Learning Probabilistic Lexicalized Grammars for Natural Language Processing

Rebecca Hwa
Harvard University
rebecca@eecs.harvard.edu

Grammar Induction

- Learning a tractable representation of linguistic structures for natural language processing applications
  - How should we represent the grammar for processing English?
  - How can the grammar be learned automatically and efficiently?

Our Approach

- Grammar representation
  - Use probabilistic and lexicalized grammar formalisms
- Learning algorithm
  - Induce grammars automatically from large corpora (Expectation-Maximization)
- Efficient training process
  - Reduce human effort in annotating training data

Outline of the Talk

- Overview
  - Tasks that need grammars
  - Properties of a good representation
  - Probabilistic grammar formalisms
    - Probabilistic Lexicalized Tree Insertion Grammars (PLTIGs)
- Grammar induction algorithm for PLTIGs
- Efficient training techniques
- Conclusion and future directions

Grammars

- Syntactic description of the language
  - List of allowable sentences
  - Finite-state network
  - Context-free grammars
  - Tree-rewriting grammars
- Tasks requiring grammars
  - Error Correction
    - speech recognition, context-dependent spelling correction
  - Parsing
    - information extraction, machine translation

Challenges in Constructing Grammars

- Disambiguation
  - Lexical disambiguation
  - Structural disambiguation
    - Prepositional Phrase Attachment
- Complexity
  - Exceptions to the rules
  - Many lexical dependencies
- Manual Grammar Construction
  - Limited coverage
  - Difficult to maintain

Meeting these Challenges: Probabilistic Grammars

Grammar rules parameterized with probabilities

Compute the likelihood of sentences in the language

- Disambiguation?
  - Resolve local ambiguities with global likelihood
- Complexity?
  - Choose the right formalism (Tree Insertion Grammars)
- Manual Grammar Construction?
  - Automatic induction from large corpora

Error Correction with Probabilistic Grammars

- Language Modeling
  - How likely is sentence O to appear in the language (as modeled by grammar G)?
- Lexical disambiguation:

$$\arg\max_{O_i} \Pr(O_i \mid G)$$

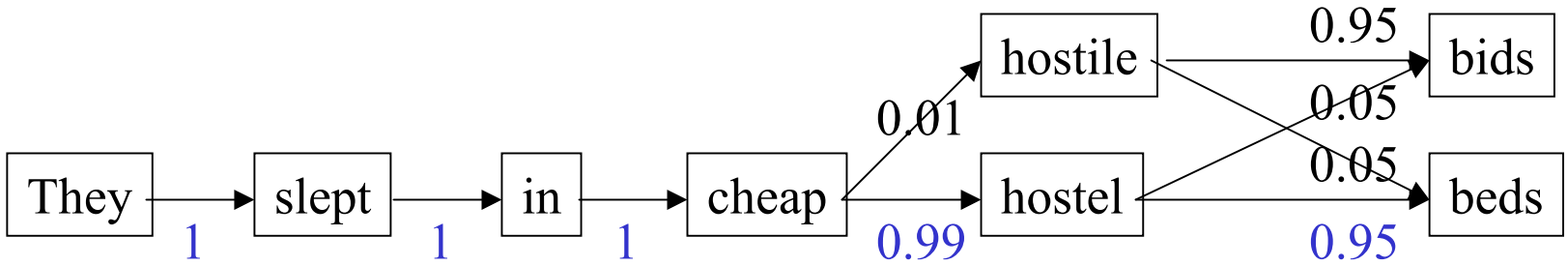
Pr("They slept in cheap hostel beds" | G) >

Pr("They slept in cheap hostile bids." | G)

