

# A Continuum of Bootstrapping Methods for Parsing Natural Languages

*University of Rochester*  
*Dec. 12, 2003*

Rebecca Hwa  
University of Pittsburgh  
*hwa@cs.pitt.edu*

# The Role of Parsing in Language Applications...

- As a stand-alone application
  - Grammar checker
- As a pre-processing step
  - Q&A, information extraction, dialogue systems
- As an integral part of a model
  - Speech Recognition
    - language models
  - Machine Translation
    - word alignment

# Challenges in Building Parsers

- Disambiguation
  - Lexical disambiguation
  - Structural disambiguation
- Rule Exceptions
  - Many lexical dependencies
- Manual Grammar Construction
  - Limited coverage
  - Difficult to maintain

# Meeting these Challenges: Statistical Parsing

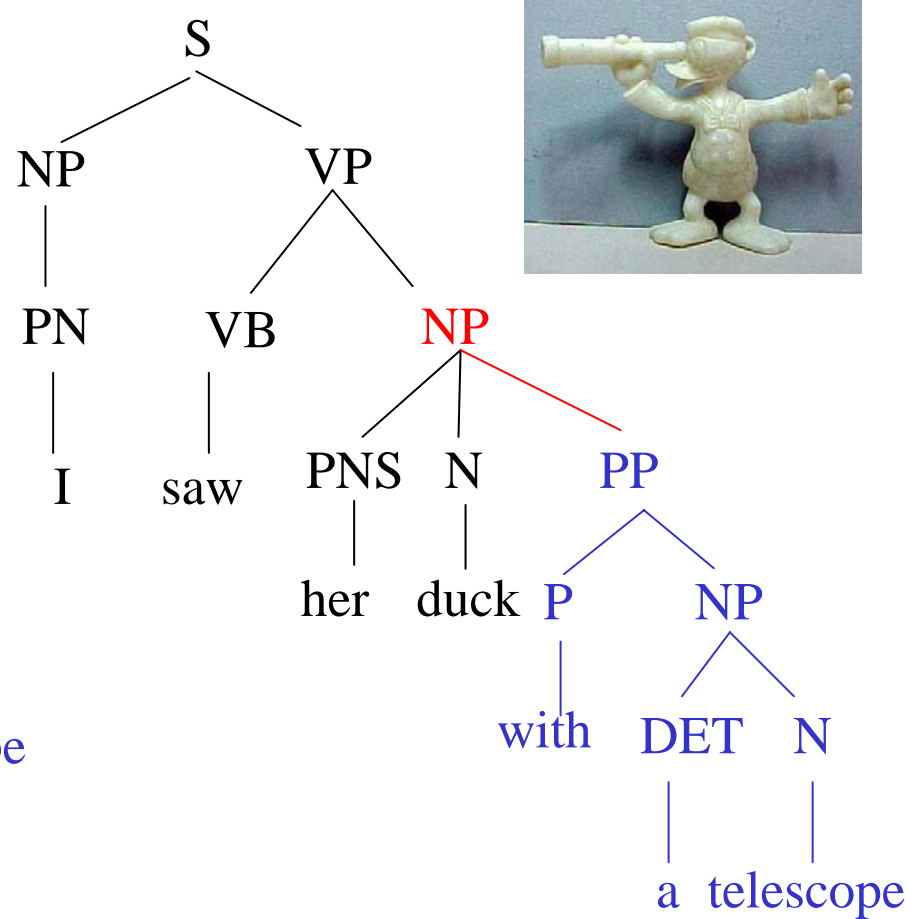
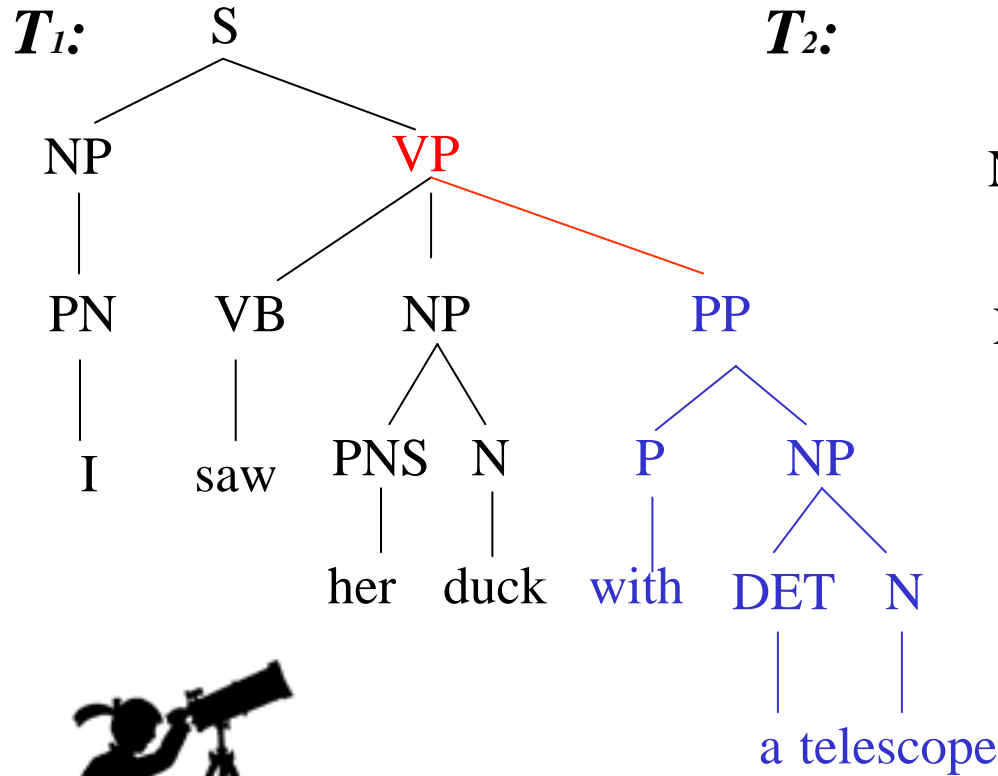
- Disambiguation?
  - Resolve local ambiguities with global likelihood
- Rule Exceptions?
  - Lexicalized representation
- Manual Grammar Construction?
  - Automatic induction from large corpora
  - **A new challenge: how to obtain enough suitable training corpora?**
  - **Make better use of both annotated and unprocessed text through an iterative process**

# Roadmap

- Parsing as a learning problem
- Three bootstrapping methods
  - Sample selection
  - Co-training
  - Corrected co-training
- Conclusion and further directions

