CSEP 517
Natural Language Processing
Autumn 2013

Unsupervised and Semi-supervised Learning

Luke Zettlemoyer - University of Washington

[Many slides from Dan Klein and Michael Collins]

Overview

- Unsupervised and Semi-supervised Learning
- Discrete classification
  - k-Means
  - EM for Naïve Bayes
- EM in General (see notes for more math)
- Sequence Models
  - EM for HMMs
  - Semi-supervised learning

Clustering vs. Classification

- Classification: we specify which pattern we want, features uncorrelated with that pattern are idle

P(w|sports)	P(w|politics)	P(w|headline)	P(w|story)
the 0.1	the 0.1	the 0.05	the 0.1
game 0.02	game 0.005	game 0.01	game 0.01
win 0.02	win 0.01	win 0.01	win 0.01

- Clustering: the clustering procedure locks on to whichever pattern is most salient, statistically
  - P(content words | class) will learn topics
  - P(length, function words | class) will learn style
  - P(characters | class) will learn “language”

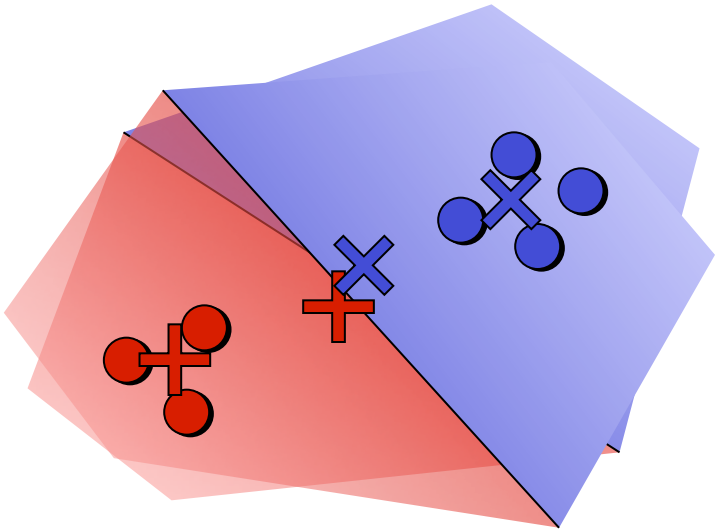
Learning Models with EM

- Hard EM: alternate between

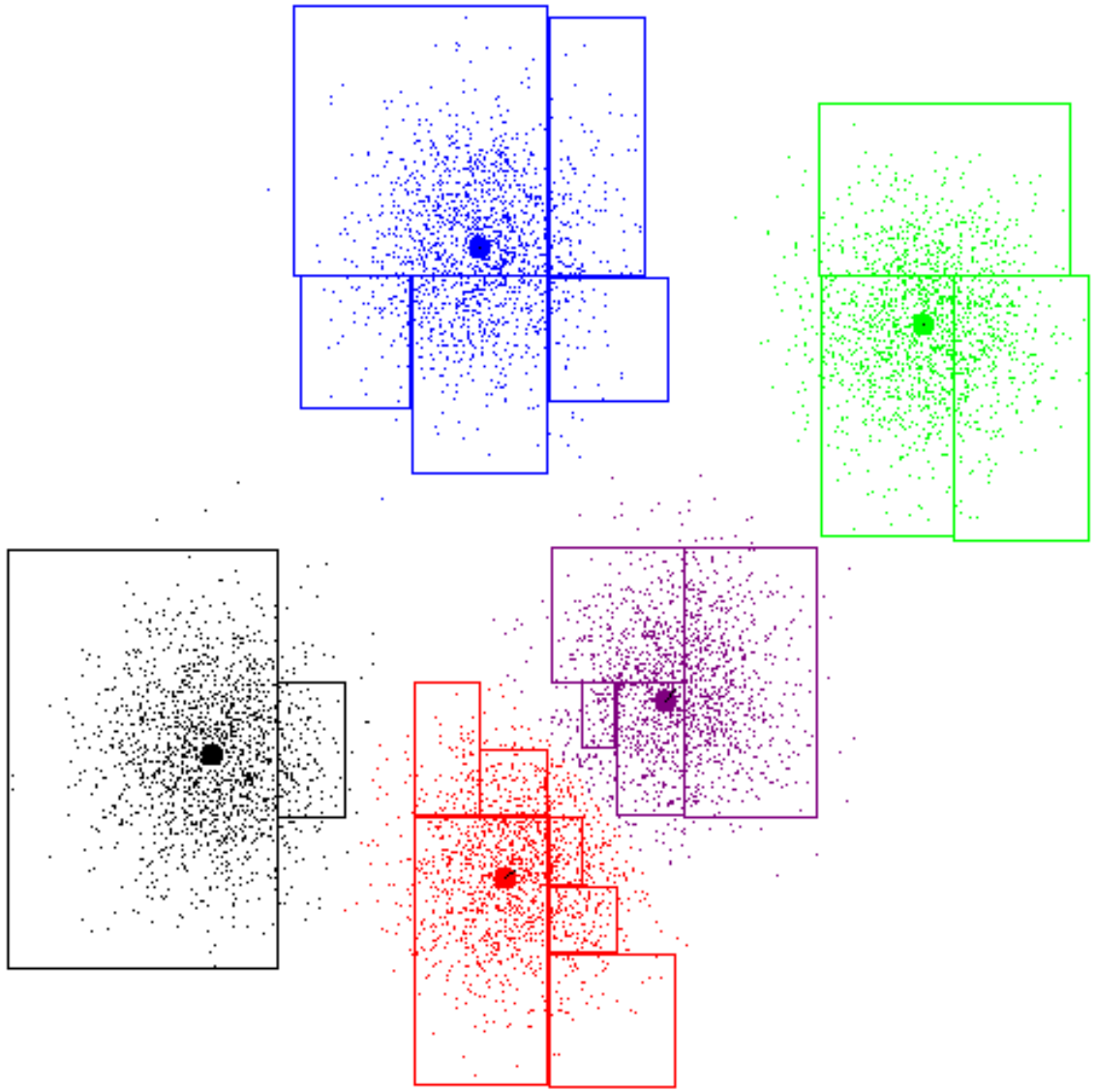
  E-step: Find best "completions" Y for fixed θ