N-Grams

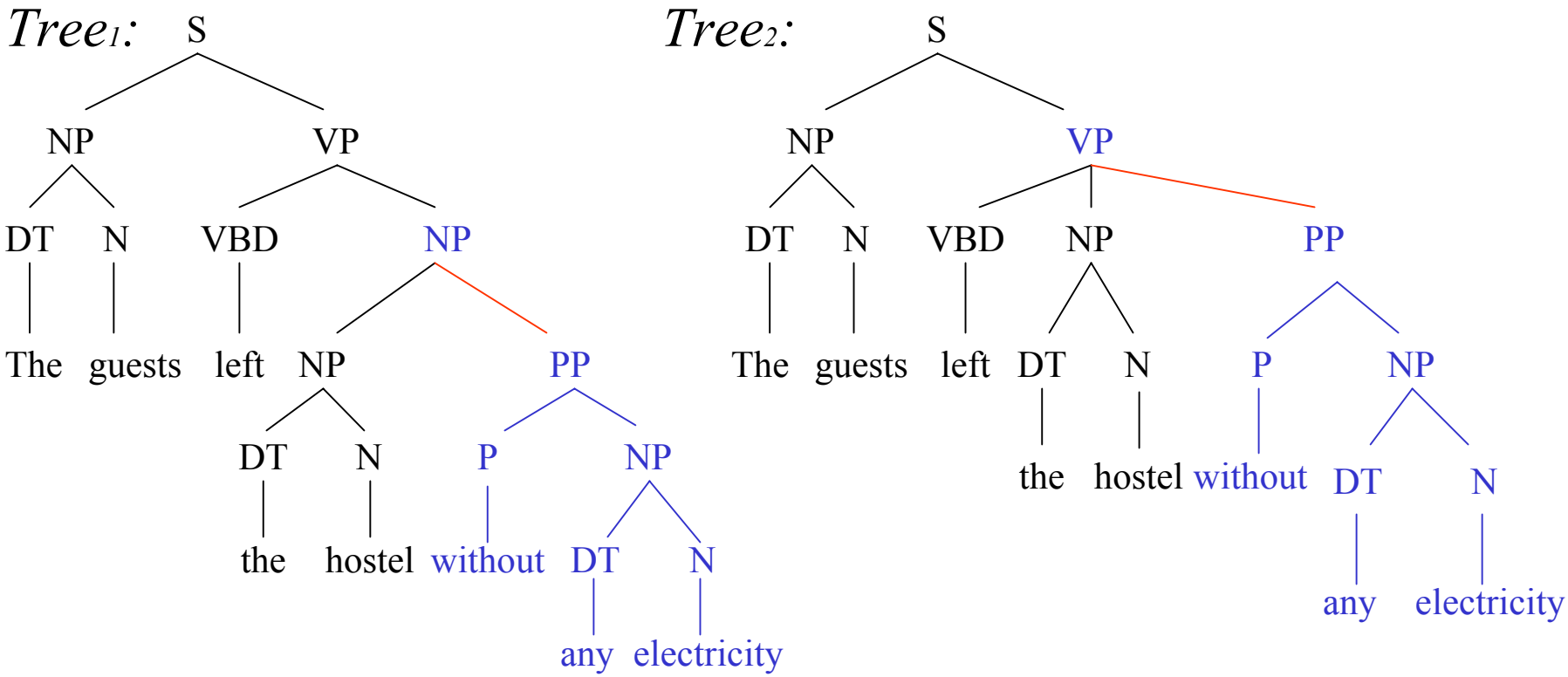
Example of Bi-Gram:

Pr(They slept in cheap hostel beds) = 0.99*0.95 = 0.94



- Finite state network, each state is a history of N-1 words.
- No hierarchical structures
- Models lexical relationships between neighboring words
- Training data sparsity problem

Parsing with Probabilistic Grammars



$$\Pr(Tree_1 \mid O, G) > \Pr(Tree_2 \mid O, G)$$

- What is the likelihood that sentence O has tree T as its parse?
- Structural disambiguation: $\arg\max_{T_i \in Trees(O)} \Pr(T_i \mid O, G)$

Probabilistic Context-Free Grammars (PCFGs)

- Associate probabilities with production rules
- More expressive than N-Grams
- Models structural relationships
- Not good for language modeling [*c.f. Chelba et. al., 1998*]

Example of PCFG rules:

0.7 NP ⇒ DT N
0.3 NP ⇒ PRO

0.5 DT ⇒ a
0.1 DT ⇒ an
0.4 DT ⇒ the

...

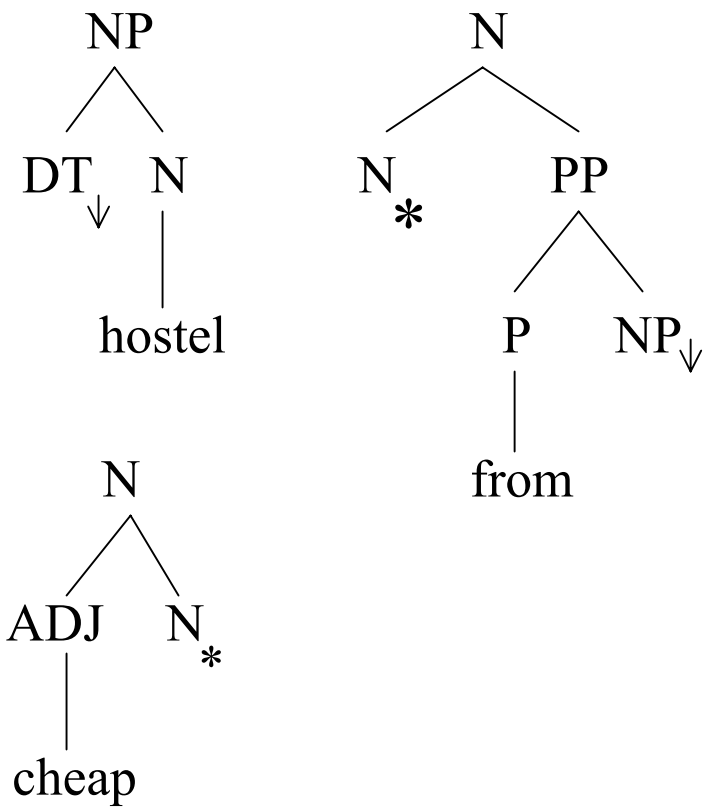
$$\Pr(T \mid G) = \prod \Pr(R_i \overset{*}{\Rightarrow} O)$$

