Parsing with Tree-Adjoining Grammars

Anoop Sarkar

Simon Fraser University

anoop@cs.sfu.ca

ACL/HCSNet Advanced Program in

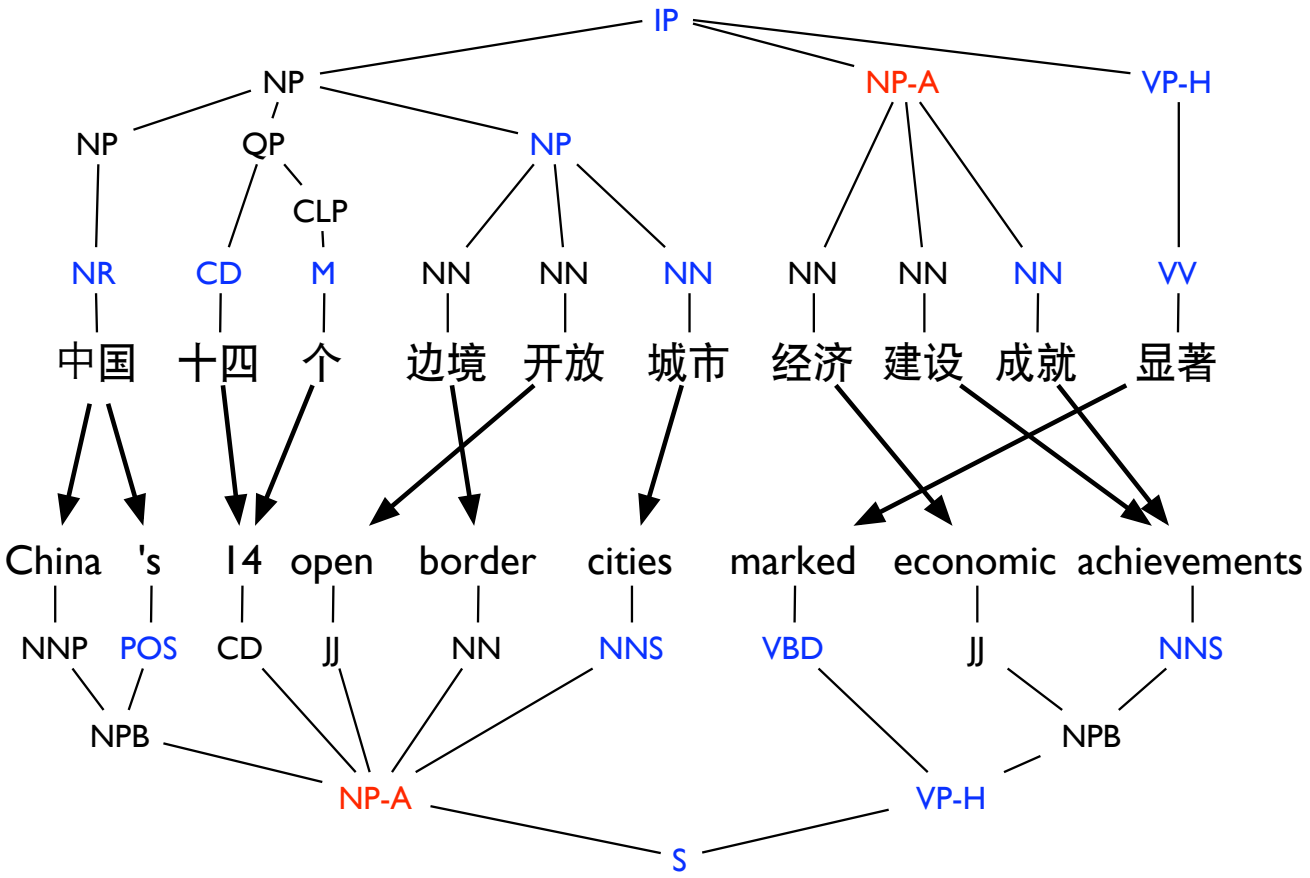
Natural Language Processing

Melbourne, Australia July 13-14, 2006

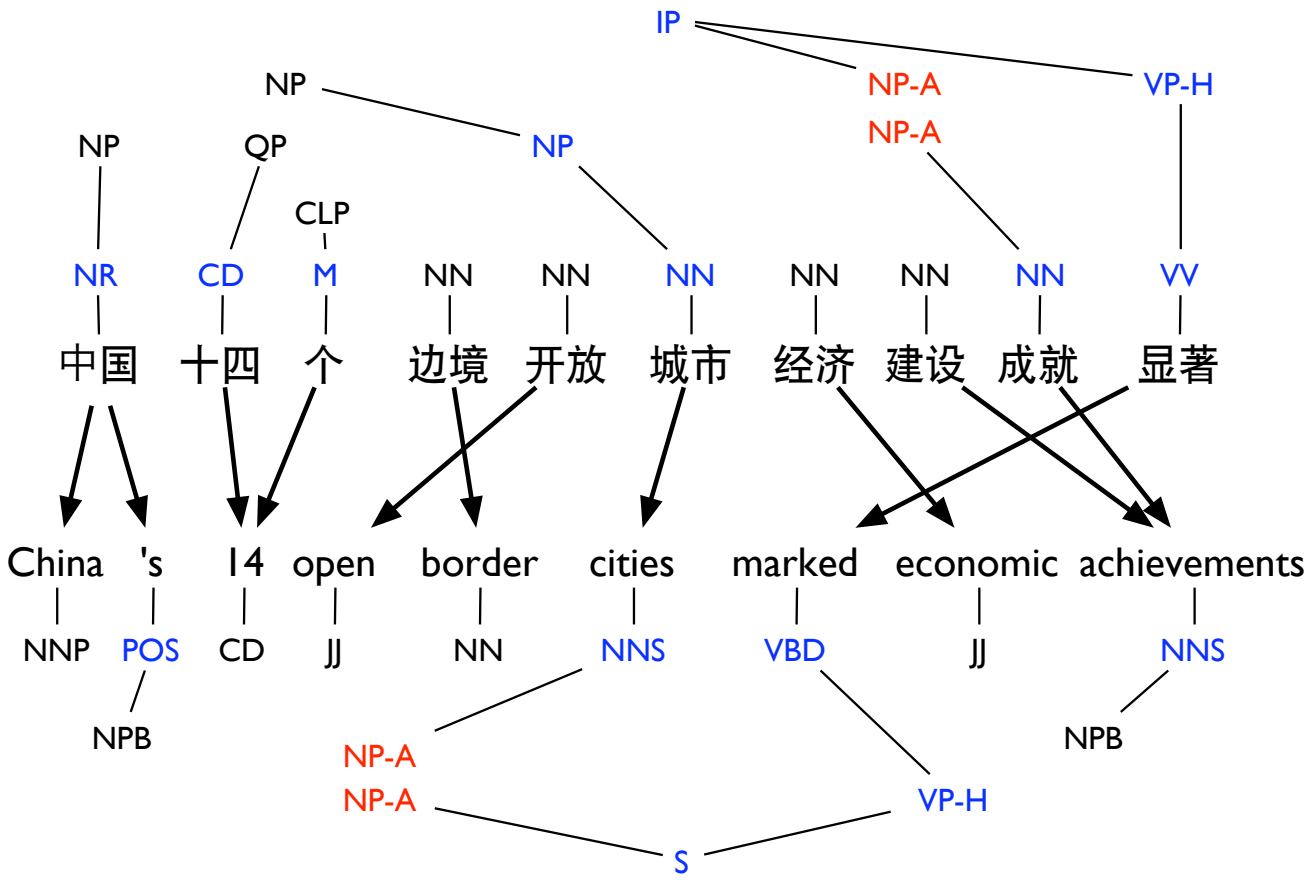
Overview

- **TAGs from word-aligned parallel corpora**
- Supertagging and robust, shallow parsing (slides taken from B. Srinivas)
- Statistical Parsing with LTAG
- Bootstrapping between CFG and LTAG

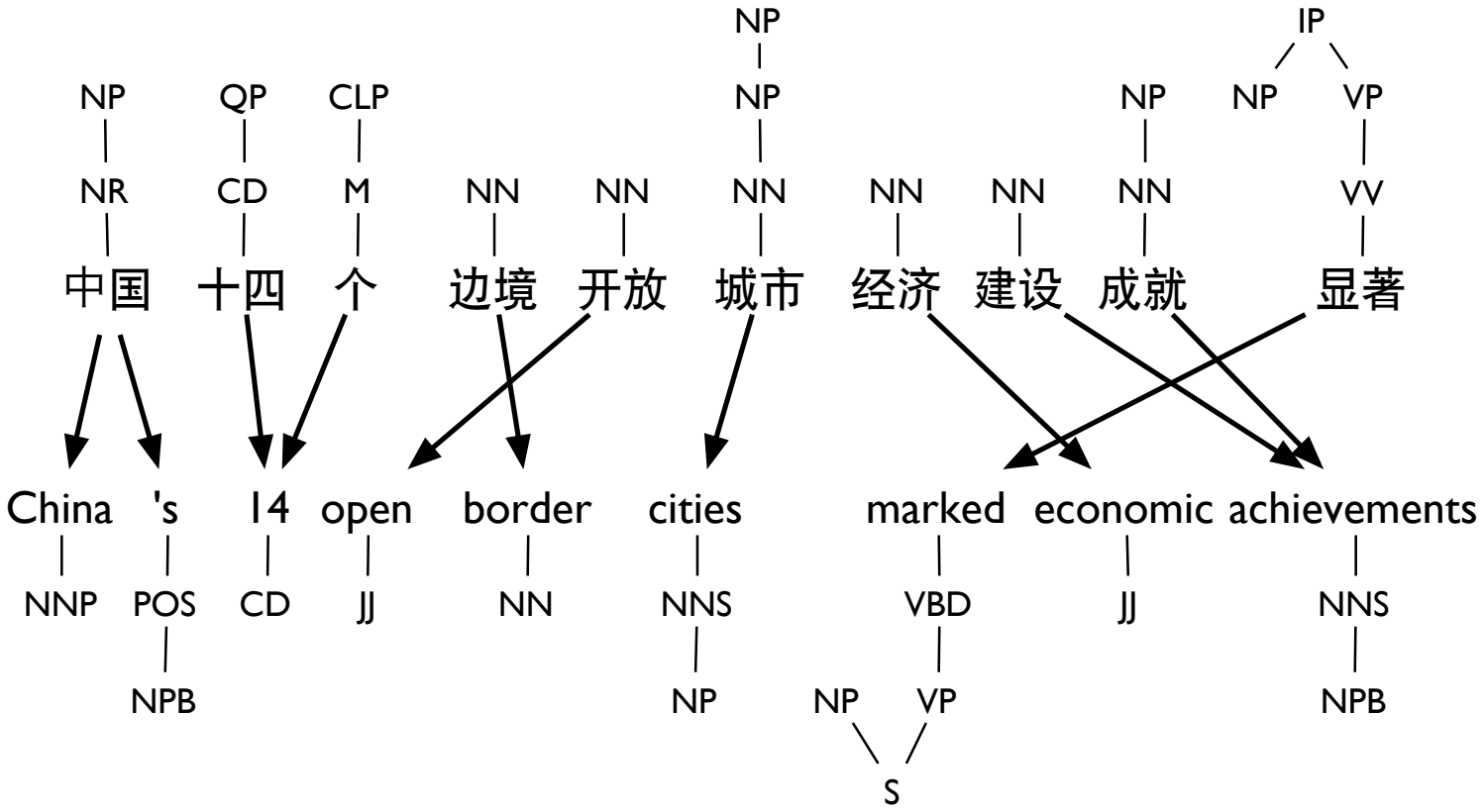
Parse trees have head and argument information



