Bootstrapping
a Classifier
Using the Yarowsky Algorithm

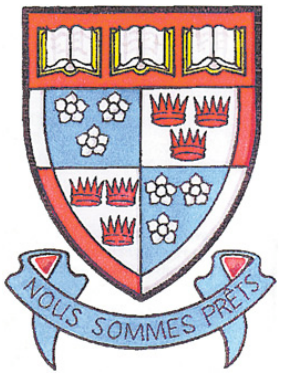
Anoop Sarkar

On Sabbatical at U. of Edinburgh (Informatics 4.18b)

Simon Fraser University

Vancouver, Canada

natlang.cs.sfu.ca



October 2, 2009

Acknowledgements

- This is joint work with my students Gholamreza Haffari (Ph.D.) and Max Whitney (B.Sc.) at SFU.
- Thanks to Michael Collins for providing the named-entity dataset and answering our questions.
- Thanks to Damianos Karakos and Jason Eisner for providing the word sense dataset and answering our questions.

Bootstrapping

Self-Training

1. A base model is trained with a small/large amount of labeled data.
2. The base model is then used to classify the unlabeled data.
3. Only the most confident unlabeled points, along with the predicted labels, are incorporated into the labeled training set (*pseudo-labeled data*).
4. The base model is re-trained, and the process is repeated.

Self-Training

- It can be applied to *any* base learning algorithm: only need confidence weights for predictions.

- Differences with EM:
  - Self-training only uses the mode of prediction distribution.
  - Unlike hard-EM, it can abstain: "I do not know the label".

- Differences with Co-training:
  - In co-training there are two views, in each of which a model is learned.
  - The model in one view trains the model in another view by providing pseudo-labeled examples.

Bootstrapping

- Start with a few *seed rules* (typically high precision, low recall). Build initial classifier.
- Use classifier to label unlabeled data.
- Extract new rules from *pseudo-labeled data* and build classifier for next iteration.
- Exit if labels for unlabeled data are unchanged. Else, apply classifier to unlabeled data and continue.

Decision List (DL)

- A *Decision List* is an ordered set of rules.
  - Given an instance x, the first applicable rule determines the class label.
- Instead of ordering the rules, we can give weight to them.
  - Among all applicable rules to an instance x, apply the rule which has the highest weight.
- The parameters are the weights which specify the ordering of the rules.

Rules: If x has feature f → class k , $\theta_{f,k}$

parameters

DL for Word Sense Disambiguation

(Yarowsky 1995)

- WSD: Specify the most appropriate sense (meaning) of a word in a given sentence.

➢ Consider these two sentences:

- … company said the plant is still operating.

(company, factory, operating) sense +

- … and divide life into plant and animal kingdom.

(life, living, animal) sense -

If company	➔	+1 ,	confidence weight .97
If life	➔	–1 ,	confidence weight .96
…			

Sorted

Example: disambiguate 2 senses of sentence

- Seed rules:

  If context contains *served*, label +1, conf = 1.0

  If context contains *reads*, label -1, conf = 1.0

- Seed rules label 8 out of 303 unlabeled examples

- These 8 pseudo-labeled examples provide 6 rules above 0.95 threshold (including the original seed rules) e.g.

  If context contains *read*, label -1, conf = 0.953

- These 6 rules label 151 out of 303 unlabeled examples

Example: disambiguate 2 senses of sentence

- These 151 pseudo-labeled examples provide 60 rules above the threshold, e.g.

  If context contains *prison*, label +1, conf = 0.989

  If prev word is *life*, label +1, conf = 0.986

  If prev word is *his*, label +1, conf = 0.983

  If next word is *from*, label -1, conf = 0.982

  If context contains *relevant*, label -1, conf = 0.953

  If context contains *page*, label -1, conf = 0.953

- After 5 iterations, 297/303 unlabeled examples are permanently labeled (no changes possible)
- Building final classifier gives 67% accuracy on test set of 515 sentences. With some "tricks" we can get 76% accuracy.

Brief History of Bootstrapping

- (Yarowsky 1995) used it with Decision List base classifier for Word Sense Disambiguation (WSD) task.
  - It achieved the same performance level as the supervised algorithm, using only a few seed examples as labeled training data.
- (Collins & Singer 1999) used it for Named Entity Classification task with Decision List base classifier.
  - Using only 7 initial rules, it achieved 91% accuracy.
  - It achieved the same performance level as Co-training (no need for 2 views).
- (Abney ACL 2002) in a paper about co-training contrasts it with the Yarowsky algorithm. Initial analysis abandoned later.

Brief History of Bootstrapping

- (Abney CL 2004) provided a new analysis of the Yarowsky algorithm.
  - It could not mathematically analyze the original Yarowsky algorithm, but introduced new variants (we will see them later).

- (Haffari & Sarkar UAI 2007) advanced Abney's analysis and gave a general framework that showed how the Yarowsky algorithm introduced by Abney is related to other SSL methods.

- (Eisner and Karakos 2005) examines the construction of seed rules for bootstrapping.

Analysis of the Yarowsky Algorithm

Original Yarowsky Algorithm

(Yarowsky 1995)

- The Yarowsky algorithm is a bootstrapping algorithm with a Decision List base classifier.
- The predicted label is k* if the confidence of the applied rule is above some threshold η.
- An instance may become unlabeled in future iterations.

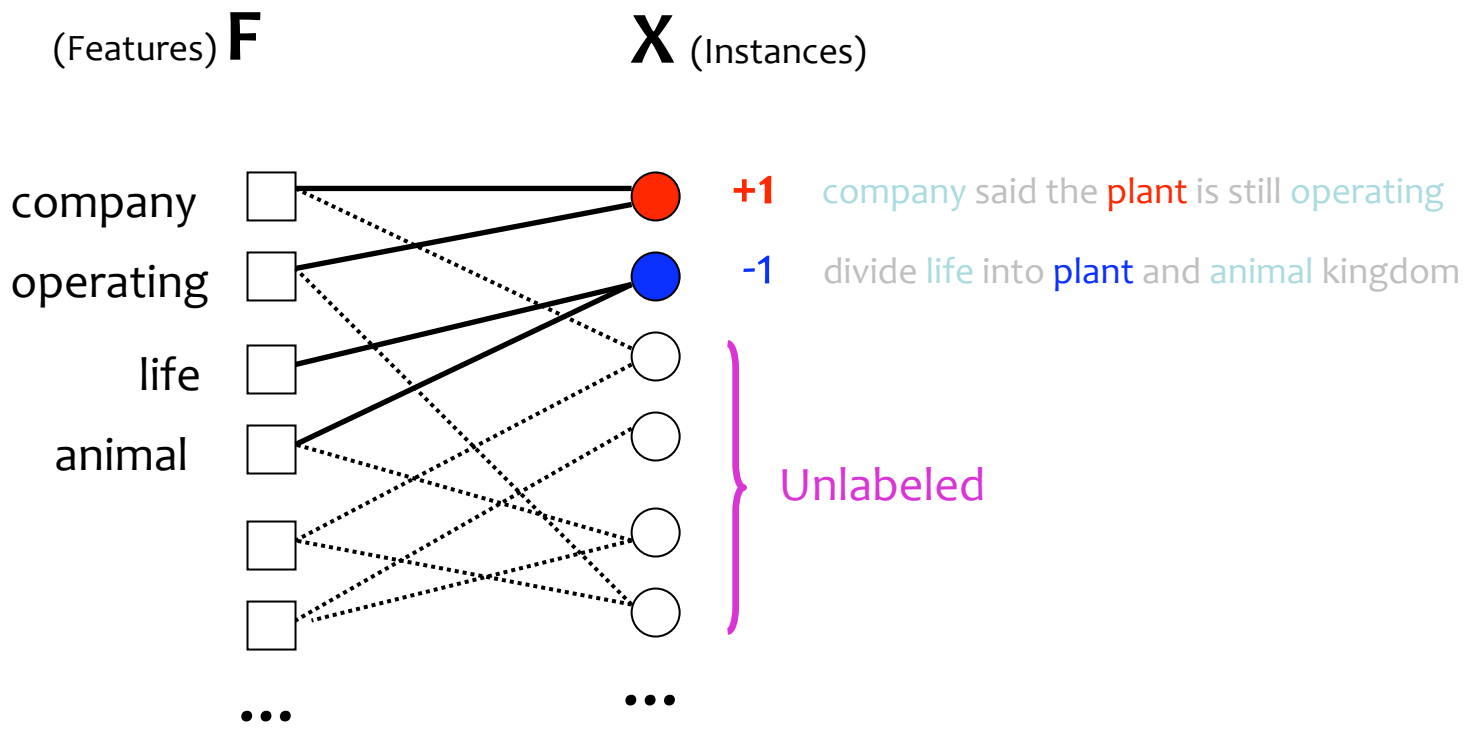
Modified Yarowsky Algorithm

(Abney 2004)

- Instead of the feature with the max score we use the sum of the scores of all features active for an example to be labeled.
- The predicted label is k* if the confidence of the applied rule is above the threshold 1/K.
  - K: is the number of labels.
- An instance must stay labeled once it becomes labeled, but the label may change.
- These are the conditions in all the algorithms we will analyze in the rest of the talk.
  - Analyzing the original Yarowsky algorithm is still an open question.