Formalism Trade-Offs

Kind of Tasks	*Metric*	*Good Grammar Formalism*
Language Modeling	Minimizing Entropy	N-Grams
Parsing	Maximizing Accuracy	PCFGs

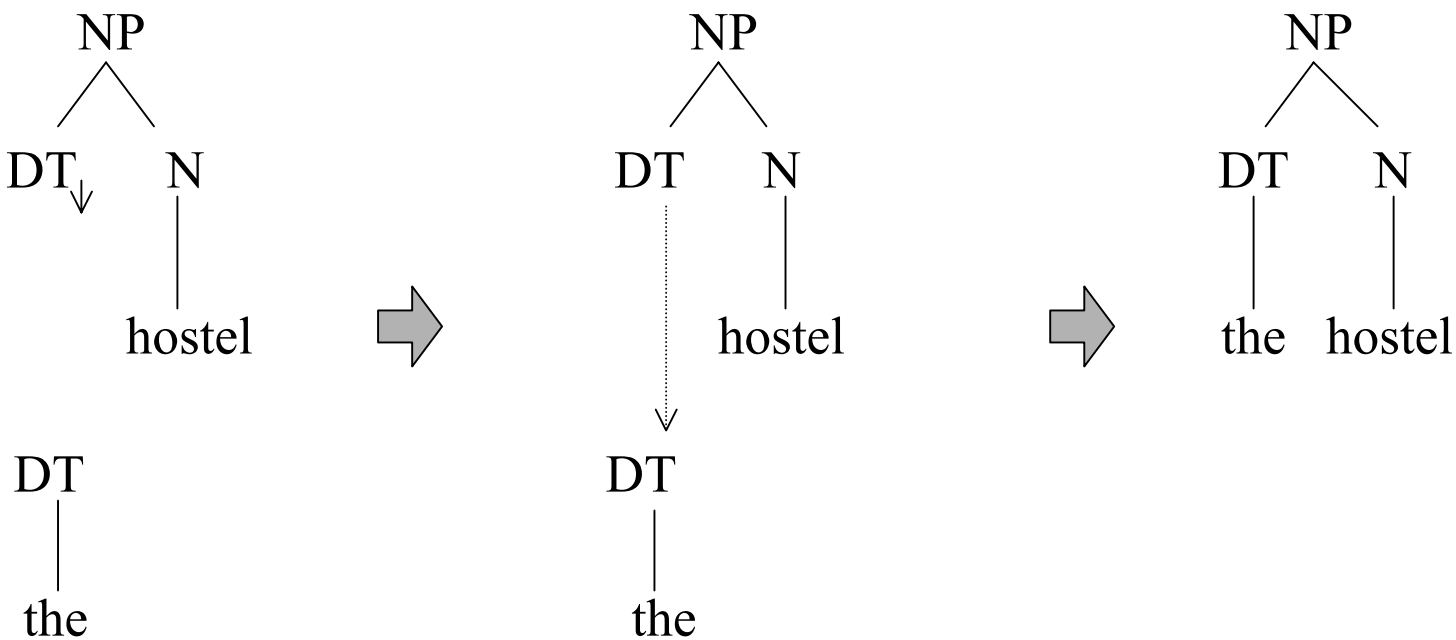
Proposed Grammar Representation

- Probabilistic Lexicalized Tree Insertion Grammars (PLTIGs) [*Schabes and Waters, 1993*]
  - Represent rules as tree fragments
  - Each tree contains a lexical word
  - Has the expressive power of Context-Free Grammars