Parse trees have head and argument information



Each word gets a tree: elementary tree



Models over alignments of elementary trees

- Trained using SRI LM Toolkit using 60K aligned parse trees: 1300 elementary tree templates each for Chinese/English
- Unigram model over alignments: $\prod_i P(f_i, t_{f_i}, e_i, t_{e_i})$
- Conditional model: $\prod_i P(e_i, t_{e_i} \mid f_i, t_{f_i}) \times P(f_{i+1}, t_{f_{i+1}} \mid f_i, t_{f_i})$
- IBM Model 1 on aligned elementary trees (Kenji)

SMT Re-ranking Results

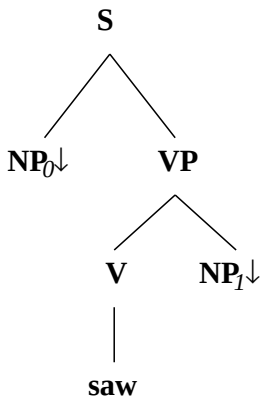
Method	BLEU[%]
Model 1 on elementary trees	31.6
Unigram model over aligned elementary trees	31.7
Conditional bigram model over aligned elementary trees	31.9

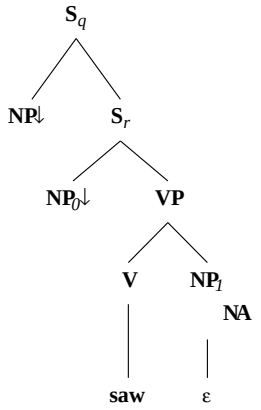
Overview

- TAGs from word-aligned parallel corpora
- **Supertagging and robust, shallow parsing** (slides taken from B. Srinivas)
- Statistical Parsing with LTAG
- Bootstrapping between CFG and LTAG

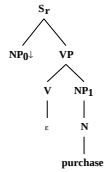
Supertags

- Elementary trees are called Supertags.
- Localize head-complement and filler-gap dependencies.

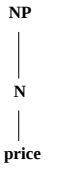




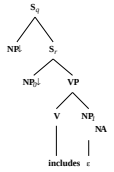
- Supertags
  - more complex than part-of-speech tags
  - more supertags associated with word than part-of-speech tags



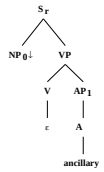
α_1



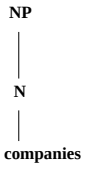
α_2



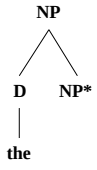
α_3



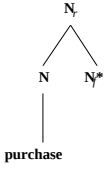
α_4



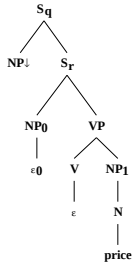
α_5



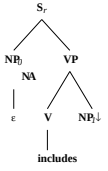
β_1



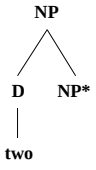
β_2



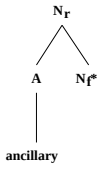
α_6



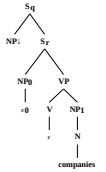
α_7



β_3

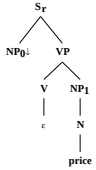


β_4

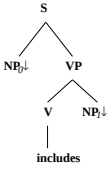


α_8

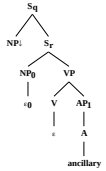
α_9



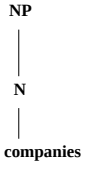
α_{10}



α_{11}



α_{12}



α_{13}

⋮ ⋮ ⋮ ⋮ ⋮

the purchase price includes two ancillary companies.

Supertagging

Sent:	the	purchase	price	includes	two	ancillary	companies.
Initial		α_1	α_2	α_3		α_4	α_5
Assig.	β_1	β_2	α_6	α_7	β_3	β_4	α_8
		α_9	α_{10}	α_{11}		α_{12}	α_{13}
		⋮	⋮	⋮		⋮	⋮
Final Assig.	β_1	β_2	α_2	α_{11}	β_3	β_4	α_{13}

- Supertagging: Select most appropriate supertag for each word.
- Supertag disambiguation before parsing.
- Supertag disambiguation results in an "almost parse".

Models for Supertag Disambiguation

- N-gram models
  - Trigram model
  - Head trigram model
- Dependency based model (COLING 94)
  - More like full parsing

Trigram Model for Supertagging

- Find the most likely Supertag sequence for a given word sequence.

$$\hat{T} = \text{argmax}_T \ \Pr(T_1,T_2,\ldots,T_N|W_1,W_2,\ldots,W_N)$$

- By Bayes Rule

$$\hat{T} = \text{argmax}_T \ \frac{\Pr(W_1,W_2,\ldots,W_N|T_1,T_2,\ldots,T_N)*\Pr(T_1,T_2,\ldots,T_N)}{\Pr(W_1,W_2,\ldots,W_N)}$$

- Since the word sequence is given

$$\hat{T} = \text{argmax}_T \ \Pr(W_1,W_2,\ldots,W_N|T_1,T_2,\ldots,T_N)*\Pr(T_1,T_2,\ldots,T_N)$$

Trigram Model for Supertagging

- Contextual probability

$$\text{Pr}(T_1,T_2,\ldots,T_N) \approx \Pi_{i=1}^{N} \text{Pr}(T_i \mid T_{i-2}, T_{i-1})$$

- Word Emit probability

$$\text{Pr}(W_1,W_2,\ldots,W_N|T_1,T_2,\ldots,T_N) \approx \Pi_{i=1}^{N} \text{Pr}(W_i \mid T_i)$$

- Trigram Model

$$\hat{T} = \text{argmax}_T \; \Pi_{i=1}^{N} \text{Pr}(T_i \mid T_{i-2}, T_{i-1}) * \text{Pr}(W_i \mid T_i)$$

where T_i is the supertag for word W_i.

- Unseen events
  - Good-Turing discounting with Katz's Back-off Model.

Training and Test Data

- Training Set A:
  - 200,000 word-supertag pairs
  - collected by bootstrapping and hand correction.
  - WSJ sections 15 through 18
- Training Set B:
  - 1,000,000 word-supertag pairs
  - collected by heuristically mapping from Penn Treebank
  - WSJ sections 0-19 and 21-24
- Test Set: section 20 of WSJ.

Performance of Trigram Supertagger

- Performance of the supertagger on the WSJ corpus
- Correct supertag implies that a word is assigned the same supertag as it would be in the correct parse of the sentence.

Size of training corpus	Size of test corpus	# of words correctly supertagged	% correct
Baseline	47,000	35,391	75.3%
200,000	47,000	42,723	90.9%
1 Million	47,000	43,334	92.2%

- Errors:
  - PP attachment
  - Verbs with more than two complements.

Dependency Based Model

- A supertag is **dependent** on another supertag if the former substitutes or adjoins into the latter.
- Training Data
  - LTAG derivation trees of 5000 Wall Street Journal sentences parsed using the XTAG system.
  - Using (part-of-speech,supertag) pairs.
- Test Data
  - 100 Wall Street Journal Sentences

Dependency Based Model

- Data Representation

(P.O.S,Supertag)	Direction of Dependent Supertag	Ordinal position	Dependent Supertag	Prob
(D,α_1)	()	-	-	-
(N,α_{13})	()	-	-	-
(N,α_2)	$(-)$	-1	α_1	0.975
(V,α_{15})	$(-, +)$	-1	α_{13}	0.700
(V,α_{15})	$(-, +)$	1	α_{13}	0.420

- For example, the fourth entry reads
  - the supertag α_{15}, anchored by a verb (V)
  - has a left and a right dependent $(-,+)$
  - the first word to the left (-1) with the supertag α_{13} serves as a dependent and
  - the strength of this association is represented by the probability 0.700

Dependency Based Model

Sent:	the	purchase	price	includes	two	ancillary	companies.
POS:	D	N	N	V	D	A	N
Initial	α_1	α_2	α_3	α_4	β_1	α_5	α_6
Assig.	α_7	β_2	α_8	α_9	α_{10}	β_3	α_{11}
	α_{12}	α_{13}	α_{14}	α_{15}	α_{16}	α_{17}	α_{18}
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Final Assig.	α_1	β_2	α_3	α_{15}	α_{10}	β_3	α_6

- Every anchor must find its dependents.
- Every dependent must be linked to a anchor.
- No two dependency arcs may cross one another.

Dependency Based Model

- Performance results on Wall Street Journal (WSJ) sentences

Criterion	Total number	Number correct	% correct
Supertags	915	707	77.26%
Dependency links	815	620	76.07%

- Issues:
  - Needs a parsed corpus as training material
  - Attempts at getting a complete linkage
  - Worst-case complexity: $O(n^3)$
  - Lots of parameters to train: $O(S^{2*D^A})$
  - More like parsing than not

Stapler

- Stapler combines the words of a supertagged sentence to yield a dependency linkage.
- Two Approaches
  - XTAG (Earley) parser as a stapler
  - Lightweight Dependency Analyzer as a stapler

Stapler

- Broad coverage grammar slows down the parser tremendously
- Supertagging even before parsing begins
  - Speeds up the parser significantly (by a factor of 30).
  - All words must be tagged with the correct supertag to get a parse.

Lightweight Dependency Analyzer

- Information associated with supertags:
  - Slots: substitution and foot nodes
- Fillers of substitution nodes are argument words and fillers of foot nodes are modified words.
- Two pass algorithm:
  - Establish dependencies for auxiliary supertags
  - Mark all the words that serve as arguments as unavailable for the next pass
  - Establish dependencies for initial supertags.
- Establish dependencies – local search
  - first supertag with root node same as the argument type.