Bipartite Graph Representation

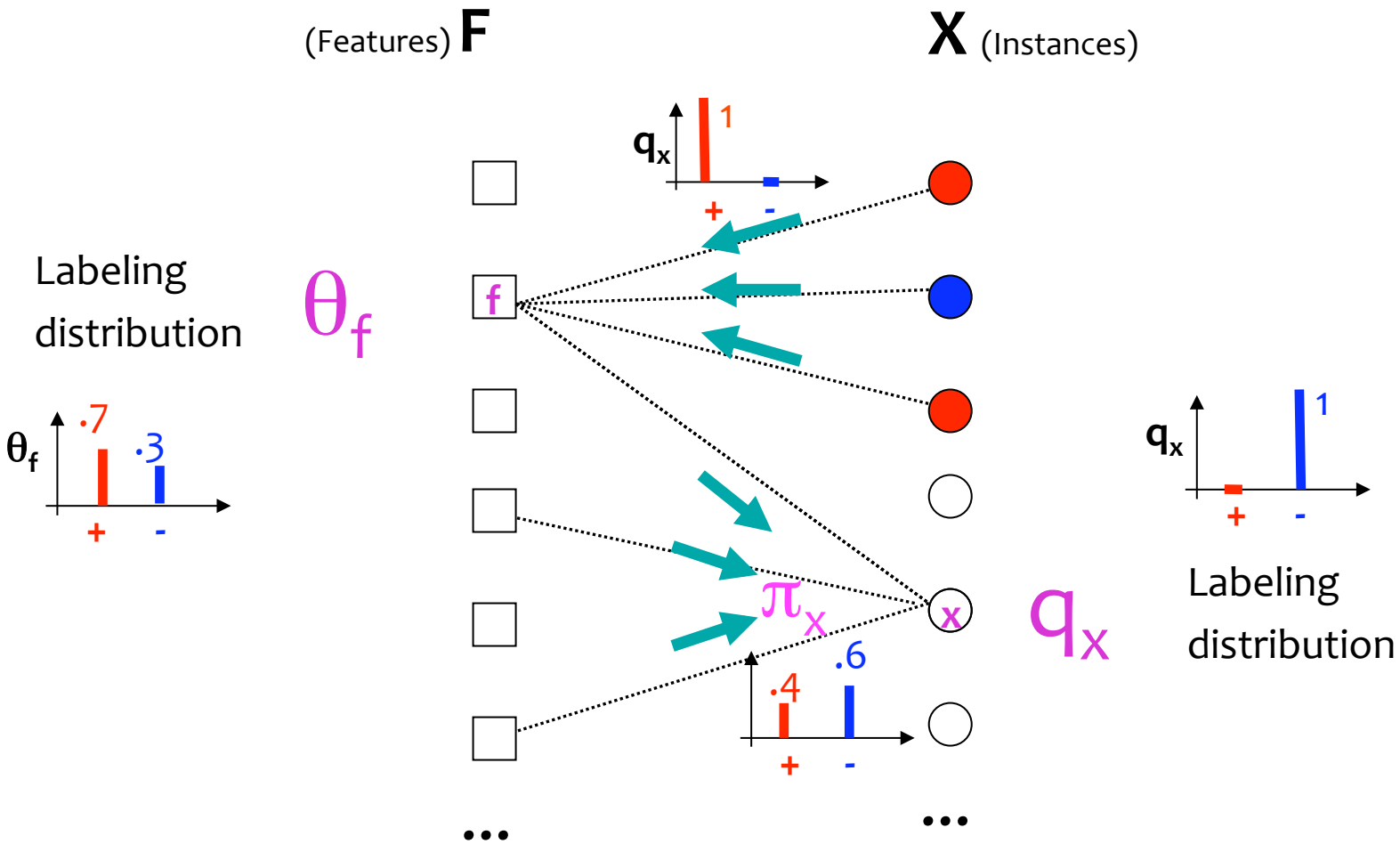
(Cordunneanu 2006, Haffari & Sarkar 2007)



We propose to view bootstrapping as propagating the labels of initially labeled nodes to the rest of the graph nodes.

Self-Training on the Graph

(Haffari & Sarkar 2007)



Goals of the Analysis

- To find *reasonable* objective functions for the self-training algorithms on the bipartite graph.

- The objective functions may shed light to the empirical success of different DL-based self-training algorithms.
  - It can tell us the kind of properties in the data which are well exploited and captured by the algorithms.
  - It is also useful in proving the convergence of the algorithms.

Objective Function

- KL-divergence is a measure of distance between two probability distributions:

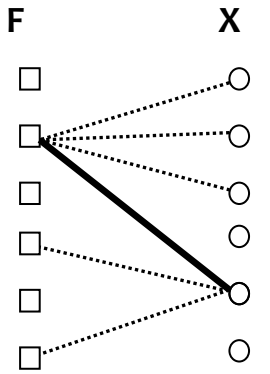
$$KL(\alpha \parallel \beta) := \sum_i \alpha_i \log \frac{\alpha_i}{\beta_i}$$

- Entropy H is a measure of randomness in a distribution:

$$H(\alpha) := -\sum_i \alpha_i \log \alpha_i$$

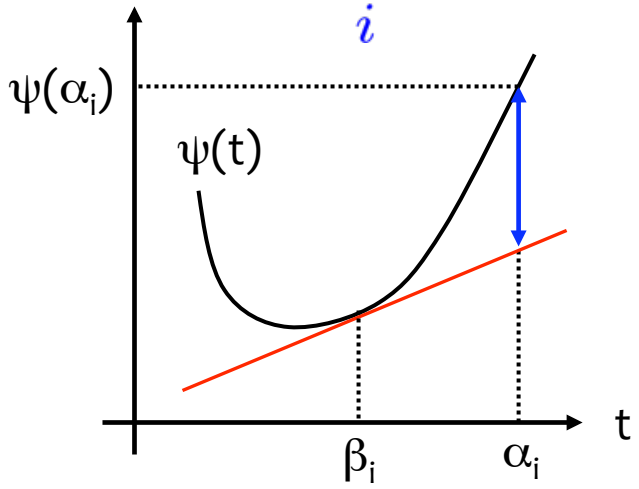
- The objective function:

$$\mathcal{K}(q, \theta) := \sum_{(f,x) \in \text{Edges}} \left[KL(q_x \parallel \theta_f) \right] + H(q_x) + H(\theta_f)$$



The Bregman Distance

$$B_\psi(\alpha,\beta) := \sum_i \psi(\alpha_i) - \psi(\beta_i) - \psi'(\beta_i)(\alpha_i - \beta_i)$$



- Examples:
  - If $\psi(t) = t \log t$ Then $B_\psi(\alpha,\beta) = KL(\alpha,\beta)$
  - If $\psi(t) = t^2$ Then $B_\psi(\alpha,\beta) = \Sigma_i (\alpha_i - \beta_i)^2$

