Semi-Supervised Training for Statistical Parsing

Mark Steedman, Steven Baker, Jeremiah Crim,
Stephen Clark, Julia Hockenmaier, Rebecca Hwa,
Miles Osborne, Paul Ruhlen, Anoop Sarkar
(Edinburgh, Penn, UMD, JHU, Cornell)

August 21, 2002

Introduction

Mark Steedman

The Team

Name	E-Mail	Affiliation
Steedman, Mark	steedman@cogsci.ed.ac.uk	University of Edinburgh
Sarkar, Anoop	anoop@linc.cis.upenn.edu	University of Pennsylvania
Osborne, Miles	osborne@cogsci.ed.ac.uk	University of Edinburgh
Hwa, Rebecca	hwa@umiacs.umd.edu	University of Maryland
Clark, Stephen	stephenc@cogsci.ed.ac.uk	University of Edinburgh
Hockenmaier, Julia	julia@cogsci.ed.ac.uk	University of Edinburgh
Ruhlen, Paul	ruhlen@cs.jhu.edu	Johns Hopkins University
Baker, Steven	sdb22@cornell.edu	Cornell University
Crim, Jeremiah	jcrim@jhu.edu	Johns Hopkins University

Associates

Name	E-Mail	Affiliation
John Henderson	jhndrsn@mitre.org	Mitre Corp.
Chris Callison-Burch	callison-burch@ed.ac.uk	Stanford/Edinburgh

Statistical parsing: the State of the Art

- Parsers trained on the Penn Treebank have improved rapidly . . .
- . . . and have been widely and usefully applied.
- Generalizing to other genres of text and other languages is highly desirable.
- But we will probably never get another 1M words of **any** kind of human-annotated text
- Automatically parsed text such as the **BLLIP Parsed Corpus** (30M words) is too noisy.
- **Co-training** has been shown to support bootstrapping respectable performance from **small labeled corpora** (Sarkar 2001)

What is Co-training?

- **Co-Training** (Blum and Mitchell 1998) is a weakly supervised method for **bootstrapping** a model from **a (small) seed set of labeled examples**, augmented with **a much larger set of unlabeled examples** by exploiting redundancy among multiple statistical models.

- A **set of models** is trained on the same labeled seed material.

- The whole set of models is then **run on a larger amount of unlabeled data**. Novel parses from any parser that are deemed **reliable** under some measure are used as **additional labeled data** for retraining the other models.

What is Co-training? (Contd.)

- Co-training can be thought of as seeking to optimize an **objective function** that measures the degree of agreement between the predictions for the unlabeled data based on the two views (see below).

- Crucially, **Co-Training** is training on **others'** output, in contrast to **Self-training**, or training on own output (cf. Charniak 1997).

- The theory of co-training has been developed in application to **classifiers**. The present project is an empirical study of its applicability to **parsers**, following (Sarkar 2001)

Potential Payoffs are Large

- **Payoff 1:**
  - Improved performance of **Wide-Coverage Parsers**
- **Payoff 2:**
  - Methods for building **Very Large Treebanks** (larger and less noisy than BLLIP) Useful for numerous speech/language applications
- **Payoff 3:**
  - Methods for bootstrapping parsers for **novel genres** and **new languages** from **small labeled datasets**

Co-training: Project Details

- **Three Distinct Parsers**
  - **Treebank CFG** parsers (Collins 1999, 2000, . . .)
  - **LTAG** parser (Sarkar)
  - **CCG** parsers (and CCG supertaggers) (Clark, Hockenmaier)
- **Approaches**
  - **Co-training** with **supertaggers** and **parsers** (CCG)
  - **Co-training** with **different parsers** (CFG and LTAG)
  - **Co-training Rerankers** (Collins 2000)
- **Data:**
  - Use **labeled** WSJ (**1M** words) and Brown (**440K** words) Penn Treebank
  - Then use **unlabeled** North American News Text corpus: **500M** words

Workshop Goals

- Identify criteria for **Parse Selection** that exclude noise and maximize co-training effects;
- Explore the way co-training effects **depend on size of labeled seed set**;
- Explore effectiveness of co-training for **porting parsers to new genres** by **training on Brown** Corpus, **co-training** on unlabeled WSJ and held-out PT-WSJ labeled secn. 00, ultimately evaluating on PT-WSJ secn. 23.
- Explore effectiveness of co-training on parsers **trained** on all of PT-WSJ 2-22, **co-trained** on unlabeled WSJ and labeled secn. 00, ultimately evaluating on secn. 23.

What We Have to Show

- We will show that **co-training enhances performance** for parsers and taggers trained on **small (500—10,000 sentences) amounts of labeled data**.
- We will also show that **co-training can be used for porting** parsers trained on one genre to parse on another **without any new human-labeled data at all**, improving on state-of-the-art for this task.
- We will also show that **even tiny amounts of human-labelled data** for the target genre enhace porting via co-training.
- We will show a number of preliminary results on ways to deliver similar improvements for **parsers trained on large (Penn WSJ Treebank) labeled datasets** and **expressive grammars**.
- These include a number of novel methods for **Parse Selection**.

Outline

• Data Management	(5 min, Steven Baker)
• Parse Selection	(15 min, Rebecca Hwa)
• Experiments I: Collins-CFG/LTAG	(25 min, Anoop Sarkar)
• Experiments II: CCG/CCG-Supertagger	(30 min, Stephen Clark)
• Student Presentation: Oracle experiments	(10 min, Steven Baker)
• **Coffee Break**	(30min)
• Student Proposal:Smoothing for Co-training	(15 min, Julia Hockenmaier)
• Experiments III: Rerankers	(10 min, Jeremiah Crim)
• Student Proposal: Cotraining Rerankers	(5 min, Jeremiah Crim)
• Conclusions	(5 min, Miles Osborne)
• **Questions and Discussion**	(30 min)

Co-training Architecture and Parse Selection

Steve Baker, Rebecca Hwa, and Paul Ruhlen

Co-training: The Algorithm

- Re uires:
  - Two learners with different **views** of the task
  - Cache Manager (CM) to interface with the disparate learners
  - A small set of **labelled seed data** and a larger pool of **unlabelled data**

- Pseudo-Code:
  - **Init:** Train both learners with labelled seed data
  - **Loop:**
    - CM picks unlabelled data to add to cache
    - Both learners label cache
    - CM selects newly labelled data to add to the learners respective training sets
    - Learners re-train

Co-training: Theoretical Considerations

- Co-training works (provably) when the views are **independent**.
  - Blum and Mitchell, 1998
  - Nigam and Ghani, 2000
  - Dasgupta et al., 2001
- To apply theory to practice, we would like to explicitly optimise an **objective function** that measures the degree of agreement between the predictions for the unlabelled data based on the two views.

Parse Selection: Practical Heuristics

- Computing the objective function is impractical.

- We use **parse selection heuristics** to determine which labelled examples to add.
  - Each parser assigns a **score** for every sentence, estimating the goodness of its best parse.
  - Cache Manager uses these scores to select the newly labelled data for both parsers according to some **selection method**.

Scoring Phase

- Ideally, we want the parser to return its true accuracy for the parses it produced (e.g., *parseval scores*).
- In practice, we can approximate these scores with con dence metrics.
  - log probability of the best parse
  - entropy of the n-best parses (tree-entropy)
  - number of parses, sentence length
  - . . .

Selection Phase

- Want to select training examples for one parser (student) labelled by the other (teacher) so as to **minimise noise** and **maximise training utility**.
  - **Top-n:** Choose the n examples for which the teacher assigned the highest scores.
  - **Difference:** Choose the examples for which the teacher assigned a higher score than the student by some threshold.
  - **Intersection:** Choose the examples that received high scores from the teacher but low scores from the student.
  - **Disagreement:** Choose the examples for which the two parsers provided different analyses and the teacher assigned a higher score than the student.

Two Pilot Studies

- What scoring function might work well
  - Experiment with different scoring functions while using the same selection method.
- What selection method might work well
  - Experiment with different selection method while using the same scoring method.

xperimental Setup

- Unlabelled training data: derived from Penn Treebank WSJ sec02-21.
- Cache size: 50 sentences per iteration.
- Development test set: WSJ sec00.
- Evaluation metric: parseval
  - Precision (P) $\frac{\text{of correctly guessed labelled constituents}}{\text{of guessed labelled consituents}}$
  - Recall (R) $\frac{\text{of correctly guessed labelled constituents}}{\text{of labelled constituents in gold standard}}$
  - F2 $\frac{2 \times P \times R}{P + R}$

Scoring Function Variation Study

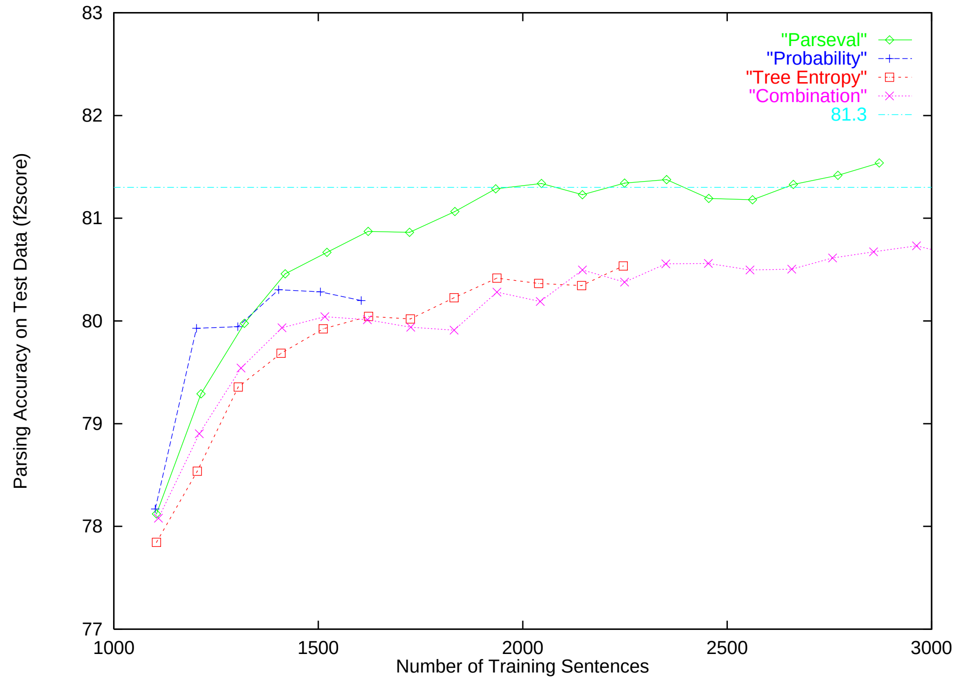
- Setting: Two versions of the same parser trained on different initial labelled seed data (1000 sentence subsets of Penn Treebank; resp. w/o conjunctions, and w/o prepositions).

- Scoring functions: *parseval*, best parse probability, tree-entropy, combination.

- Selection method: intersection, picks 20 sentences per iteration.

Scoring Function Variation Learning Curve

- All practical scoring functions performed similarly

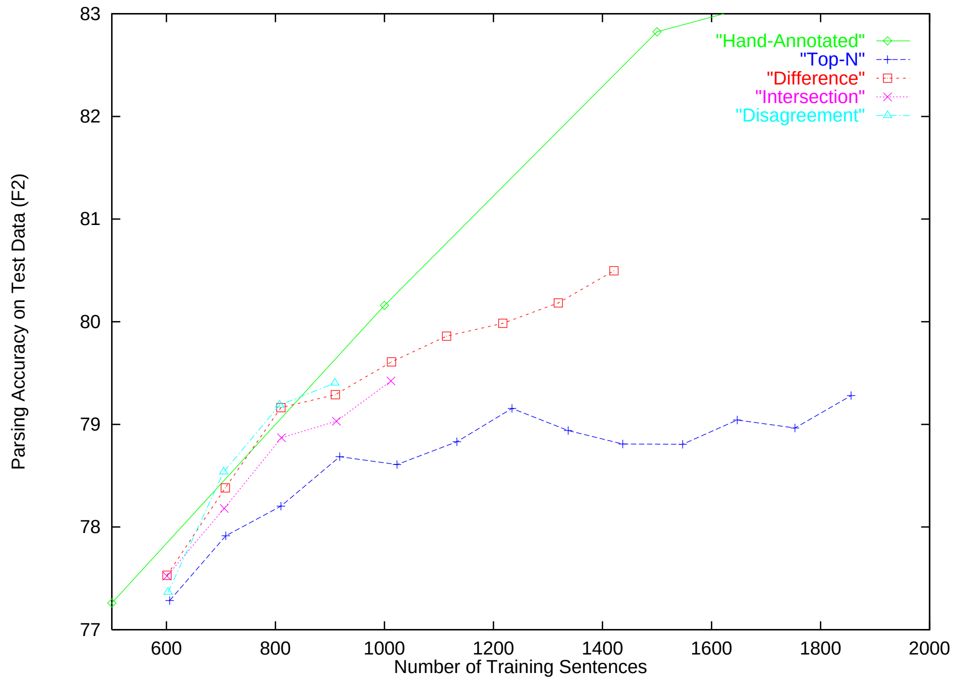


Selection Method Variation Study

- Setting: Two different parsers (CFG and LTAG) trained on the same initial labelled seed data (500, 1000 sentences from Penn Treebank)
- Scoring function: $parseval$
- Selection methods: Top-n, Difference, Intersection, Disagreement

Selection Method Variation (500 examples)

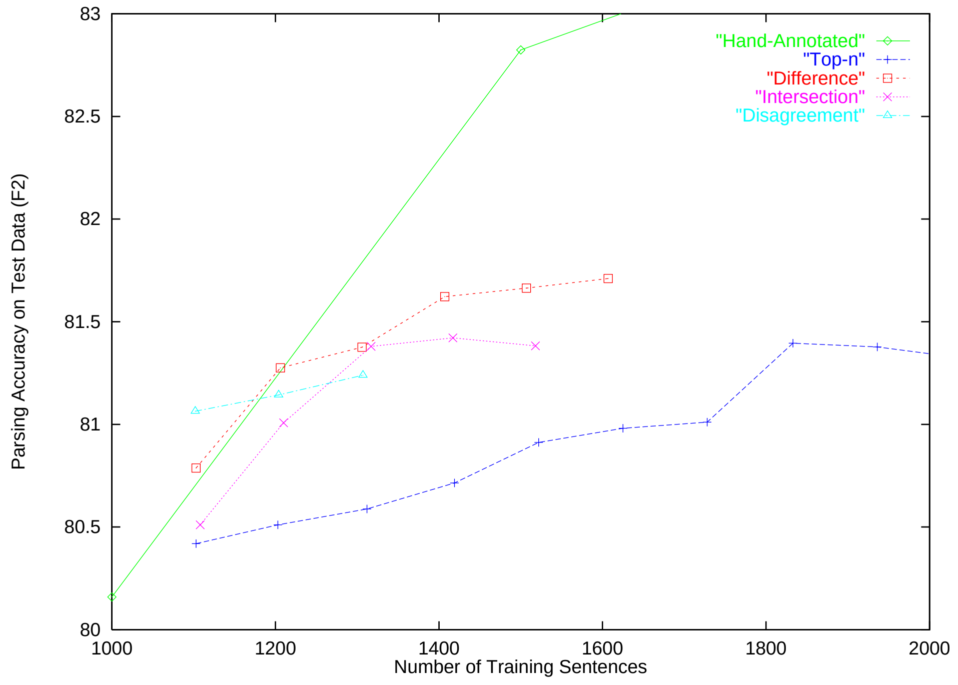
- Smart selection methods do better than Top- n
- None is as good as human annotated data



Selection Method Variation (1000 examples)

- Similar trend for experiment with larger initial labelled set

- Disparity between selection methods and hand-annotated upper-bound is bigger



Summary

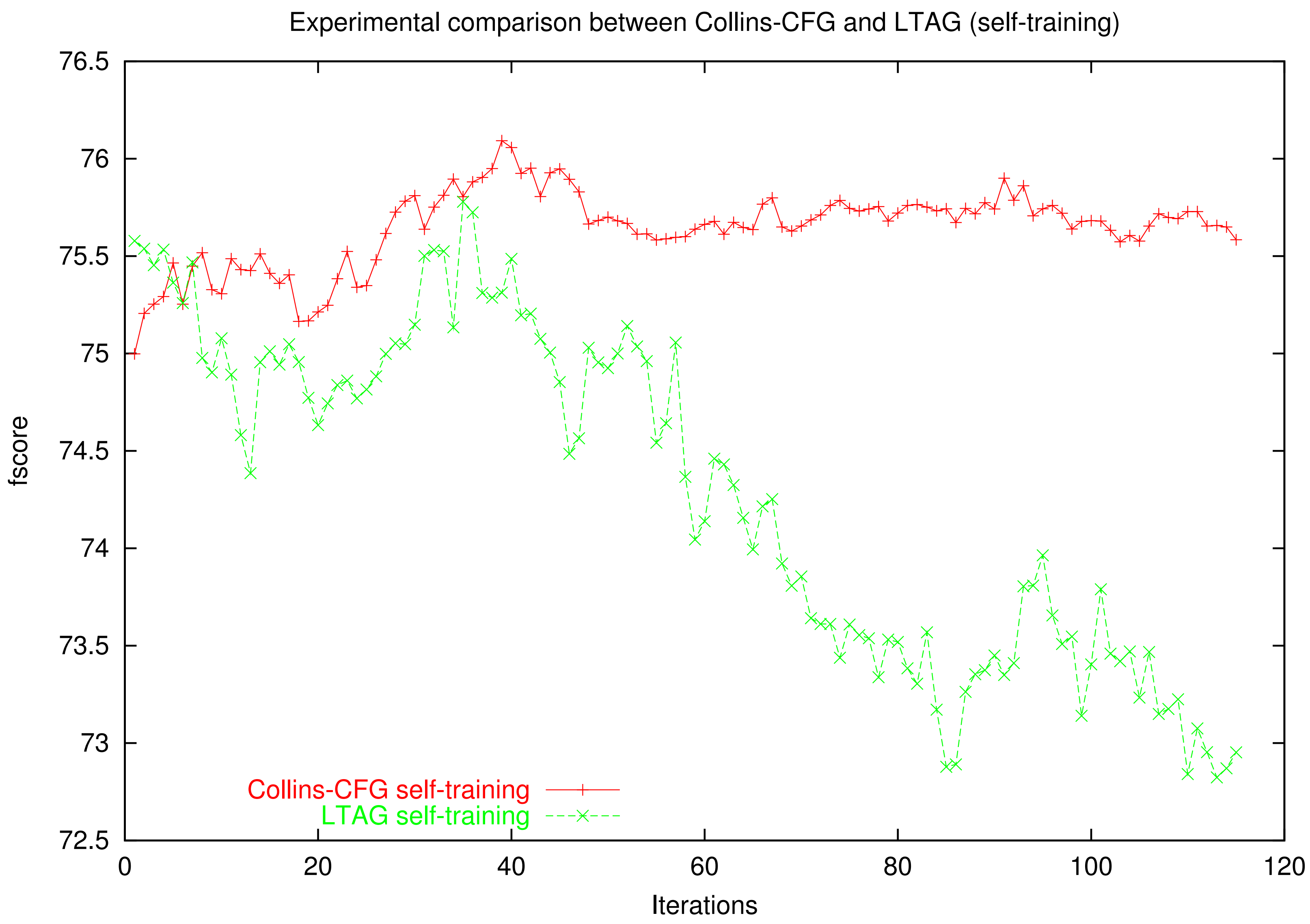
- Use parse selection heuristics to approximate **objective function**.
- Scoring function differences are inconclusive – use log probability for large-scale experiments.
- Selection methods that use scores from both parsers (Difference, Intersection, Disagreement) seem more reliable.