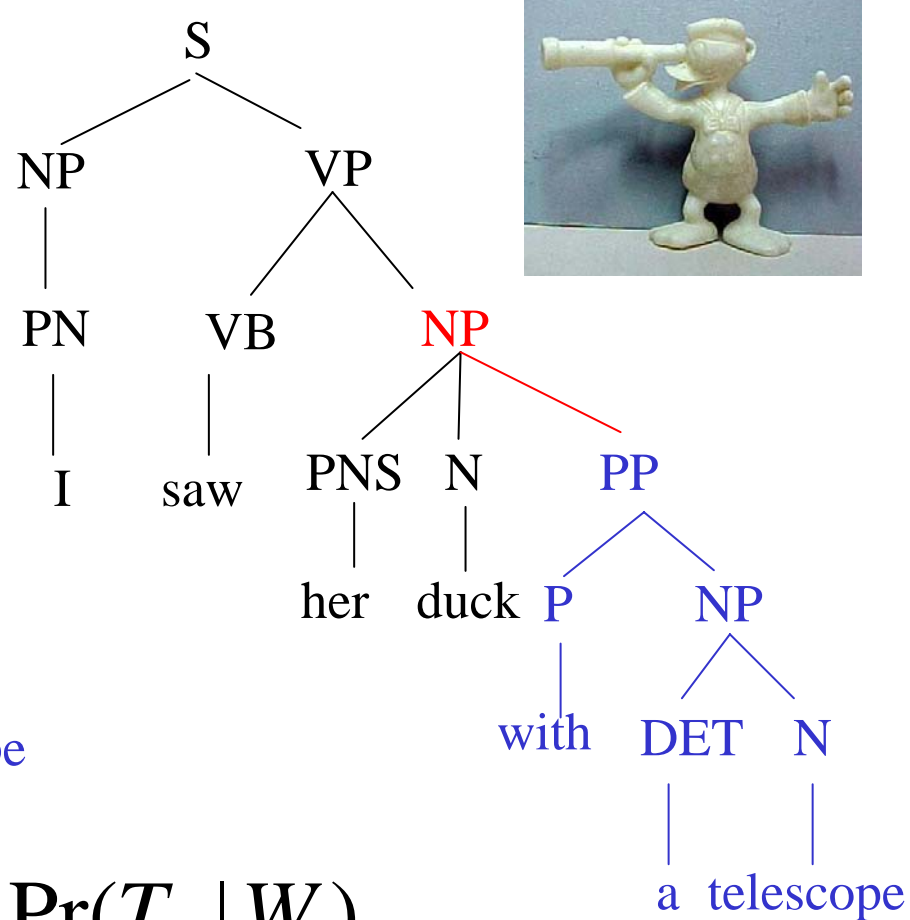
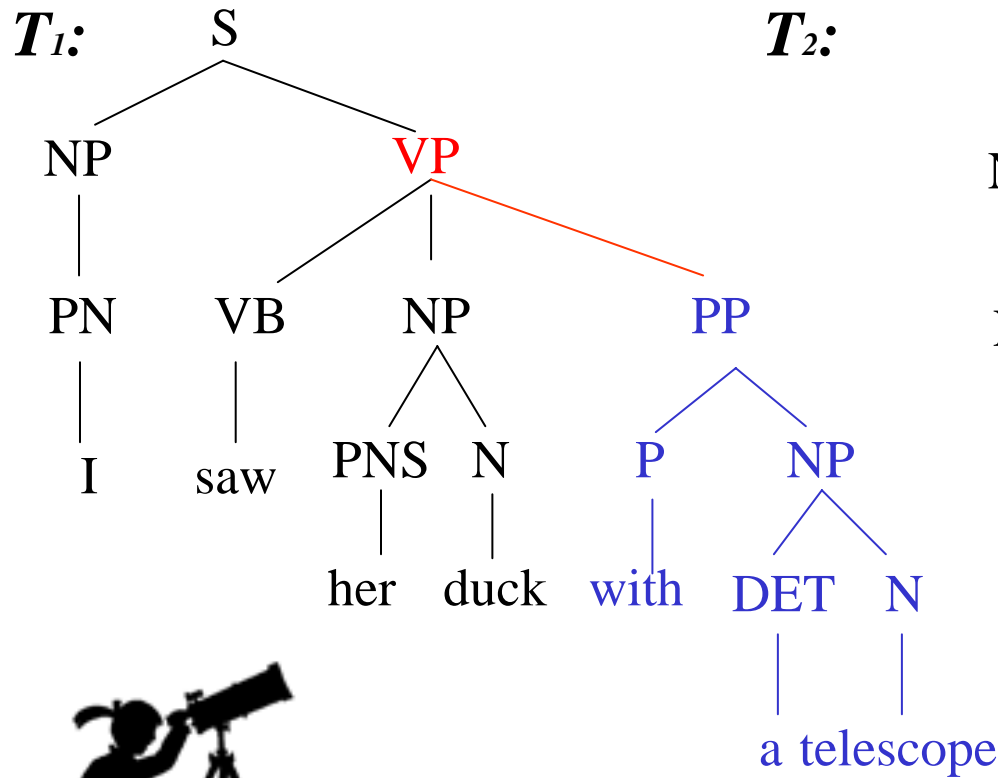
# Parsing Ambiguities

**Input:** “*I saw her duck with a telescope*”



# Disambiguation with Statistical Parsing

$W = \text{"I saw her duck with a telescope"}$



$$\Pr(T_1 | W) > \Pr(T_2 | W)$$

# A Statistical Parsing Model

- Probabilistic Context-Free Grammar (PCFG)
- Associate probabilities with production rules
- Likelihood of the parse is computed from the rules used
- Learn rule probabilities from training data

*Example of PCFG rules:*

0.7 NP → DET N

0.3 NP → PN

0.5 DET → a

0.1 DET → an

0.4 DET → the

...

$$\arg \max_{T_i \in \text{Trees}(W)} \Pr(T_i | W) = \arg \max_{T_i \in \text{Trees}(W)} \frac{\Pr(T_i, W)}{\Pr(W)}$$

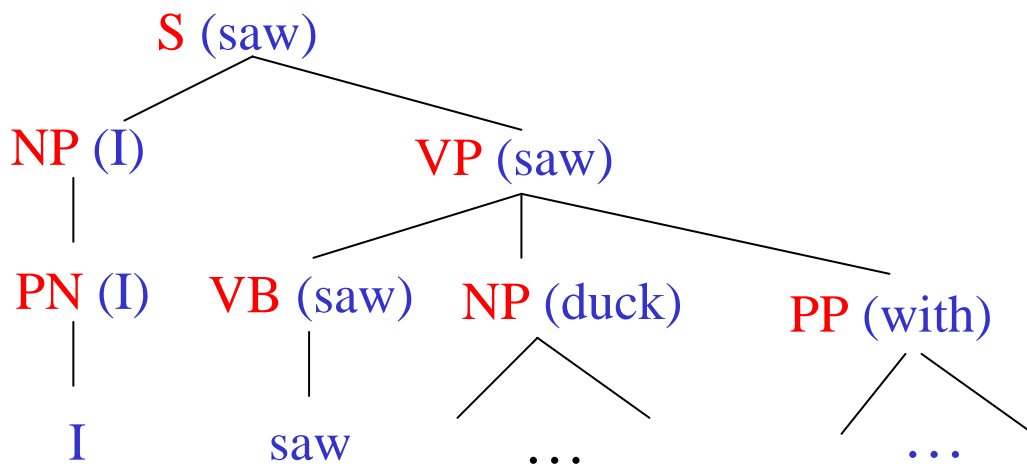
$$\Pr(T_i, W) = \prod_r \Pr(RHS_r | LHS_r)$$

# Handle Rule Exceptions with Lexicalized Representations

- Model relationship between words as well as structures
  - Modify the production rules to include words
    - Greibach Normal Form
  - Represent rules as tree fragments anchored by words
    - Lexicalized Tree Grammars
  - Parameterize the production rules with words
    - Collins Parsing Model

# Example: Collins Parsing Model

- Rule probabilities are composed of probabilities of bi-lexical dependencies



$S(\text{saw}) \Rightarrow NP(I) VP(\text{saw})$

$$\begin{aligned} \Pr(S[\text{saw}] \rightarrow NP[I], VP[\text{saw}]) = & \\ & \Pr_H(VP | S, \text{saw}, VB) \times \\ & \Pr_L(NP(I) | S, VP, \text{saw}, VB) \times \\ & \Pr_L(STOP | S, VP, \text{saw}, VB) \times \\ & \Pr_R(STOP | S, VP, \text{saw}, VB) \end{aligned}$$

# Machine Learning Avoids Manual Construction

- Supervised training
  - Training examples are pairs of **problems (instances)** and **answers (labels)**
  - Training examples for parsing: a collection of **sentence, parse tree** pairs (**Treebank**)
- **New challenge:** treebanks are difficult to obtain
  - Needs human experts
  - Takes years to complete

# Learning to Classify

Train a model to decide: should a **prepositional phrase** modify the **verb** before it or the **noun**?