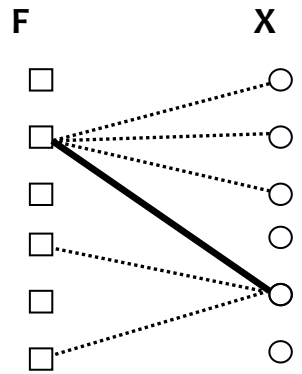
Generalizing the Objective Function

- Given a strictly convex function ψ, the Bregman distance B_ψ between two probability distributions is defined as:

$$B_\psi(\alpha \parallel \beta) := \sum_i \psi(\alpha_i) - \psi(\beta_i) - \psi'(\beta_i)(\alpha_i - \beta_i)$$

- The ψ-entropy H_ψ is defined as:

$$H_\psi(\alpha) := -\sum_i \psi(\alpha_i)$$



- The generalized objective function:

$$\mathcal{K}_\psi(q,\theta) := \sum_{(f,x)\in \text{Edges}} \left[B_\psi(q_x \parallel \theta_f)\right] + H_\psi(q_x) + H_\psi(\theta_f)$$

Optimizing the Objective Functions

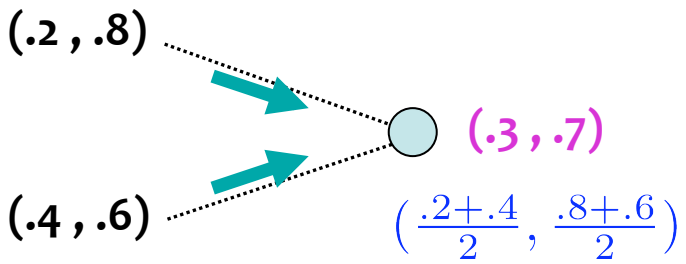
- In what follows, we mention some specific objective functions together with their optimization algorithms.

- These optimization algorithms correspond to some variants of the modified Yarowsky algorithm.

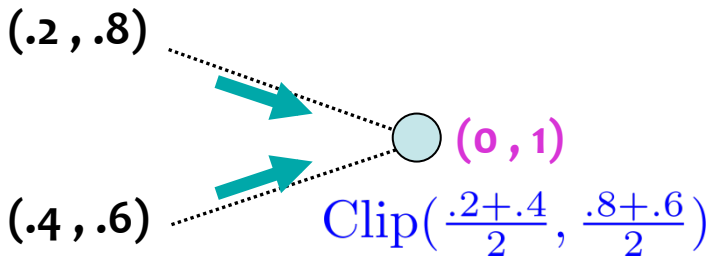
- It is not easy to come up with algorithms for directly optimizing the generalized objective functions.

Useful Operations

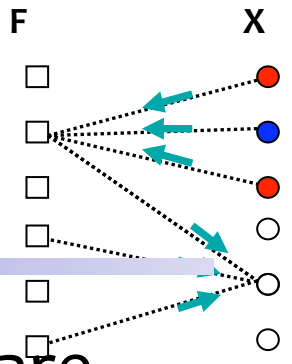
- Average: takes the average distribution of the neighbors



- Majority: takes the majority label of the neighbors



Analyzing Self-Training



- Theorem. The following objective functions are optimized by the corresponding label propagation algorithms on the bipartite graph:

Related to graph-based SS learning (Zhu et al 2003)

Algorithm	Features	Instances	Objective Function
Avg-Avg	soft	soft	$\mathcal{K}^1_{t^2} := \sum_{(f,x)\in \text{Edges}} B_{t^2}(\theta_f \parallel q_x)$
Avg-Maj	soft	hard	$\mathcal{K}^2_{t^2} := \sum_{(f,x)\in \text{Edges}} B_{t^2}(\theta_f \parallel q_x) + H_{t^2}(q_x)$
Maj-Avg	hard	soft	$\mathcal{K}^3_{t^2} := \sum_{(f,x)\in \text{Edges}} B_{t^2}(\theta_f \parallel q_x) + H_{t^2}(\theta_f)$
Maj-Maj	hard	hard	$\mathcal{K}^4_{t^2} := \sum_{(f,x)\in \text{Edges}} B_{t^2}(\theta_f \parallel q_x) + H_{t^2}(q_x) + H_{t^2}(\theta_f)$

Converges in Poly time $O(|F|^2\,|X^{|2|}|)$

Abney's variant of Yarowsky algorithm

where:

$$B_{t^2}(\alpha \parallel \beta) = \sum_i (\alpha_i - \beta_i)^2 \qquad H_{t^2}(\alpha) = -\sum_i \alpha_i^2$$

What about Log-Likelihood ?

- Initially, the labeling distribution is uniform for unlabeled vertices and a δ-like distribution for labeled vertices.

- By learning the parameters, we would like to reduce the uncertainty in the labeling distribution while respecting the labeled data:

$$\mathcal{L}_\psi := \sum_{x \in X} B_\psi(q_x, \pi_x) + H_\psi(q_x)$$

$$= \sum_{x \in L} B_\psi(q_x, \pi_x) + \sum_{x \in U} B_\psi(q_x, \pi_x) + H_\psi(q_x)$$

Negative log-Likelihood of the old and newly labeled data

Connection between the two Analyses

- Lemma. If m is the number of features connected to an instance, then:

$$\mathcal{L}_{t^2} \leq \frac{1}{m}\mathcal{K}^2_{t^2}$$

$$\sum_{x \in X} B_{t^2}(q_x, \pi_x) + H_{t^2}(q_x) \qquad \sum_{(f,x) \in \text{Edges}} \left[B_{t^2}(q_x, \theta_f)\right] + H_{t^2}(q_x)$$

Compare with Conditional Entropy Regularization (Grandvalet and Bengio 2005)

$$\sum_{x \in L} KL(q_x || \pi_x) + \gamma \sum_{x \in U} H(\pi_x)$$

Experiments

Named Entity Classification

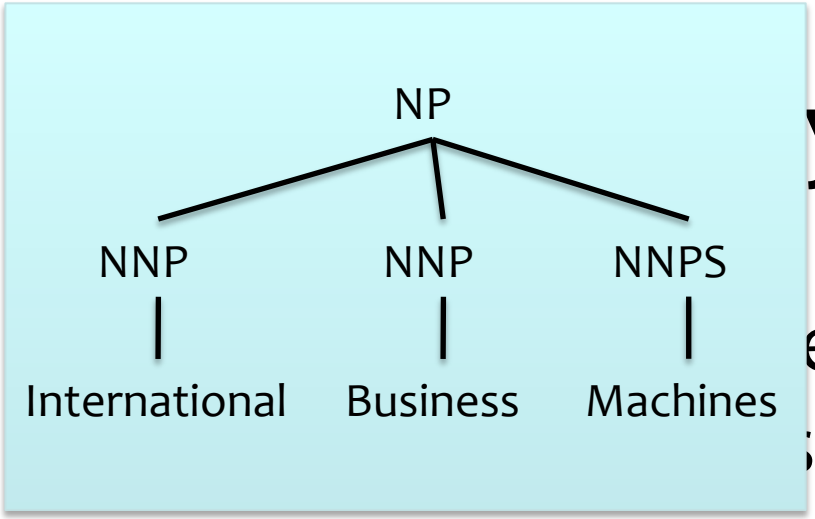
(Collins and Singer, 1999)

- 971,476 sentences from the NYT were parsed with the Collins parser
- The task is to identify three types of named entities:
  1. Location (LOC)
  2. Person (PER)
  3. Organization (ORG)
  -1. not a NE or "don't know"

Named Entity Classification

(Collins and Singer, 1999)

- Noun phrases were extracted that met the following conditions
  1. The NP contained only words tagged as proper nouns
  2. The NP appeared in the following two syntactic contexts:
     - Modified by an appositive whose head is a singular noun
     - In a prepositional phrase modifying an NP whose head is a singular noun



y Classification

(Collins and Singer, 1999)

extracted that met the

1. The NP contained only words tagged as proper nouns
2. The NP appeared in the following two

…, fraud related to work on a federally funded sewage [$_{CONTEXT}$ plant in] [$_{NE}$ Georgia]

…, says [$_{NE}$ Maury Cooper] , a vice [$_{CONTEXT}$ president] at S. & P.

