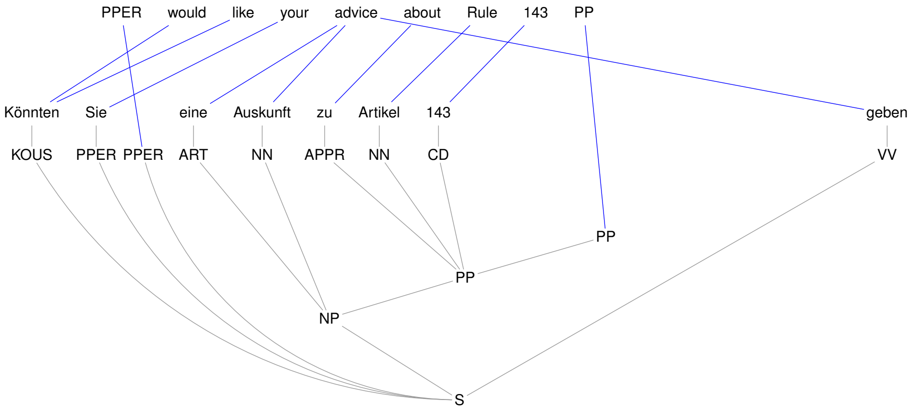
Rule extraction

Removal of extractable rule:



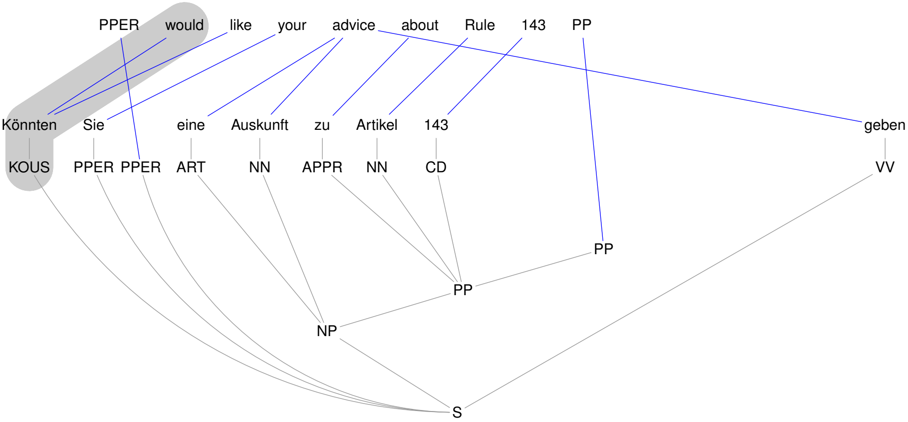
Rule extraction

Repeated rule extraction:



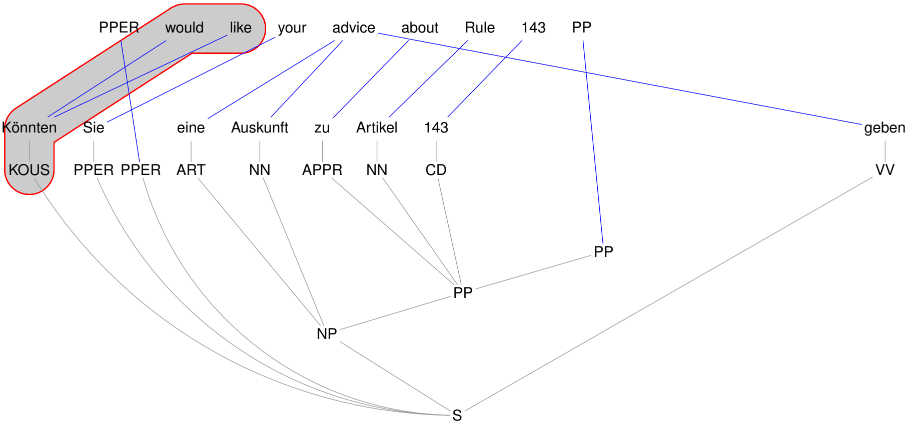
Rule extraction

Repeated rule extraction:



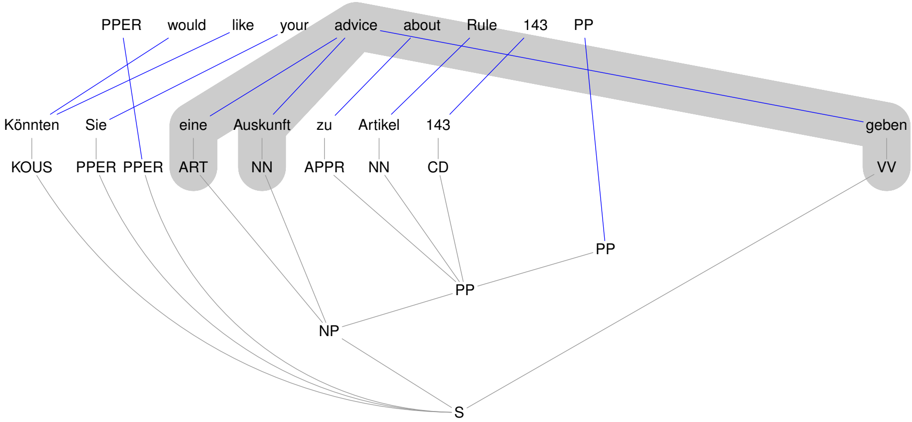
Rule extraction

Repeated rule extraction:



extractable rules marked in red

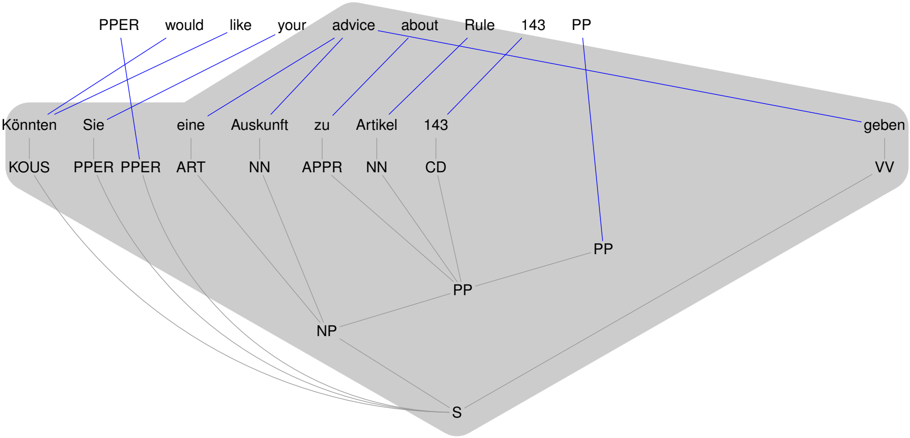
Repeated rule extraction:



extractable rules marked in red

Rule extraction

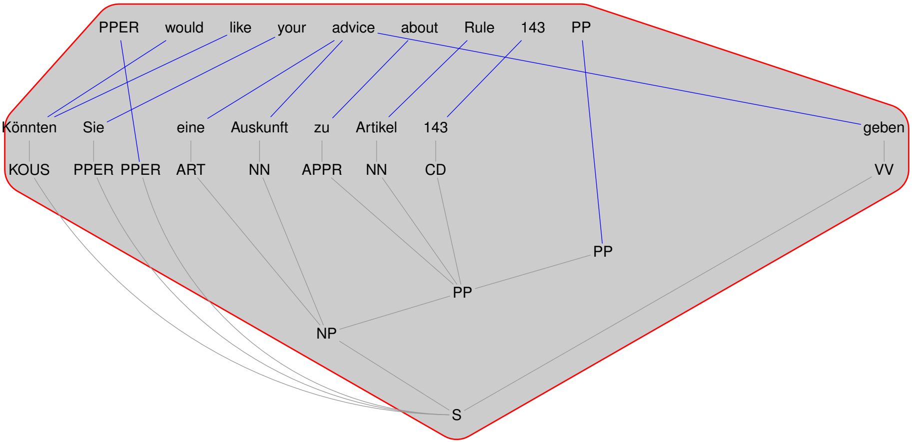
Repeated rule extraction:



extractable rules marked in red

Rule extraction

Repeated rule extraction:



extractable rules marked in red

Extended Tree Transducer

Advantages

- very simple
- implemented in MOSES [KOEHN et al., '07]
- "context-free"

Extended Tree Transducer

Advantages

- very simple
- implemented in MOSES [KOEHN et al., '07]
- “context-free”

Disadvantages

- problems with discontinuities
- composition and binarization not possible [M. et al., '09] and [ZHANG et al., '06]
- “context-free”

Extended Tree Transducer

Remarks

- synchronization breaks almost all existing constructions
  (e.g., the normalization construction)

→ the basic grammar model very important

Extended Tree Transducer

Remarks

- synchronization breaks almost all existing constructions
  (e.g., the normalization construction)

→ the basic grammar model very important

- tree-to-tree models use trees on both sides

Extended Tree Transducer

Major (tree-to-tree) models

1. linear top-down tree transducer (with look-ahead)
   - input-side: tree automaton
   - output-side: regular tree grammar
   - synchronization: mapping output NT to input NT

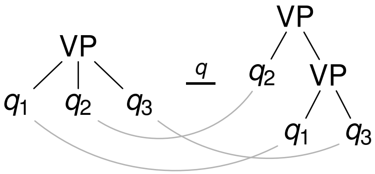
Extended Tree Transducer

Major (tree-to-tree) models

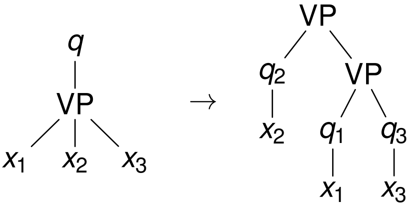
1. linear top-down tree transducer (with look-ahead)
   - input-side: tree automaton
   - output-side: regular tree grammar
   - synchronization: mapping output NT to input NT
2. linear extended top-down tree transducer (w. look-ahead)
   - input-side: regular tree grammar
   - output-side: regular tree grammar
   - synchronization: mapping output NT to input NT