*Training examples:*

(v, saw, duck, with, telescope)

(n, saw, duck, with, feathers)

(v, saw, stars, with, telescope)

(n, saw, stars, with, Oscars)

...

# Learning to Parse

Train a model to decide: what is the most likely **parse** for a **sentence W**?

```
[S [NP-SBJ [NNP Ford] [NNP Motor] [NNP Co.]]  
  [VP [VBD acquired]  
      [NP [NP [CD 5] [NN %]]  
          [PP [IN of]  
              [NP [NP [DT the] [NNS shares]]  
                  [PP [IN in]  
                      [NP [NNP Jaguar]  
                          [NNP PLC]]]]]]]] . ]
```

```
[S [NP-SBJ [NNP Pierre] [NNP Vinken]]  
  [VP [MD will]  
      [VP [VB join]  
          [NP [DT the] [NN board]]  
          [PP [IN as] [NP [DT a] [NN director]]]] . ]
```

...

# Building Treebanks

Language	Size of Treebank	Time to Develop	Parser Performance
English (WSJ)	1M words 40k sent.	~5 years	~90%
Chinese (Xinhua News)	100K words 4k sent.	~2 years	~75%
Others (e.g., Hindi, Cebuano)	?	?	?

# Our Approach

- Make use of both **labeled** and **unlabeled** data during training
- Bootstrapping
  - Improve the parsing model(s) iteratively
- Three methods:
  - Sample selection
    - Machine picks unlabeled data for human to label
  - Co-training
    - Machines label data for each other
  - Corrected Co-training
    - Combines sample selection and co-training

# Roadmap

- Parsing as a learning problem
- Three bootstrapping methods
  - Sample selection
  - Co-training
  - Corrected Co-training
- Conclusion and further directions

# Sample Selection Algorithm

## Initialize

Train the parser on a *small* treebank (seed data) to get the initial parameter values.

## Repeat

Create a candidate set by randomly sample a large unlabeled pool.

Estimate the *Training Utility Value* of each sentence in the candidate set with a **scoring function,  $f$** .

Pick the  $n$  sentences with the highest score (according to  $f$ ).

Human labels these  $n$  sentences and add them to training set.

Re-train the parser with the updated training set.

**Until** (no more data) or (human stops).

# Scoring Function

- Approximate the TUV of each sentence
  - True TUVs are not known
- Need relative ranking
- Ranking criteria
  - Knowledge about the domain
    - e.g., sentence clusters, sentence length, ...
  - Output of the hypothesis
    - e.g., error-rate of the parse, uncertainty of the parse, ...
  - ....

# Proposed Scoring Functions

- Using domain knowledge
  - $f_{len}$  long sentences tend to be complex
- Uncertainty about the output of the parser
  - $f_{te}$  tree entropy
- Minimize mistakes made by the parser
  - $f_{error}$  use an **oracle** scoring function find sentences with the most parsing inaccuracies

# Entropy

- Measure of uncertainty in a distribution
  - Uniform distribution  $\Rightarrow$  very uncertain
  - Spike distribution  $\Rightarrow$  very certain
- Expected number of bits for encoding a probability distribution,  $X$

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

# Tree Entropy Scoring Function

- Distribution over parse trees for sentence  $W$ :

$$\sum_{T_i \in \text{Trees}(W)} \Pr(T_i | W) = 1$$

- Tree entropy: uncertainty of the parse distribution

$$TE(W) = - \sum_{T_i \in \text{Trees}(W)} \Pr(T_i | W) \log \Pr(T_i | W)$$

- Scoring function: ratio of actual parse tree entropy to that of a uniform distribution

$$f_{te} = \frac{TE(W)}{\log(|\text{Trees}(W)|)}$$

# Oracle Scoring Function

- $f_{error}$  1 - the accuracy rate of the most-likely parse
- Parse accuracy metric: f-score

f-score = harmonic mean of precision and recall

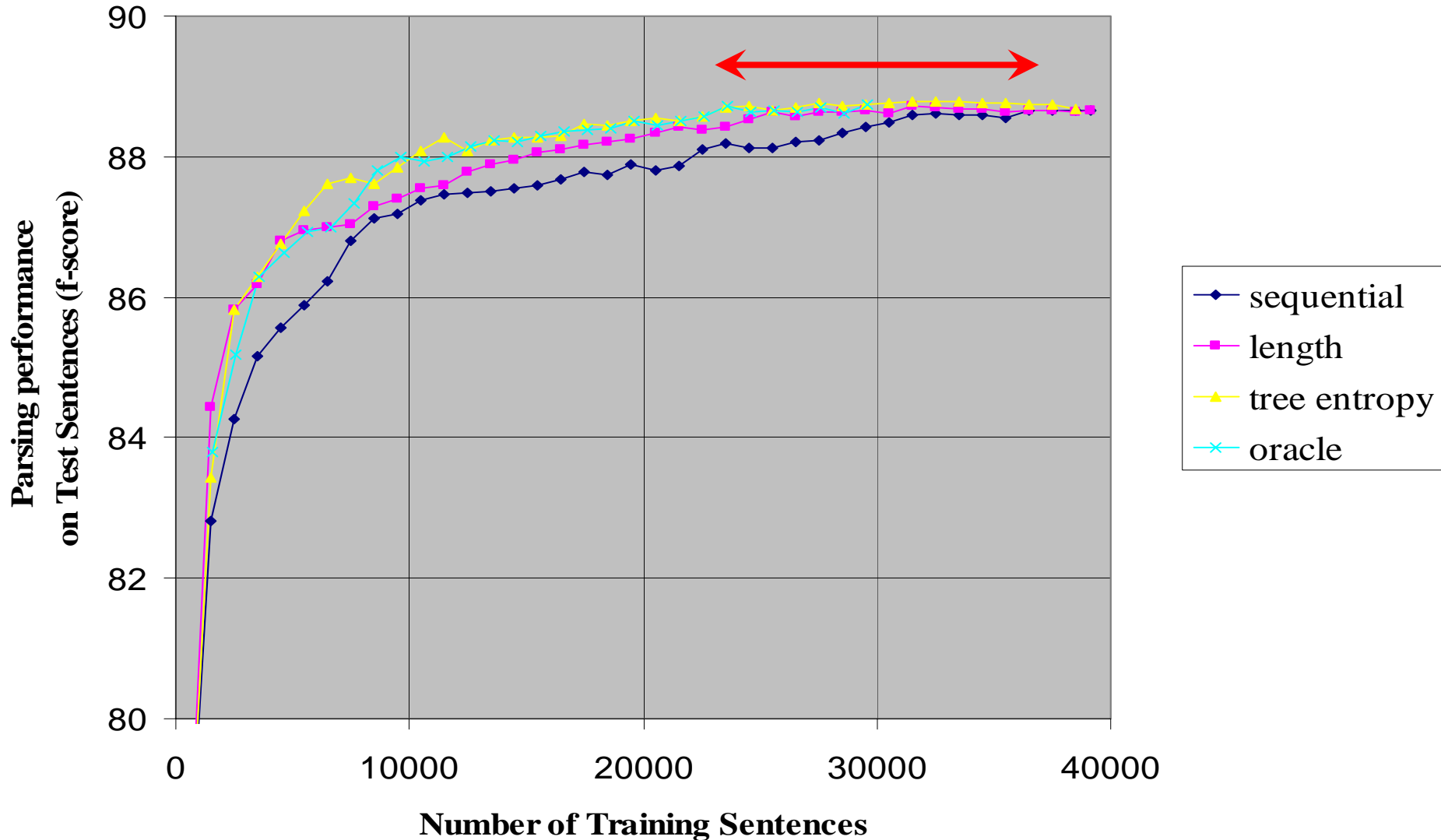
$$\text{Precision} = \frac{\# \text{ of correctly labeled constituents}}{\# \text{ of constituents generated}}$$