Named Entity Classification

(Collins and Singer, 1999)

- The task: classify NPs into LOC, PER, ORG
- 89,305 training examples with 68,475 distinct feature types
  - 88,962 was used in CS99 experiments
- 1000 test data examples (includes NPs that are not LOC, PER or ORG)
  - Month names are easily identifiable as not named entities: leaves 962 examples
  - Still 85 NPs that are not LOC, PER, ORG.
  - Clean accuracy over 877; Noisy over 962

Yarowsky Variants

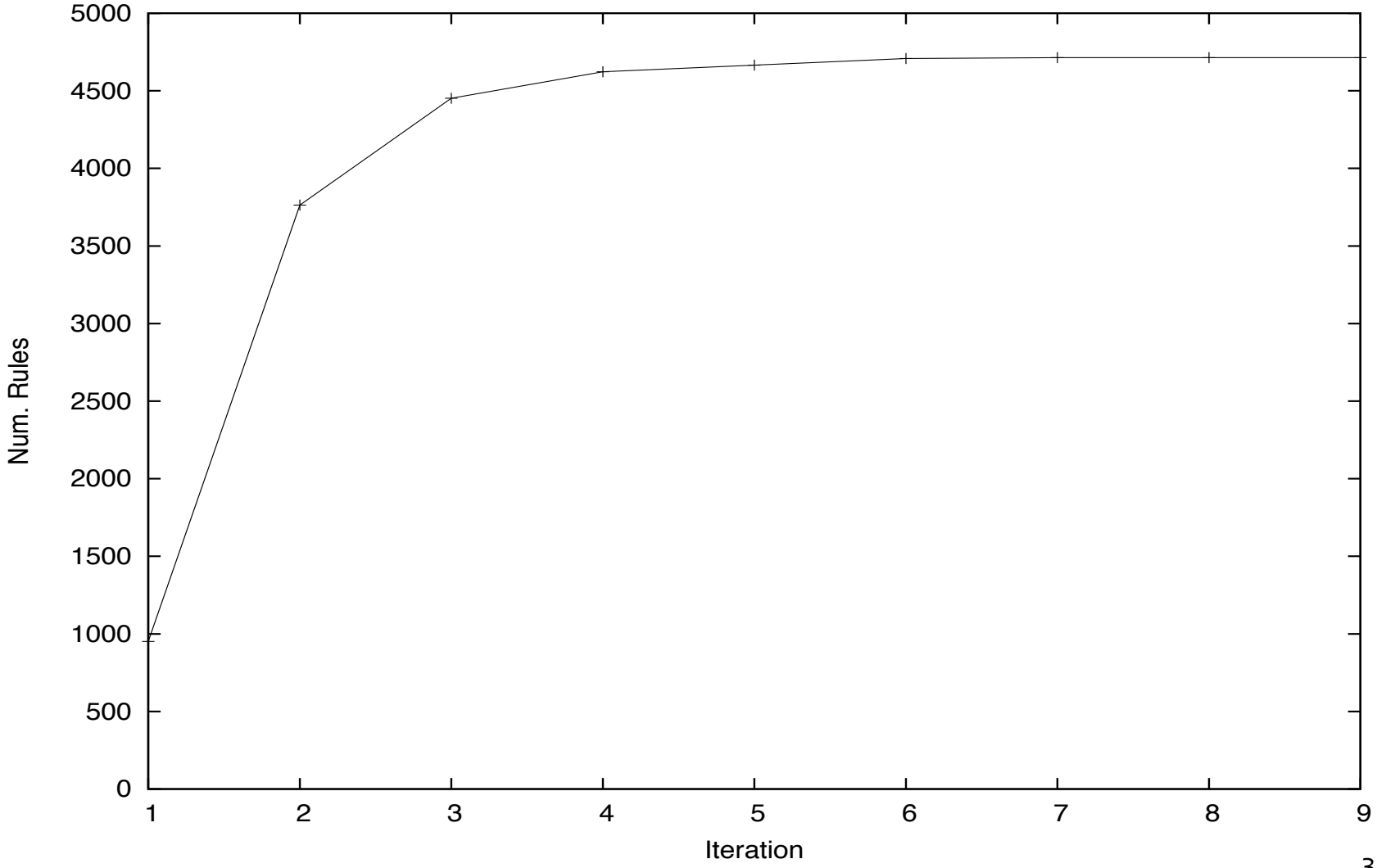
(Abney 2004, Collins and Singer, 1999)

- A trick from the Co-training paper (Blum and Mitchell 1998) is to be cautious. Don't add all rules above the 0.95 threshold
- Add only n rules *per label* (say 5) and increase this amount by n in each iteration
- Changes the dynamics of learning in the algorithm but not the objective fn
- Two variants: Yarowsky (basic), Yarowsky (cautious)
- Without a threshold: Yarowsky (no threshold)

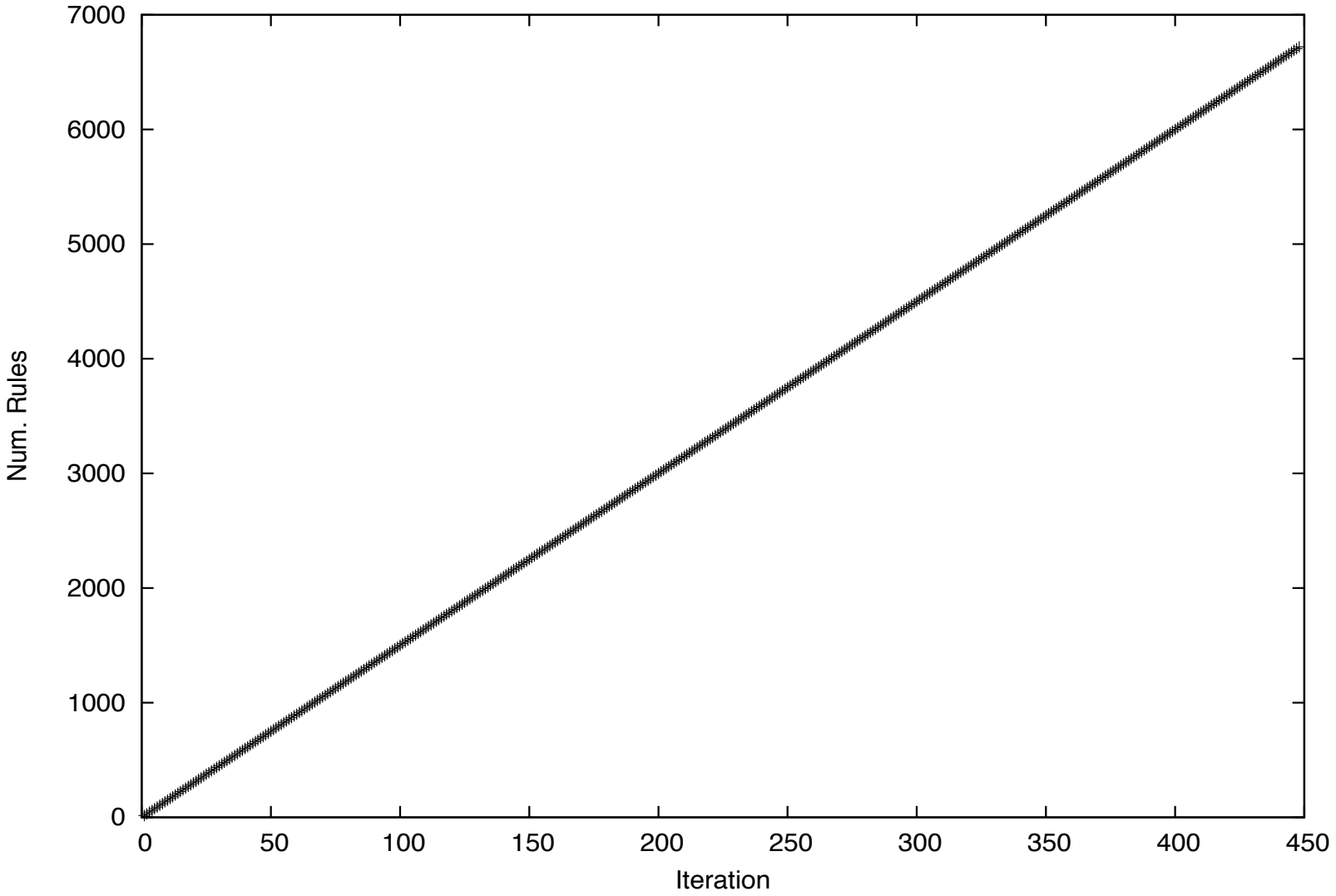
Results

Learning Algorithm	Accuracy (Clean)	Accuracy (Noisy)
Baseline (all organization)	45.8	41.8
EM	83.1	75.8
Yarowsky (basic)	80.7	73.5
Yarowsky (no threshold)	80.3	73.2
Yarowsky (cautious)	**91**	**83**
DL-CoTrain	**91**	**83**