Lightweight Dependency Analyzer

The implicit interior state of the iteration over the hash table entries has dynamic extent

Pos	Word	Supertag	Slot req.	Pass 1	Pass 2	Dep Links
0	The	α_1	–	–	–	–
1	implicit	β_2	+N*	2*		2*
2	interior	β_2	+N*	3*		3*
3	state	α_2	-D.		0.	0.
4	of	β_1	-NP* +NP.	3* 6.		3* 6.
5	the	α_1	–	–	–	–
6	iteration	α_2	-D.		5.	5.
7	over	β_1	-NP* +NP.	6* 11.		6* 11.
8	the	α_1	–	–	–	–
9	hash	β_3	+N*	10*		10*
10	table	β_3	+N*	11*		11*
11	entries	α_2	-D.		8.	8.
12	has	α_3	+NP. -NP.		3. 14.	3. 14.
13	dynamic	β_2	+N*	14*		14*
14	extent	α_4	–	–	–	–

Lightweight Dependency Analyzer

- Trigram supertagger trained on one million supertagged WSJ words.
- Performance on pairwise dependency links
  - A link in output must be in gold standard

Corpus	# of dependency links	# produced by LDA	# correct	Recall	Precision
Brown	140,280	126,493	112,420	80.1%	88.8%
WSJ	47,333	41,009	38,480	82.3%	93.8%

Lightweight Dependency Analyzer

- Test corpus was parsed using the XTAG system
- Performance on pairwise dependency links

Training Size (words)	Test Size (words)	Recall	Precision
200,000	12,000	83.6%	83.5%
1,000,000	12,000	85.0%	85.0%

- Performance at the sentence level
  (Matching against XTAG derivation trees)

	% sentences with 0 errors	% sentences with $\leq$1 error	% sentences with $\leq$2 errors	% sentences with $\leq$3 errors
200K	35%	60.3%	78%	89.8%
1M	40%	63.0%	80.1%	91.0%

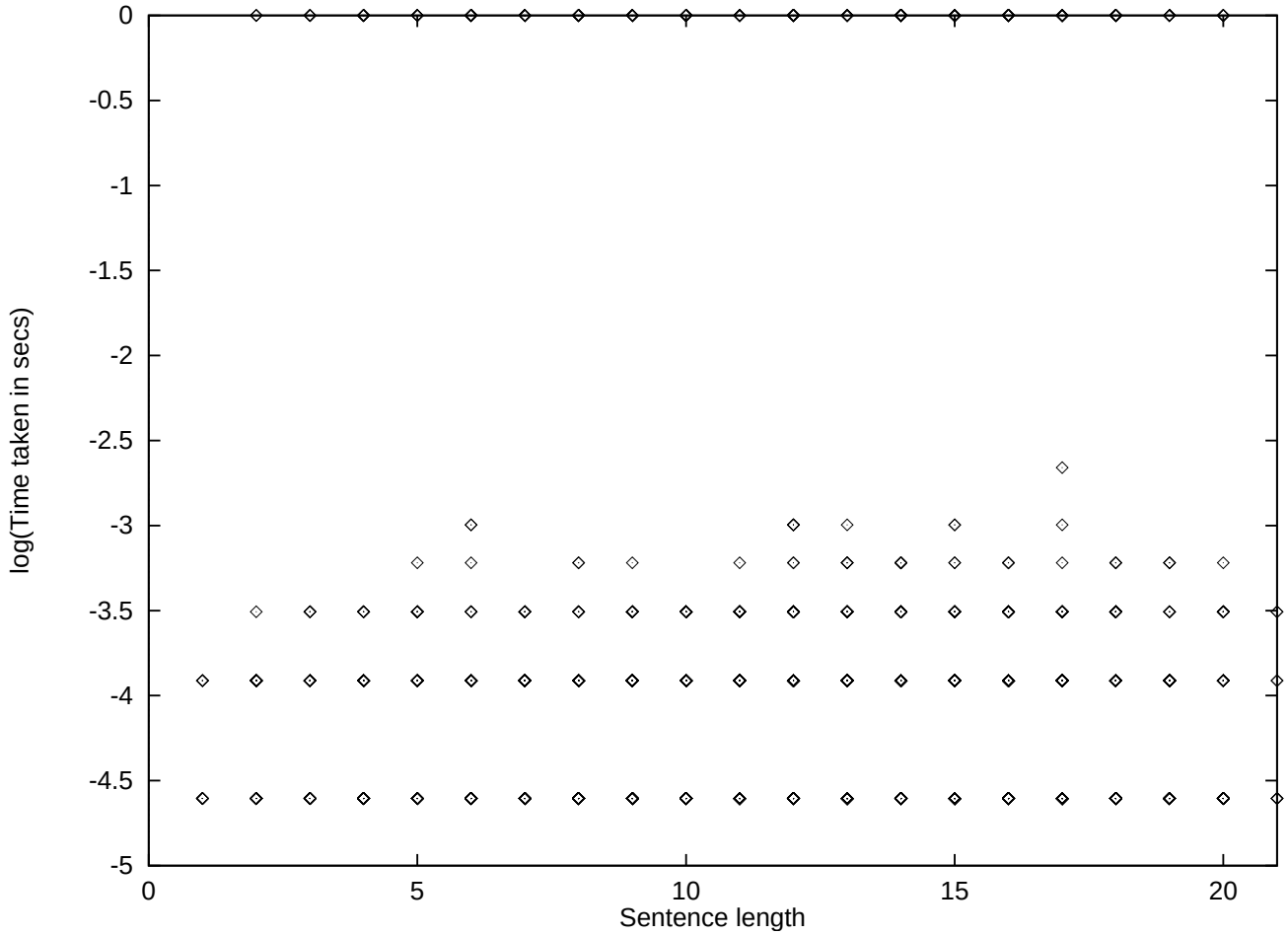
Lightweight Dependency Analyzer

- Evaluation is more strict than evaluation on skeletally bracketed corpus.
- LTAG derivation trees
  - contain internal structure for noun and verb groups.
  - distinguish between arguments and adjuncts.
  - distinguish between predicative and equative readings.
- Derivation tree is much more closer to semantic interpretation than a phrase structure.

SuperTagging and Parsing

- Extract SuperTags from TreeBank
- Oracle experiment: correct SuperTag provided for each word
- Parsed 2250 sentences from the Penn TreeBank
- For all-parses parsing: without SuperTagging time taken was 548K seconds, and in this oracle experiment, 31.2 seconds

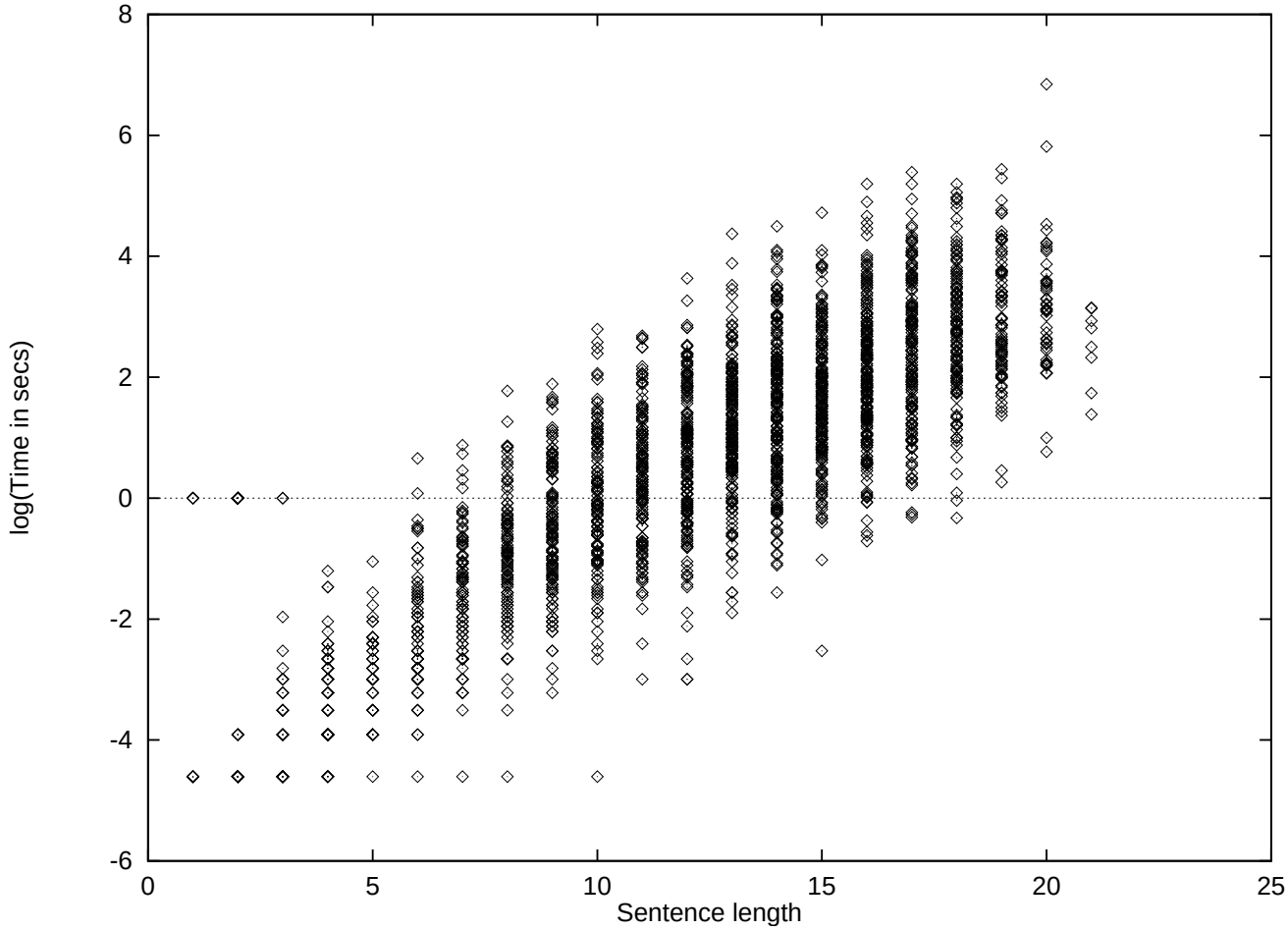
SuperTagging and Parsing



n-best SuperTagging and Parsing

- Extract SuperTags from TreeBank
- n-best experiment: top 60 SuperTags provided for each word using a trigram SuperTagger
- Parsed 2250 sentences from the Penn TreeBank
- For all-parses parsing: without SuperTagging time taken was 548K seconds, and in this n-best experiment, 21K seconds

n-best SuperTagging and Parsing



Overview

- TAGs from word-aligned parallel corpora
- Supertagging and robust, shallow parsing (slides taken from B. Srinivas)
- **Statistical Parsing with LTAG**
- Bootstrapping between CFG and LTAG

Overview

- Task: find the most likely parse for natural language sentences
- Approach: rank alternative parses with statistical methods trained on data annotated by experts (labelled data)
- Focus of this talk:
  1. Machine learning by combining different methods in parsing: PCFG and Tree-adjoining grammar
  2. Weakly supervised learning: combine labelled data with unlabelled data to improve performance in parsing using co-training