$$\text{Recall} = \frac{\# \text{ of correctly labeled constituents}}{\# \text{ of constituents in correct answer}}$$

# Experimental Setup

- Parsing model:
  - Collins Model 2
- Candidate pool
  - WSJ sec 02-21, with the annotation stripped
  - Initial labeled examples: 500 sentences
  - Per iteration: add 100 sentences
- Testing metric: f-score (precision/recall)
- Test data:
  - ~2000 unseen sentences (from WSJ sec 00)
- Baseline
  - Annotate data in sequential order

# Training Examples Vs. Parsing Performance



# Parsing Performance Vs. Constituents Labeled

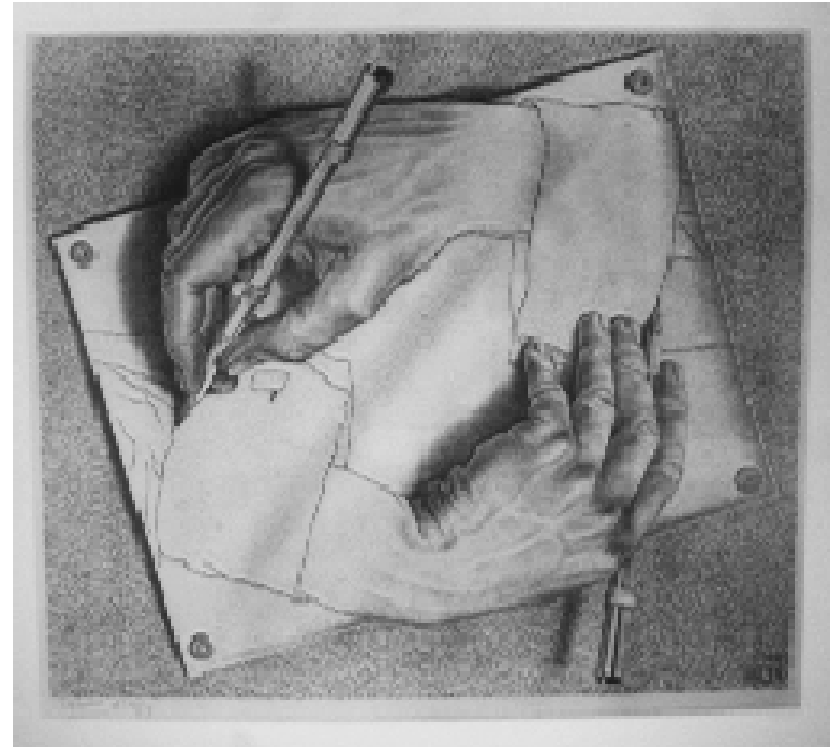


# Roadmap

- Parsing as a learning problem
- Three bootstrapping methods
  - Sample selection
  - Co-training [*joint work with Steedman et al.*]
  - Corrected Co-training
- Conclusion and further directions

# Co-Training

- Assumptions
  - Have a small treebank
  - No further human assistance
  - Have two different kinds of parsers
- A subset of each parser's output becomes new training data for the other
- Goal:
  - select sentences that are labeled with confidence by one parser but labeled with uncertainty by the other parser.



# Algorithm

## Initialize

Train two parsers on a *small* treebank (seed data) to get the initial models.

## Repeat

Create a candidate set by randomly sample a large unlabeled pool.

Each parser labels the candidate set and estimates *the accuracy of its output* with **scoring function,  $f$** .

Choose examples according to some **selection method,  $S$**  (using the scores from  $f$ ).

Add them to the parsers' training sets.

Re-train parsers with the updated training sets.

**Until** (no more data).

# Scoring Functions

- Evaluates the quality of each parser's output
- Ideally, function measures accuracy
  - Oracle  $f_{\text{F-score}}$ 
    - combined prec./rec. of the parse
- Practical scoring functions
  - Conditional probability  $f_{\text{cprob}}$ 
    - Prob(parse | sentence)
  - Others (joint probability, entropy, etc.)

# Selection Methods

- Above- $n$ :  $S_{\text{above-}n}$  [Blum & Mitchell, 1998]
  - The score of the teacher's parse is greater than  $n$
- Difference:  $S_{\text{diff-}n}$ 
  - The score of the teacher's parse is greater than that of the student's parse by  $n$
- Intersection:  $S_{\text{int-}n}$ 
  - The score of the teacher's parse is one of its  $n\%$  highest while the score of the student's parse for the same sentence is one of the student's  $n\%$  lowest