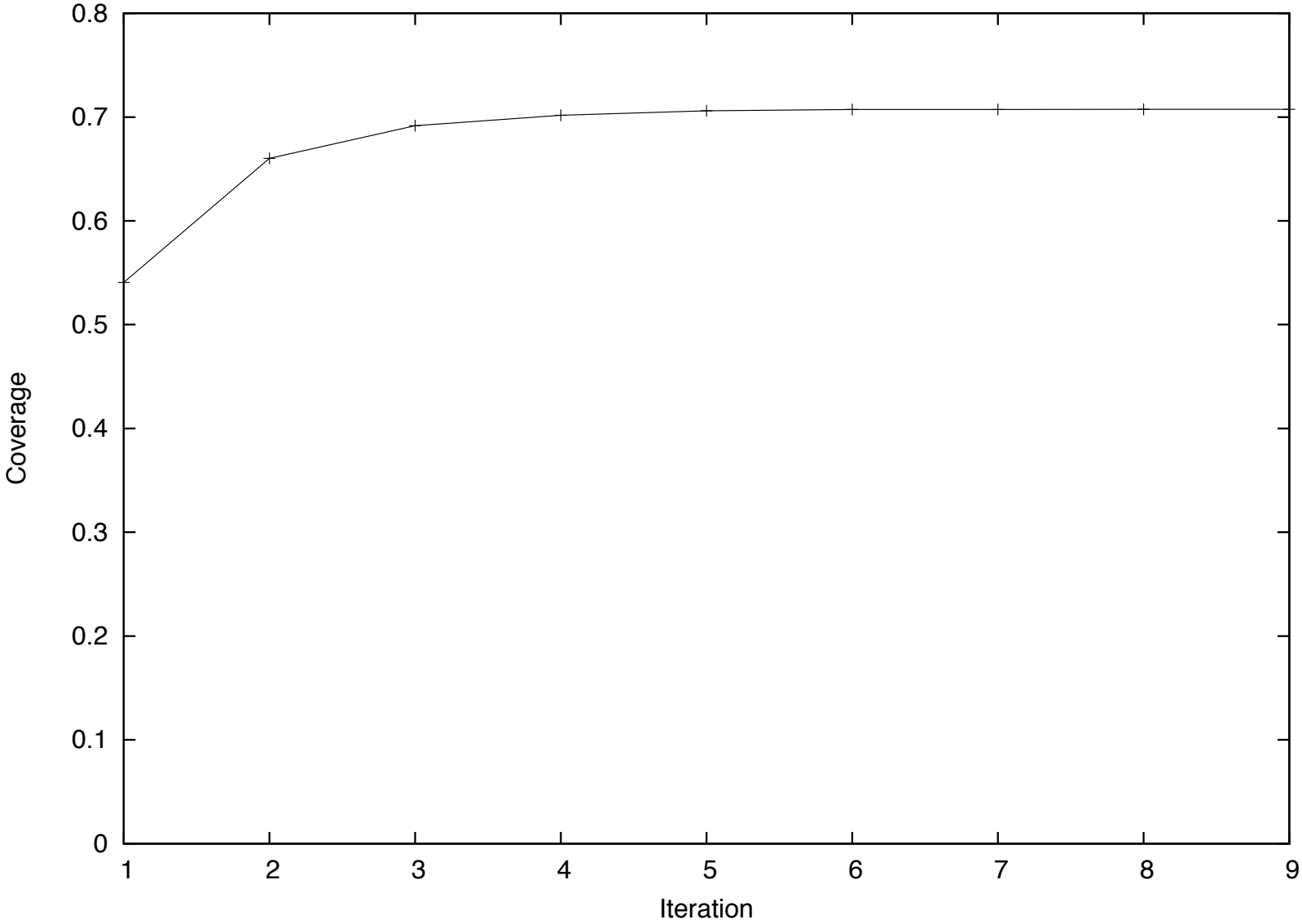
Number of Rules (basic)



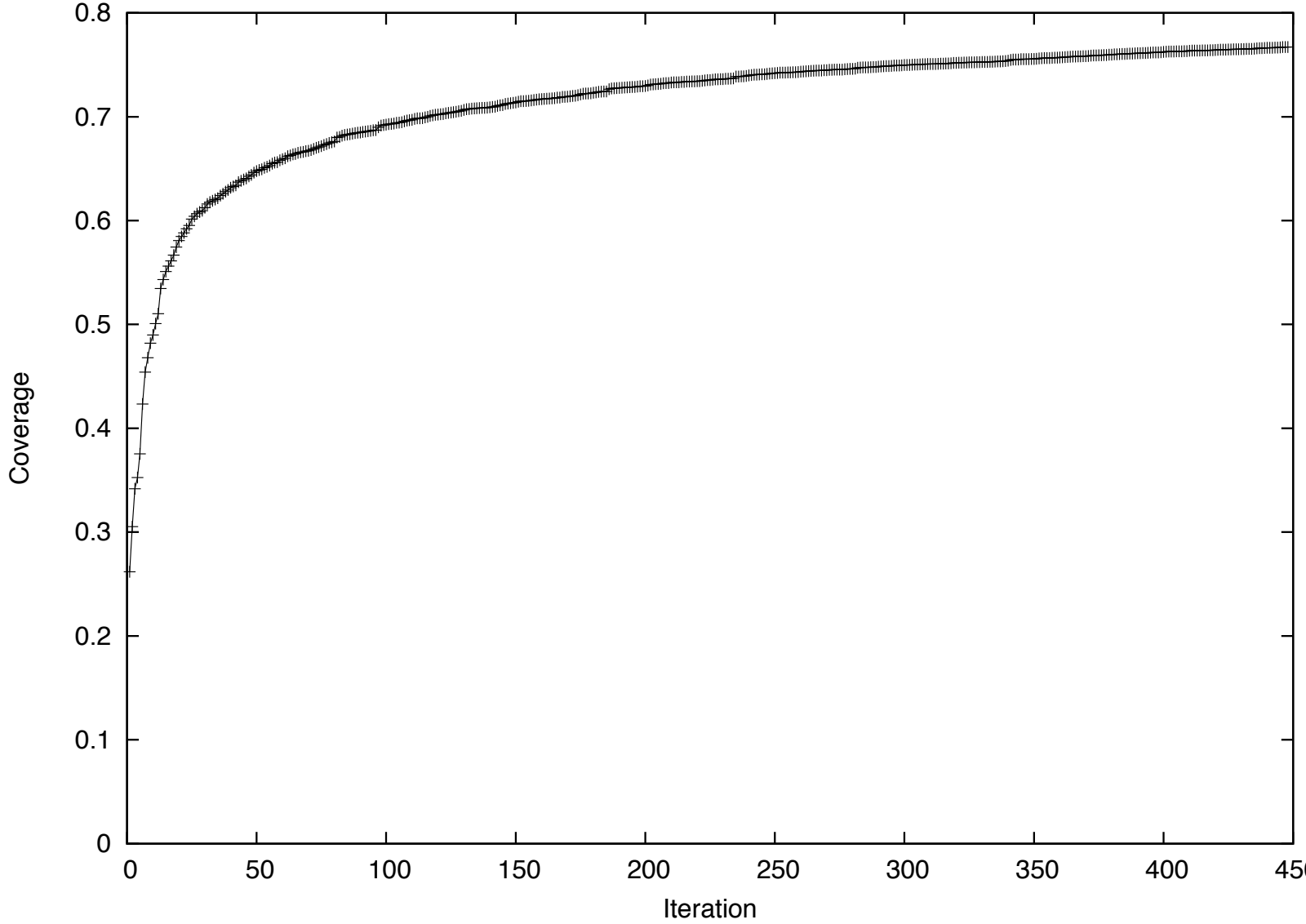
Number of Rules (cautious)