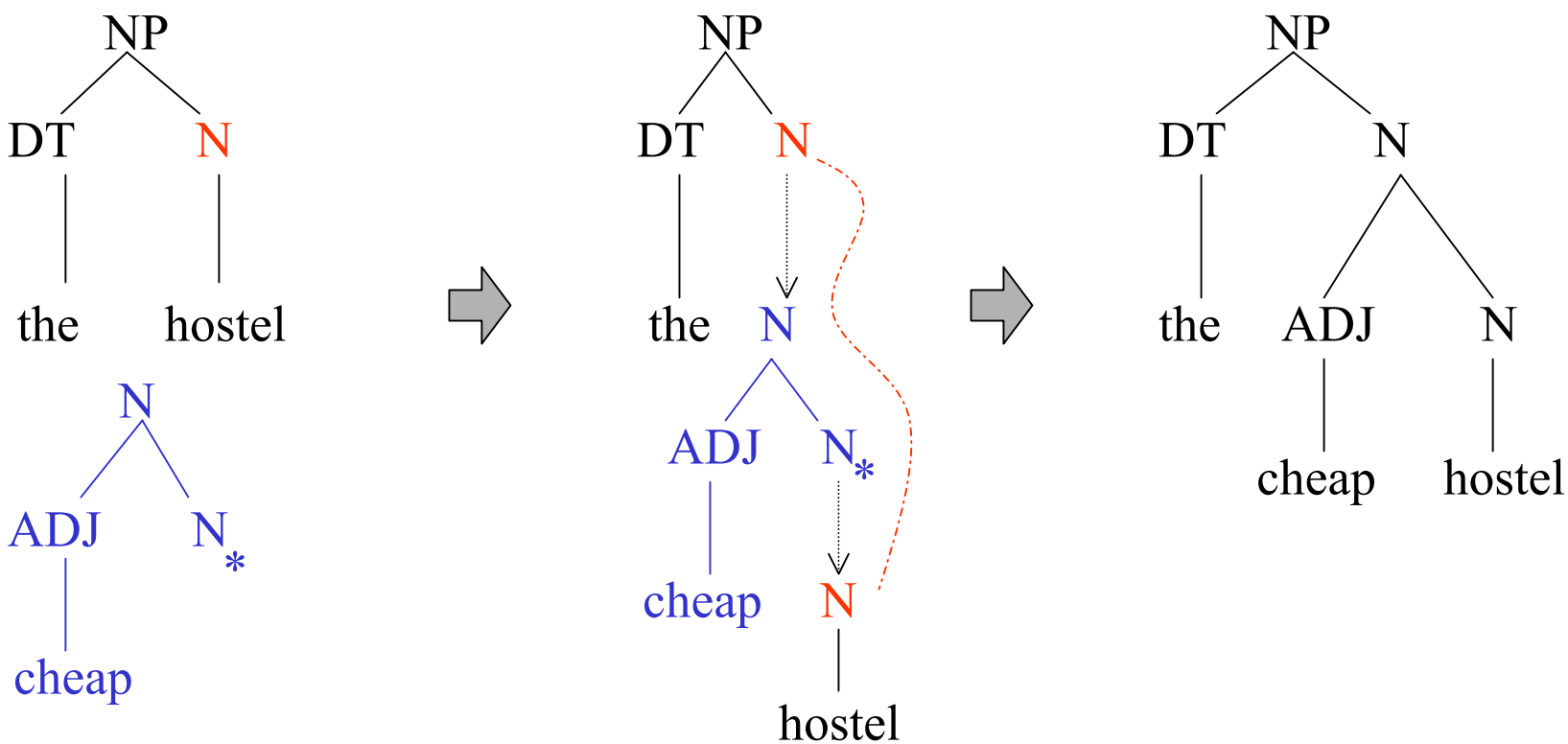
PLTIG operation: substitution

- Substitute one tree into the frontier node of another tree



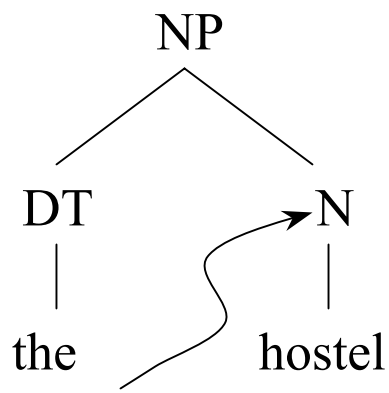
PLTIG Operation: Adjunction

- Adjoin an auxiliary tree into the interior node of another tree



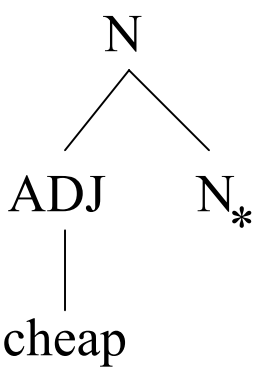
PLTIG Parameters

- associate a probability value for each operation between a tree and a node

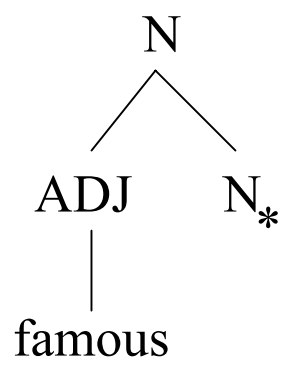


$\Pr(\text{no adj. } \eta_{N_{hostel}}) = 0.4$

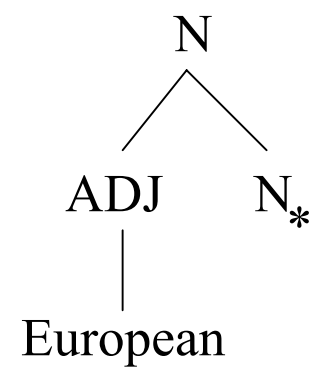
$\Pr(\rho_{cheap} \rightarrow \eta_{N_{hostel}}) = 0.3$



$\Pr(\rho_{famous} \rightarrow \eta_{N_{hostel}}) = 0.2$



$\Pr(\rho_{European} \rightarrow \eta_{N_{hostel}}) = 0.1$



Why PLTIGs?

- Why lexicalized tree representation?
  - Models structural and lexical relations
- Why adjunction?
  - Models long distance dependencies
  - Extends the domain of locality
- Why insertion constraints?
  - Computational efficiency
- Why probabilities?
  - Parameterize the grammar

Outline of the Talk