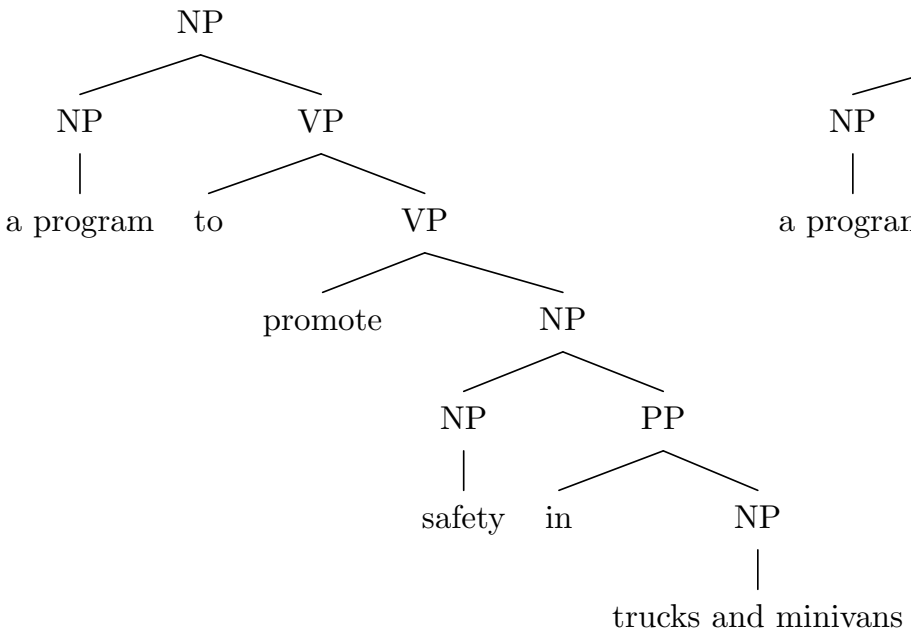
A Key Problem in Processing Language: Ambiguity: (Church and Patil 1982; Collins 1999)

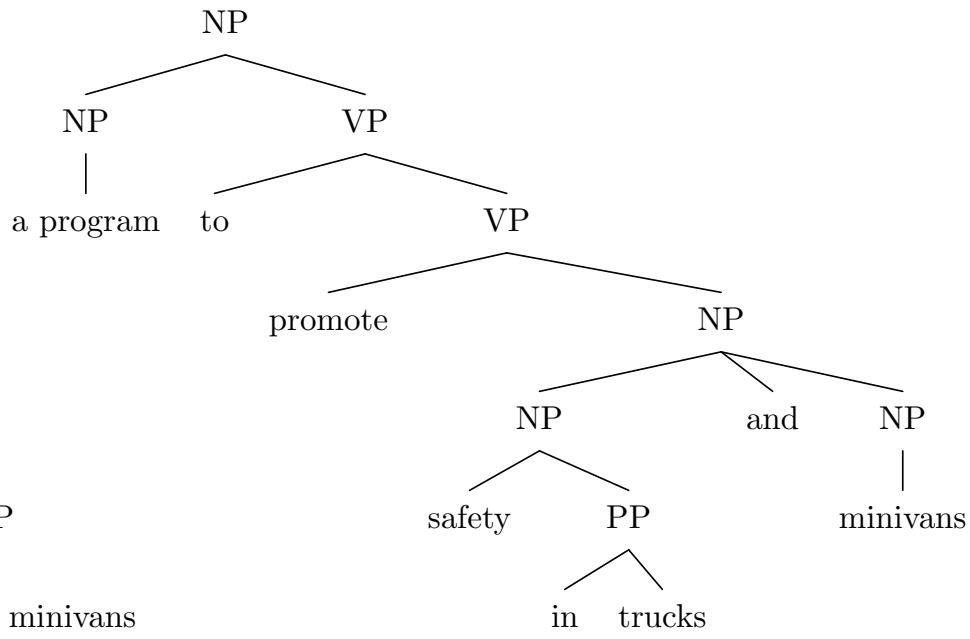
- Part of Speech ambiguity
  `saw` → noun
  `saw` → verb

- Structural ambiguity: Prepositional Phrases
  `I saw (the man) with the telescope`
  `I saw (the man with the telescope)`

- Structural ambiguity: Coordination
  `a program to promote safety in ((trucks) and (minivans))`
  `a program to promote ((safety in trucks) and (minivans))`
  `((a program to promote safety in trucks) and (minivans))`

Ambiguity ← attachment choice in alternative parses





Parsing as a machine learning problem

- S = a sentence
  T = a parse tree
  A statistical parsing model defines $P(T \mid S)$

- Find best parse: $\arg\max_T P(T \mid S)$

- $P(T \mid S) = \frac{P(T,S)}{P(S)} = P(T, S)$

- Best parse: $\arg\max_T P(T, S)$

- e.g. for PCFGs: $P(T, S) = \prod_{i=1\ldots n} P(\mathrm{RHS}_i \mid \mathrm{LHS}_i)$

Parsing as a machine learning problem

- Training data: the Penn WSJ Treebank (Marcus et al. 1993)
- Learn probabilistic grammar from training data
- Evaluate accuracy on test data
- A standard evaluation:
  Train on 40,000 sentences
  Test on 2,300 sentences
- The simplest technique: PCFGs perform badly
  Reason: not sensitive to the words

Machine Learning for ambiguity resolution: prepositional phrases

- What is right analysis for:
  Calvin saw the car on the hill with the telescope

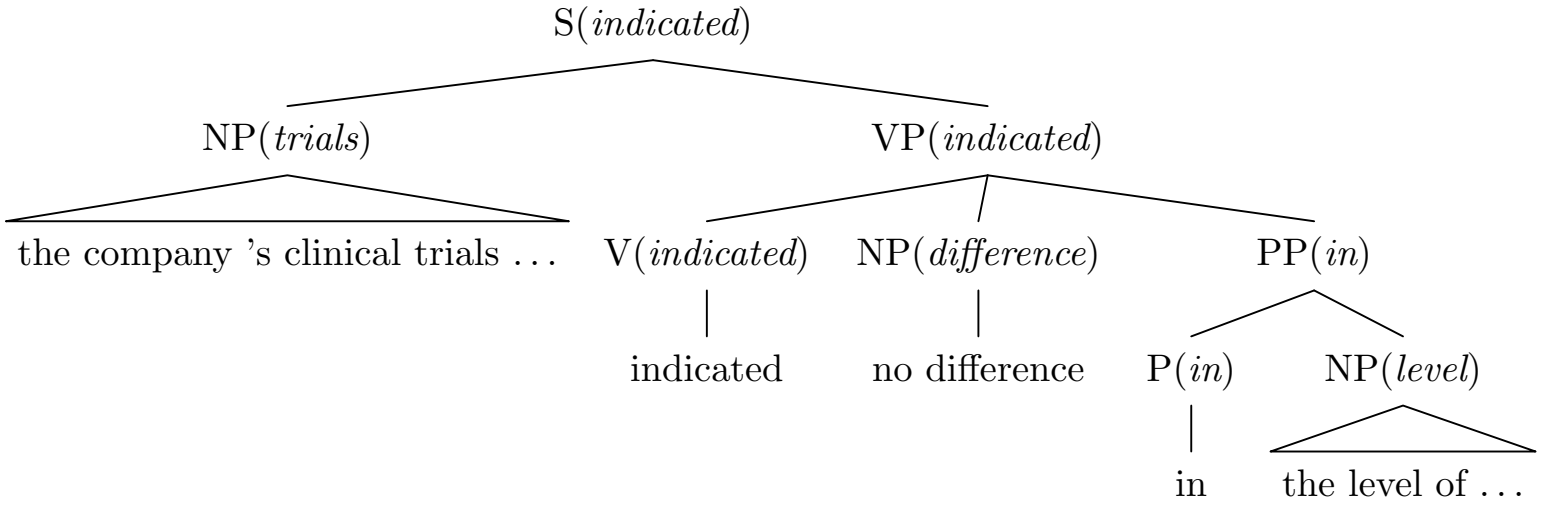
- Compare with:
  Calvin bought the car with anti-lock brakes and
  Calvin bought the car with a loan

- *(bought, with, brakes)* and *(bought, with, loan)* are useful features to solve this apparently AI-complete problem

Method	Accuracy
Always noun attachment	59.0
Most likely for each preposition	72.2
Average Human (4 head words only)	88.2
Average Human (whole sentence)	93.2
Lexicalized Model (Collins and Brooks 1995)	84.5
Lexicalized Model + Wordnet (Stetina and Nagao 1998)	88.0

Statistical Parsing

the company 's clinical trials of both its animal and human-based insulins indicated no difference in the level of hypoglycemia between users of either product



Use a probabilistic *lexicalized* grammar from the Penn WSJ Treebank for parsing ...

Bilexical CFG (Collins-CFG): dependencies between pairs of words

- Full context-free rule:
  VP(*indicated*) → V-hd(*indicated*) NP(*difference*) PP(*in*)

- Each rule is generated in three steps (Collins 1999):

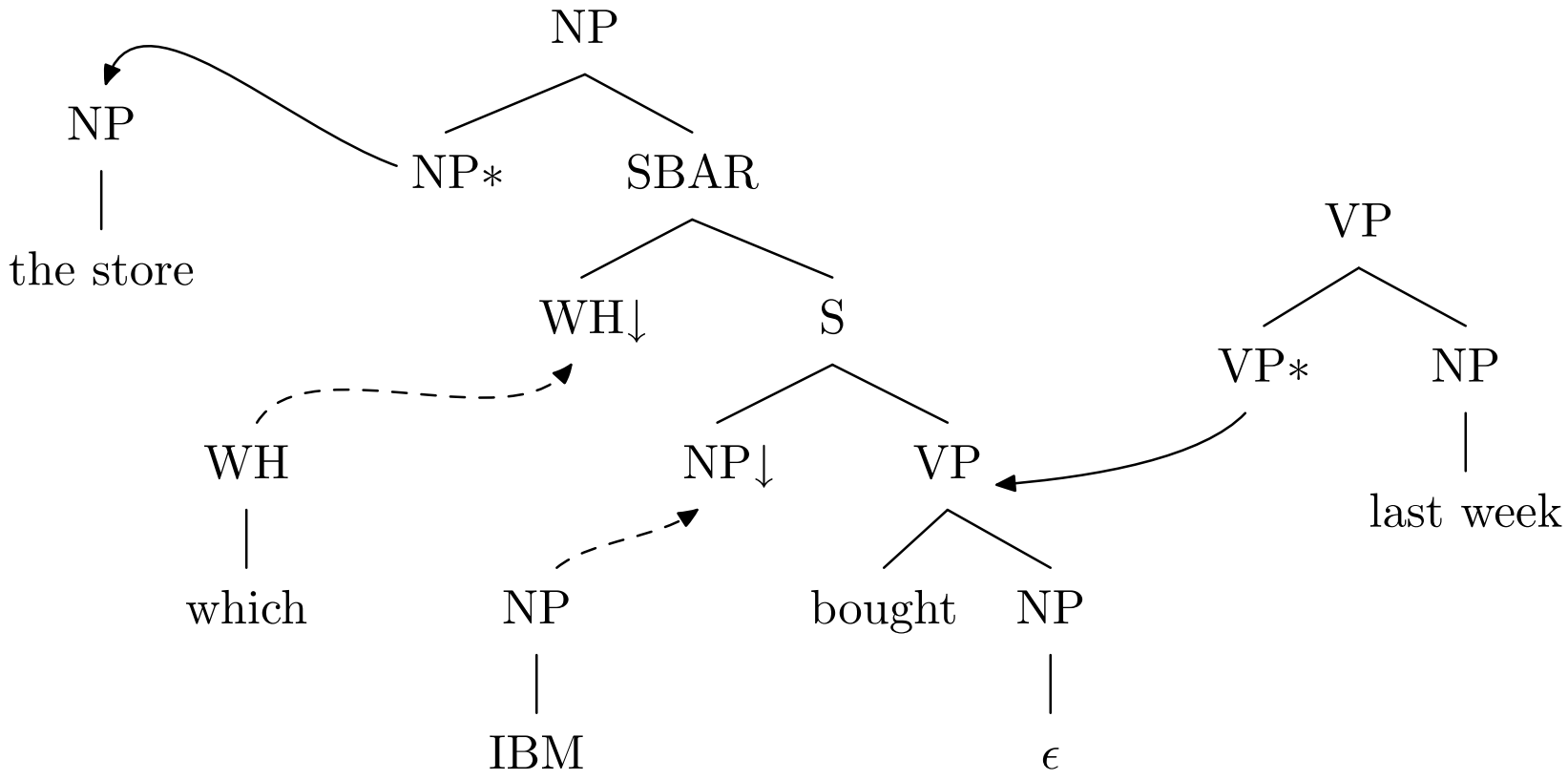
  1. Generate head daughter of LHS: VP(*indicated*) → V-hd(*indicated*)