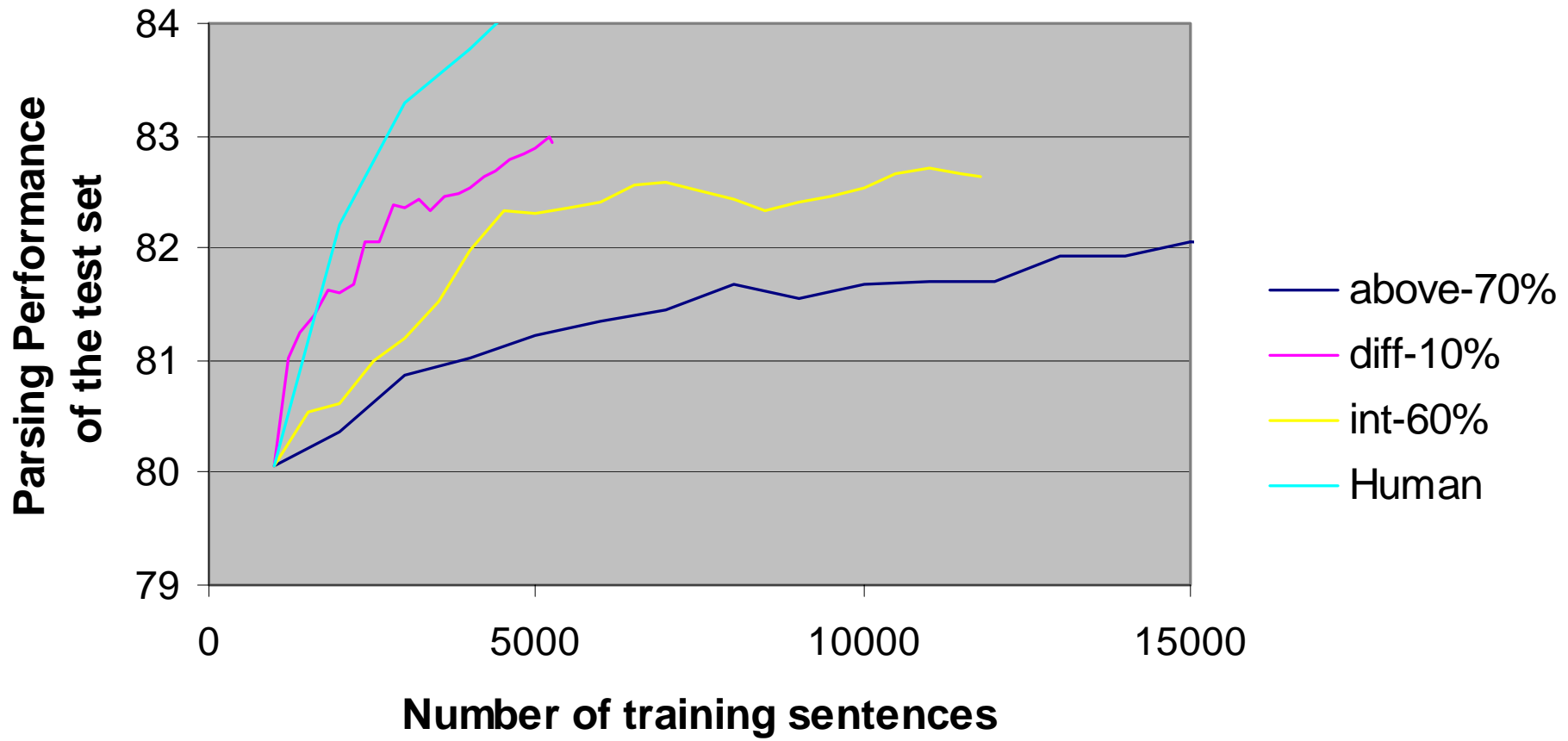
# Experimental Setup

- Co-training parsers:
  - Lexicalized Tree Adjoining Grammar parser [*Sarkar, 2002*]
  - Lexicalized Context Free Grammar parser [*Collins, 1997*]
- Seed data: 1000 parsed sentences from WSJ sec02
- Unlabeled pool: rest of the WSJ sec02-21, stripped
- Consider 500 unlabeled sentences per iteration
- Development set: WSJ sec00
- Test set: WSJ sec23
- Results: graphs for the Collins parser

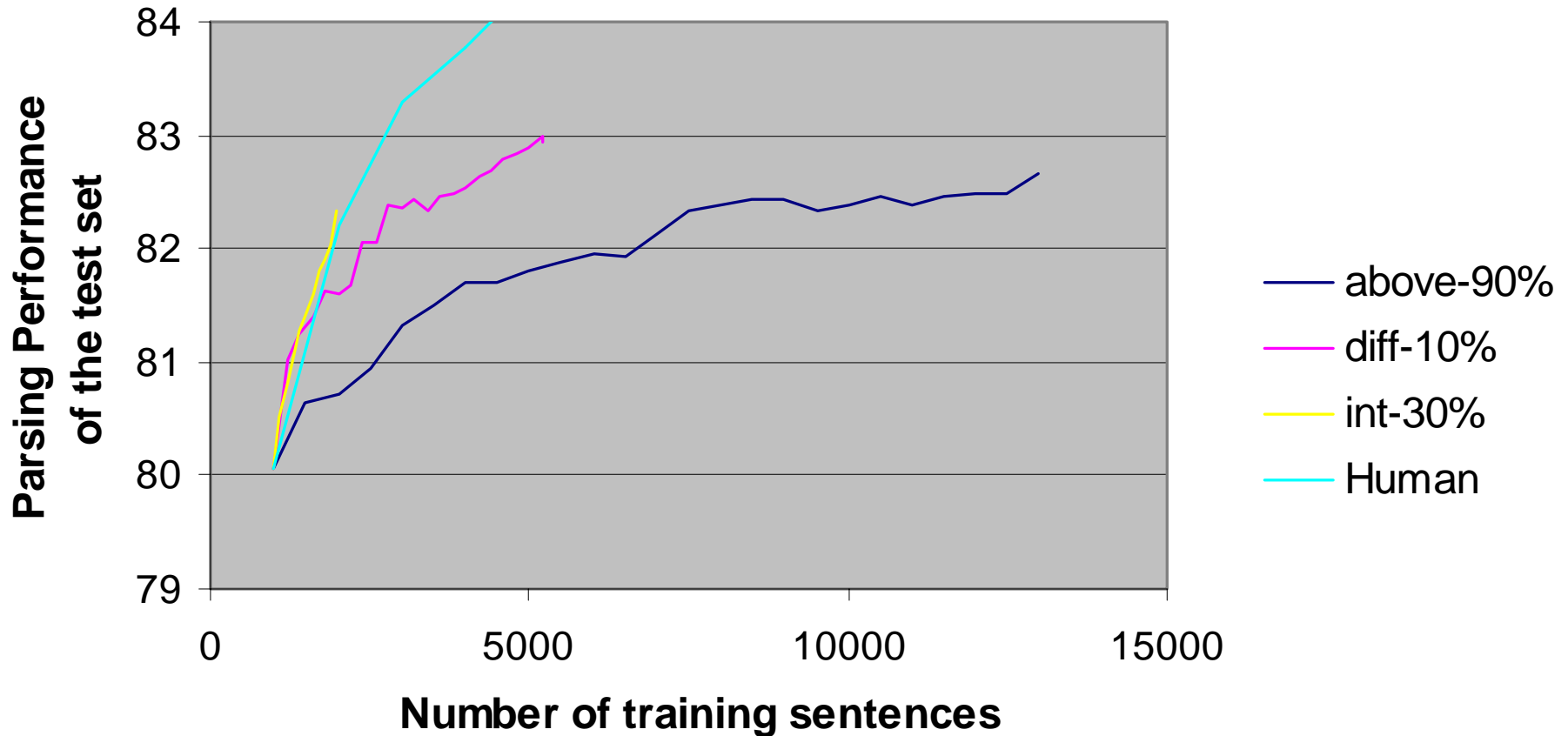
# Selection Methods and Co-Training

- Two scoring functions:  $f_{\text{F-score}}$ ,  $f_{\text{cprob}}$
- Multiple view selection vs. one view selection
  - Three selection methods:  $S_{\text{above-}n}$ ,  $S_{\text{diff-}n}$ ,  $S_{\text{int-}n}$
- Maximizing utility vs. minimizing error
  - For  $f_{\text{F-score}}$ , we vary  $n$  to control accuracy rate of the training data
  - Loose control
    - More sentences (avg. F-score of train set = 85%)
  - Tight control
    - Fewer sentences (avg. F-score of train set = 95%)