- Overview
  - Definitions of PLTIGs
- Inducing PLTIGs [*Hwa, 1998*]
  - Learning algorithm
    - Expectation-Maximization based algorithm
  - Comparison with other formalisms
- Efficient training techniques
- Conclusion and future directions

Training a PLTIG

- Grammar Induction
  - Unsupervised learning is difficult.

    Ford Motor Co. acquired 5 % of the shares in Jaguar PLC.
  - Supervised learning requires human effort.

    [[S [NP-SBJ Ford Motor Co.] [VP acquired [NP [NP 5 %] [PP of [NP [NP the shares] [PP-LOC in [NP Jaguar PLC]]]]]]] .]
- Expectation-Maximization (EM)
  - Heuristic search to find locally optimal grammars.
  - Supervision is not required, but can help to guide the search.

EM-based Learning Algorithm

Initialize with a PLTIG in *canonical form*

Repeat

E-Step:

Compute the *expected* likelihood of the training data using the current grammar model.

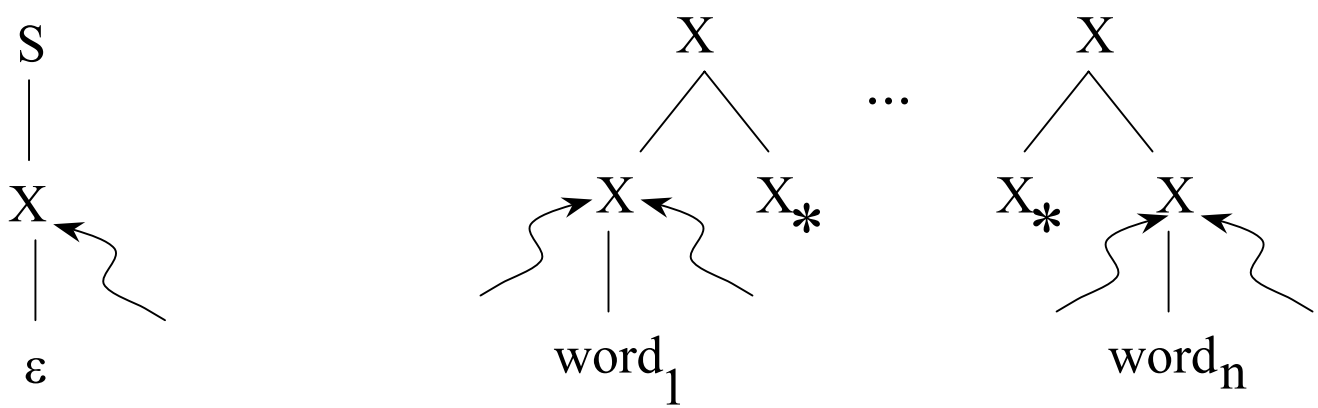
M-Step:

Update the parameters of the grammar to *maximize* the likelihood of it generating the training data.