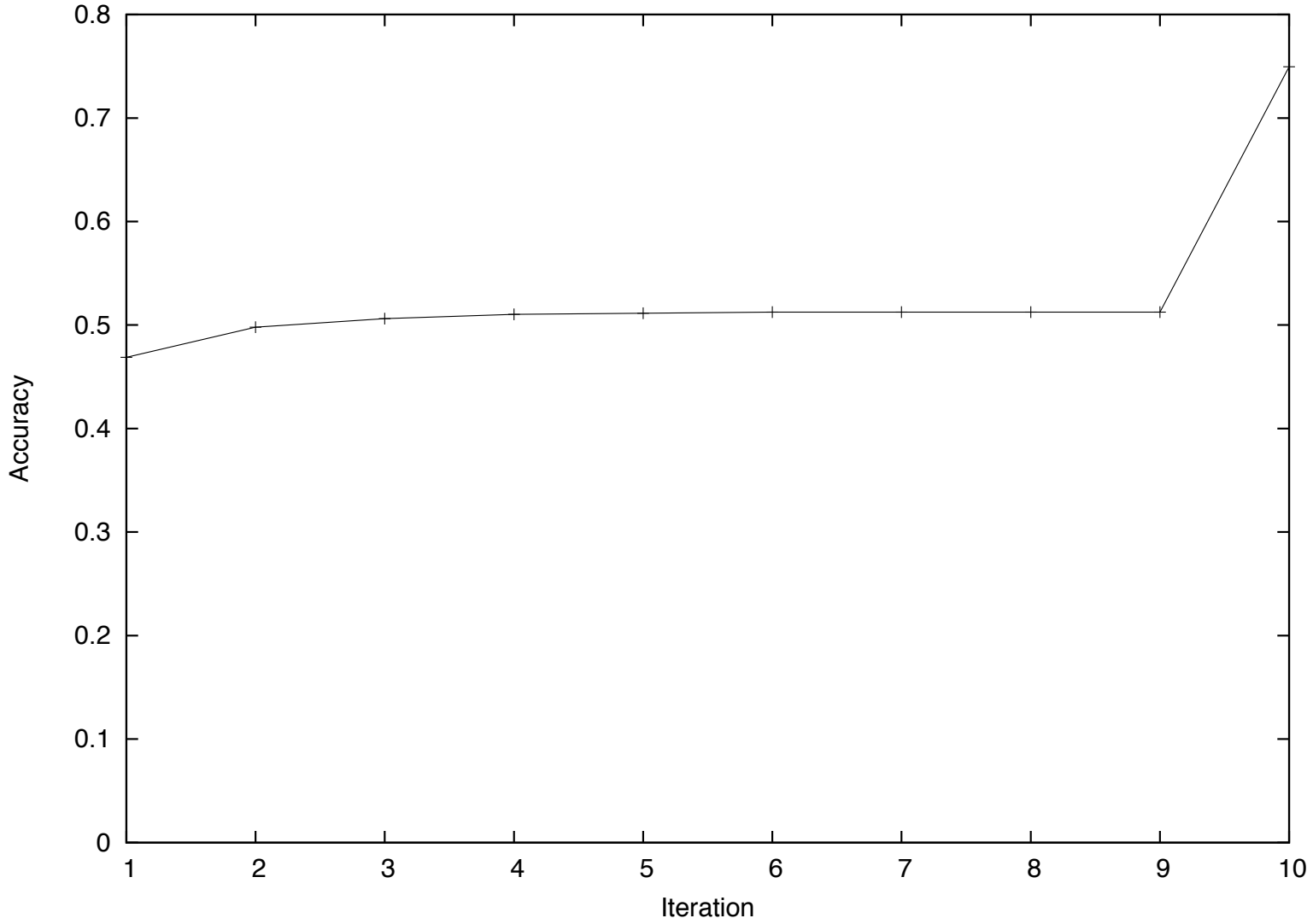
Coverage (basic)



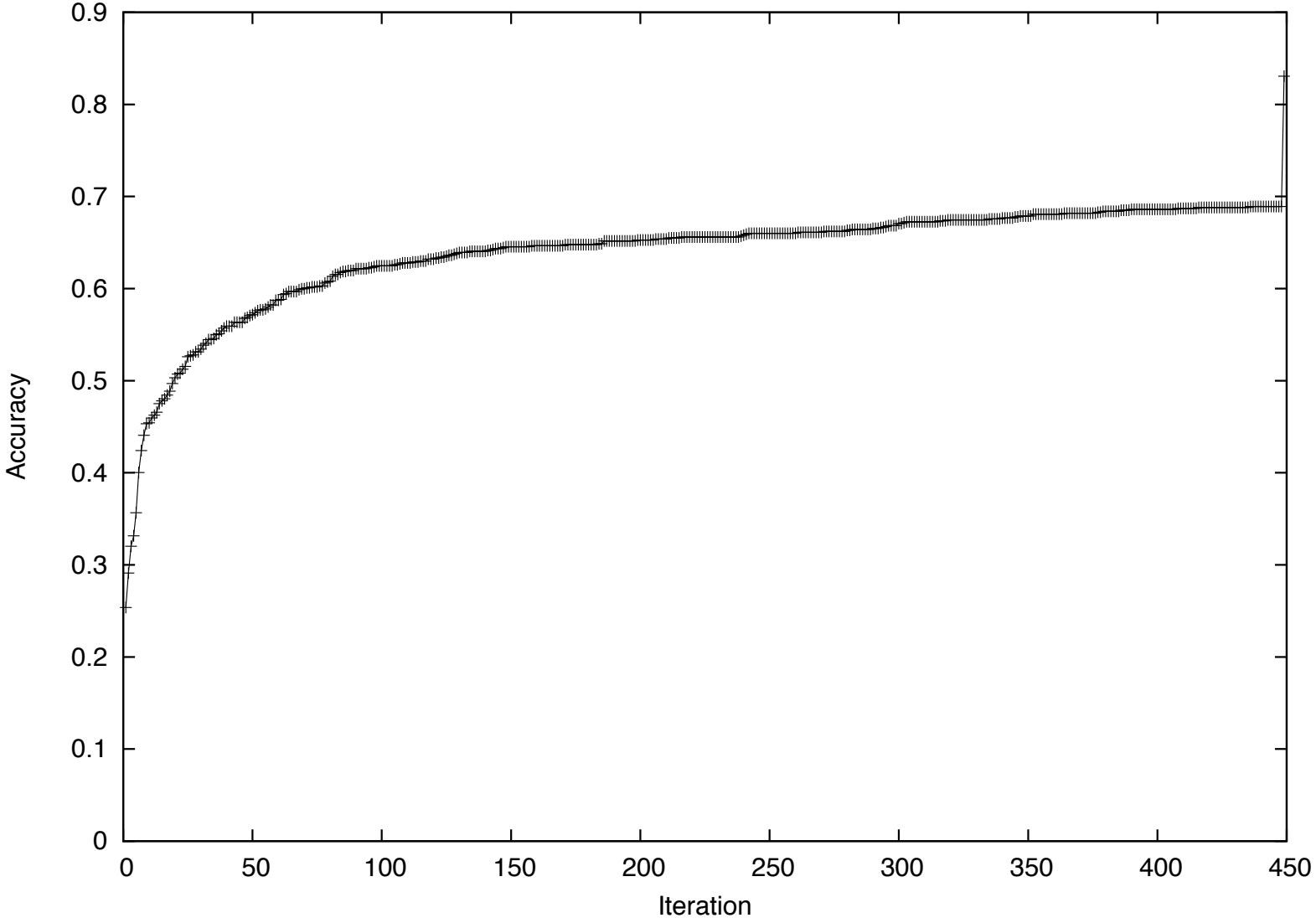
Coverage (cautious)