  2. Generate non-terminals to *left* of head daughter: STOP . . .
     V-hd(*indicated*)

3. Generate non-terminals to *right* of head daughter:

   - V-hd(*indicated*) . . . NP(*difference*)
   - V-hd(*indicated*) . . . PP(*in*)
   - V-hd(*indicated*) . . . STOP

Lexicalized Tree Adjoining Grammars (LTAG):
Different Modeling of Bilexical Dependencies



Performance of supervised statistical parsers

System	$\leq 40 wds$ LP	$\leq 40 wds$ LR	$\leq 100 wds$ LP	$\leq 100 wds$ LR
PCFG (Collins 99)	88.5	88.7	88.1	88.3
LTAG (Chiang 02)	88.63	88.59	87.72	87.66
PCFG (Charniak 97)	87.5	87.4	86.7	86.6
PCFG (Charniak 99)	90.1	90.1	89.6	89.5

- Labelled Precision $= \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in proposed parse}}$

- Labelled Recall $= \frac{\text{number of correct constituents in proposed parse}}{\text{number of constituents in treebank parse}}$

Overview

- TAGs from word-aligned parallel corpora
- Supertagging and robust, shallow parsing (slides taken from B. Srinivas)
- Statistical Parsing with LTAG
- **Bootstrapping between CFG and LTAG**

Bootstrapping

- Current state-of-the-art in parsing on the Penn WSJ Treebank dataset is approx 90% accuracy

- However this accuracy is obtained with 1M words of human annotated data (40K sentences)

- Exploring methods that can exploit unlabelled data is an important goal:

  – What about different languages? The Penn Treebank took several years with many linguistic experts and millions of dollars to produce. Unlikely to happen for all other languages of interest.

  - What about different genres? Porting a parser trained on newspaper text and using it on fiction is a challenge.

  - Combining labelled and unlabelled data is an interesting challenge for machine learning.

- In this talk, we will consider *bootstrapping* using unlabelled data.

- Bootstrapping refers to a problem setting in which one is given a small set of labelled data and a large set of unlabelled data, and the task is to extract new labelled instances from the unlabelled data.

- The noise introduced by the new automatically labelled instances has to be offset by the utility of training on those instances.

Multiple Learners and the Bootstrapping problem

- With a single learner, the simplest method of bootstrapping is called *self-training*.
- The high precision output of a classifier can be treated as new labelled instances (Yarowsky, 1995).
- With multiple learners, we can exploit the fact that they might:
  - Pay attention to different features in the labelled data.
  - Be confident about different examples in the unlabelled data.
  - Combine multiple learners using the *co-training* algorithm.

Co-training

- Pick two “views” of a classification problem.
- Build separate models for each of these “views” and train each model on a small set of labelled data.
- Sample an unlabelled data set and to find examples that each model independently labels with high confidence.
- Pick confidently labelled examples and add to labelled data. Iterate.
- Each model labels examples for the other in each iteration.

An Example: (Blum and Mitchell 1998)

- Task: Build a classifier that categorizes web pages into two classes, $+$: *is a course web page*, $-$: *is not a course web page*

- Usual model: build a Naive Bayes model:

$$
\begin{aligned}
P[C = c_k \mid X = \mathbf{x}] &= \frac{P(c_k) \times P(\mathbf{x} \mid c_k)}{P(\mathbf{x})} \\
P(\mathbf{x} \mid c_k) &= \prod_{x_j \in \mathbf{x}} P(x_j \mid c_k)
\end{aligned}
$$

- Each labelled example has two views:

  $\mathbf{x}_1$ Text in hyperlink: `<a href="...">CSE 120, Fall semester</a>`

  $\mathbf{x}_2$ Text in web page: `<html>...Assignment #1...</html>`

- Documents in the unlabelled data where $C = c_k$ is predicted with high confidence by classifier trained on view $\mathbf{x}_1$ can be used as new training data for view $\mathbf{x}_2$ and vice versa

- Each view can be used to create new labelled data for the other view.

- Combining labelled and unlabelled data in this manner outperforms using only the labelled data.

Theory behind co-training: (Abney, 2002)

- For each instance x, we have two views $X_1(x) = x_1, X_2(x) = x_2$. x_1, x_2 satisfy *view independence* if:

$$\begin{aligned} Pr[X_1 = x_1 \mid X_2 = x_2, Y = y] &= Pr[X_1 = x_1 \mid Y = y] \\ Pr[X_2 = x_2 \mid X_1 = x_1, Y = y] &= Pr[X_2 = x_2 \mid Y = y] \end{aligned}$$

- If $\mathcal{H}_1, \mathcal{H}_2$ are rules that use only X_1, X_2 respectively, then *rule independence* is:

$$Pr[F = u \mid G = v, Y = y] = Pr[F = u \mid Y = y]$$

where $F \in \mathcal{H}_1$ and $G \in \mathcal{H}_2$ (note that view independence implies rule independence)

Theory behind co-training: (Abney, 2002)

- Deviation from conditional independence:

$$d_y = \frac{1}{2} \sum_{u,v} \mid Pr[G = v \mid Y = y, F = u] - Pr[G = v \mid Y = y] \mid$$

- For all $F \in \mathcal{H}_1, G \in \mathcal{H}_2$ such that

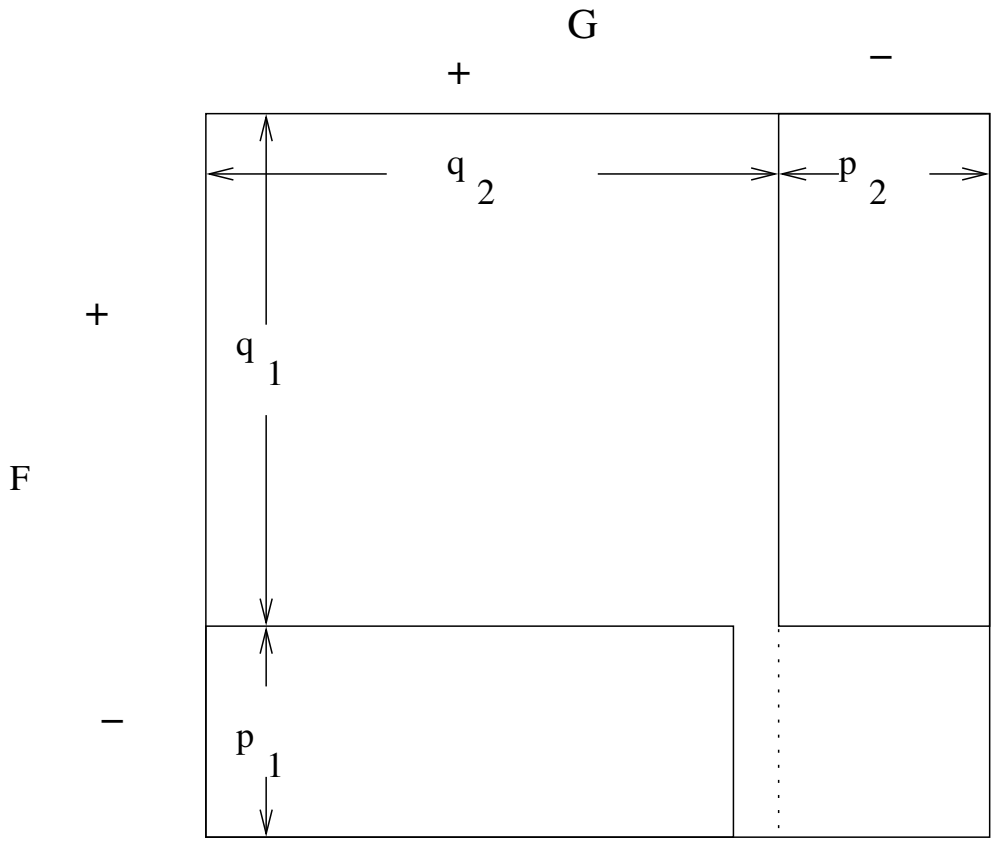
$$d_y \leq p_2 \frac{q_1 - p_1}{2 p_1 q_1}$$

and $\min_u Pr[F = u] > Pr[F \neq G]$ then

$$\begin{aligned} Pr[F \neq Y] &\leq Pr[F \neq G] \\ Pr[\bar{F} \neq Y] &\leq Pr[F \neq G] \end{aligned}$$

we can choose between F and $\bar{F}$ using seed labelled data

Theory behind co-training: $Pr[F \neq Y] \leq Pr[F \neq G]$



Positive Correlation, Y = +

Theory behind co-training

- (Blum and Mitchell, 1998) prove that, when the two views are *conditionally independent* given the label, and each view is sufficient for learning the task, co-training can improve an initial weak learner using unlabelled data.

- (Dasgupta et al, 2002) show that maximising the agreement over the unlabelled data between two learners leads to few generalisation errors (same independence assumption).

- (Abney, 2002) argues that the independence assumption is extremely restrictive and typically violated in the data. He proposes a weaker independence assumption and a greedy algorithm that maximises agreement on unlabelled data.