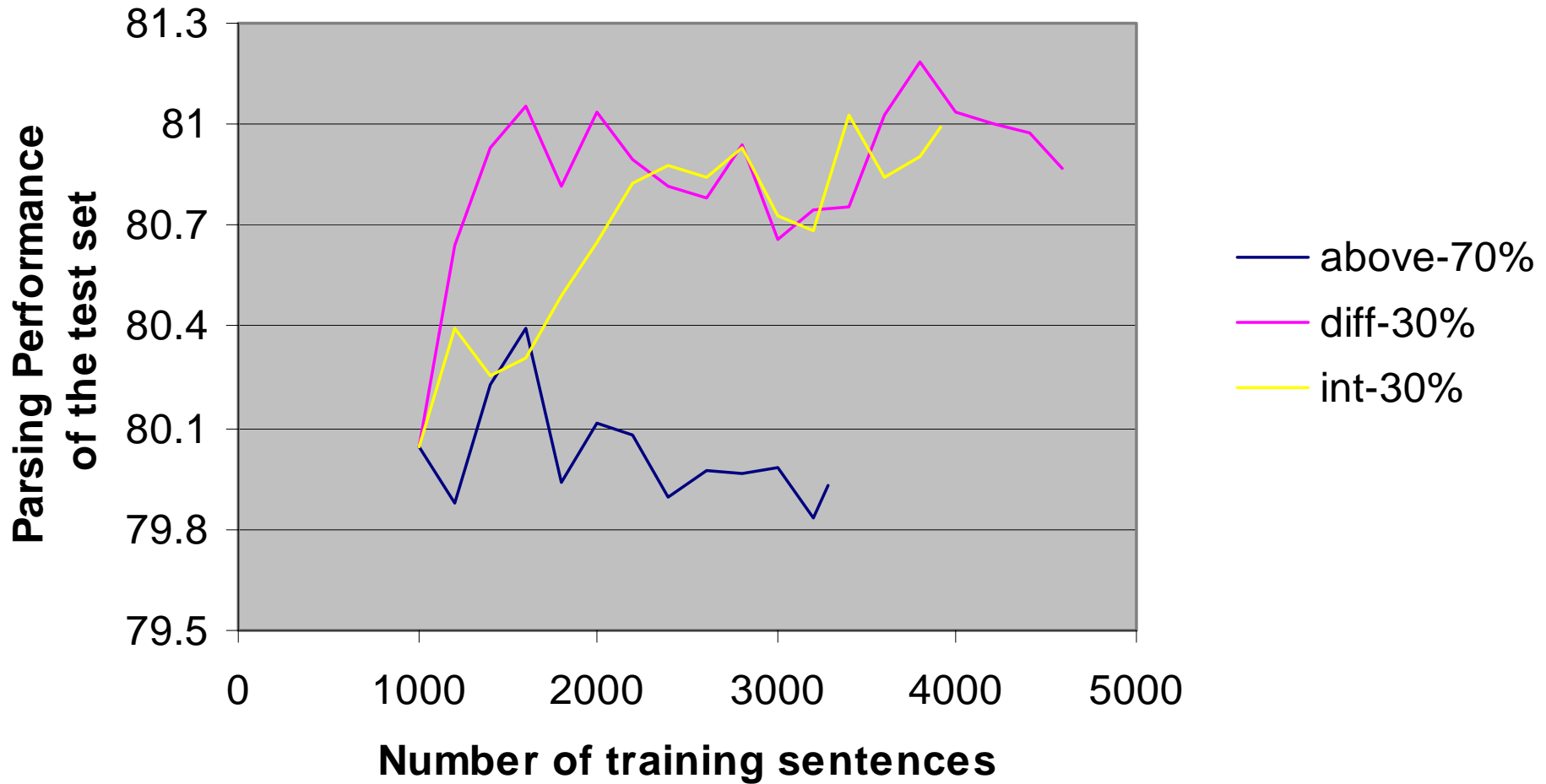
# Co-Training using $f_{F\text{-score}}$ with Loose Control



# Co-Training using $f_{F\text{-score}}$ with Tight Control



# Co-Training using $f_{\text{cprob}}$



# Roadmap

- Parsing as a learning problem
- **Three bootstrapping methods**
  - Sample selection
  - Co-training
  - **Corrected Co-training**
- Conclusion and further directions

# Corrected Co-Training

- Human **reviews** and **corrects** the machine outputs before they are added to the training set
- Can be seen as a variant of sample selection [cf. *Muslea et al., 2000*]
- Has been applied to Base NP detection [ *Pierce & Cardie, 2001* ]

# Algorithm

## Initialize:

Train two parsers on a *small* treebank (seed data) to get the initial models.

## Repeat

Create a candidate set by randomly sample a large unlabeled pool. Each parser labels the candidate set and evaluates its output with scoring function,  $f$ .

Choose examples according to some selection method,  $S$  (using the scores from  $f$ ).

Human reviews and corrects the chosen examples.

Add them to the parsers' training sets.

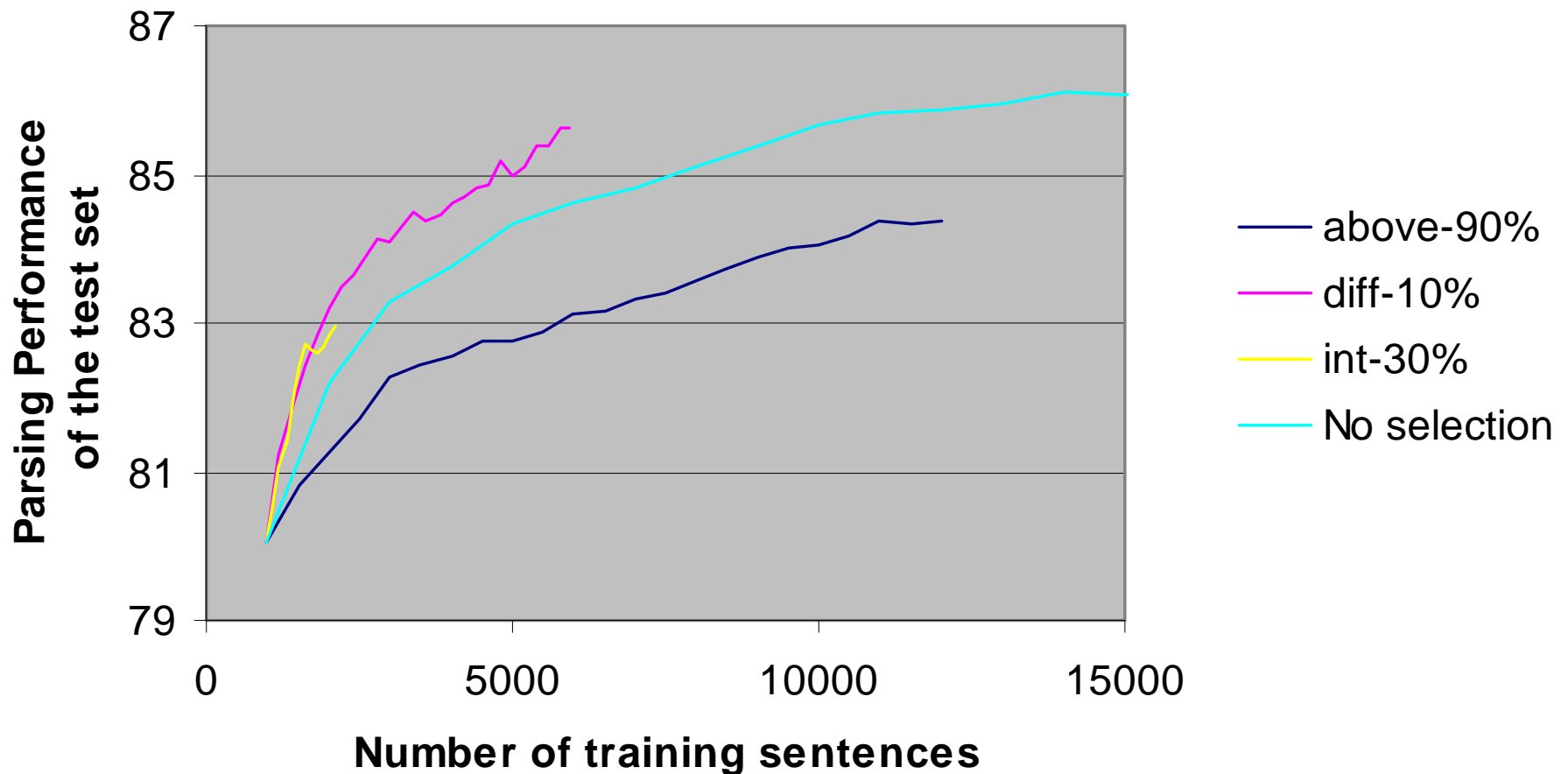
Re-train parsers with the updated training sets.