Co-training between Collins-CFG and LTAG

Rebecca Hwa, Miles Osborne and Anoop Sarkar

Collins-CFG and LTAG: Different Views for Co-Training

Collins-CFG	LTAG
Bi-lexical dependencies are between nonterminals	Bi-lexical dependencies are between elementary trees
Can produce novel elementary trees for the LTAG	Can produce novel bi-lexical dependencies for Collins-CFG
Learning curves show convergence on 1M words labeled data	Learning curves show LTAG needs relatively more labeled data
When using small amounts of seed data, abstains less often than LTAG	When using small amounts of seed data, abstains more often than Collins-CFG


Experimental comparison between Collins-CFG and LTAG (self-training)
fscore
Iterations
76.5
76
75.5
75
74.5
74
73.5
73
72.5
0
20
40
60
80
100
120
Collins-CFG self-training
LTAG self-training

Workshop Goals

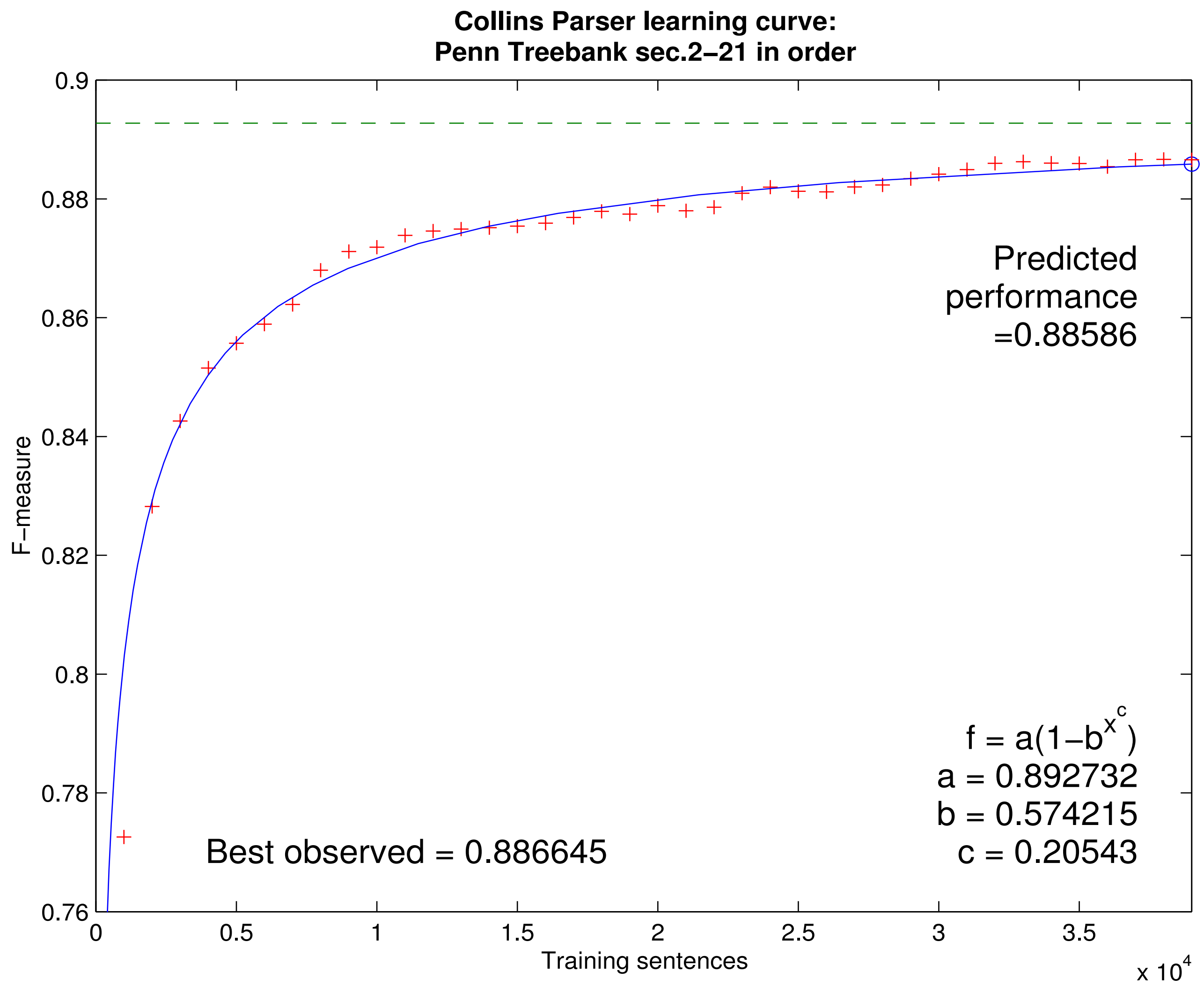
- Use co-training to boost performance, when faced with small seed data
  $\longrightarrow$ Use small subsets of WSJ labeled data as seed data

- Use co-training to port parsers to new genres
  $\longrightarrow$ Use Brown corpus as seed data, co-train and test on WSJ

- Use a large set of labeled data and use unlabeled data to improve parsing performance
  $\longrightarrow$ Use Penn Treebank (40K sents) as seed data

Experiments on Small Labeled Seed Data

- Motivating the size of the initial seed data set
- We plotted learning curves, tracking parser accuracy while varying the amount of labeled data
- Find the “elbow” in the curve where the payoff will occur
- This was done for both the Collins-CFG and the LTAG parser



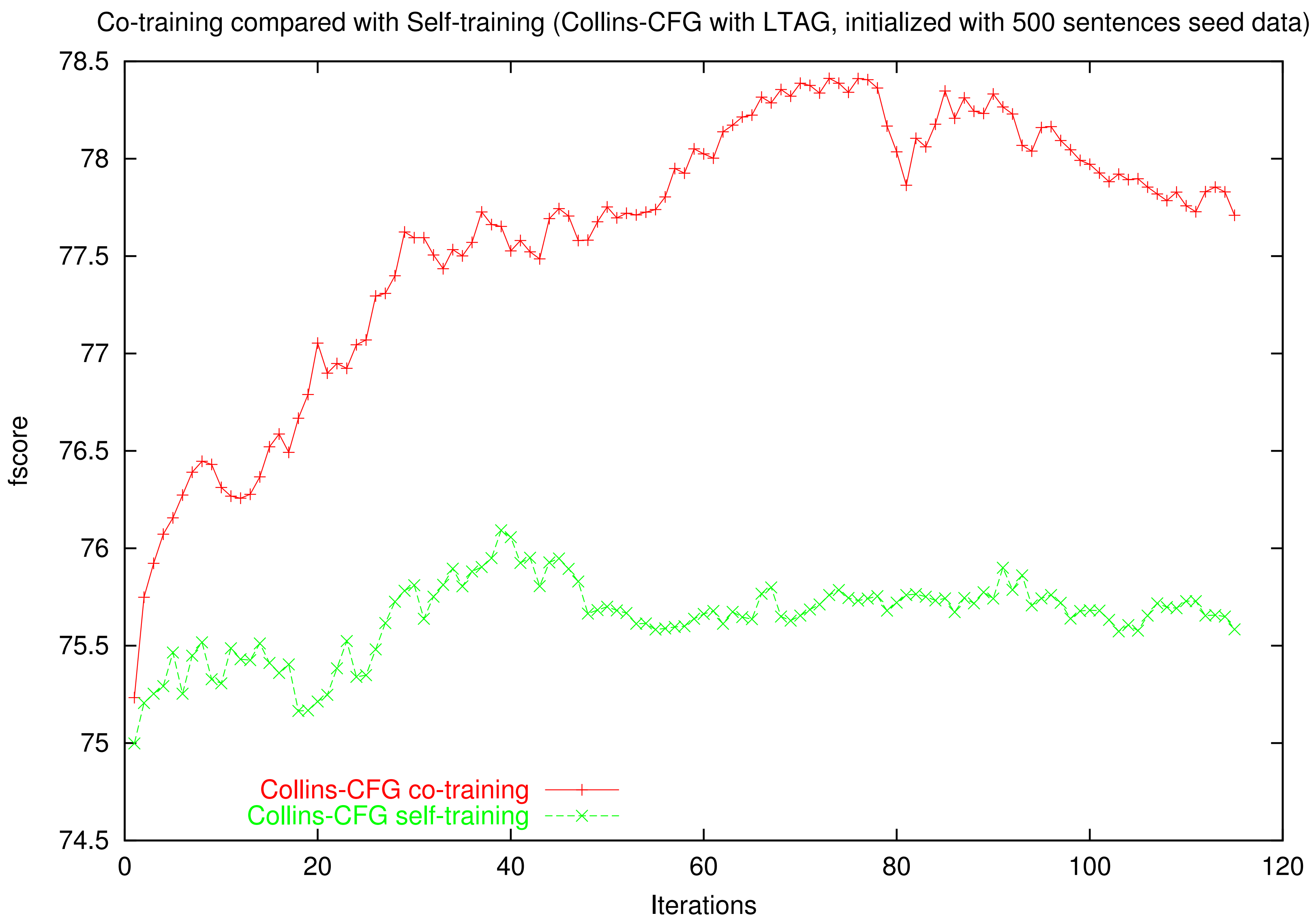
Upper bound = 89.3% Projected to Treebank of size 80K sentences = 88.9%   400K = 89.2%

Workshop Goals

- Use co-training to boost performance, when faced with small seed data
  $\longrightarrow$ Use 500 sentences of WSJ labeled data as seed data
  $\longrightarrow$ Compare performance of co-training vs. self-training

- Use co-training to port parsers to new genres
  $\longrightarrow$ Use Brown corpus as seed data, co-train and test on WSJ

- Use a large set of labeled data and use unlabeled data to improve parsing performance
  $\longrightarrow$ Use Penn Treebank (40K sents) as seed data

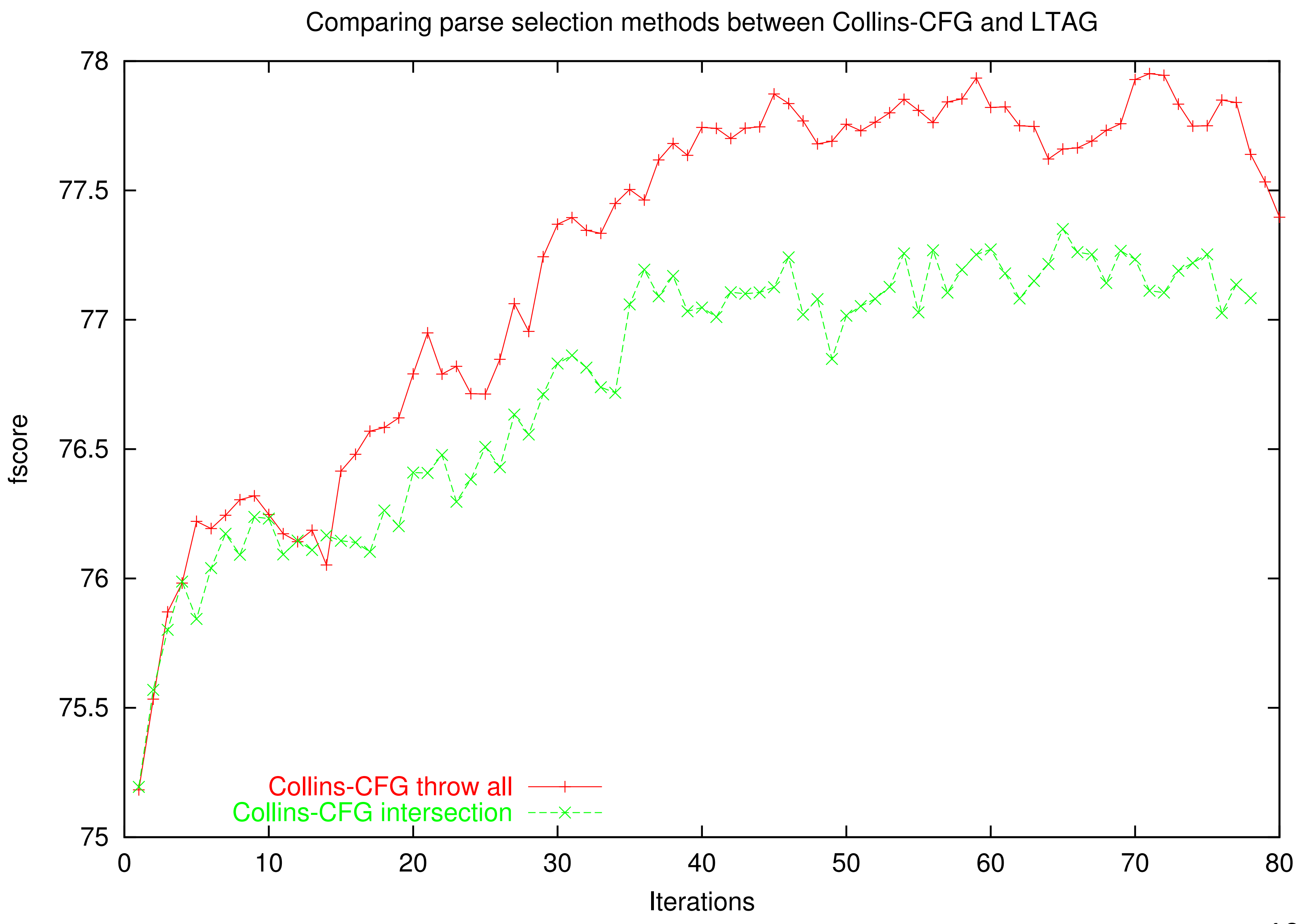

Co-training compared with Self-training (Collins-CFG with LTAG, initialized with 500 sentences seed data)
fscore
Iterations
78.5
78
77.5
77
76.5
76
75.5
75
74.5
0
20
40
60
80
100
120
Collins-CFG co-training
Collins-CFG self-training

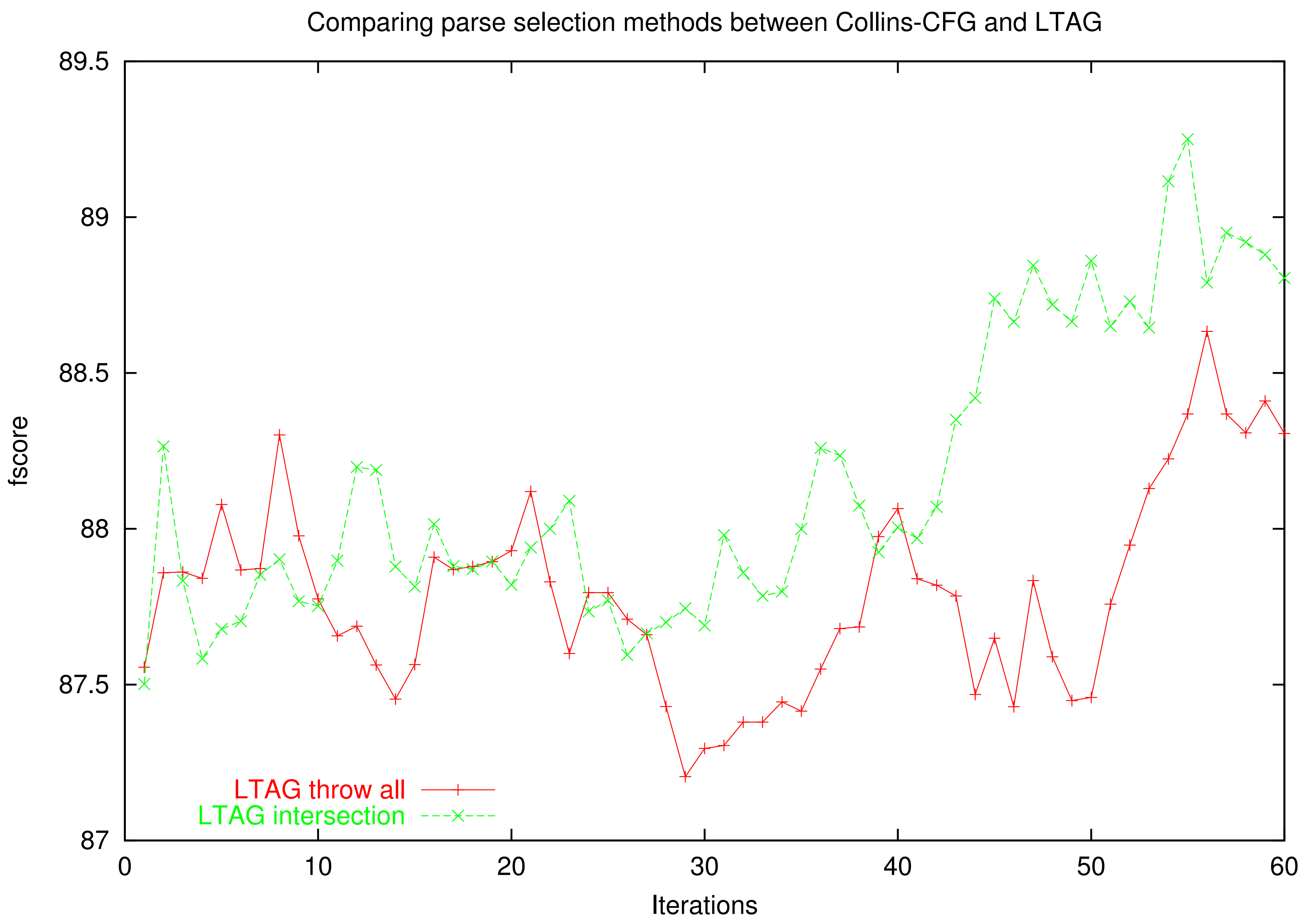
Workshop Goals

- Use co-training to boost performance, when faced with small seed data
  $\longrightarrow$ Co-training beats self-training with 500 sentence seed data
  $\longrightarrow$ Compare parse selection methods for different parser views