  M-step: Find best parameters θ for fixed Y

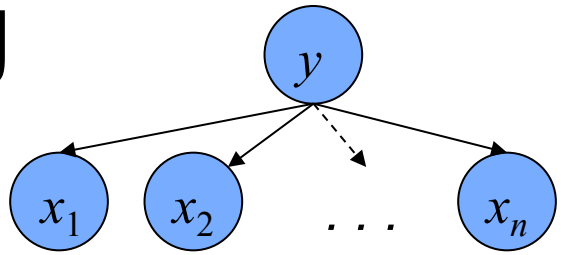
- Example: K-Means



K-Means Example



Model-Based Clustering



- Clustering with probabilistic models:

Unobserved (Y)	Observed (X)
??	LONDON -- Soccer team wins match
??	NEW YORK – Stocks close up 3%
??	Investing in the stock market has …
??	The first game of the world series …

Build a model of the domain: $P(x, y, \theta)$

Often: find θ to maximize: $P(x|\theta) = \sum_y P(x, y|\theta)$

- Problem 2: The relationship between the structure of your model and the kinds of patterns it will detect is complex.

EM in General

- We'll use EM over and over again to fill in missing data, e.g. we want P(x,y) but our training data has only xs (no y labels)
  - Convenience Scenario: we want P(x), including y just makes the model simpler (e.g. mixing weights for language models)
  - Induction Scenario: we actually want to know y (e.g. clustering)
  - NLP differs from much of statistics / machine learning in that we often want to interpret or use the induced variables (which is tricky at best)

- General approach: alternately update y and θ
  - E-step: compute posteriors $P(y|x,\theta)$
    - This means scoring all completions with the current parameters
    - Usually, we do this implicitly with dynamic programming
  - M-step: fit θ to these completions
    - This is usually the easy part – treat the completions as (fractional) complete data
  - Initialization: start with some noisy labelings and the noise adjusts into patterns based on the data and the model
  - We'll see lots of examples in this course

- EM is only locally optimal (why?)

Hard EM for Naïve-Bayes

- For t = 1..T

  1) [E-step] calculate most probable class for each training example i:

$$\delta(y|i) = \begin{cases} 1 & \text{if } \; y = \arg\max_{y'} p(y', \underline{x}^{(i)}; \underline{\theta}^{t-1}) \\ 0 & \text{otherwise} \end{cases}$$

  2) [M-step] compute maximum likelihood estimates, given counts

$$q^t(y) = \frac{1}{n} \sum_{i=1}^{n} \delta(y|i) \qquad q_j^t(x|y) = \frac{\sum_{i:x_j^{(i)}=x} \delta(y|i)}{\sum_i \delta(y|i)}$$

  where $\underline{\theta}^t$ is a concatenation of the Naïve Bayes parameters $q^t(y)$ and $q_j^t(x|y)$ at iteration t.