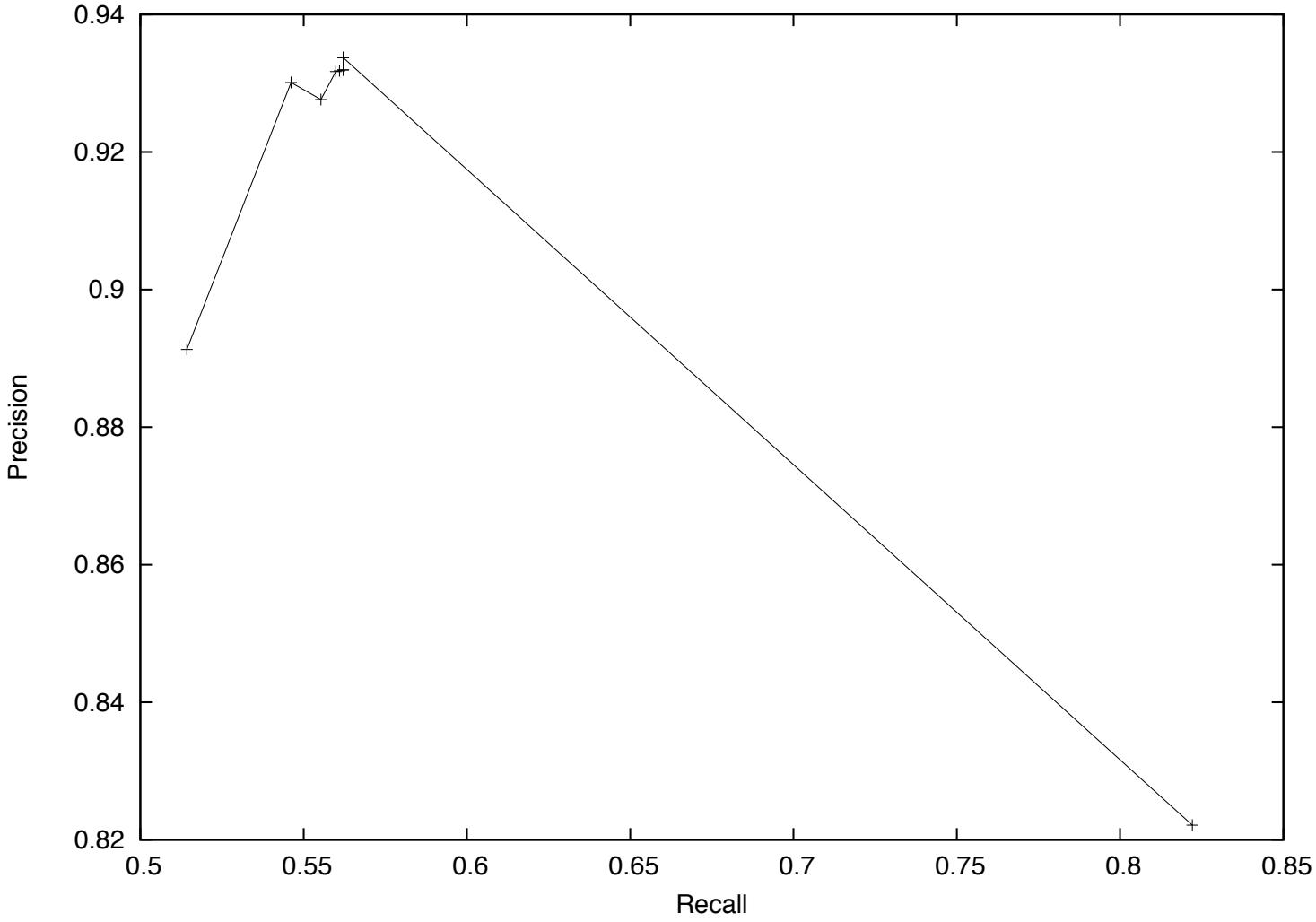
Accuracy (basic)



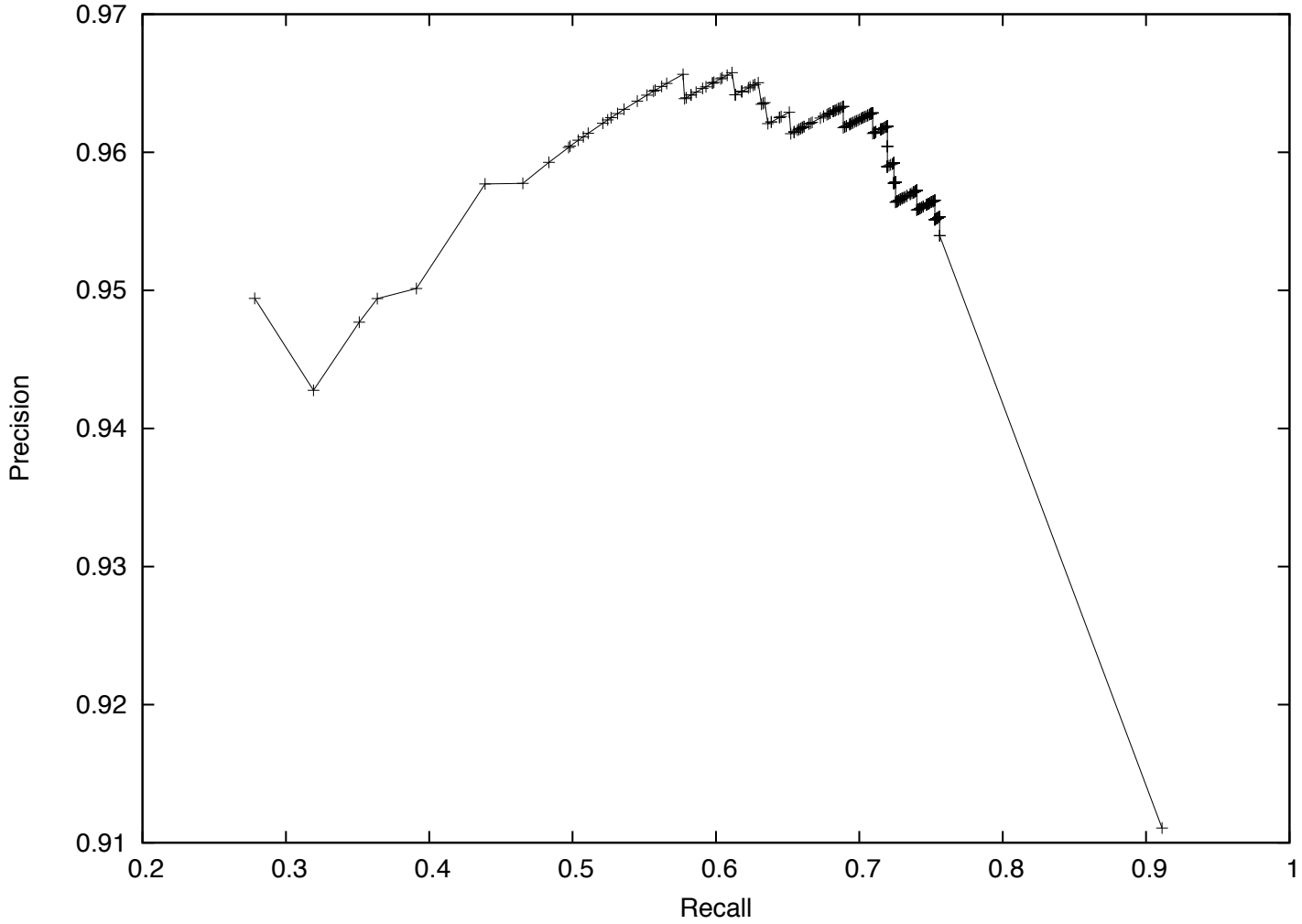
Accuracy (cautious)



Precision-Recall (basic)



Precision-Recall (cautious)



Seeds

(Eisner and Karakos 2005, Zagibalov and Carroll 2008)

- Selecting seed rules: what is a good strategy?
  - Frequency: sort by frequency of feature occurrence
  - Contexts: sort by number of other features a feature was observed with
  - Weighted: sort by a weighted count of other features observed with feature. Weight(f) = count (f) / Σ_f, count(f')

Seeds

- In each case the frequencies were taken from the unlabeled training data
- Seeds were extracted from the sorted list of features by manual inspection and assigned a label (the entire example was used)
- Location (LOC) features appear infrequently in all three orderings
- It is possible that some good LOC seeds were missed