- Use co-training to port parsers to new genres
  $\longrightarrow$ Use Brown corpus as seed data, co-train and test on WSJ

- Use a large set of labeled data and use unlabeled data to improve parsing performance
  $\longrightarrow$ Use Penn Treebank (40K sents) as seed data


Comparing parse selection methods between Collins-CFG and LTAG
fscore
Iterations
78
77.5
77
76.5
76
75.5
75
0
10
20
30
40
50
60
70
80
Collins-CFG throw all
Collins-CFG intersection

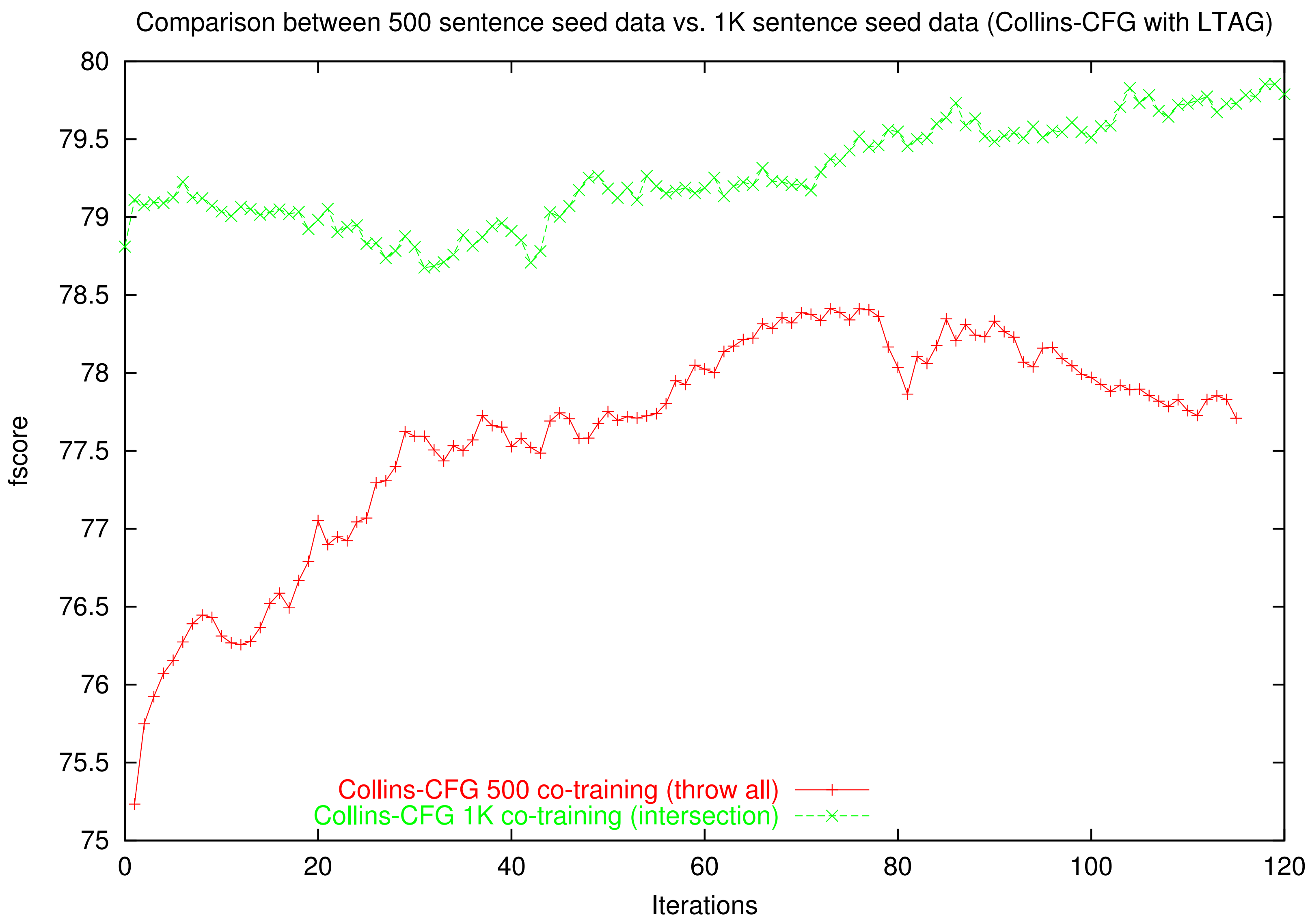

Comparing parse selection methods between Collins-CFG and LTAG
fscore
Iterations
89.5
89
88.5
88
87.5
87
0
10
20
30
40
50
60
LTAG throw all
LTAG intersection

Workshop Goals

- Use co-training to boost performance, when faced with small seed data
  $\longrightarrow$ Co-training beats self-training with 500 sentence seed data
  $\longrightarrow$ Different parse selection methods better for different parser views
  $\longrightarrow$ Compare performance when seed data is doubled to 1K sentences

- Use co-training to port parsers to new genres
  $\longrightarrow$ Use Brown corpus as seed data, co-train and test on WSJ

- Use a large set of labeled data and use unlabeled data to improve parsing performance
  $\longrightarrow$ Use Penn Treebank (40K sents) as seed data

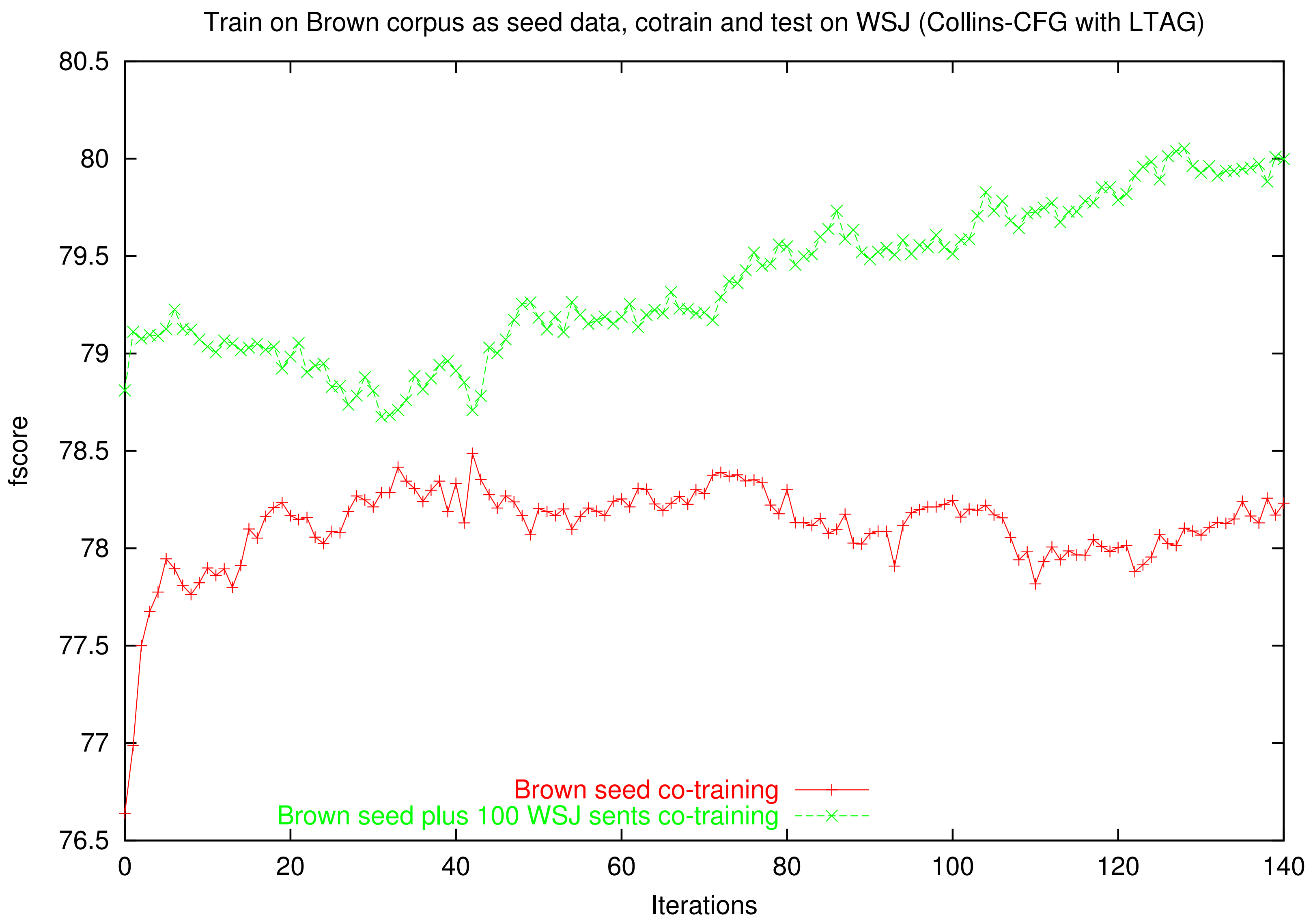

Comparison between 500 sentence seed data vs. 1K sentence seed data (Collins-CFG with LTAG)
fscore
Iterations
75
75.5
76
76.5
77
77.5
78
78.5
79
79.5
80
0
20
40
60
80
100
120
Collins-CFG 500 co-training (throw all)
Collins-CFG 1K co-training (intersection)

Workshop Goals

- Use co-training to boost performance, when faced with small seed data
  $\longrightarrow$ Co-training beats self-training with 500 sentence seed data
  $\longrightarrow$ Different parse selection methods better for different parser views
  $\longrightarrow$ Co-training still improves performance with 1K sentence seed data

- Use co-training to port parsers to new genres
  $\longrightarrow$ Use Brown corpus as seed data, co-train and test on WSJ

- Use a large set of labeled data and use unlabeled data to improve parsing performance
  $\longrightarrow$ Use Penn Treebank (40K sents) as seed data


Train on Brown corpus as seed data, cotrain and test on WSJ (Collins-CFG with LTAG)
fscore
80.5
80
79.5
79
78.5
78
77.5
77
76.5
0
20
40
60
80
100
120
140
Iterations
Brown seed co-training
Brown seed plus 100 WSJ sents co-training

Workshop Goals

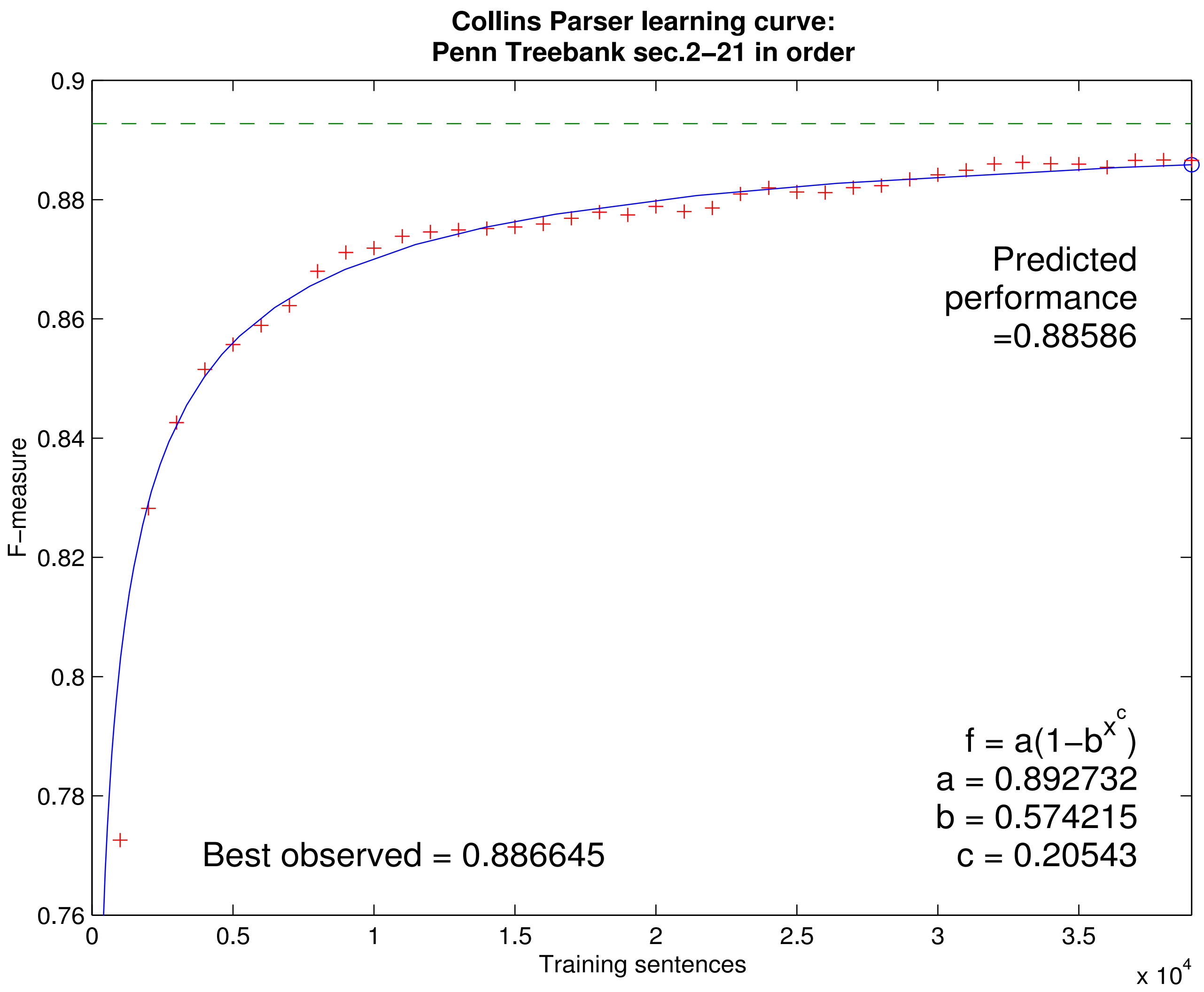
- Use co-training to boost performance, when faced with small seed data
  $\longrightarrow$ Co-training beats self-training with 500 sentence seed data
  $\longrightarrow$ Different parse selection methods better for different parser views
  $\longrightarrow$ Co-training still improves performance with 1K sentence seed data

- Use co-training to port parsers to new genres
  $\longrightarrow$ Co-training improves performance significantly when porting from one genre (Brown) to another (WSJ)

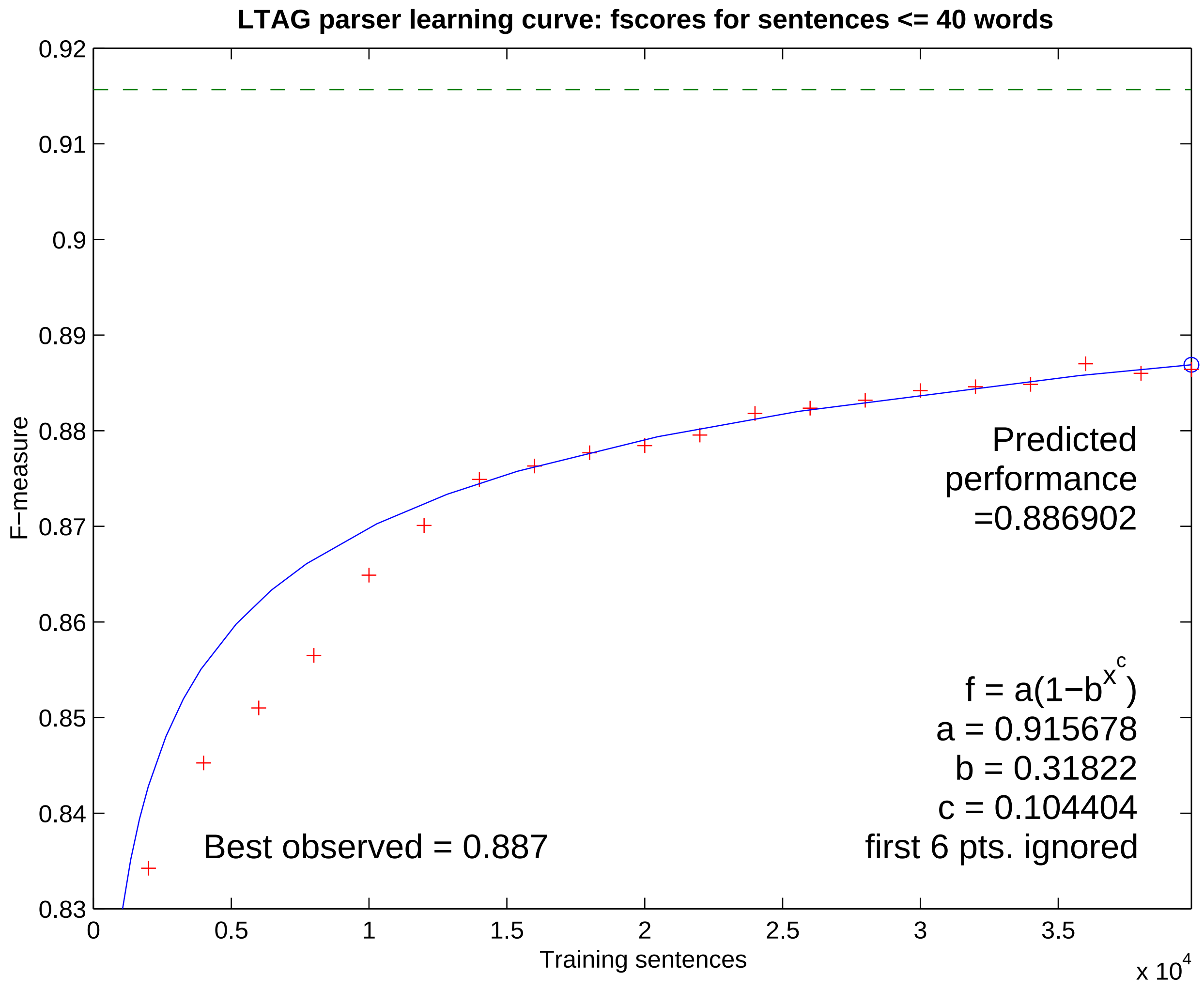
- Use a large set of labeled data and use unlabeled data to improve parsing performance
  $\longrightarrow$ Use Penn Treebank (40K sents) as seed data