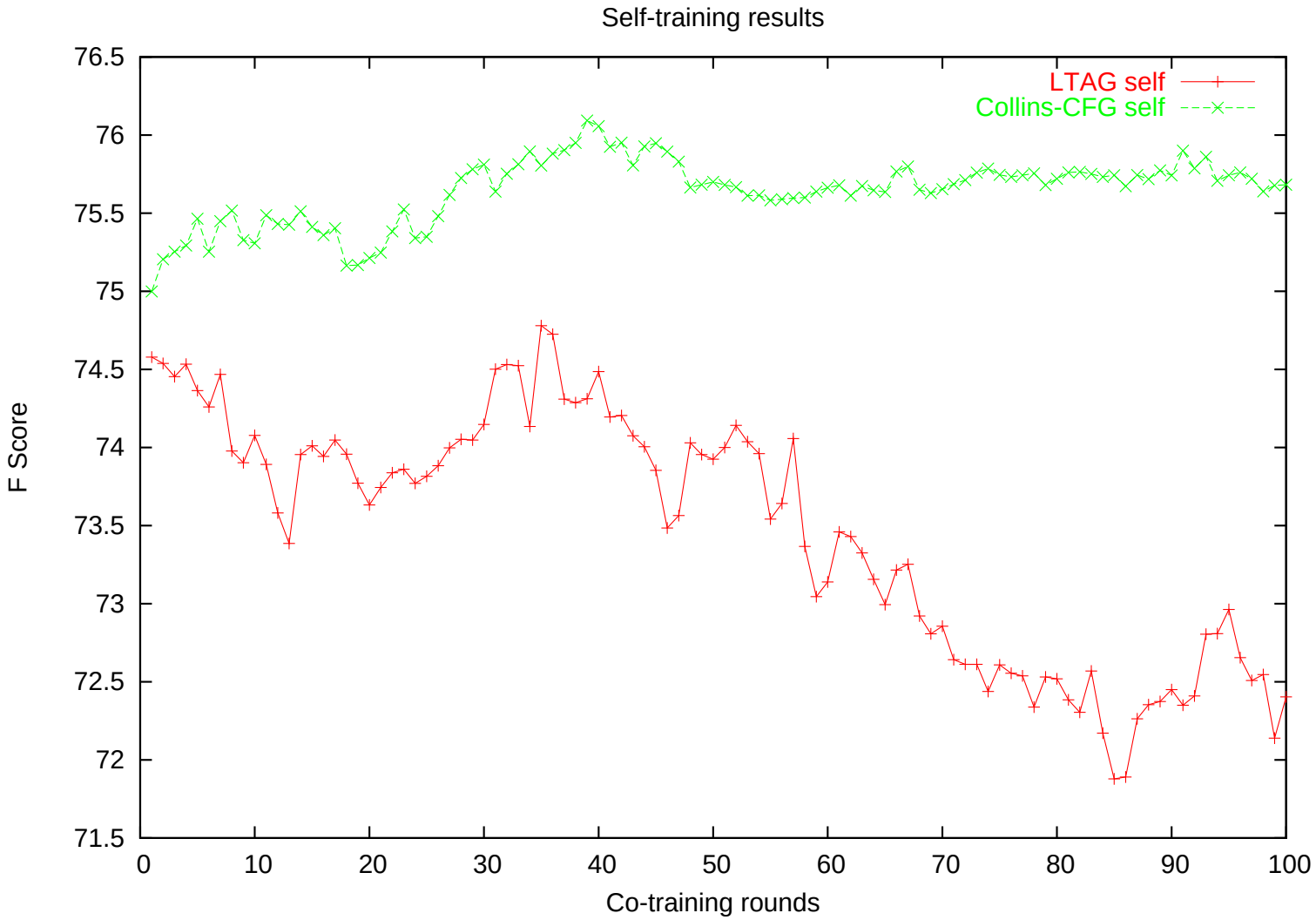
Co-training for statistical parsing In order to conduct co-training experiments between statistical parsers, it was necessary to choose two parsers that generate comparable output but use different statistical models.

1. The Collins lexicalized PCFG parser (Collins, 1999), model 2. Some code for (re)training this parser was added to make the co-training experiments possible. We refer to this parser as **Collins-CFG**.

2. The Lexicalized Tree Adjoining Grammar (LTAG) parser of (Sarkar, 2001), which we refer to as the **LTAG** parser.

Summary of the Different Views

Collins-CFG	LTAG
Bi-lexical dependencies are between lexicalized nonterminals	Bi-lexical dependencies are between elementary trees
Can produce novel elementary trees for the LTAG parser	Can produce novel bi-lexical dependencies for Collins-CFG
Using small amounts of seed data, abstains less often than LTAG	Using small amounts of seed data, abstains more often than Collins-CFG


Self-training results
LTAG self
Collins-CFG self
F Score
Co-training rounds
76.5
76
75.5
75
74.5
74
73.5
73
72.5
72
71.5
0
10
20
30
40
50
60
70
80
90
100

The pseudo-code for the co-training algorithm

A and B are two different parsers.
M_A^i and M_B^i are models of A and B at step i.
U is a large pool of unlabelled sentences.
U^i is a small cache holding subset of U at step i.
L is the manually labelled seed data.
L_A^i and L_B^i are the labelled training examples
for A and B at step i.

Initialize:

$$L_A^0 \leftarrow L_B^0 \leftarrow L.$$
$$M_A^0 \leftarrow Train(A, L_A^0)$$
$$M_B^0 \leftarrow Train(B, L_B^0)$$

Loop:

$U^i \leftarrow$ Add unlabelled sentences from U.

M_A^i and M_B^i parse the sentences in U^i
and assign scores to them according to
their scoring functions f_A and f_B.

Select new parses $\{P_A\}$ and $\{P_B\}$
according to some selection method S,
which uses the scores from f_A and f_B.

L_A^{i+1} is L_A^i augmented with $\{P_B\}$

L_B^{i+1} is L_B^i augmented with $\{P_A\}$

$M_A^{i+1} \leftarrow Train(A, L_A^{i+1})$

$M_B^{i+1} \leftarrow Train(B, L_B^{i+1})$

Experiments

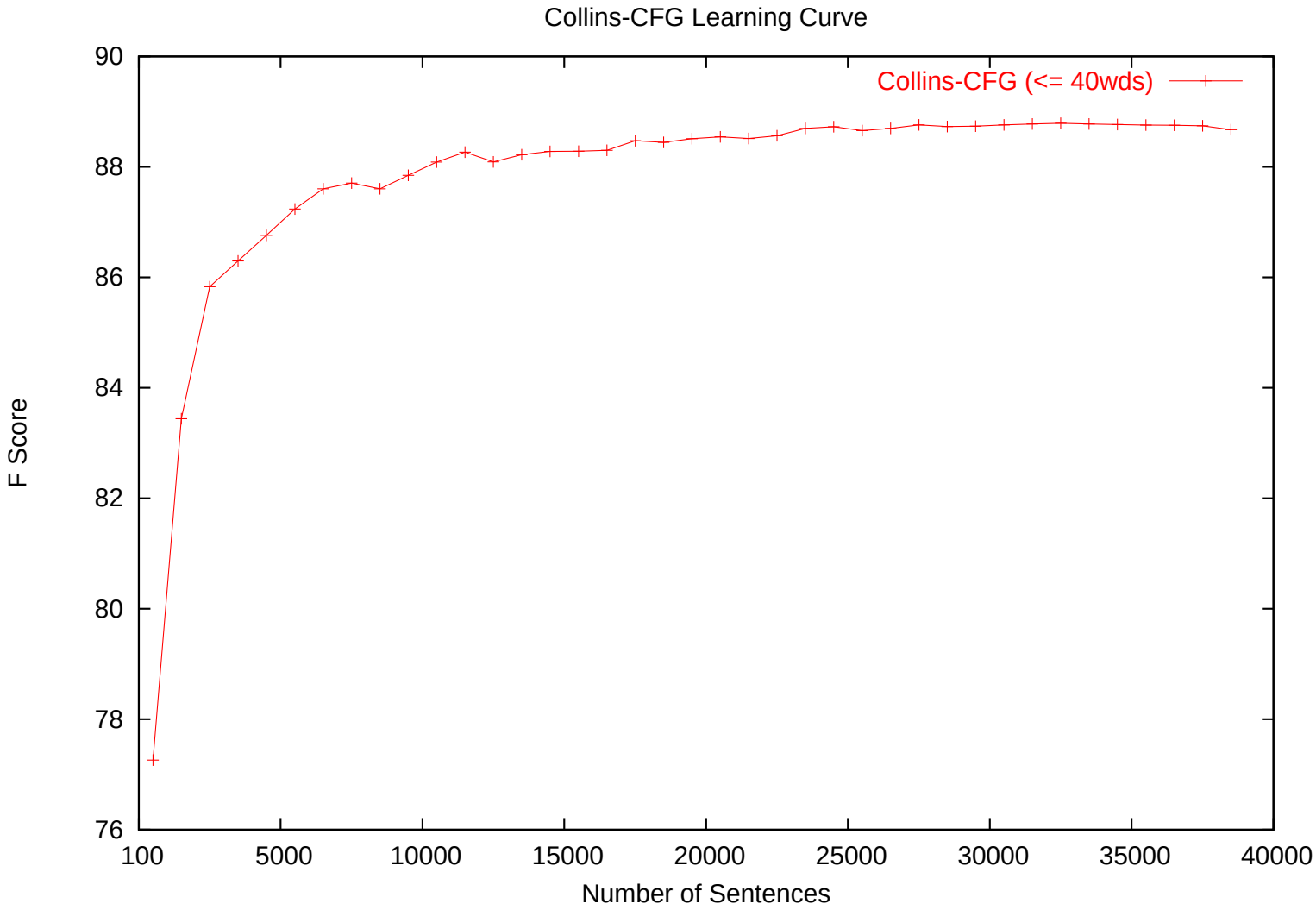
- Use co-training to boost performance, when faced with small seed data
  → Use small subsets of WSJ labelled data as seed data

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

- Use a large set of labelled data and use unlabelled data to improve parsing performance
  → Use Penn Treebank (40K sents) as seed data