Seeds

Number of Rules	**Frequency**		**Contexts**		**Weighted**	
(n/3) rules/label	Clean	Noisy	Clean	Noisy	Clean	Noisy
3	84	77	84	77	88	80
9	**91**	**83**	90	82	82	74
15	**91**	**83**	**91**	**83**	85	77
7 (CS99)	**91**		**83**			

Word Sense Disambiguation

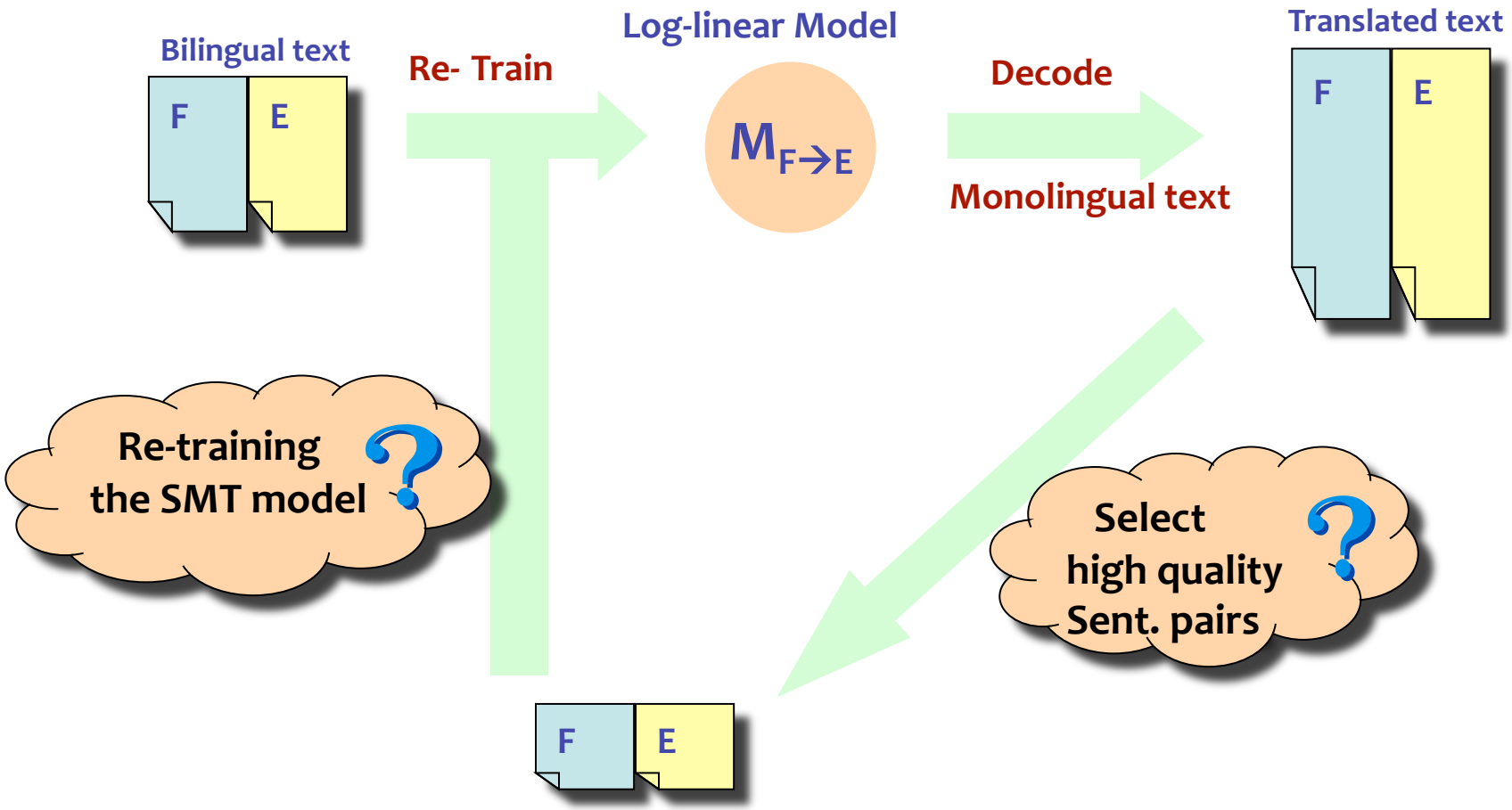
- Data from (Eisner and Karakos 2005)
- Disambiguate two senses each for drug, duty, land, language, position, sentence (Gale et. al. 1992)
- Source of unlabeled data: 14M word Canadian Hansards (English only)
- Two seed rules for each disambiguation task from (Eisner and Karakos 2005)

Results

Learning Algorithm	drug		land		sentence	
Seeds ➔	*alcohol*	*medical*	*acres*	*courts*	*served*	*reads*
Train / Test size ➔	134 / 386		1604 / 1488		303 / 515	
Yarowsky (basic)	53.3		79.3		67.7	
Yarowsky (no threshold)	52		**79**		64.8	
Yarowsky (cautious)	**55.9**		**79**		**76.1**	
DL-CoTrain (2 views = long distance v.s. immediate context)	53.1		77.7		75.9	

Self-training for Machine Translation

Self-Training for SMT



Selecting Sentence Pairs

- First give scores:
  - Use normalized decoder's score
  - Confidence estimation method (Ueffing & Ney 2007)

- Then select based on the scores:
  - Importance sampling: $P_{\text{select}}(s,t) \propto \exp\left(\text{score}(t|s)\right)$
  - Those whose score is above a threshold
  - Keep all sentence pairs

Re-training the SMT Model

- Use new sentence pairs to train an additional phrase table and use it as a new feature function in the SMT log-linear model
  - One phrase table trained on sentences for which we have the true translations
  - One phrase table trained on sentences with their generated translations

Phrase Table 1

Phrase Table 2

Chinese to English (Transductive)

NIST Eval-2004: train = 8.2M, test = 1788 (4 refs)

Selection	Scoring	BLEU%
Baseline		31.8 ± .7
Keep all		*33.1*
Importance Sampling	Norm. score	***33.5***
	Confidence	33.2
Threshold	Norm. score	***33.5***
	confidence	***33.5***

Train: news, magazines, laws + UN

Test: newswire, editorials, political speeches

Bold: best result, *italic*: significantly better

We use Portage from NRC as the underlying SMT system (Ueffing et al, 2007)

Chinese to English (Inductive)

Using importance sampling

system		BLEU%
Baseline		31.8 ± .7
Add Chinese data	Iter 1	**32.8**
	Iter 4	32.6
	Iter 10	32.5

	Before	After
editorials	30.7	31.3
newswire	30.0	*31.1*
speeches	36.1	*37.3*

Bold: best result, *italic*: significantly better

Why does it work?

- Reinforces parts of the phrase translation model which are relevant for test corpus
- Glue phrases from test data used to compose new phrases (most phrases still from original data)

eval-04	editorials	newswire	speeches
sentences	449	901	438
selected translations	101	187	113
size of adapted phrase table	1,981	3,591	2,321
adapted phrases used	707	1,314	815
new phrases	679	1,359	657
new phrases used	23	47	25

Why does it work?

Table X. Translation examples[a] from the 2006 GALE corpus.

baseline	[the report said] [that the] [united states] [is] [a potential] [problem] [, the] [practice of] [china 's] [foreign policy] [is] [likely to] [*weaken us*] [*influence*] [.]
adapted	[the report] [said that] [this is] [a potential] [problem] [in] [the united states] [,] [china] [is] [likely to] [**weaken**] [**the impact of**] [**american foreign policy**] [.]
reference	the report said that this is a potential problem for america . china 's course of action could possibly weaken the influence of american foreign policy .
baseline	[*what we advocate*] [*his*] [*name*]
adapted	[**we**] [**advocate**] [**him**] [**.**]
reference	we advocate him .

Summary

- Should we ever use Co-training for Bootstrapping?
- Per-label cautiousness leads to effective bootstrapping.
  - Exploited in Yarowsky algo., DL-CoTrain, Co-Boosting
- These dynamics can/should be examined more closely.
  - Perhaps using tools from the analysis of feature induction.
- Bootstrapping and self-training may be more effective than you may have thought.