Until convergence

Initialization

- Initial grammar must be able to generate any string
- Express the lexical trees in a canonical form (like the Chomsky Normal Form for PCFGs)



Head of sentence

Lexical trees

E-Step

- Compute the probability that each lexical tree is used in parsing training sentences

$$\Pr(\rho \overset{*}{\Rightarrow} w_i \ldots w_j, S \overset{*}{\Rightarrow} w_1 \ldots w_{i-1} \rho w_{j+1} \ldots w_n \mid G)$$

  - Bottom-up chart parsing computes: $\Pr(\rho \overset{*}{\Rightarrow} w_i \ldots w_j \mid G)$
  - Top-down pass to compute: $\Pr(S \overset{*}{\Rightarrow} w_1 \ldots w_{i-1} \rho w_{j+1} \ldots w_n \mid G)$
- Get expected number of times that each lexical tree is used to parse training sentences.

M-Step

- Re-estimate the parameters based on E-Step
- Re-estimation leads to over-fitting
  - Unseen adjunction between two trees gets zero probability
  - Not a problem for PCFG because it is not lexicalized.
- "Smooth" the distribution
  - deleted-interpolation

$$\overline{P}_{\rho\to\eta} = \lambda \frac{c(\rho \to \eta)}{\sum_{\rho_i} c(\rho_i \to \eta)} + (1-\lambda) \frac{\sum_{\eta_j} c(\rho \to \eta_j)}{\sum_{\eta_j}\sum_{\rho_i} c(\rho_i \to \eta_j)}$$

Comparison Study

- Formalisms
  - N-Grams (N=2, N=3)
  - PCFG
  - PLTIG
- WSJ Corpus
  - 13242 training sentences

    [[Ford Motor Co.] [acquired [[5 %] [of [[the shares] [in [Jaguar PLC]]]]]] .]
  - 2245 testing sentences
  - 48 part-of-speech tags as words

Comparison Study

- Tasks
  - Language modeling
    - Minimizing entropy (uncertainty)
  - Statistical parsing
    - Maximizing accuracy

$$\text{Accuracy} = \frac{\text{\# of brackets consistent with true brackets}}{\text{\# of brackets guessed}}$$

True Parse: [[Ford Motor Co.][acquired [the shares]].]
Guessed Parse: [[Ford [Motor Co.]][[acquired the] shares].]

Comparison Results

	Bi-Gram (2,400)*	**Tri-Gram** (115,296)	**PCFG** (13,271)	**PLTIG** (14,210)
Iterations to convergence	--	--	70	**30**
Real-time convergence (hrs)	--	--	511	**41**
Entropy	3.39	**3.20**	3.63	3.32
Parsing accuracy (%)	49.44	49.44	79.30	**82.43**

*Number of parameters of the grammar

- PLTIGs have the language modeling capability of N-grams and the parsing quality of PCFGs.

Outline of the Talk

- Overview
- Inducing PLTIGs
- Efficient training techniques
  - Sample selection
    - Evaluation functions
      - Sentence Length
      - Tree Entropy
- Conclusion and future directions

Supervised Training

- Training example with structure information

  [[Ford Motor Co.] [acquired [[5 %] [of [[the shares] [in [Jaguar PLC]]]]]] .]

- Annotation is labor intensive
- Minimize annotations
  - By using fewer examples [*Hwa, in preparation*]
  - By using fewer brackets per example [*Hwa,1999*]

Sample Selection

- Annotate examples as needed during training
- Predict the *benefit* of annotation
  - *Training Utility Value (TUV)*
    - the improvement of the hypothesis if a candidate is labeled and added to the training set
  - Annotate the candidates with the highest TUVs
- Can be applied to grammar induction

Sample Selection for Prepositional Phrase Attachment

- Training examples:
  - quintuple (attachment, v, n1, p, n2)

  (v, left, hostel, without, energy)
  (n, left, hostel, without, electricity)
- Sample selection reduces the number of training examples by 46% with <1% loss of accuracy.

Approximation of TUVs

- True TUVs are not known without labeling
- Don’t need to know the absolute TUVs
  - only need to approximate the relative ordering between sentences.
- Evaluation functions
  - heuristics to approximate TUVs

Sample Selection Algorithm

Initialize:

Train on small set of labeled examples to get initial hypothesis

Repeat

Using current hypothesis and an evaluation function, *f*, pick *n* examples from unlabeled pool.

Ask human to label those *n* examples.

Add them to training set.

Re-train to get a new hypothesis.

Until (hypothesis good enough) or (human stops).

Uncertainty

- Conjecture: reduction in uncertainty improves the grammar
  - A grammar is uncertain about a sentence if it has multiple, equally-likely parses for the sentence.
- Evaluation Functions
  - Heuristics to model uncertainties in the parse-trees

Proposed Evaluation Functions

- Random f_{rand}
  - baseline
- Sentence length f_{len}
  - in general, longer sentences have more parses.
- Tree entropy f_{te}
  - use entropy to measure a grammar's uncertainty over possible parse trees.