Co-training using a large labeled seed set

- Experiments using 40K sentences Penn Treebank WSJ sentences as seed data for co-training did not produce a positive result
- Even after adding 260K sentences of unlabeled data using co-training did not significantly improve performance over the baseline
- However, we plan to do more experiments in the future which leverage more recent work on parse selection and the difference between the Collins-CFG and LTAG views



Upper bound = 89.3% Projected to Treebank of size 80K sentences = 88.9% 400K = 89.2%



Upper bound = 91.6% Projected to Treebank of size 80K sentences = 89.3%   400K = 90.4%

Summary

- Use co-training to boost performance, when faced with small seed data
  $\longrightarrow$ Co-training beats self-training with 500 sentence seed data
  $\longrightarrow$ Different parse selection methods better for different parser views
  $\longrightarrow$ Co-training still improves performance with 1K sentence seed data

- Use co-training to port parsers to new genres
  $\longrightarrow$ Co-training improves performance significantly when porting from one genre (Brown) to another (WSJ)

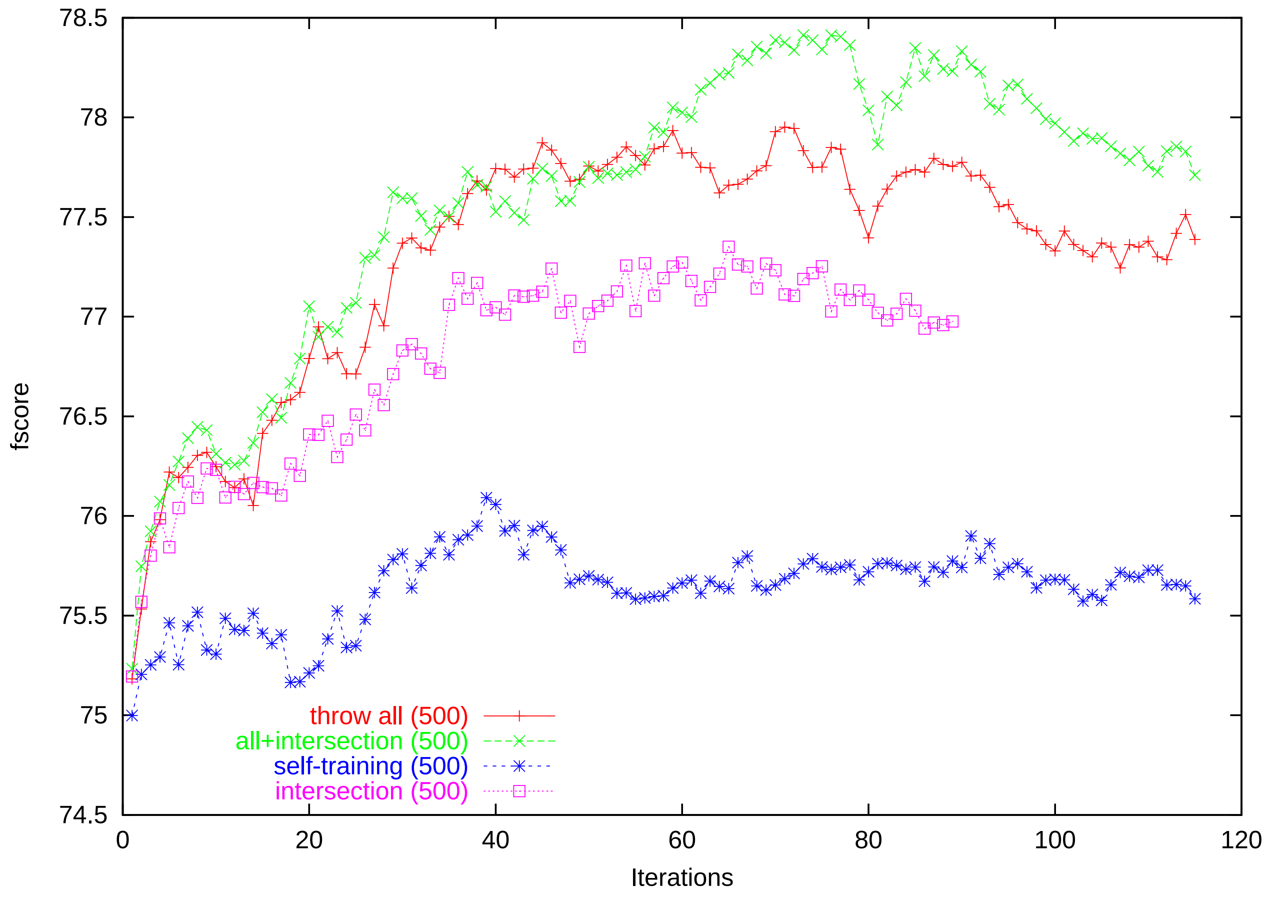
Summary

- Use a large set of labeled data and use unlabeled data to improve parsing performance
  $\longrightarrow$ Using 40K sentences of Penn Treebank as seed data showed no improvement over the baseline. Future work: improving LTAG performance

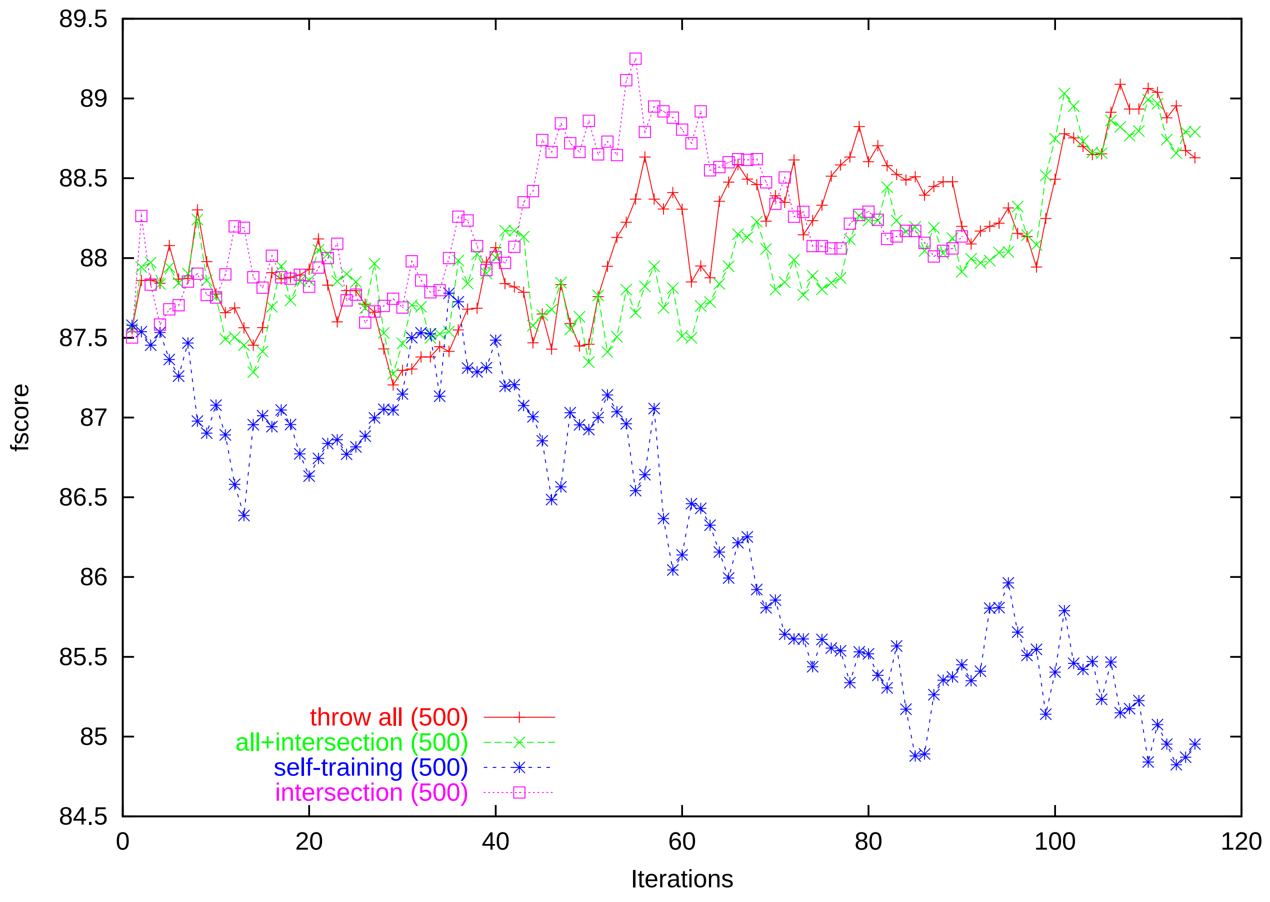
Summary of experiments done during the workshop

Seed Type	Seed Size	Improvement start	Improvement end	added/cache	total added	Upper Bound Gold	Upper Bound Oracle	Select
CFG WSJ	500	75	78	20/30	3000	85	80	all
	1000	79	80	20/30	3000	86	82	int
	all(40K)	88	88	1000/2000	260K	-	-	all
LTAG WSJ	500	87.5	89.5	20/30	3000	-	-	int
	1000	89	90	20/30	3000	-	-	int
	all(40K)	-	-	-	-	-	-	-
CFG Brown	1000	75	78	20/30	3000	-	81	all
	all(24K)	80.7	80.8	200/300	-	-	-	int
CFG Brown +100 WSJ	1000	77	79	20/30	3000	-	82	int
	all(24K)	-	-	-	-	-	-	-