**Until** (no more data) or (human stops).

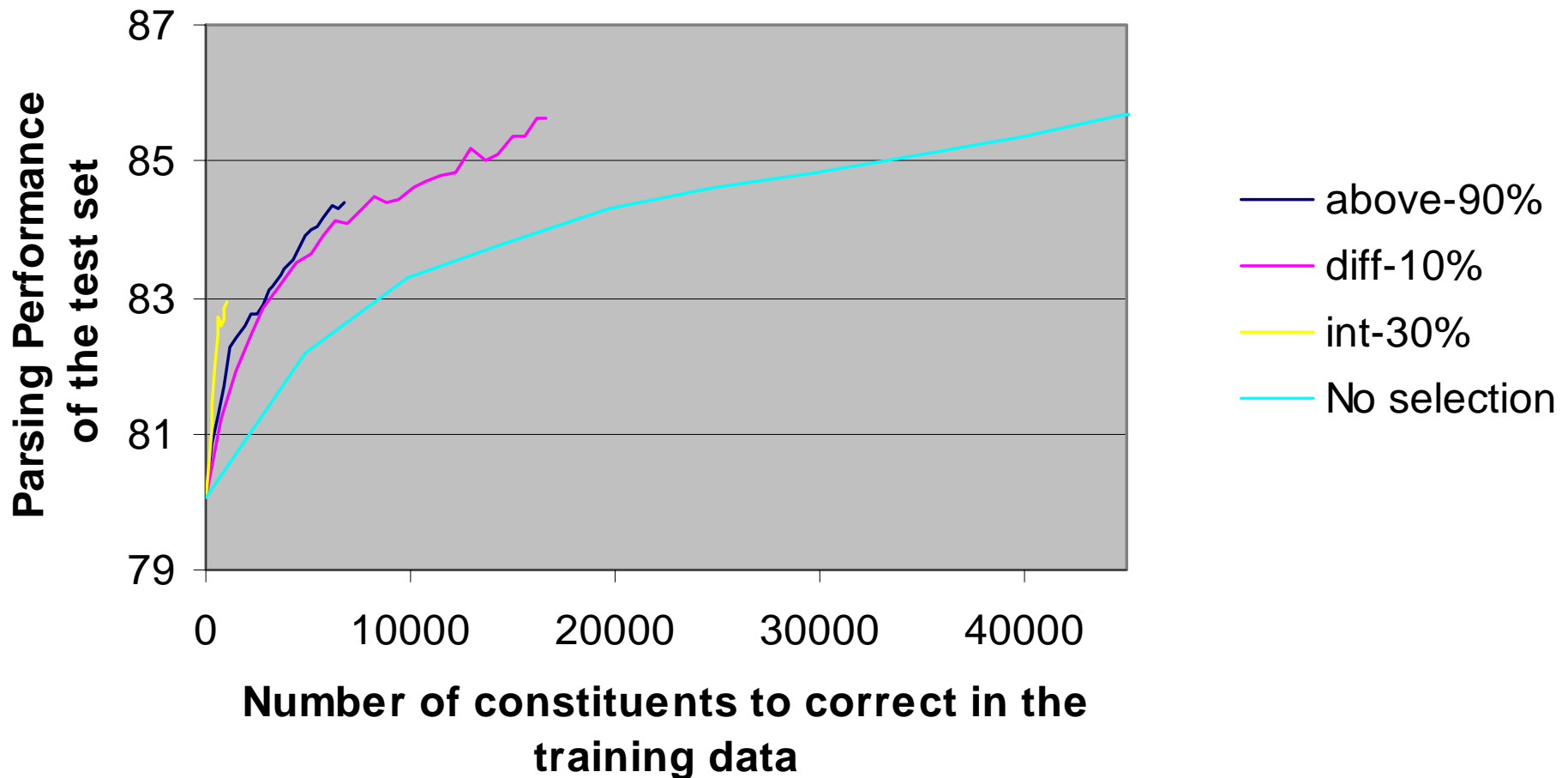
# Selection Methods and Corrected Co-Training

- Two scoring functions:  $f_{\text{F-score}}$ ,  $f_{\text{cprob}}$
- Three selection methods:  $S_{\text{above-}n}$ ,  $S_{\text{diff-}n}$ ,  $S_{\text{int-}n}$
- Balance between reviews and corrections
  - Maximize training utility: fewer sentences to review
  - Minimize error: fewer corrections to make
  - Better parsing performance?