Extended Tree Transducer

Synchronous grammar rule:



"Classical" top-down tree transducer rule:



Extended Tree Transducer

Syntactic restrictions

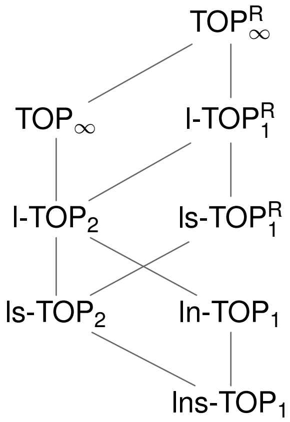
- nondeleting if synchronization bijective (in all rules)
- strict if r_1 not a nonterminal (for all rules $q \to (r, r_1)$)
- ε-free if r not a nonterminal (for all rules $q \to (r, r_1)$)

Composition (COMP)

executing transformations $\tau \subseteq T_\Sigma \times T_\Delta$ and $\tau' \subseteq T_\Delta \times T_\Gamma$ one after the other:

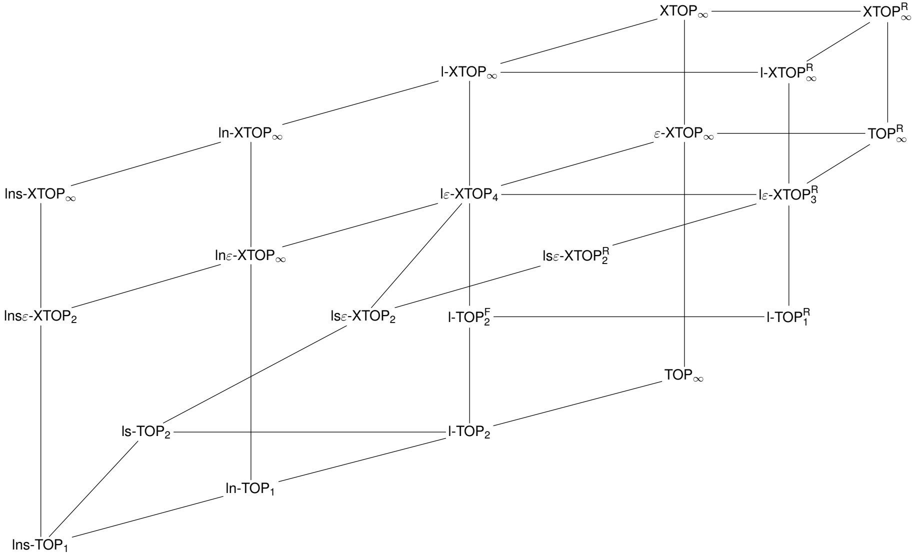
$$\tau \,;\, \tau' = \{(s, u) \mid \exists t \in T_\Delta \colon (s, t) \in \tau, (t, u) \in \tau'\}$$

Top-down Tree Transducer



composition closure indicated in subscript

Extended Tree Transducer



composition closure indicated in subscript

Machine Translation

TCS contributions

- efficient evaluation and complexity considerations
  (exact decoding, best runs, best translations, etc.)
- evaluation of expressive power
  (which linguistic phenomena can be captured?
  relationship to other models)
- model transformations
  (intersection, language model integration,
  parse forest decoding, etc.)
- very little on model induction so far
  (mostly local models so far; power of finite-state not yet explored)

Evaluation

Task	System	BLEU
English → German	STSG	15.22
	MBOT	15.90
	phrase-based	16.73
	hierarchical	16.95
	GHKM	17.10
English → Arabic	STSG	48.32
	MBOT	49.10
	phrase-based	50.27
	hierarchical	51.71
	GHKM	46.66
English → Chinese	STSG	17.69
	MBOT	18.35
	phrase-based	18.09
	hierarchical	18.49
	GHKM	18.12

from [SEEMANN et al., '15]

Selected Literature

- ENGELFRIET: *Bottom-up and Top-down Tree Transformations — A Comparison.* Math. Systems Theory 9 **'75**
- KLEIN, MANNING: *Accurate Unlexicalized Parsing* Proc. ACL **'03**
- KOEHN: *Statistical Machine Translation* Cambridge University Press **'10**
- MANNING, SCHÜTZE: *Foundations of Statistical Natural Language Processing.* MIT Press **'99**
- PETROV, BARRETT, THIBAUX AND KLEIN: *Learning Accurate, Compact, and Interpretable Tree Annotation.* Proc. ACL **'06**
- SHINDO, MIYAO, FUJINO AND NAGATA: *Bayesian Symbol-Refined Tree Substitution Grammars for Syntactic Parsing.* Proc. ACL **'12**