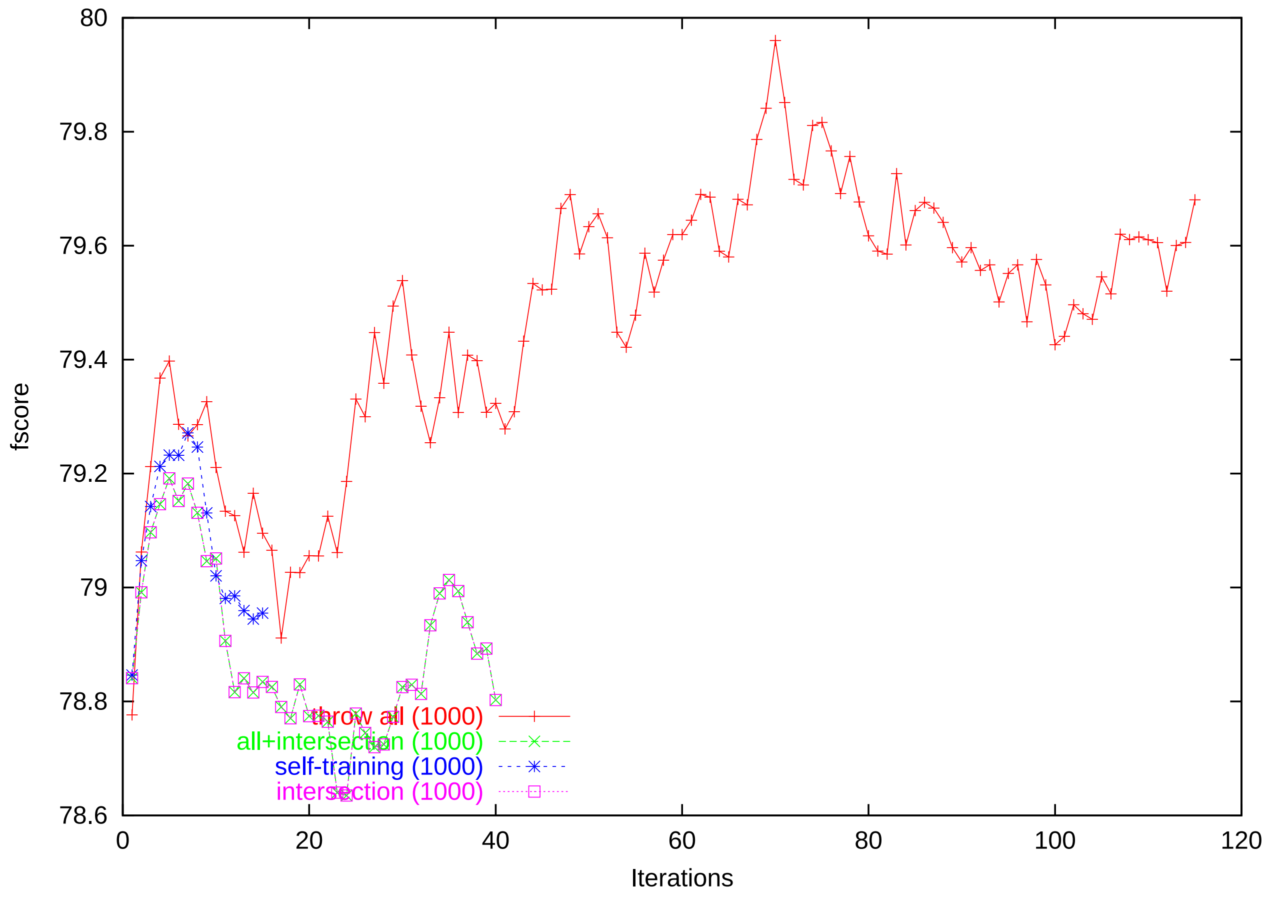
collins runs1–4



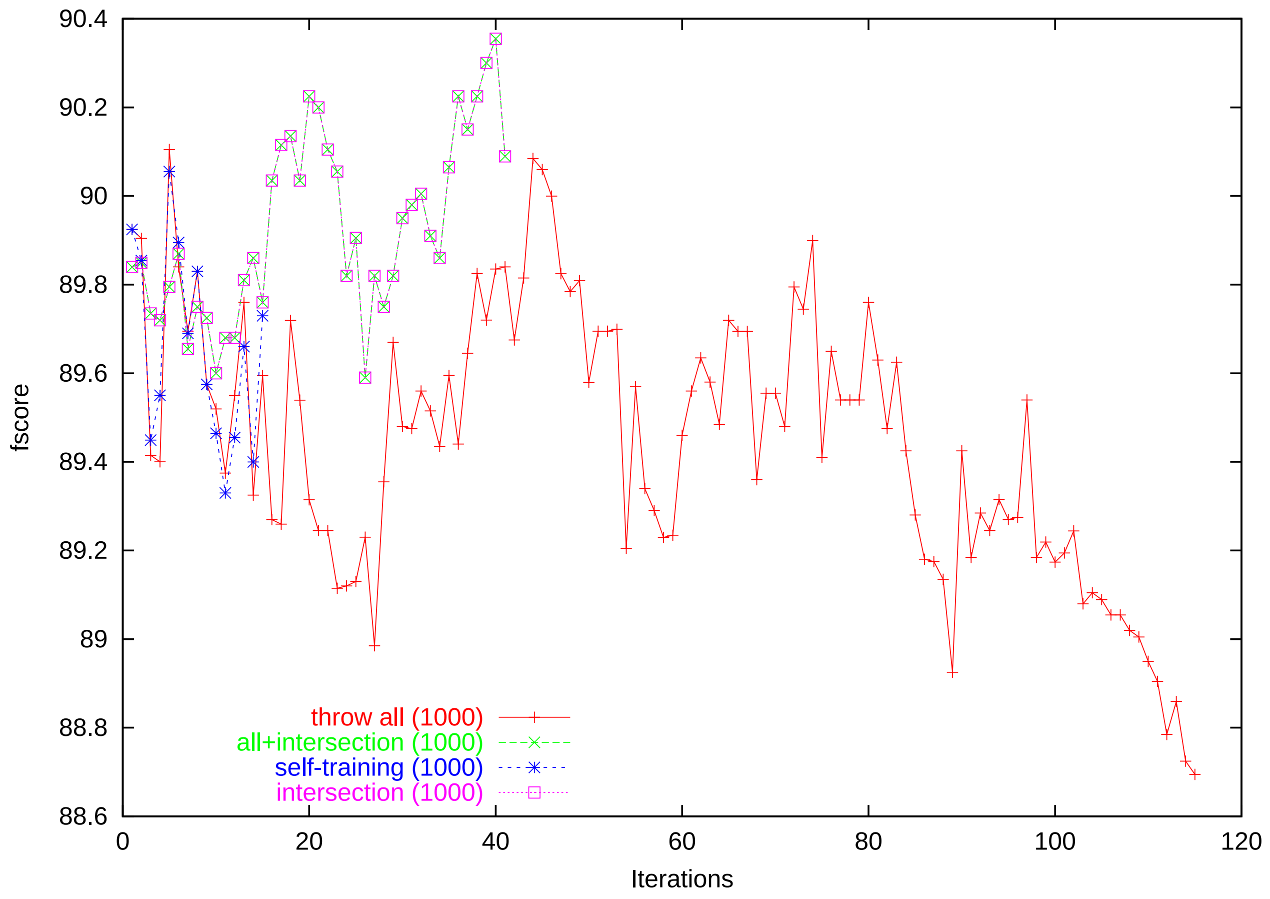
tagparser runs1–4



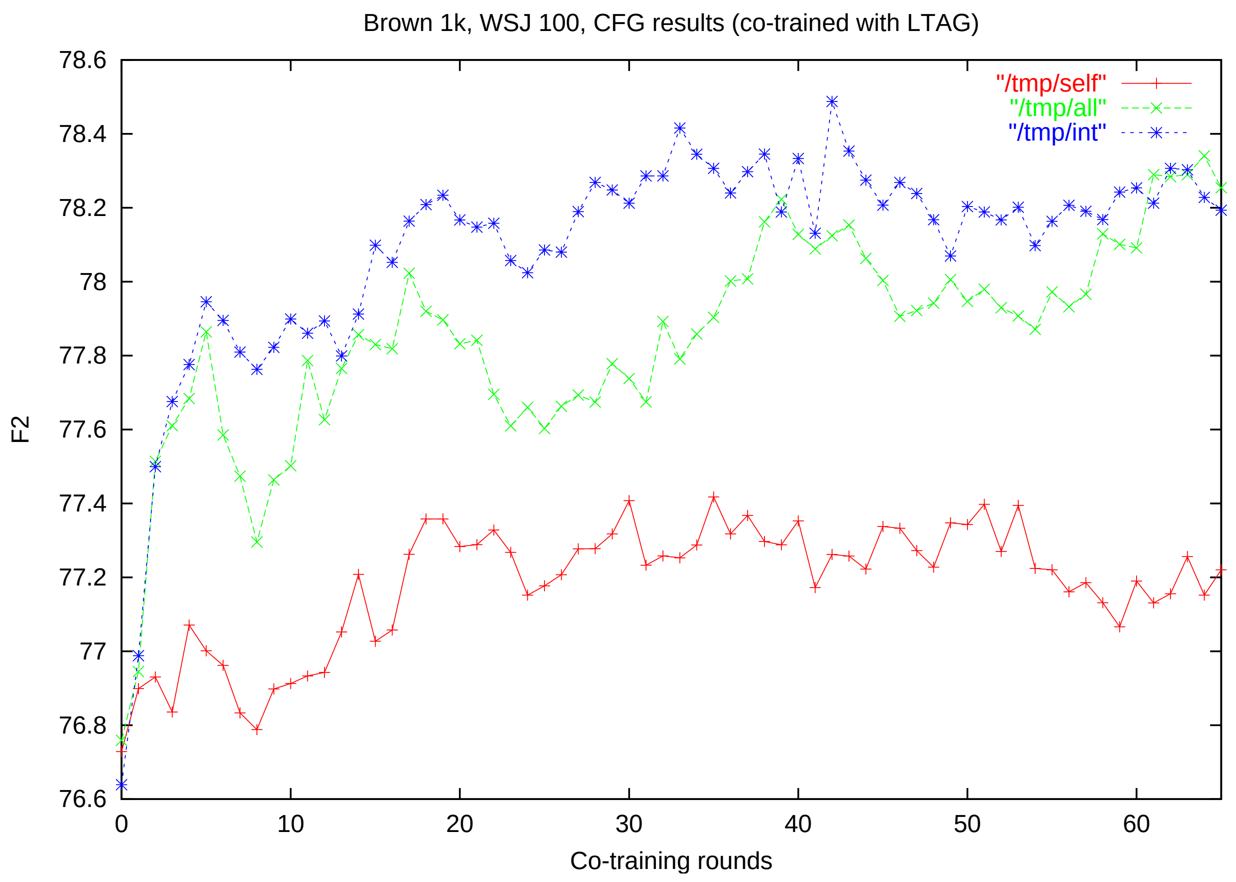
collins runs5–8



tagparser runs5–8



brown-wsj



Brown 1k, WSJ 100, CFG results (co-trained with LTAG)
"/tmp/self"
"/tmp/all"
"/tmp/int"
F2
78.6
78.4
78.2
78
77.8
77.6
77.4
77.2
77
76.8
76.6
0
10
20
30
40
50
60
Co-training rounds

Experimental Setup – Evaluating the Collins-CFG parser

- Various starting seed human labeled data sets were used (WSJ 40K, 500, 1K sents, Brown 1K, 24K sents)
- Evaluation of the newly trained model at each iteration of co-training was done on the entire Section 00 of the Penn Treebank (1921 sentences)
- The fscore graphs we show are plotted for each iteration of the co-training algorithm
- We also show self-training results where the output of the Collins-CFG parser was used to bootstrap new labeled data

Experimental Setup – Evaluating the LTAG parser

- Various starting seed human labeled data sets were used (WSJ 500, 1K sentences)

- Evaluation of the newly trained model at each iteration of co-training was done on a 250 sentence subset ($\leq$ 15 words) of the Section 00 of the Penn Treebank

- We also show self-training results where the output of the LTAG parser was used to bootstrap new labeled data

Summary: Collins-CFG co-training with LTAG

- Positive cotraining effect has been obtained for the Collins-CFG parser using output of the LTAG parser

- Training on 500-sentence and 1000-sentence seed sets and selection techniques including intersection

- Full set selection works better for very small seed data: 500 sentences

Summary: LTAG co-training with Collins-CFG

- Positive cotraining effect has been obtained for the LTAG parser using output of the Collins-CFG parser

- Training on 500-sentence and 1000-sentence seedsets and selection techniques including intersection

- Selection techniques like Intersection work better than full set selection

Co-training with Combinatory Categorial Grammar (CCG)

Steve Baker, Stephen Clark, Jay Crim, Julia Hockenmaier
Paul Ruhlen, Mark Steedman

August 2002

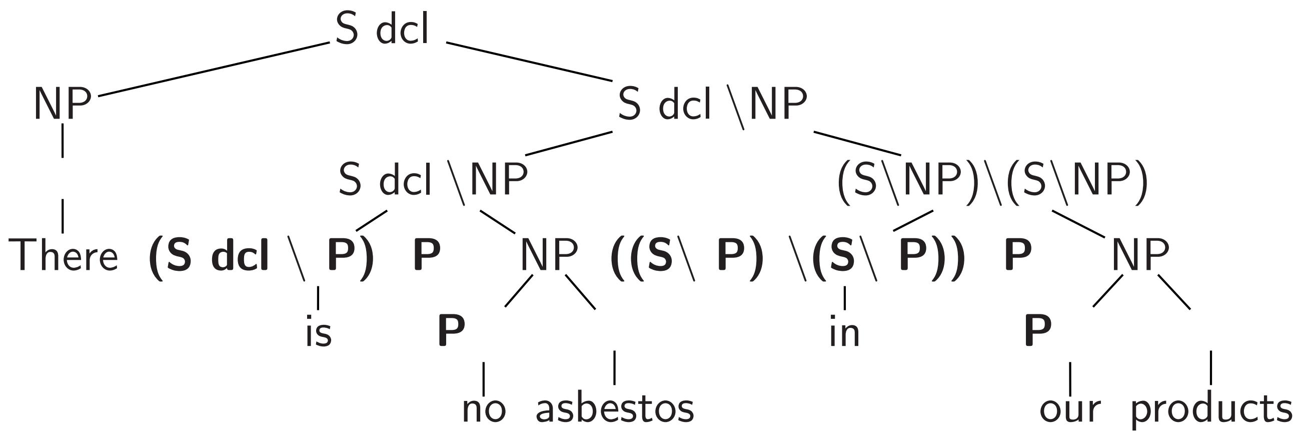
Outline

- Brief introduction to CCG
- Possible views for CCG co-training
- Co-training experiments
- Oracle experiments

Combinatory Categorial Grammar (CCG)

- CCG is a **mildly context-sensitive**, **lexicalised** grammar formalism
- CCG is based on **categories**
  - atomic categories: S, N, NP, PP
  - complex categories: $(S\backslash NP)/NP$
- Categories are combined using a small number of **combinatory rules**
  - example of **function application**: $(S\backslash NP)/NP \quad NP \rightarrow S\backslash NP$
- Combinatory rules allow unbounded dependencies to be captured for wide-coverage parsing (Clark et al., Hockenmaier and Steedman, ACL 2002)

A CCG derivation tree



- CCG trades
  - lexical category types (≈1,200 compared with ≈50 Penn Treebank pos-tags)
  - for phrase-structure rules (≈3,000 binary rule instantiations against ≈12,000 *n*-ary rules for a treebank grammar)

Data and models for CCG parsing

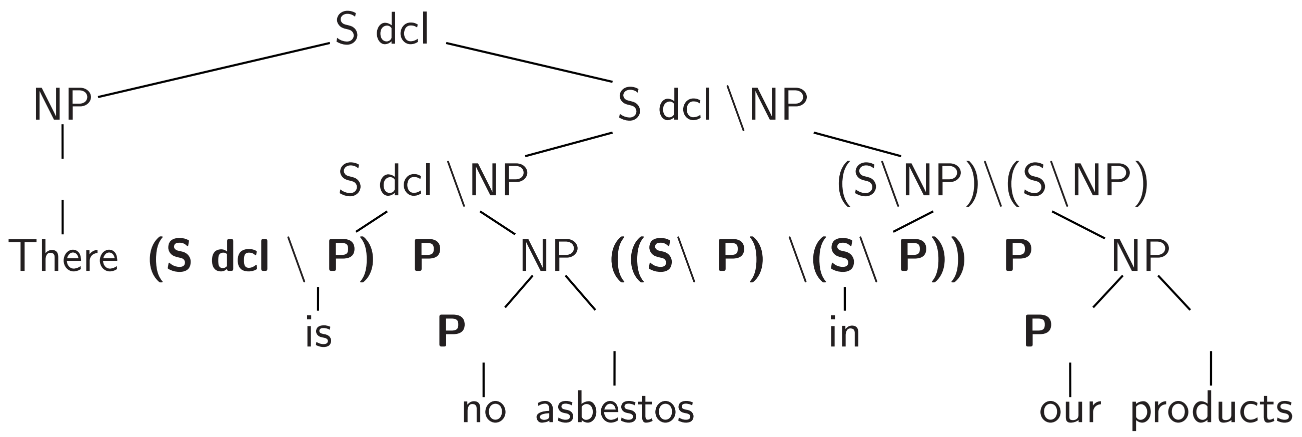
- **The corpus:** CCGbank (Hockenmaier and Steedman, LREC 2002)
  - translation of Penn Treebank phrase-structure trees to CCG derivations

- **The parser:** (Hockenmaier and Steedman, ACL 2002)
  - top-down generative probability model similar to lexicalised PCFG
  - state-of-the-art performance at (unlabelled) dependency recovery
  - CCG Parseval performance not comparable with CFG
    - ∗ binary CCG trees heavily penalised
  - but use Parseval for co-training experiments
    - ∗ consistent with other co-training experiments; relative difference important

Resolving Lexical Category Ambiguity: Supertagging

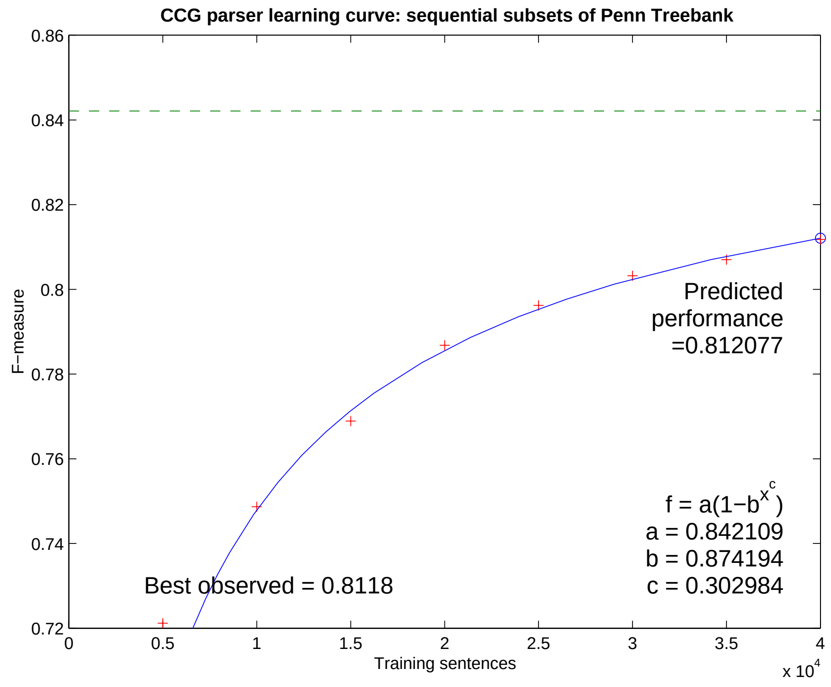
- Many words have many possible categories
- eg: the(39) man(4) is(93) an(12) executive(4) director(3)
- Prior fre uency and contextual information can help resolve ambiguity
- Standard techni ues for pos-tagging (maximum entropy models, HMMs) can be used to estimate **word-category se uence probabilities**
  - Max-Ent super-tagger gives 90 accuracy (Clark, TAG 2002)
  - use Brant s TnT tagger for co-training experiments (uick to train)

Two co-training views (Sarkar 2001)



- **generative probability of parse tree**
- **probability of lexical category se uence**
  - for parses with the same category se uence use tree probability
- Why not co-train across grammar formalisms with CCG

A learning curve for the CCG parser

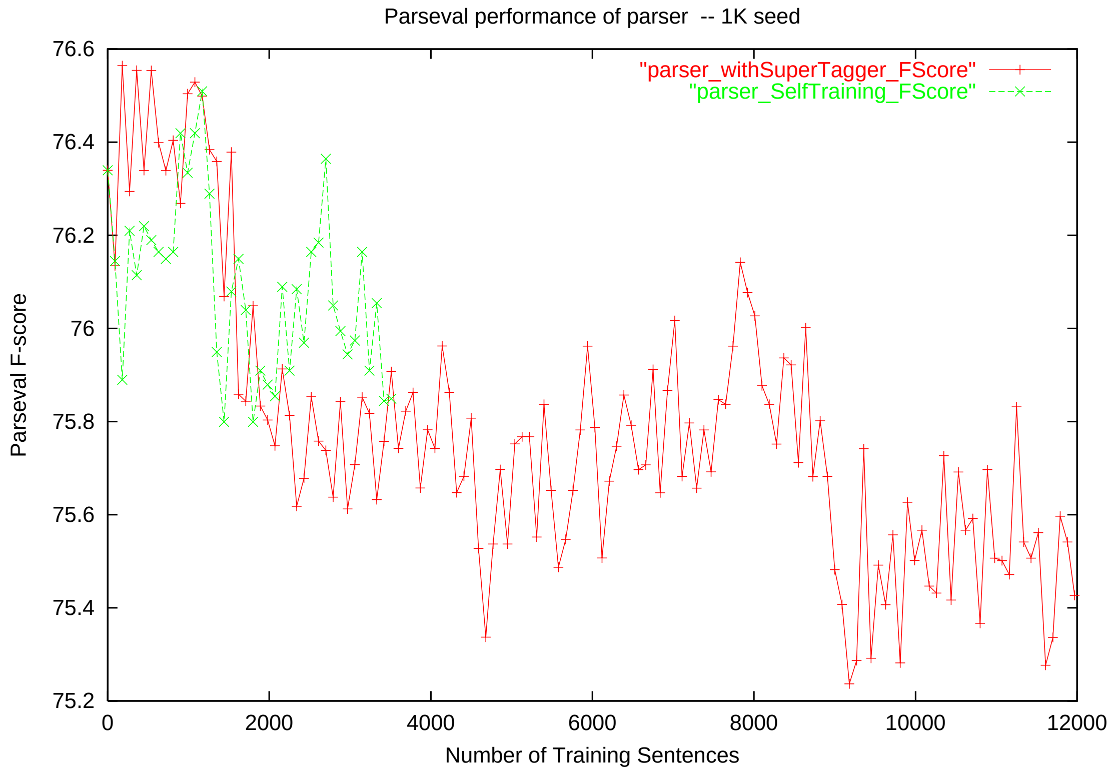


Upper bound = 84.2% Projected to Treebank of size 80K sentences = 82.8% 400K = 84.1%

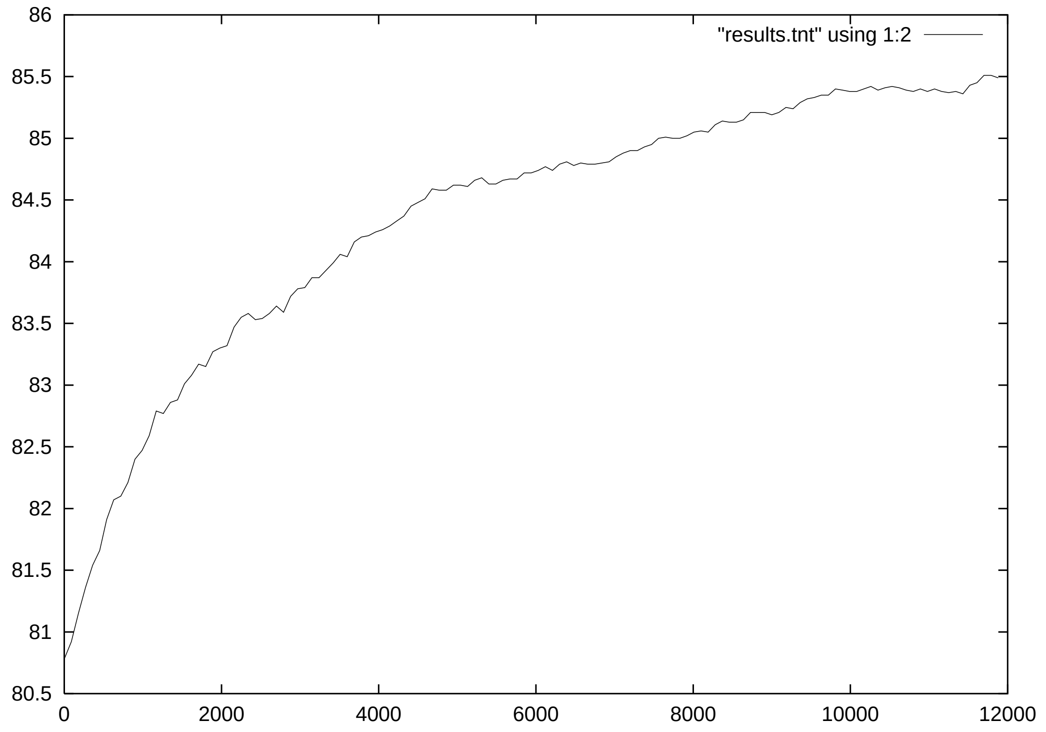
Co-training xperiments

Seed type	Seed size	added/cache	selection	evaluation
wsj	1,000	90/300	top-N	Section 00 wsj PTB
wsj	10,000	39/300		
Brown	24,000	39/300		
wsj	40,000	30m words		