Entropy

- Measure of uncertainty
- Expected number of bits for encoding a probability distribution

$$H(X) = -\sum_{x \in \chi} p(x) \log p(x)$$

Example:

Let X be a random variable with a distribution over N outcomes

- Uniform distribution:
  - H(X) = log(N)
- Pr(X=x_0) = 1
  - H(X) = log(1) = 0

Tree Entropy

- Probability distribution over all parse trees.

$$\sum_{T_i \in Trees(O)} \Pr(T_i \mid O, G) = 1$$

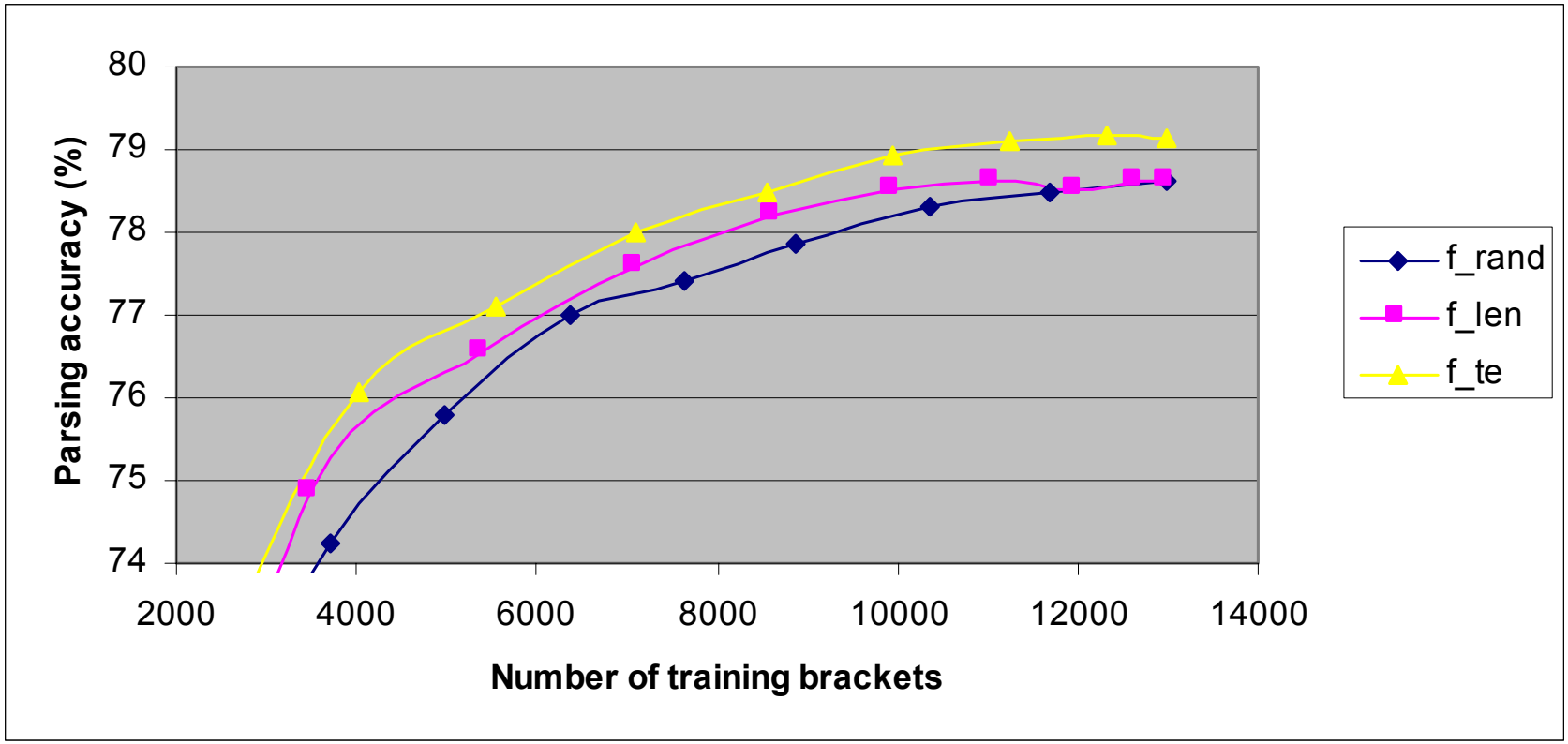
  – Uniform distribution $\Rightarrow$ very uncertain
  – Spike distribution $\Rightarrow$ very certain

$$f_{te} = -\frac{1}{len(O)} \sum_{T_i \in Trees(O)} \Pr(T_i \mid O, G) \lg \Pr(T_i \mid O, G)$$