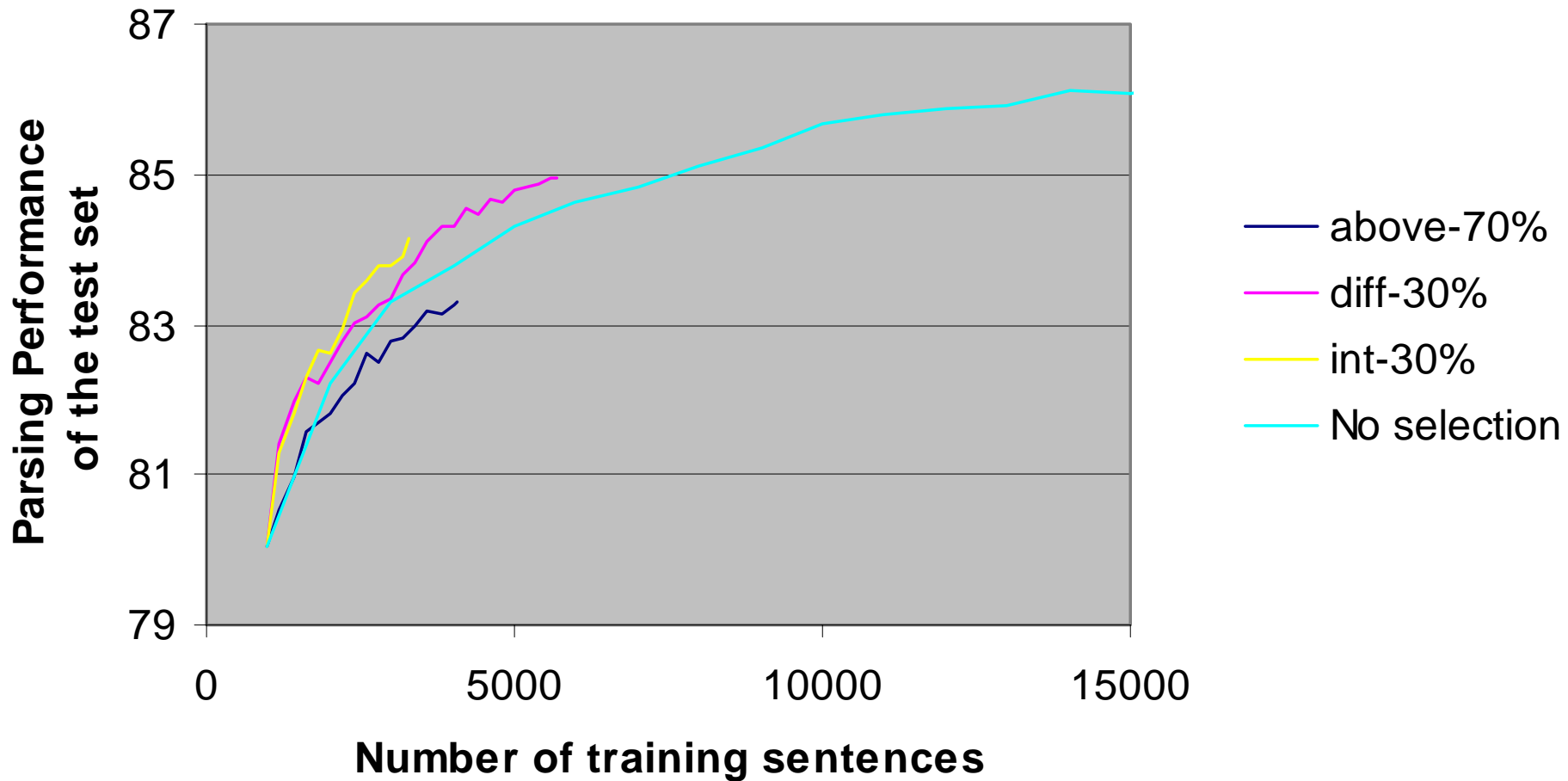
# Corrected Co-Training using $f_{F\text{-score}}$ (Reviews)



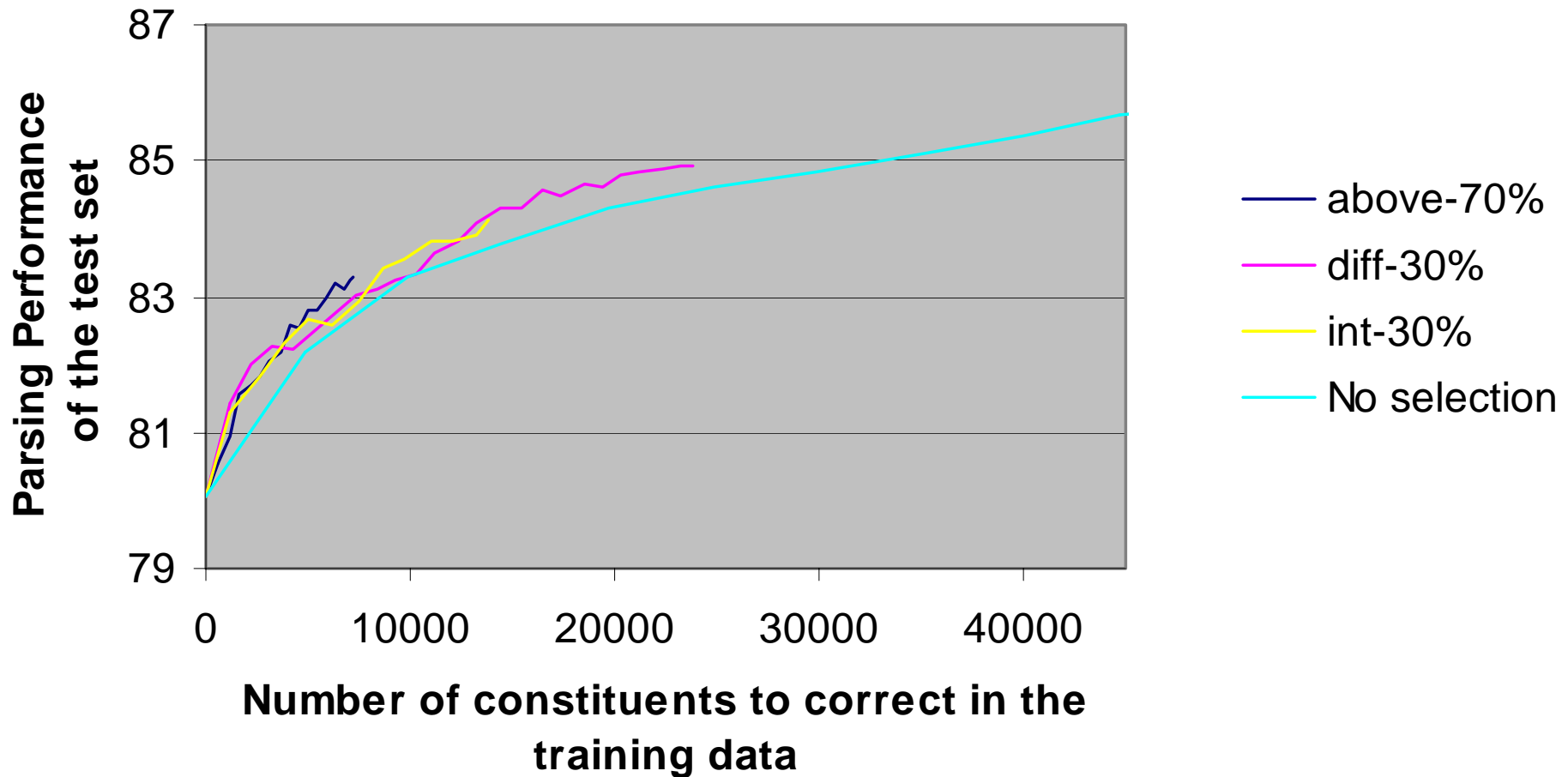
# Corrected Co-Training using $f_{F\text{-score}}$ (Corrections)



# Corrected Co-Training using $f_{\text{cprob}}$ (Reviews)



# Corrected Co-Training using $f_{\text{cprob}}$ (Corrections)



# Conclusion

- Sample selection
  - Tree-entropy reduces the number of training examples by 35% and the number of labeled constituents by 23%.
- Co-training and corrected co-training
  - Selection methods that use **multiple views** improve learning
  - Selection methods need to balance (often conflicting) criteria
    - Maximize training utility
    - Minimize error
  - Maximizing training utility is beneficial even at the potential cost of reducing training set accuracy

# Further Directions

- Machine learning methods for parsing
  - Better understanding of relationships between different learning techniques
  - Scoring functions for sample selection and co-training
  - Selection methods for co-training
  - Interaction with human in supervised training
- Applications of parsing: multilingual language processing
  - Word alignment
  - Structural correspondences
  - Machine translation