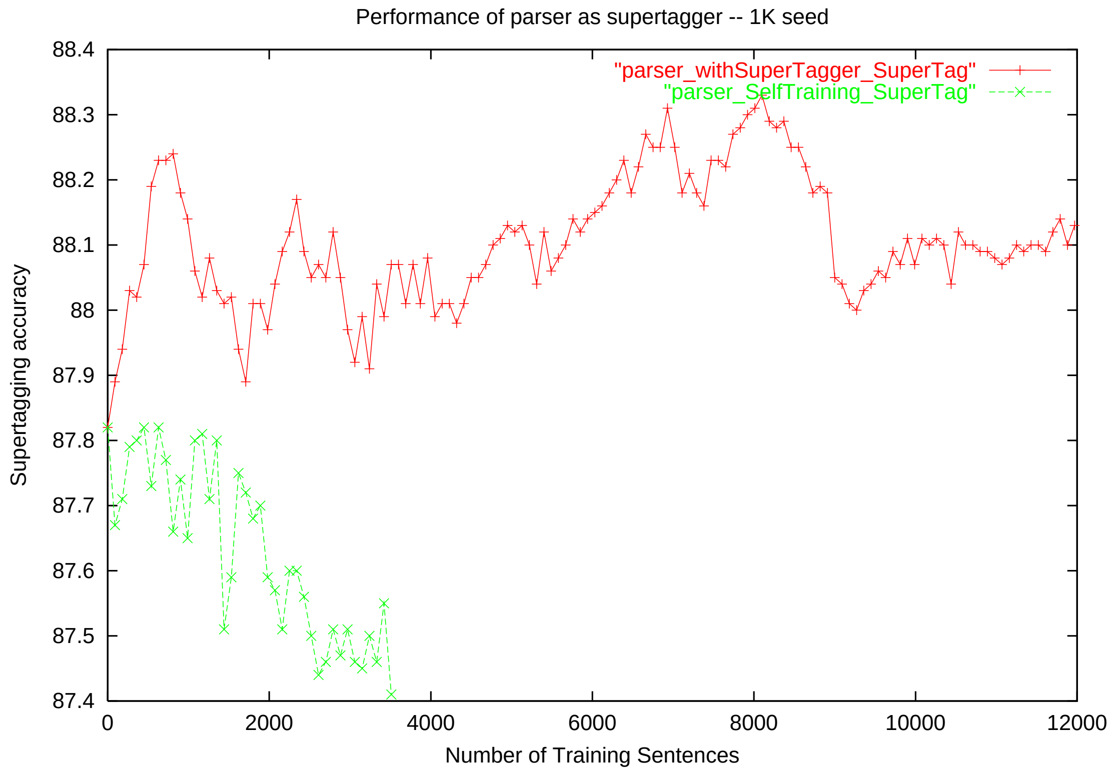
Parser scores with 1,000 sentences WSJ seed data



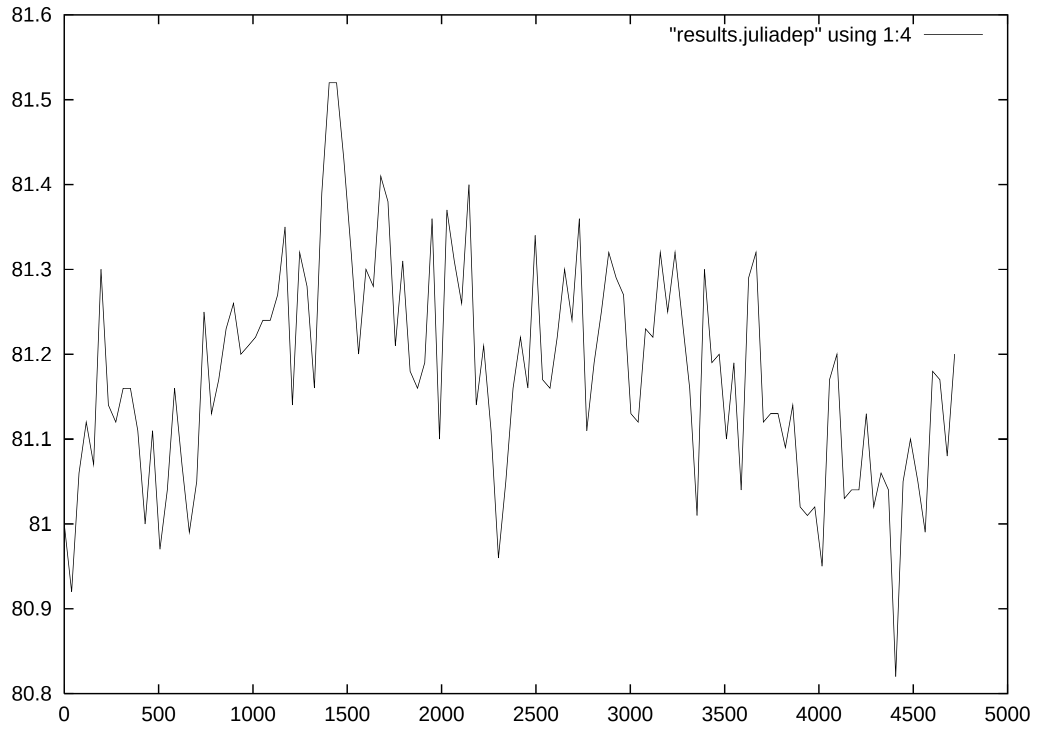
Supertagger with 1,000 sentences WSJ seed data



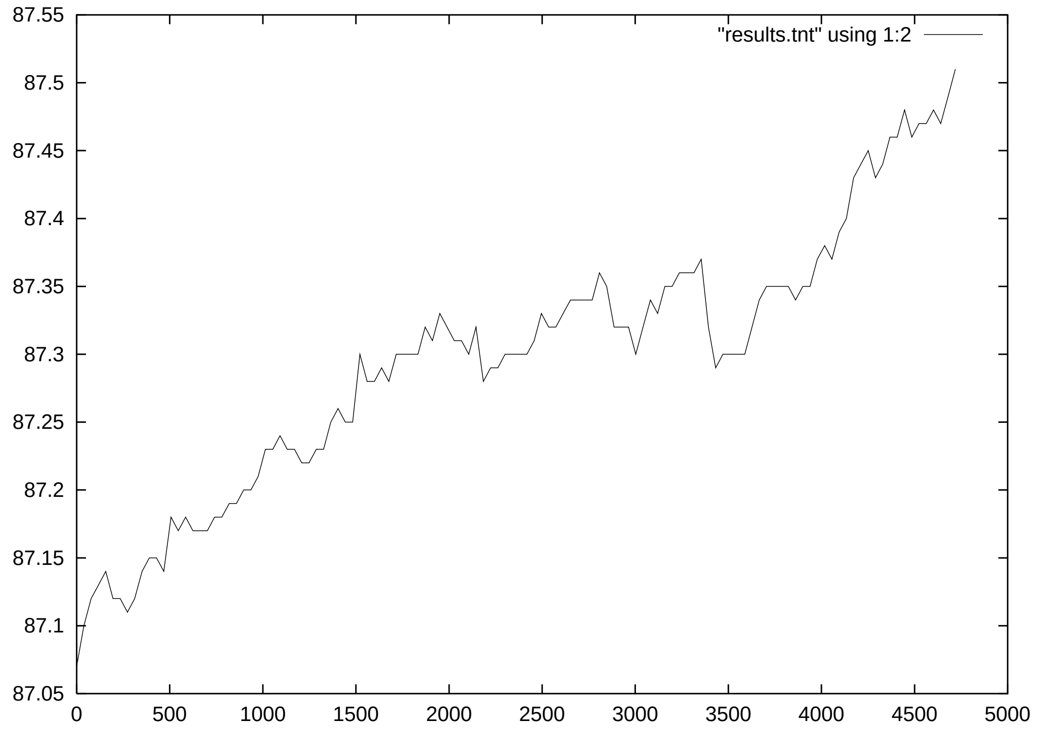
Parser as a supertagger with 1,000 WSJ seed data



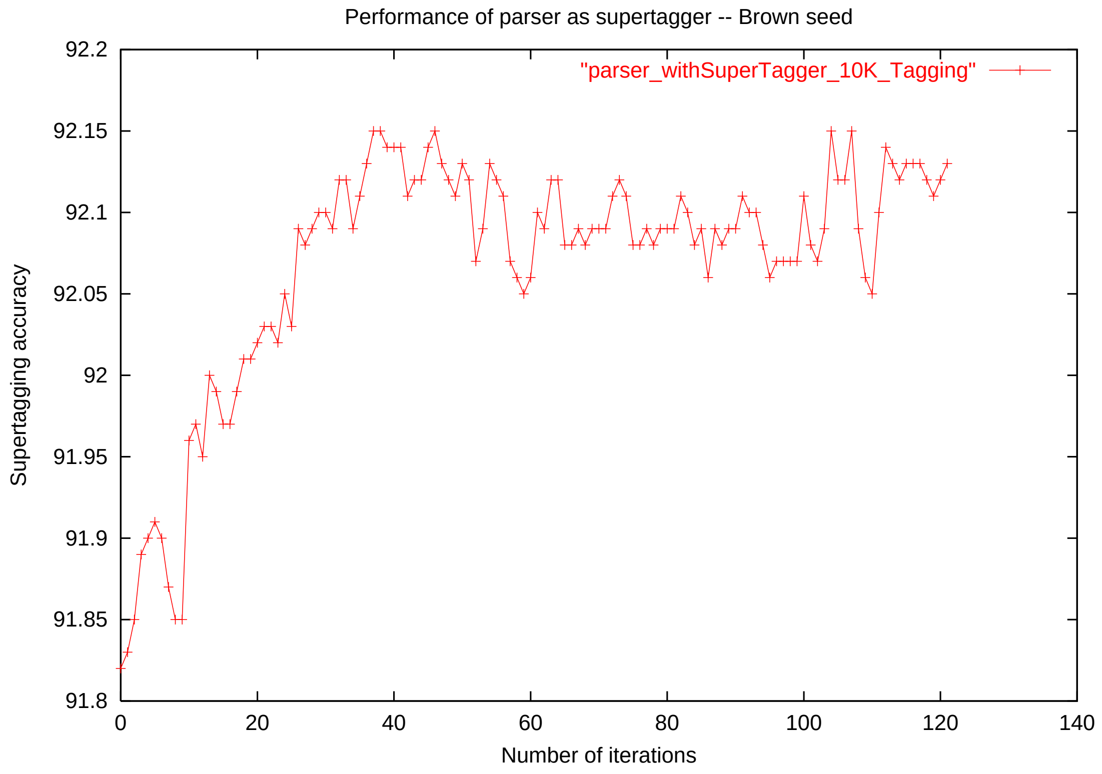
Parser scores with 10,000 sentences WSJ seed data



Supertagger with 10,000 sentences WSJ seed data



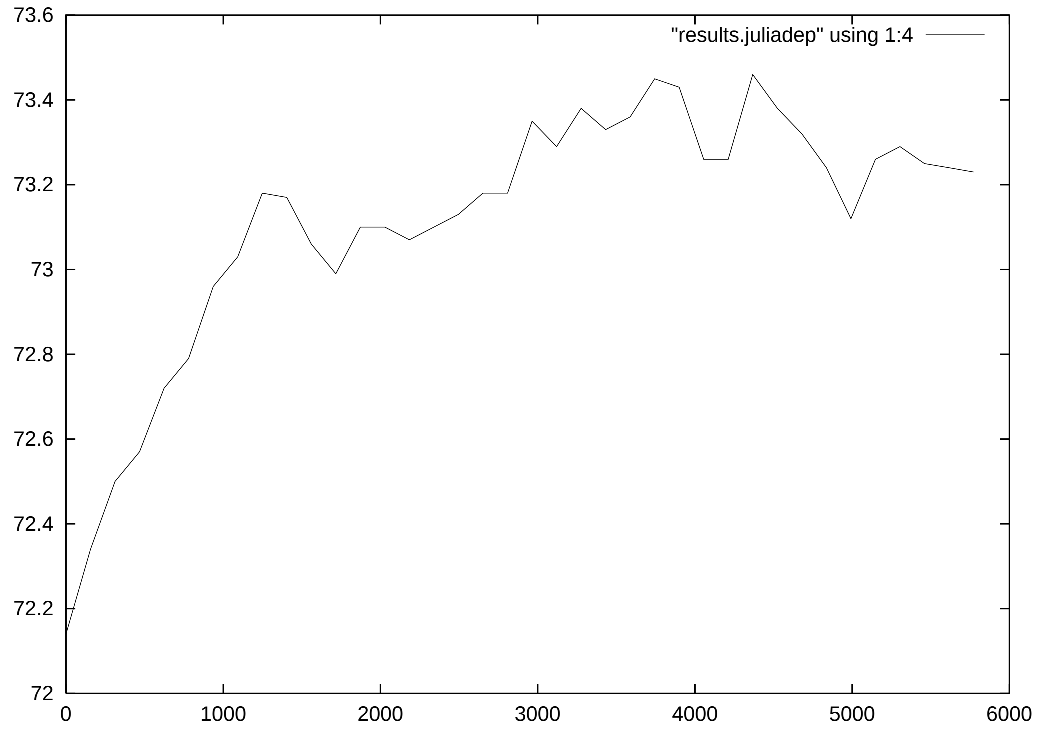
Parser as a supertagger with 10,000 WSJ seed



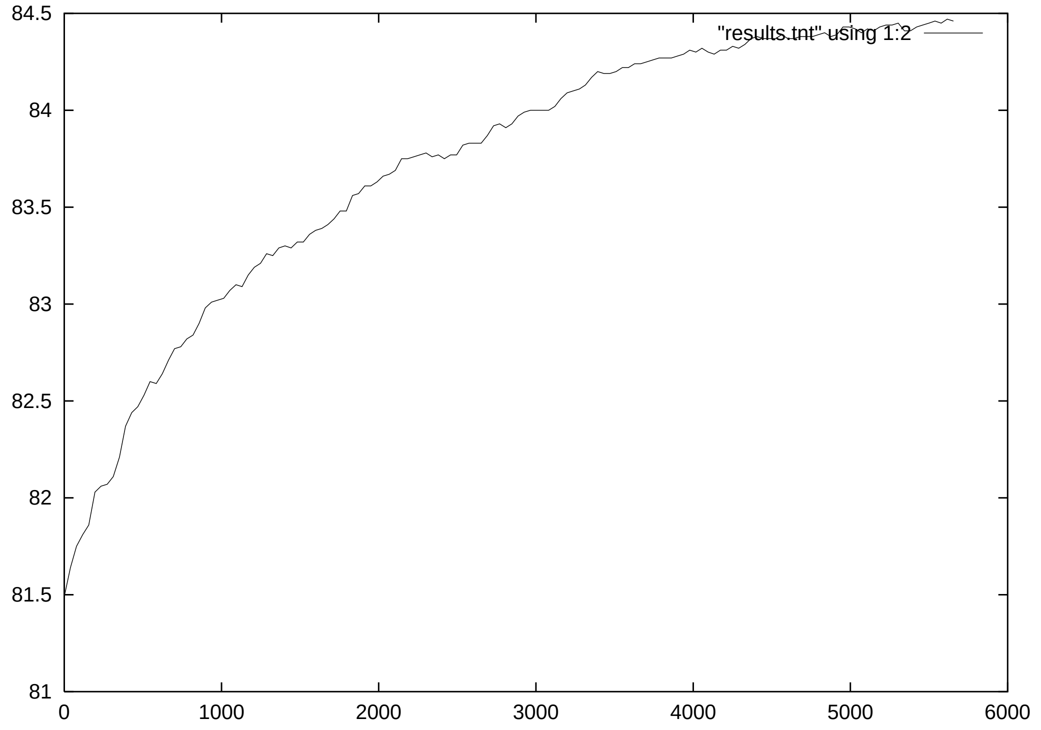
Summary of seed-WSJ experiments

- Supertagger performance improves signi cantly for 1 and 10 seed data
- Parser as a supertagger improves signi cantly for 1 and 10 seed data (performance better than supertagger itself)
- Parsing results inconclusive
  - parsing view is currently not obtaining useful information from the supertagger view (for parsing)