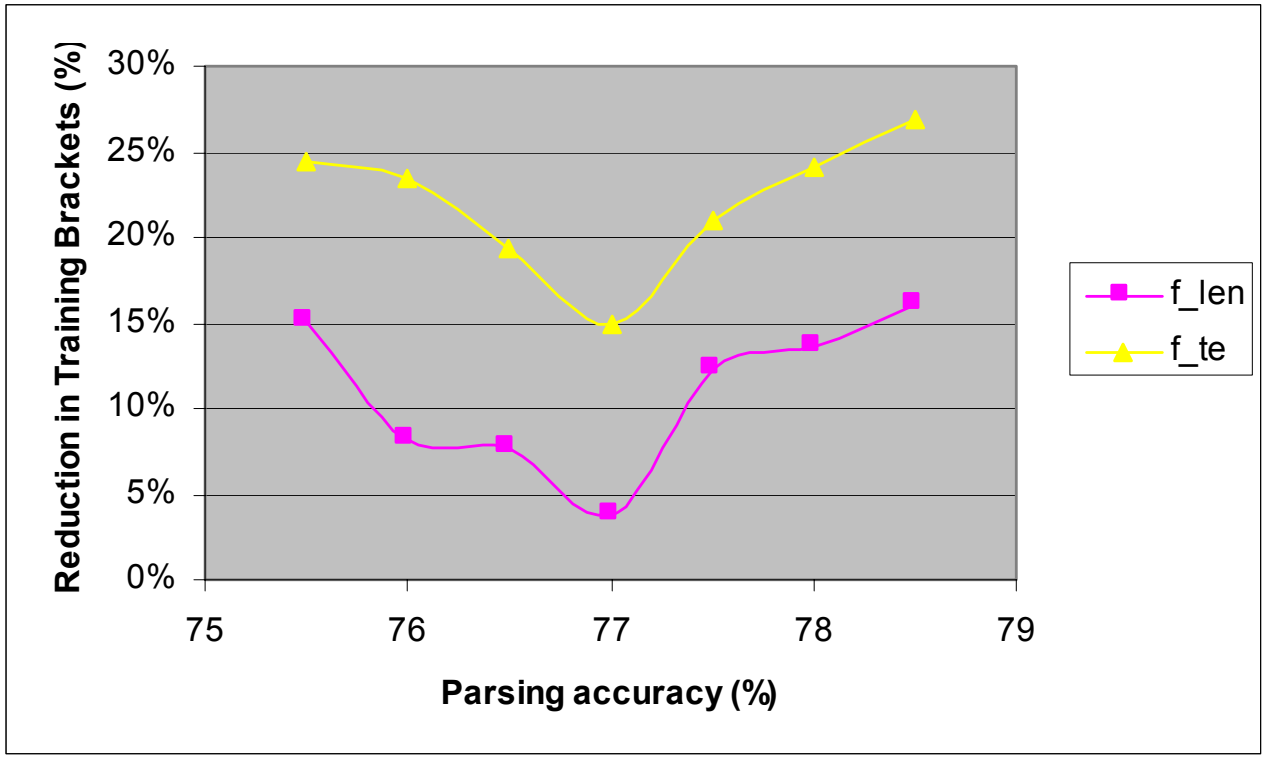
Experimental Setup

- 10 sets of randomly generated train-test pair from the WSJ Corpus
- 1000 unlabeled sentences as potential training sentences.
- In each selection phase, pick 100 highest ranked sentences
- Evaluation functions: $f_{rand}, f_{len}, f_{te}$

Results


80
79
78
77
76
75
74
Parsing accuracy (%)
2000
4000
6000
8000
10000
12000
14000
Number of training brackets
f_rand
f_len
f_te

Results



f_{te} reduces the number of training brackets by 27% at 78.5% accuracy

f_{len} reduces the number of training brackets by 16% at 78.5% accuracy

Conclusion

- Grammar representation
  - Empirical study showing PLTIGs combine the advantages of N-grams and PCFGs.
- Learning algorithm
  - PLTIGs can be trained with partial supervision
- Training efficiency
  - Sample selection
    - Reduces human effort for labeling training data
    - Uses uncertainties to approximate TUVs

Future Directions

- PLTIGs
  - Make use of more lexical features
  - Synchronous-PLTIGs for machine translation
- Minimize supervision
  - Committee-based sample selection
  - Membership queries
  - Apply to other NLP learning tasks

Future Directions (Cont'd)

- Data sparsity
  - Efficient use of existing technologies
- Human-computer interaction
  - Division of labor