Experiments on Small Labelled Seed Data

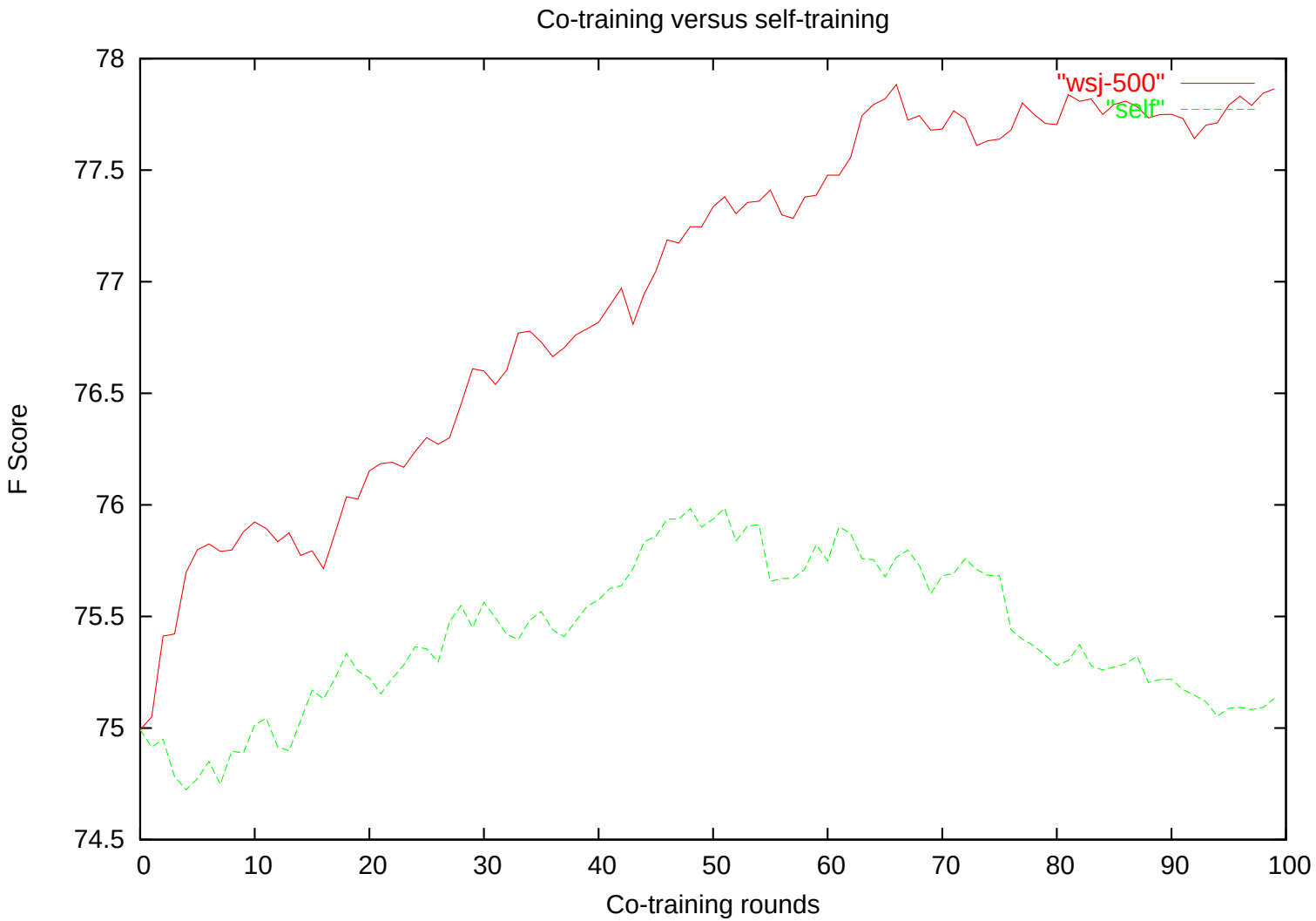
- Motivating the size of the initial seed data set
- We plotted learning curves, tracking parser accuracy while varying the amount of labelled data
- Find the “elbow” in the curve where the payoff will occur
- This was done for both the Collins-CFG and the LTAG parser
- The learning curve shows that the maximum payoff from co-training is likely to occur between 500 and 1,000 sentences.


Collins-CFG Learning Curve
Collins-CFG (<= 40wds)
F Score
Number of Sentences
90
88
86
84
82
80
78
76
100
5000
10000
15000
20000
25000
30000
35000
40000

- Use co-training to boost performance, when faced with small seed data
  → Use 500 sentences of WSJ labelled data as seed data
  → Compare performance of co-training vs. self-training

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

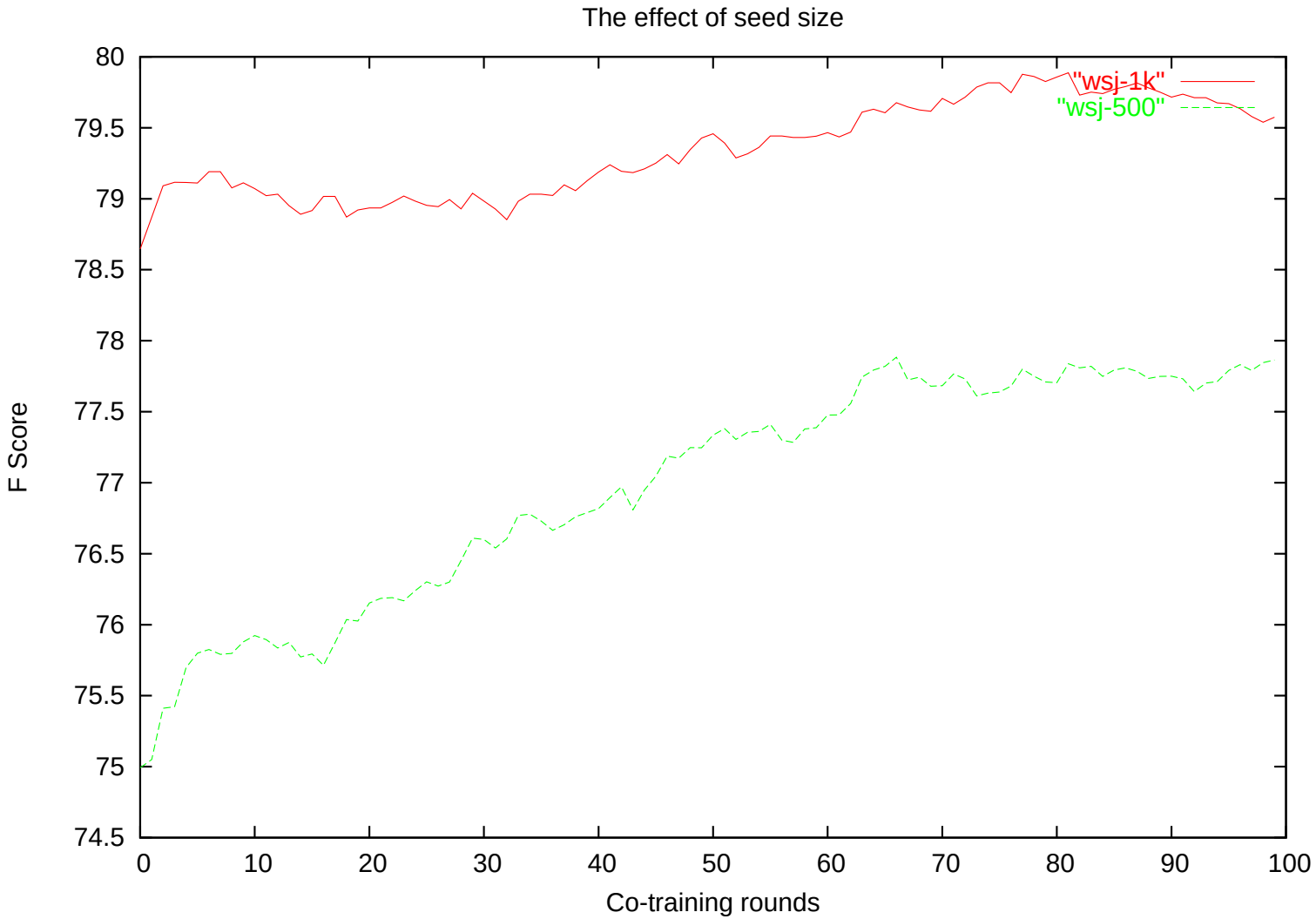
- Use a large set of labelled data and use unlabelled data to improve parsing performance
  → Use Penn Treebank (40K sents) as seed data


Co-training versus self-training
"wsj-500"
"self"
F Score
Co-training rounds
78
77.5
77
76.5
76
75.5
75
74.5
0
10
20
30
40
50
60
70
80
90
100

- Use co-training to boost performance, when faced with small seed data
  → Co-training beats self-training with 500 sentence seed data
  → Compare performance when seed data is doubled to 1K sentences

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

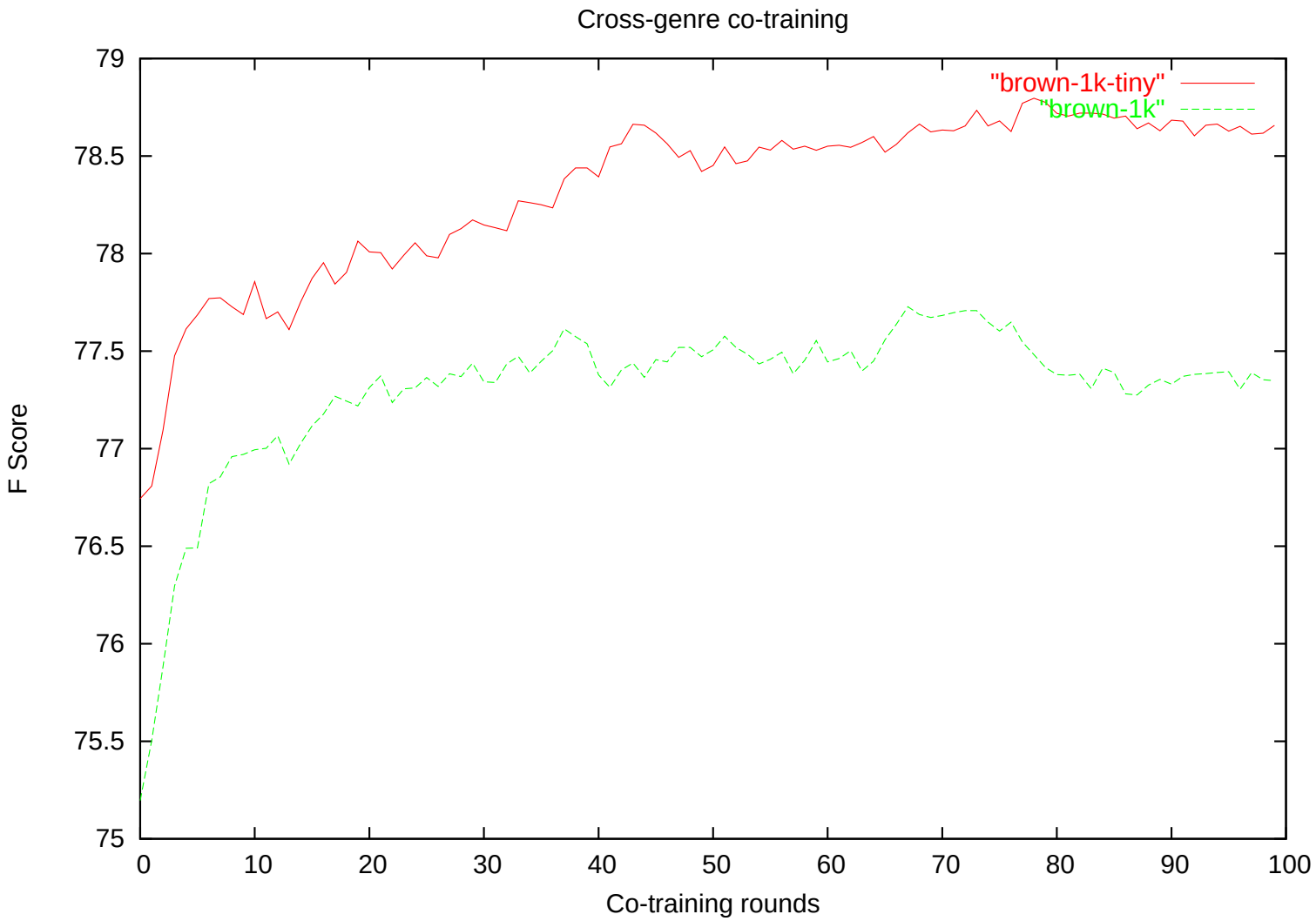
- Use a large set of labeled data and use unlabeled data to improve parsing performance
  → Use Penn Treebank (40K sents) as seed data