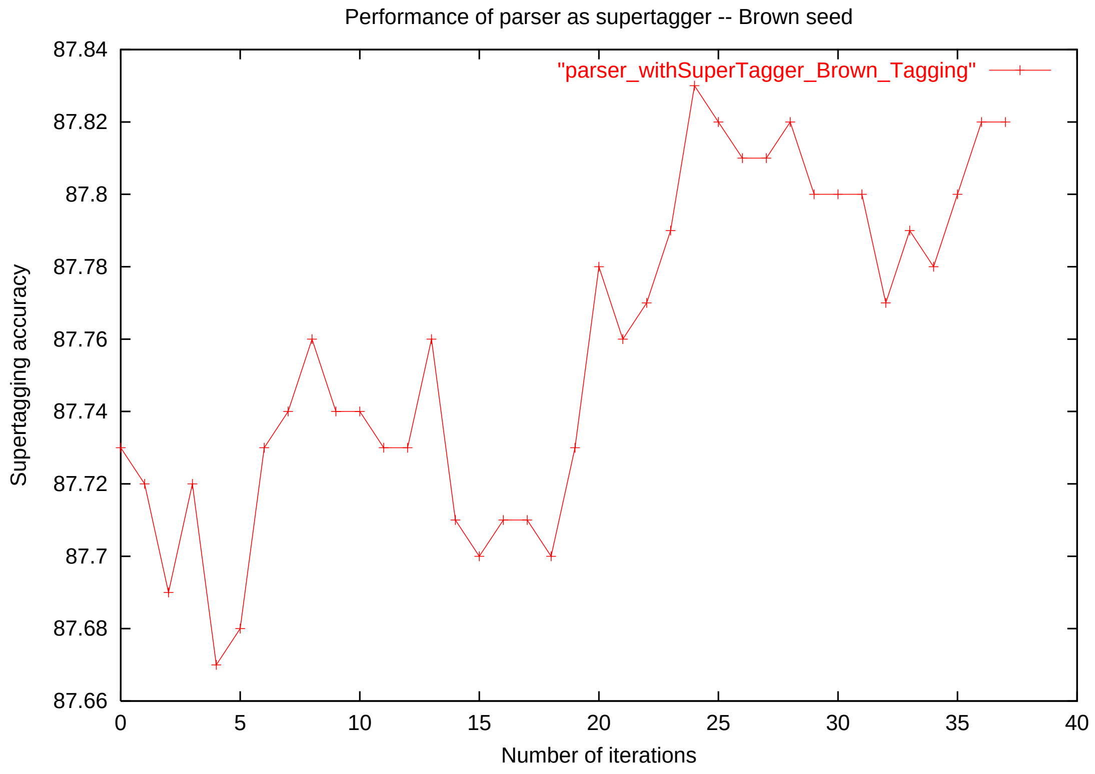
Parser scores with Brown seed data



Supertagger scores with Brown seed data



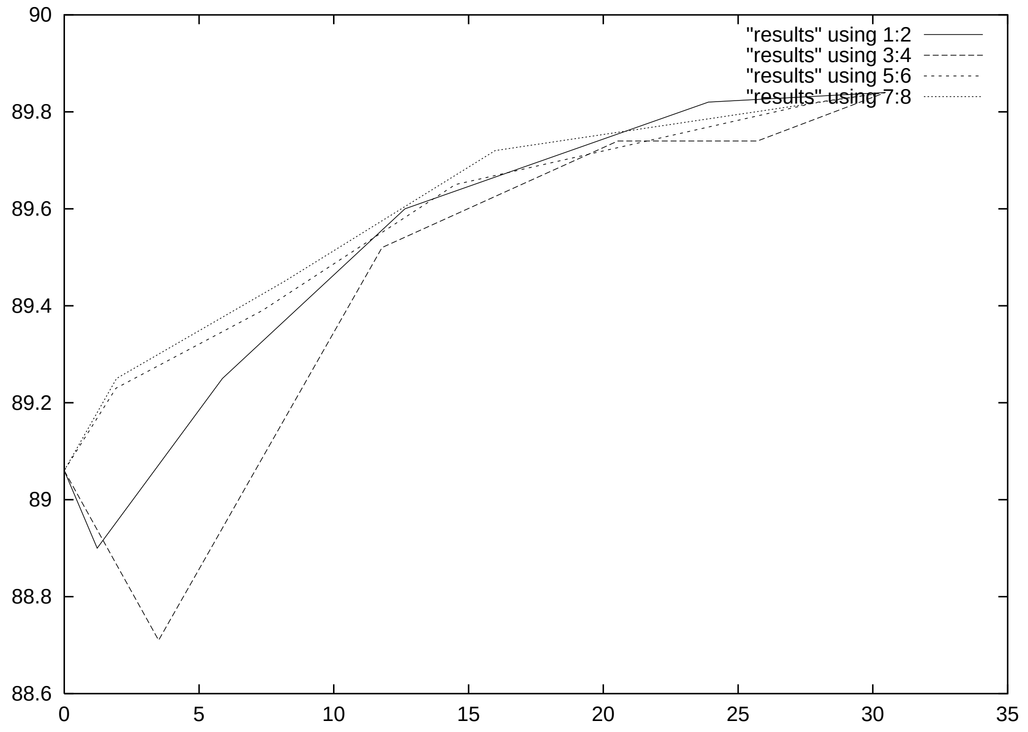
Parser as a supertagger with Brown seed data



Summary of Brown-seed experiments

- Supertagger performance improves signi cantly
- Parser as a supertagger improves signi cantly
- **Parsing results improve significantly**
  - parser is obtaining new information **about the new genre** from the supertagger view

Supertagger with lots of pre-parsed data



Summary of CCG co-training experiments

- Supertagger performance – and parser performance as a supertagger – increased signi cantly for all experiments (across all types and sizes of seed data)

- Parser performance increased for the Brown porting experiment

- Other parsing results were inconclusive; possible reasons:
  - the 2 parsing views were too similar (consider Sarkar (2001))
  - current set-up does not provide new categories for the lexicon

Summary of all CCG co-training experiments

Seed type	Seed size	Improvement: Parser start	Parser max	Supertagger start	Supertagger max	added/cache	select
wsj (dist)	1,000	71.16	72.06	80.78	81.88	30/300	top 30
wsj (dist)	1,000	71.16	72.18	80.78	84.36	90/300	top 90
wsj (dist)	1,000	71.16	71.87	80.78	81.46	300/300	all
wsj	1,000	76.34	76.56	80.78	85.49	90/300	top 90
wsj	1,000	76.34	76.41	(dist) 71.16	71.38	90/300	top 90
wsj	1,000	76.34	76.51	selftrain		90/300	top 90
wsj	10,000	81.21	81.80	87.07	87.51	39/300	top 39
wsj	10,000	80.47	80.80	87.07	87.54	39/300	intersect
brown	24,000	72.78	74.06	81.49	84.47	39/300	top 39

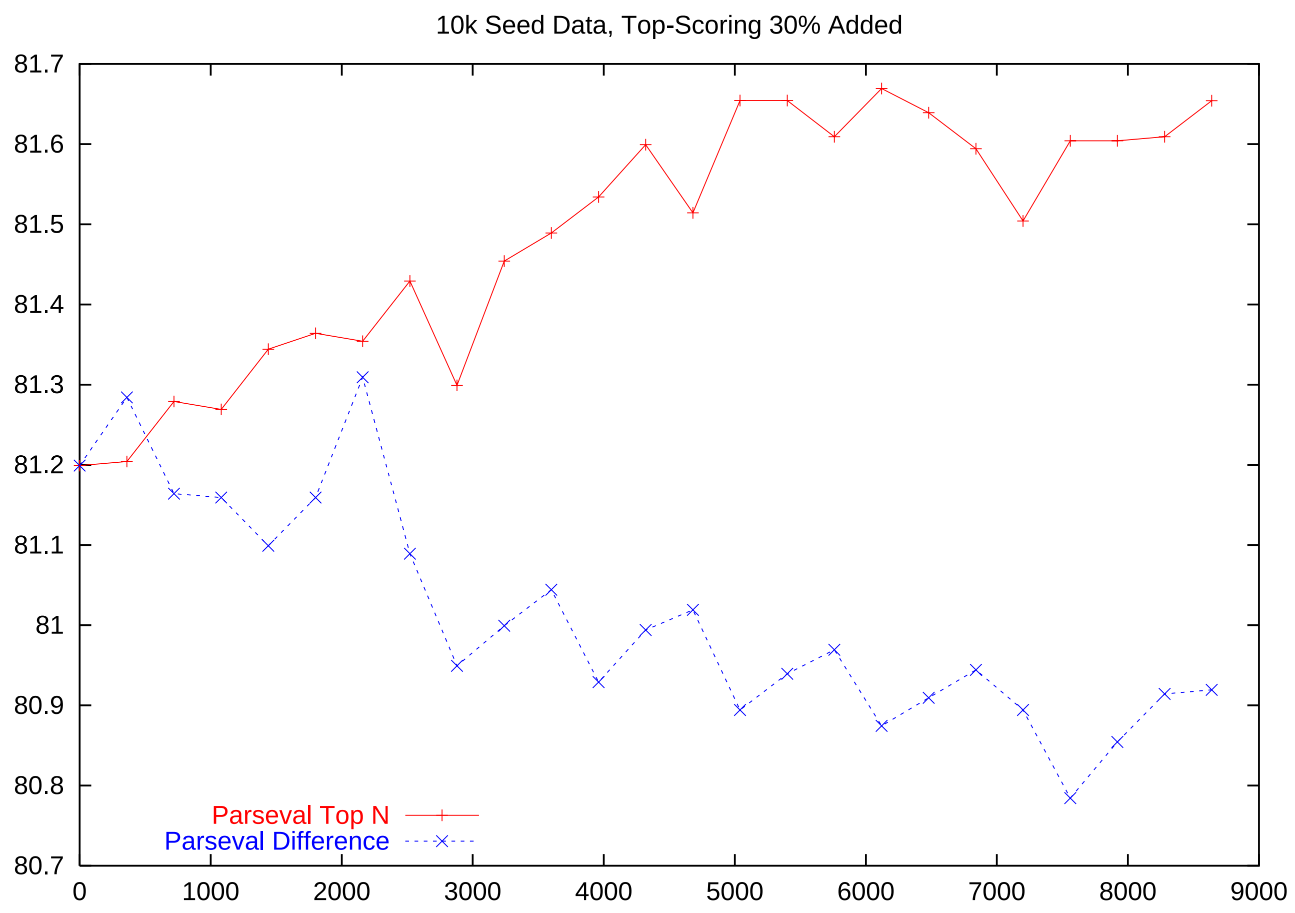
CCG Parseval Oracle xperiments

Steven Baker

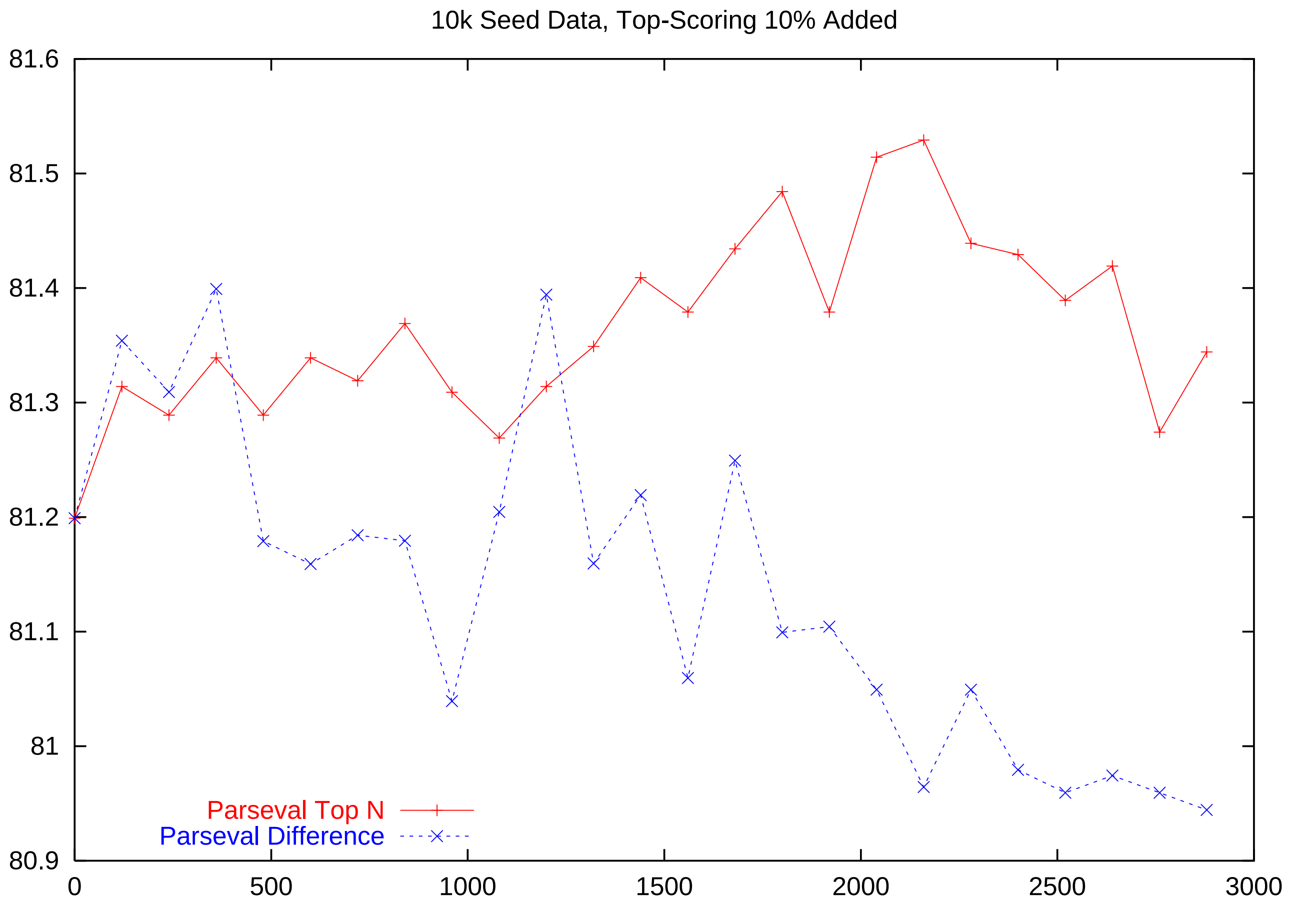
CCG Parseval Oracle xperiments

- Use Parseval as an oracle to obtain a score for newly-parsed data
- Can use this measure to investigate differences between parse selection methods without the inaccuracy of the sentence score skewing the results
- Also useful in establishing realistic upper bounds of performance when using non-oracle scores to determine parse-selection

Oracle Top- Difference Comparison, 10k Seed



Oracle Top- Difference Comparison, 10k Seed



10k Seed Training Set Results

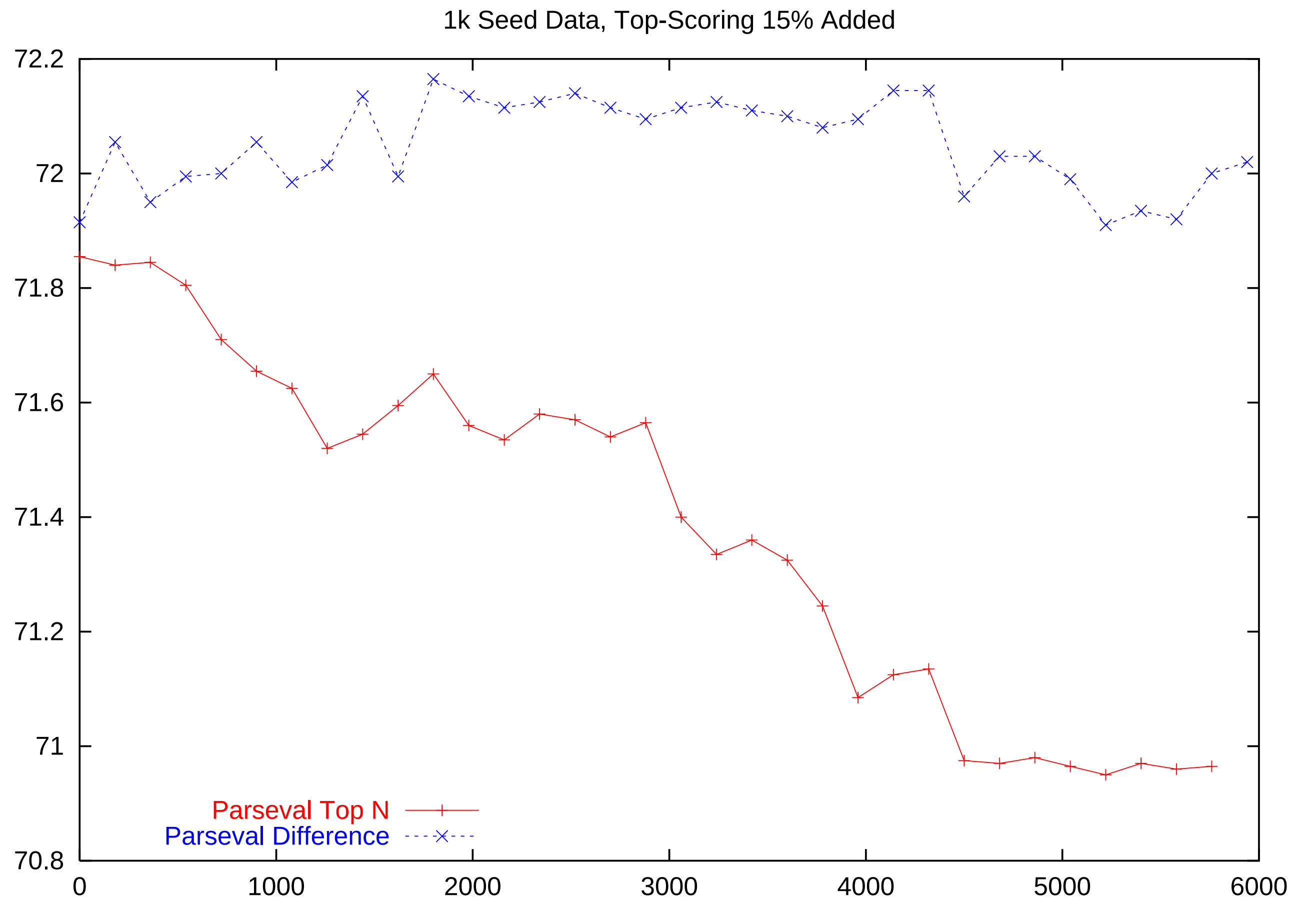
- Top-N selection method results in increased performance over Difference selection

- Top-N Method performs similarly using both yield rates, even though the higher yield rate means the selection method is less restrictive about what it adds back in

- Upon examination, with 10k initial training, almost 30 of new parses were perfect according to Parseval, so the uality of parses being added would not be changed as Top-N s yield is increased from 10 to 30 .

- Also suggests that at 10k, some improvement can be had when cotraining with the right selection method.

Oracle Top- Difference Comparison, 1k Seed



1k Seed Training Set Results

- Difference selection method performs better than Top-N selection at 1k
- Shows that there is little room for improvement, even using oracle selection methods, of current CCG parsers at 1k.