(Soft) EM for Naïve-Bayes

- For t = 1..T
  1) [E-step] calculate posteriors (soft completions) for each training example i and class y:

$$\delta(y|i) = p(y|\underline{x}^{(i)}; \underline{\theta}^{t-1}) = \frac{q^{t-1}(y) \prod_{j=1}^{d} q_j^{t-1}(x_j^{(i)}|y)}{\sum_{y=1}^{k} q^{t-1}(y) \prod_{j=1}^{d} q_j^{t-1}(x_j^{(i)}|y)}$$

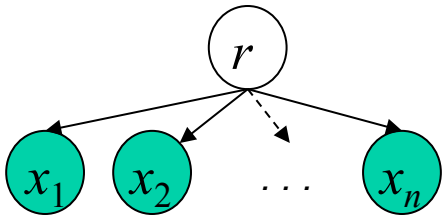
  2) [M-step] compute maximum likelihood estimates, given counts

$$q^t(y) = \frac{1}{n} \sum_{i=1}^{n} \delta(y|i) \qquad q_j^t(x|y) = \frac{\sum_{i:x_j^{(i)}=x} \delta(y|i)}{\sum_i \delta(y|i)}$$

  where $\underline{\theta}^t$ is a concatenation of the Naïve Bayes parameters $q^t(y)$ and $q_j^t(x|y)$ at iteration t.

- Can also do this when some docs are labeled

EM Example



Setup

- Cluster into 2 classes (r is binary)
- q(x|r) is a binary multinomial (bag of words)
- Only showing a subset of entries in q(x|r)

Example from: Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

(a)

docID	document text	docID	document text
1	hot chocolate cocoa beans	7	sweet sugar
2	cocoa ghana africa	8	sugar cane brazil
3	beans harvest ghana	9	sweet sugar beet
4	cocoa butter	10	sweet cake icing
5	butter truffles	11	cake black forest
6	sweet chocolate		

(b)

	Parameter	Iteration of clustering							
		0	1	2	3	4	5	15	25
E Step	α_1		0.50	0.45	0.53	0.57	0.58	0.54	0.45
	$r_{1,1}$		1.00	1.00	1.00	1.00	1.00	1.00	1.00
	$r_{2,1}$		0.50	0.79	0.99	1.00	1.00	1.00	1.00
	$r_{3,1}$		0.50	0.84	1.00	1.00	1.00	1.00	1.00
	$r_{4,1}$		0.50	0.75	0.94	1.00	1.00	1.00	1.00
	$r_{5,1}$		0.50	0.52	0.66	0.91	1.00	1.00	1.00
	$r_{6,1}$	1.00	1.00	1.00	1.00	1.00	1.00	0.83	0.00
	$r_{7,1}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$r_{8,1}$		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$r_{9,1}$		0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$r_{10,1}$		0.50	0.40	0.14	0.01	0.00	0.00	0.00
	$r_{11,1}$		0.50	0.57	0.58	0.41	0.07	0.00	0.00
M Step	$q_{\text{africa},1}$		0.000	0.100	0.134	0.158	0.158	0.169	0.200
	$q_{\text{africa},2}$		0.000	0.083	0.042	0.001	0.000	0.000	0.000
	$q_{\text{brazil},1}$		0.000	0.000	0.000	0.000	0.000	0.000	0.000
	$q_{\text{brazil},2}$		0.000	0.167	0.195	0.213	0.214	0.196	0.167
	$q_{\text{cocoa},1}$		0.000	0.400	0.432	0.465	0.474	0.508	0.600
	$q_{\text{cocoa},2}$		0.000	0.167	0.090	0.014	0.001	0.000	0.000
	$q_{\text{sugar},1}$		0.000	0.000	0.000	0.000	0.000	0.000	0.000
	$q_{\text{sugar},2}$		1.000	0.500	0.585	0.640	0.642	0.589	0.500
	$q_{\text{sweet},1}$		1.000	0.300	0.238	0.180	0.159	0.153	0.000
	$q_{\text{sweet},2}$		1.000	0.417	0.507	0.610	0.640	0.608	0.667

EM for Semi-supervised Learning

[Nigam, McCallum, Mitchel, 2006]

- Define data log likelihood to be L(y,x) + L(x), computed on labeled and unlabeled data. Find parameters that maximize the total likelihood.
- Paper also presents a number of other fancier models where the unlabeled data helps more.