The effect of seed size
F Score
Co-training rounds
"wsj-1k"
"wsj-500"
80
79.5
79
78.5
78
77.5
77
76.5
76
75.5
75
74.5
0
10
20
30
40
50
60
70
80
90
100

- Use co-training to boost performance, when faced with small seed data
  → Co-training beats self-training with 500 sentence seed data
  → Co-training still improves performance with 1K sentence seed data

- Use co-training to port parsers to new genres
  → 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
  → Use Penn Treebank (40K sents) as seed data


Cross-genre co-training
"brown-1k-tiny"
"brown-1k"
F Score
Co-training rounds
79
78.5
78
77.5
77
76.5
76
75.5
75
0
10
20
30
40
50
60
70
80
90
100

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

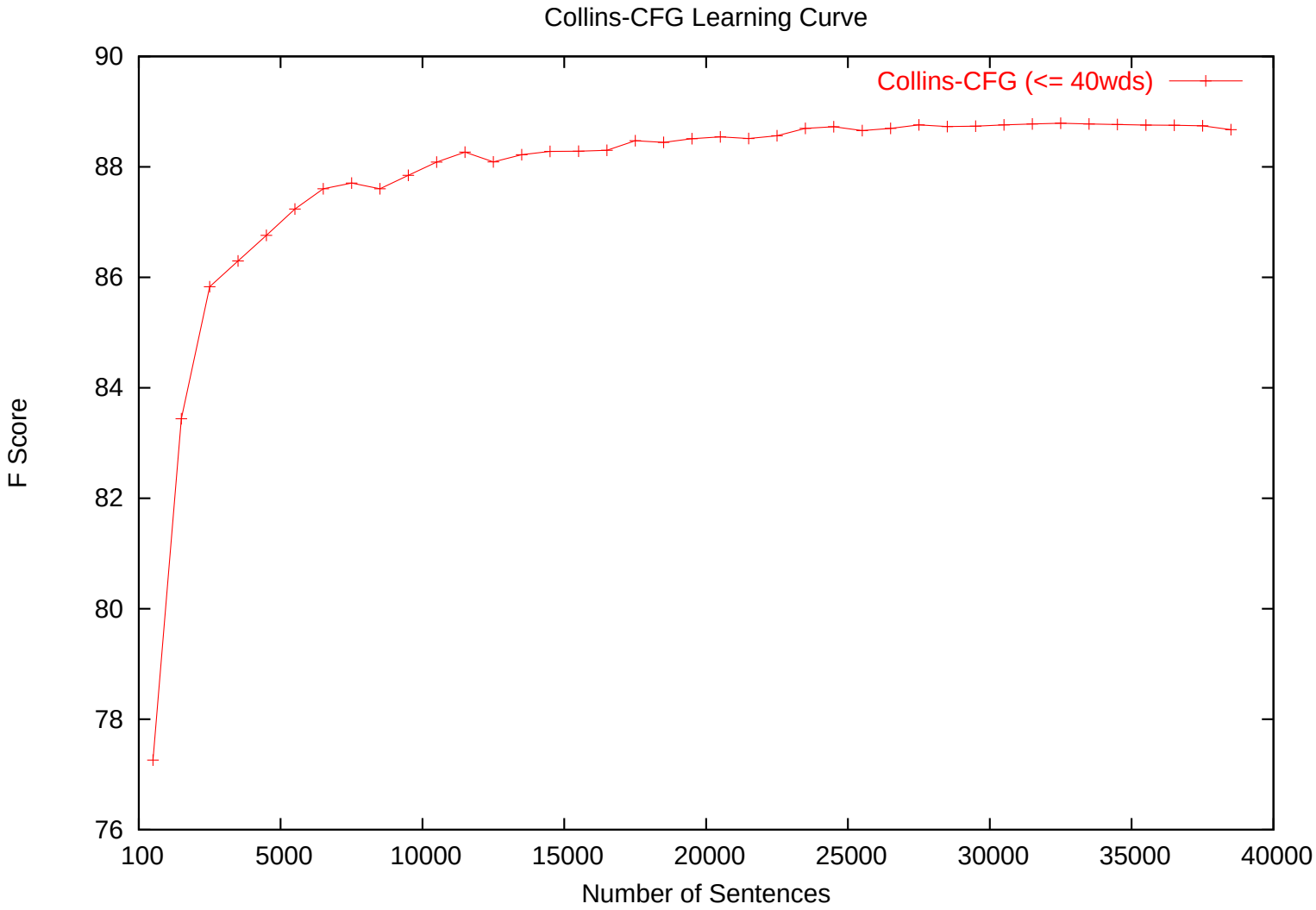
- Use co-training to port parsers to new genres
  → Co-training improves performance significantly when porting from one genre (Brown) to another (WSJ)

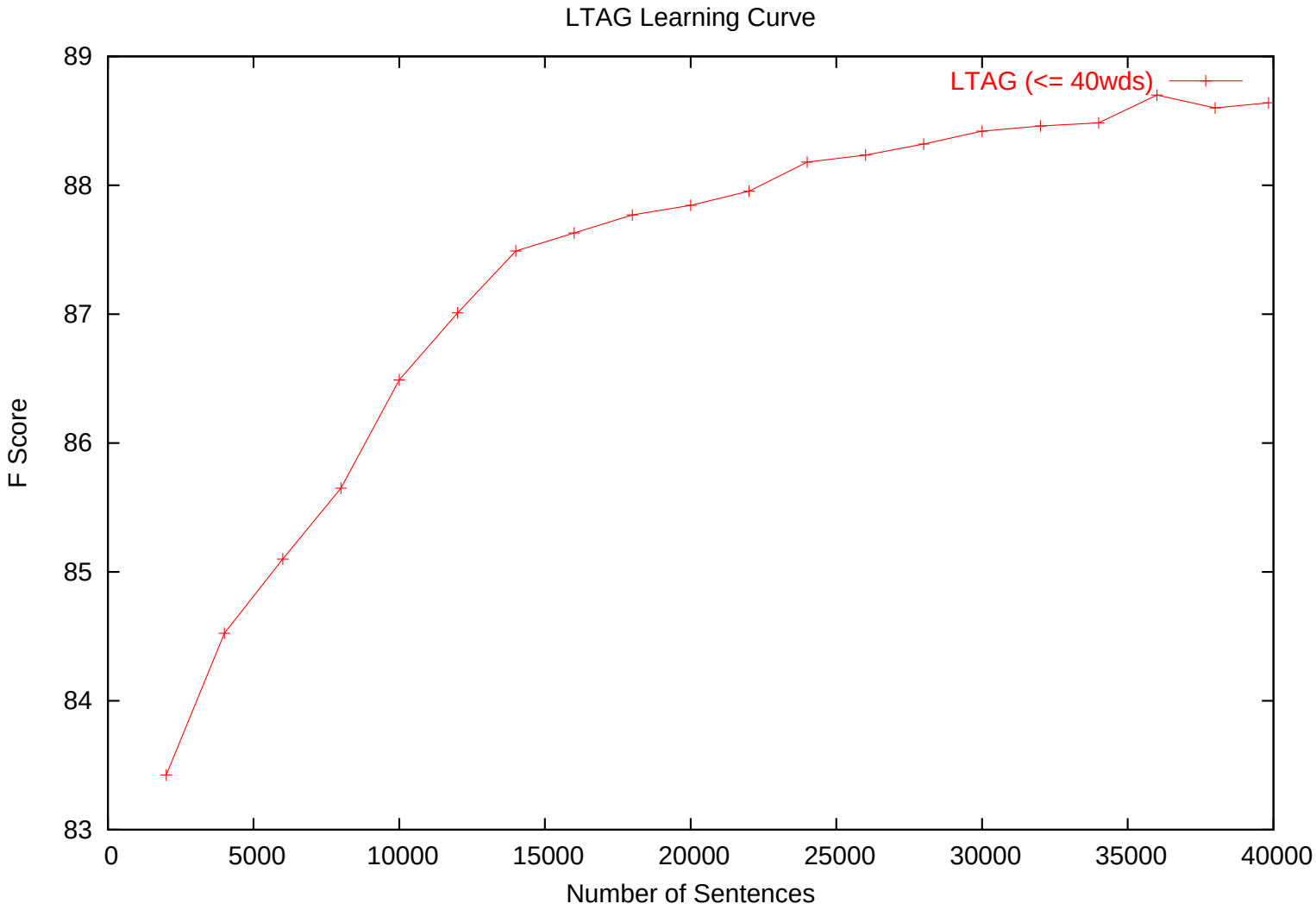
- Use a large set of labeled data and use unlabeled data to improve parsing performance
  → Use Penn Treebank (40K sents) as seed data

- 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


Collins-CFG Learning Curve
Collins-CFG (<= 40wds)
F Score
Number of Sentences
90
88
86
84
82
80
78
76
100
5000
10000
15000
20000
25000
30000
35000
40000


LTAG Learning Curve
LTAG (<= 40wds)
F Score
Number of Sentences
89
88
87
86
85
84
83
0
5000
10000
15000
20000
25000
30000
35000
40000

Summary

Experiment	Before(Sec 23)	After(Sec 23)
WSJ Self-training	74.4	74.3
WSJ (500) Co-training	74.4	76.9
WSJ (1k) Co-training	78.6	79.0
Brown co-training	73.6	76.8
Brown+ small WSJ co-training	75.4	78.2

- Use co-training to boost performance, when faced with small seed data
  → Co-training beats self-training with 500 sentence seed data
  → Co-training still improves performance with 1K sentence seed data

- Use co-training to port parsers to new genres
  → 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
  → Using 40K sentences of Penn Treebank as seed data showed no improvement over the baseline. Future work: improving LTAG performance

- Acknowledgements

  The co-training experiments were done for the most part during the NSF/DARPA JHU Language Engineering Summer Workshop 2002. This is joint work with R. Hwa, M. Osborne, M. Steedman, S. Clark, J. Hockenmaier, P. Ruhlen, S. Baker, J. Crim

- For more details about this work:

  - Bootstrapping Statistical Parsers from Small Datasets: EACL 2003
  - Example Selection for Bootstrapping Statistical Parsers: NAACL 2003