CCG Parseval Oracle Summary

- A selection method that works well for one set of views may not work well for another
    - Top-N was the better method in the 10k experiments
    - Difference was the better method in the 1k experiment
- These results are similar to Oracle selection results seen in the CFG/LTAG domain
- Only small improvements were seen in the CCG Oracle-selection experiments, which mirrors the non-Oracle results

Smoothing to help parsers co-train

Julia Hockenmaier and Mark Steedman

August 22, 2002

Graduate student proposal

- Investigate **smoothing for statistical parsing with CCG**
  - Important general question
  - Essential to allow CCG to benefit from co-training
- Student: Julia Hockenmaier, University of Edinburgh
- Supervisor: Prof. Mark Steedman, University of Edinburgh

Why smoothing is important

- Statistical parsing will always suffer from a sparse data problem
- Statistical CCG parsers suffer from a **lexical coverage problem**
- Smoothing: assign non-zero probabilities to unseen events (eg. **unseen lexical entries**)
  - What unseen events should get a non-zero probability?
  - How do we compute this probability?

Smoothing in our current parser

- Dependency model: **linear interpolation** with baseline model, using the same technique as Collins ’99 to estimate weights
- Smoothing of **lexicalized rule probabilities**: back off from head word, but not from lexical head category
- Smoothing of **lexicon probabilities**: use POS-tag information to generate new lexical entries

Why smoothing is important for co-training

- Co-training especially beneficial with **limited amounts** of seed data
  $\Rightarrow$ Smoothing is essential in this setting

- **Collins–LTAG** co-training works because:
  - Both parsers have more levels of backoff, hence can deal with noisy input, and achieve reasonable performance with little seed data
  - Both parsers provide each other with **novel** input

- With our present setup, cotraining for **CCG** has limited effect:
  - The two views cannot provide novel input such as new **lexical categories** or new **rule instantiations**

Smoothing (lexicalized) rule probabilities

- Treebank-style:

$$P(\mathtt{S}_{\langle \mathtt{VBZ}, buys\rangle} \rightarrow \mathtt{NP}\ \ \mathtt{VP})$$

- CCG-style:

$$P(\text{S[dcl]}_{\langle \text{(S[dcl]\textbackslash NP)/NP}, buys\rangle} \rightarrow \text{NP}\ \ \text{S[dcl]\textbackslash NP})$$

$\Rightarrow$ We **cannot back off** from the lexical category!
(But might be able to back off to classes of lexical categories)
$\Rightarrow$ Can we use techniques like (Eisner 2001) or similarity-based clustering for CCG?

Smoothing lexicon probabilities

- In CCG, **lexical categories** determine the space of possible parses
- In our parsers, the lexicon consists of **observed word-category pairs**
- Even a lexicon extracted from the entire labelled training corpus is incomplete
- This is not so much a problem for unknown words, as for **low-frequency observed words**

Lexical rules

- Assume we have observed 3rd pers. sg *buys* with (S[dcl]\NP)/NP
- If we know English morphology, we also know the following entries:
  - non-3rd-pers-sg *buy*: (S[dcl]\NP)/NP
  - infinitival *buy*: (S[b]\NP)/NP
  - present participle *buying*: (S[ng]\NP)/NP
  - past participle *bought*: (S[pt]\NP)/NP
  - passive *bought*: S[pass]\NP
- We can use a morphological analyzer to generate new word forms

Smoothing for lexical rules

- How do we estimate probabilities for these new lexical entries?
- Conduct a comprehensive study similar to (Chen and Goodman, ’96)

The proposal

We propose to investigate:

- how we can make use of **lexical rules** in order to overcome the sparse data problem in the CCG **lexicon**

- which **smoothing techniques** are most effective to estimate the probabilities of these new entries

- how we can smooth the **rule probabilities**

Evaluating the effectiveness of smoothing

We propose to evaluate the effectiveness of smoothing:

- by measuring the effect on **overall parse performance**
- by measuring the effect on **co-training**

Possible payoffs

- Better CCG parsers
- Better understanding of smoothing for parsing with expressive grammars
- **Allow CCG to benefit from co-training**

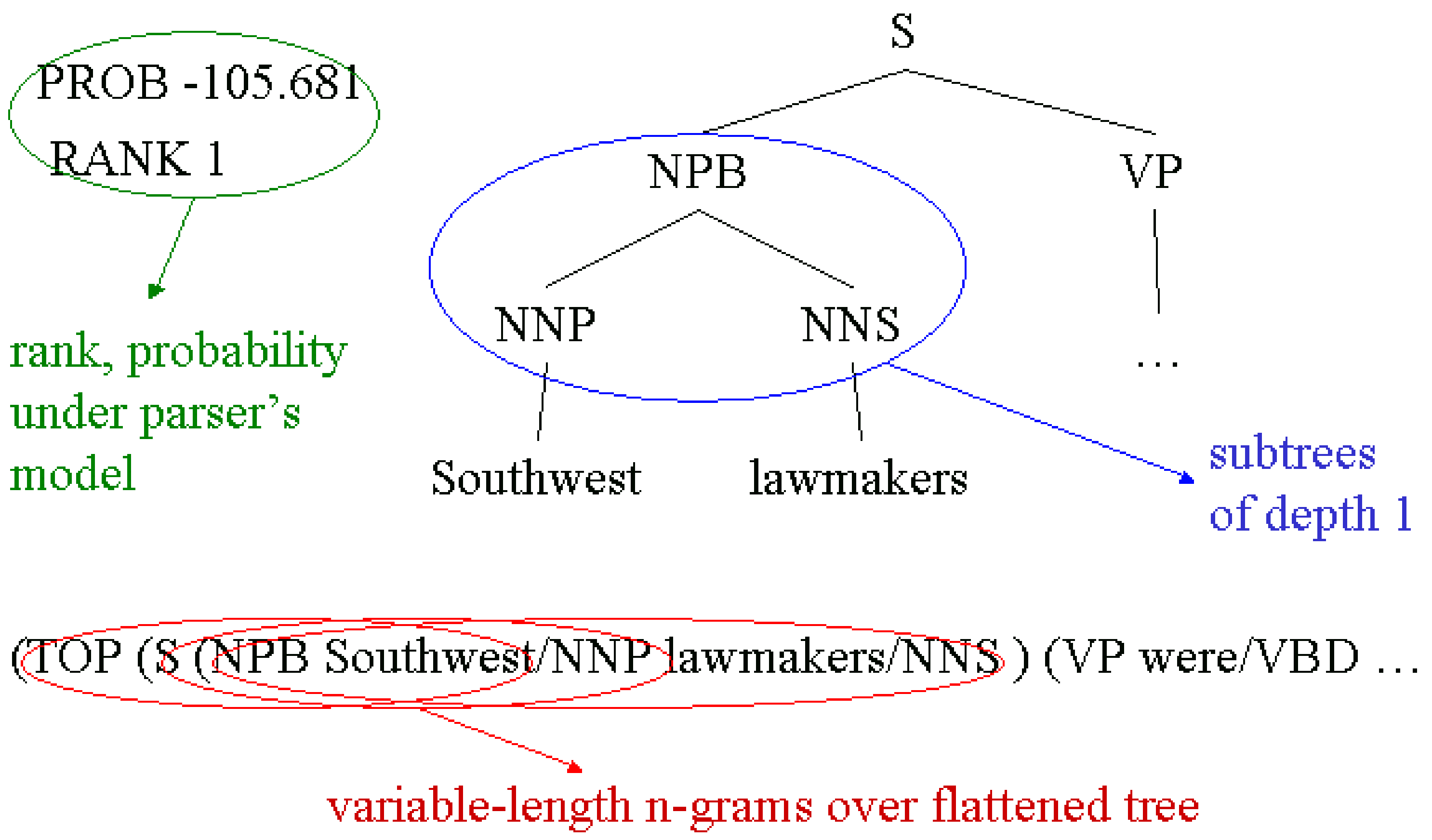
Co-training with Re-rankers

Jeremiah Crim
Johns Hopkins University

Re-ranking vs. Parsing

- A re-ranker reorders the output of an n-best (probabilistic) parser based on features of the parse
- While parsers use local features to make decisions, re-rankers use features that can span the entire tree
- Instead of co-training parsers, co-train different **re-rankers**

Which features?


PROB -105.681
RANK 1
rank, probability under parser's model
S
NPB
VP
NNP
NNS
…
Southwest
lawmakers
subtrees of depth 1
(TOP (S (NPB Southwest/NNP lawmakers/NNS) (VP were/VBD …
variable-length n-grams over flattened tree

Motivation: Why re-rankers?

- **Speed**
  - We need a large amount of unlabeled data to get an improvement
  - But parsing is slow, and we must re-parse data multiple times
  - If we are re-ranking, we only have to parse data once
  - Output will be reordered many times, but this can be done very quickly
- **Objective function**
  - The lower runtime of re-rankers allows us to explicitly maximize agreement between parses
  - Measures such as intersection and difference that attempt to approximate agreement not needed in this framework

Motivation: Why re-rankers?

- **Accuracy**
  - Re-rankers can improve performance of existing parsers
  - Collins ’00 cites a 13 percent reduction of error rate by re-ranking
- **Task closer to classification**
  - A re-ranker can be seen as a binary classifier: either a parse is the best for a sentence or it isn’t
  - This is the original domain cotraining was intended for

Reranker 1: Log linear model

- Similar to the re-ranker proposed by Collins in ”Discriminative Reranking for Natural Language Parsing”, ICML ’00
- Implemented prior to the workshop by Miles Osborne

Reranker 2: Linear perceptron

- Learn a mapping from vectors of features for a given parse to the parseval score for that parse:

- Perceptron Algorithm:

  1. Assign starting values to weights for each feature and use them to predict parseval scores for first training parse
  2. Update weights based on how far off the guess was
  3. Repeat step 2 for all training examples

- Use these weights to re-rank the n-best output of Collins' parser

Differences between Re-rankers

Log linear	Linear perceptron
Learns weights for features that predict the best parse for a sentence.	Learns weights for features that predict an expected parseval score for a parse.
Can teach perceptron about features that predict a good parse for a sentence, regardless of parseval score.	Can teach log linear model about features that will lead to a high parseval score, no matter what the rank of the parse.

Maximizing Objective Function

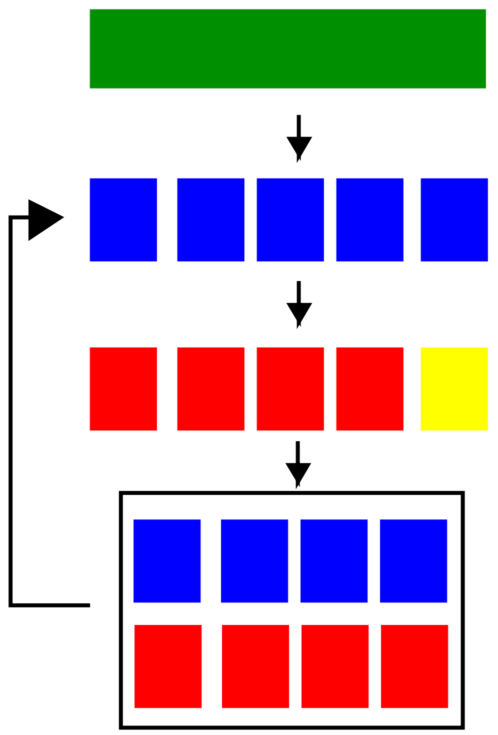
- Maximize **agreement between re-rankers**

1. Parse next sentences into cache.

2. Re-rank cache using one model and partition into sections.

3. Other model searches for best partition to retrain on.
4. Retrain on this partition.

5. Switch roles of models. Repeat.



Experiments III

- Initial small-scale experiments:
  - Train Collins parser on first 1000 sentences of Penn Treebank section 2
  - Parse remainder of PTB 02-21 using this parser
  - Train initial re-rankers on first 1000 sentences
  - Co-train re-rankers using remainder of parsed data

Baseline and Upper Bound

- Baseline parseval score (section 0):
  - Random choice from top n parses: 77.82
  - Parser’s initial choice: 82.28
  - Log linear model before co-training: 82.07
  - Linear Perceptron before co-training: 82.36

- Upper bound parseval score (section 0):
  - Gold standard: always pick best from top n: 89.50
  - Oracle: Re-rankers trained on additional ”clean” data:
    - Log linear model, +10k clean: 83.61
    - Linear perceptron, +10k clean: 82.76

Results III

- Replace only 1 partition of cache each iteration
- Log linear re-ranker improved to 82.41 from 82.07
  - 27 percent of possible error rate reduction achieved by clean data
- No improvement for linear perceptron

Summary III: Co-Training Re-rankers

Seed Data Type	Seed Set Size	Effect:		n_+/n_{cache} per iter'n	n_{total} added	Upper Bound:	
		F_0	F_n			Gold	Oracle
WSJ	1000	82.07	82.41	500/2500	10000	89.5	83.61

Post-WS02 Project Proposal

Co-training with Re-rankers

Jeremiah Crim, Johns Hopkins University
Miles Osborne, University of Edinburgh

Post-WS02 Project - Why re-rank?

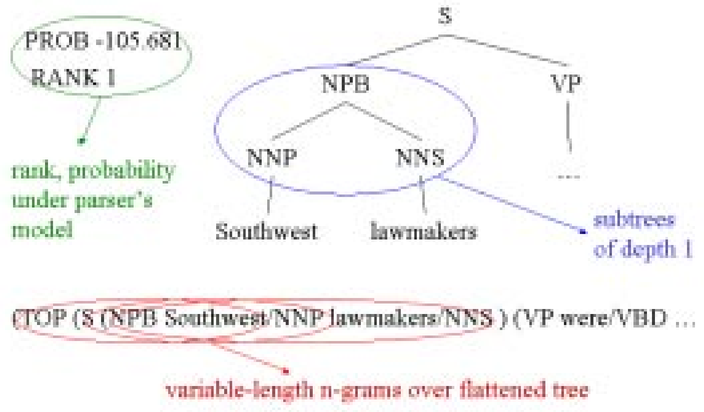
- Faster than parsing
- Can maximize agreement
- Greater accuracy
- Task closer to classification

Post-WS02 Project - More re-rankers

- We have only co-trained two re-rankers: Log Linear and Linear Perceptron
- Other re-rankers to try - Winnow, Voted Perceptron (Collins), Naive Bayes, Decision Tree, etc.
- Many have already been implemented, during the workshop (Winnow, Perceptron) or before, and co-training infrastructure is in place

Post-WS02 Project - Other feature sets

- We've only worked with one feature set so far


PROB -105.681
RANK 1
rank, probability under parser's model
S
NPB
VP
NNP
NNS
Southwest
lawmakers
...
subtrees of depth 1
(TOP (S (NPB Southwest/NNP lawmakers/NNS) (VP were/VBD ...
variable-length n-grams over flattened tree

- There are many other ways that features could be assigned to parses
  - ie: use all nonterminal subtrees (see Collins and Duffy 02) or lexical nonterminals for subtrees of depth 1
  - features from other parsers?

Post-WS02 Project - Maximizing agreement

- We use Spearman's rank to measure similarity between output of re-rankers
  - This treats all ranks as equally important
  - Explore non-uniform weighting extensions

Post-WS02 Project - Maximizing agreement

- Exploring huge search space... must intelligently choose what to retrain on

- Current method greedily optimizes agreement one re-ranker at a time

- But a slightly suboptimal set for one re-ranker may allow a larger improvement in the other

- Explore alternative search methods
  - ie: Best first search, etc.

Post-WS02 Project - Possible Payoffs

- Beat section 23?

  (improving state-of-the-art parsing results)

  - After learning best parameters for co-training on small sets, train parser and re-rankers on all of PTB
  - Retrain on up to 500m additional words (NANC) - large portion already pre-processed during workshop
  - Better chance of improvement than continuing to tweak parsers that have already almost converged
  - Room for improvement:
    - Current best parser: 89.7
    - Oracle that picks best parse from top 50: 95 +

Summary

August 22, 2002

Summary 1

- Co-training worked in a variety of situations.
- Main results:
  - Co-training is beneficial when the seed data is limited.
  - Co-training is less helpful (but not harmful) when the seed data is plentiful.
  - Novel parse selection techniques were developed, and were crucial for co-training with larger amounts of seed data.

Workshop Goals

- Explore the way co-training effects depend on size of labelled seed set.
- Explore effectiveness of co-training for porting parsers to new genres.
- Explore effectiveness of co-training on parsers seeded with all available labelled material.

Small labelled seed sets

Co-training enhances performance:

- For CFG/LTAG:
  - CFG always improves.
  - LTAG always improves.
- For Supertagger/CCG:
  - Supertagger – and parser as a Supertagger – always improves (not yet converged).
  - CCG parsers inconclusive.

Co-training allows rapid development of statistical parsers for languages with limited available resources.

Porting parsers to new genres

Co-training can help porting:

- For CFG/LTAG:
  - CFG always improves.
  - LTAG always improves.
- For Supertagger/CCG:
  - Supertagger always improves.
  - CCG parsers improve.

Co-training allows more effective porting to novel domains.

Improving state-of-the-art parsing performance

Not there yet:

- Collins-CFG Parser (nearly) converged already.
- CCG requires enhanced smoothing for co-training to be effective.

Solutions:

- Collins-CFG/LTAG re-ranking experiments have started.
- CCG smoothing will be implemented (as described by Julia).

Future work 1

Need for analysis:

- Workshop has been largely experimental.
- Understand relationship between view independence and parser configurations.
- Understand link between seed labelled set size and co-training.

Future work 2

- For Collins-CFG/LTAG:
  - Better scoring functions for parse selection.
  - Large-scale experiments have started.
  - Develop re-rankers.

- For CCG:
  - Extend statistical CCG modelling such that it can benefit from co-training.
  - Do large-scale experiments.

Conclusion

- This is the **largest experimental study to date** on the use of unlabelled data for improving parser performance.
- **Co-training enhances performance** for parsers and taggers trained on **small (500—10,000 sentences) amounts of labeled data**.
- **Co-training can be used for porting** parsers trained on one genre to parse on another **without any new human-labeled data at all**, improving on state-of-the-art for this task.
- **Even tiny amounts of human-labelled data** for the target genre enhace porting via co-training.
- New methods for **Parse Selection** have been developed, and play a crucial role.