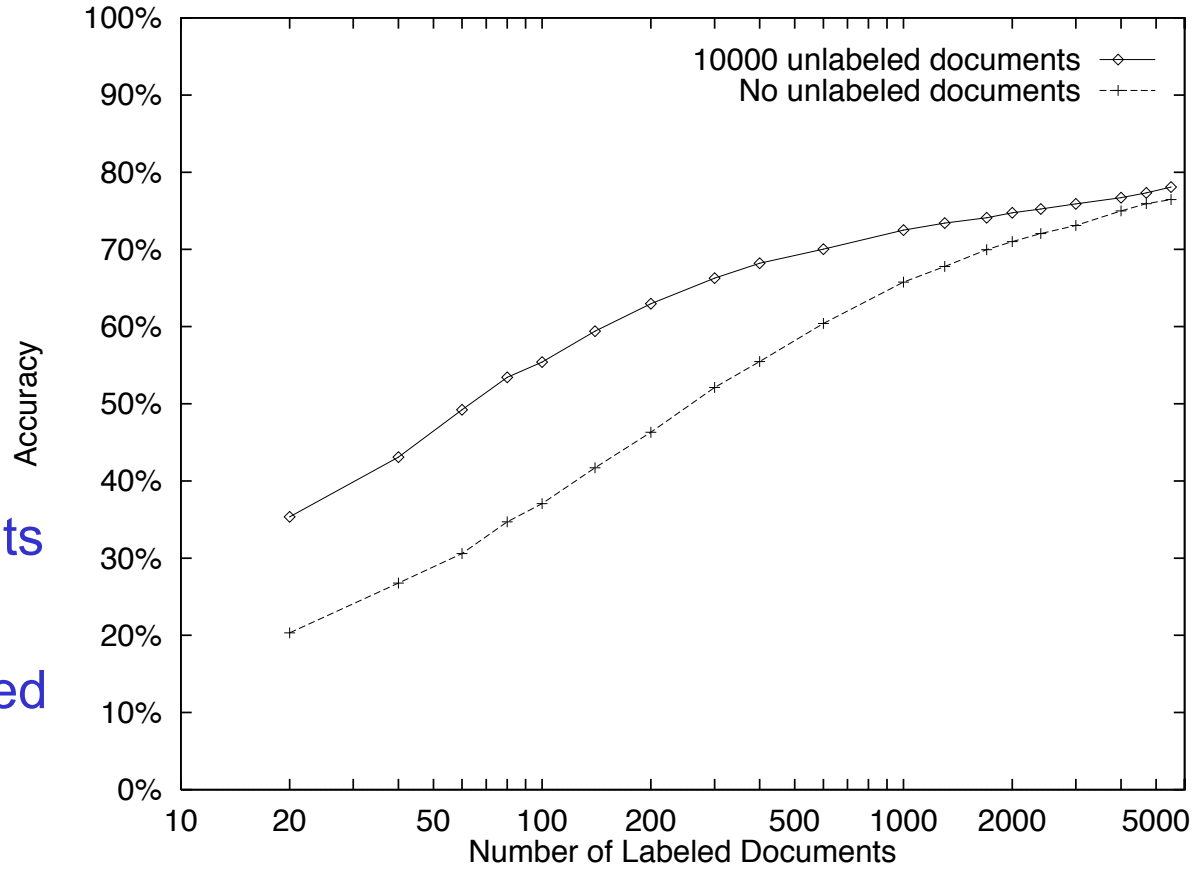


Figure 3.1 Classification accuracy on the 20 Newsgroups data set, both with and without 10,000 unlabeled documents. With small amounts of training data, using EM yields more accurate classifiers. With large amounts of labeled training data, accurate parameter estimates can be obtained without the use of unlabeled data, and classification accuracies of the two methods begin to converge.

Unsupervised Tagging?

- AKA part-of-speech induction
- Task:
  - Raw sentences in
  - Tagged sentences out
- Obvious thing to do:
  - Start with a (mostly) uniform HMM
  - Run EM
  - Inspect results

EM for HMMs: Process

- ML Estimate (only possible with full supervision):

$$q_{ML}(y_i|y_{i-1}) = \frac{c(y_{i-1}, y_i)}{c(y_{i-1})} \qquad e_{ML}(x|y) = \frac{c(y, x)}{c(y)}$$

- Instead, alternate between recomputing distributions over hidden variables (the tags) and reestimating parameters
- Crucial step: we want to tally up (fractional) counts

$$c^*(y) = \sum_{j:y_j=y} p(x_1 \dots x_m, y_j) \qquad c^*(y, x) = \sum_{j:x=x_j, y=y_j} p(y_j|x_1 \dots x_m)$$

$$c^*(y, y') = \sum_{j:y'=y_j, y=y_{j-1}} p(y_j, y_{j-1}|x_1 \dots x_m)$$

- We can do this with the forward backward algorithm!!!

Forward, Backward, Again…

$$p(x_1 \ldots x_n, y_i) = p(x_1 \ldots x_i, y_i) p(x_{i+1} \ldots x_n | y_i)$$

- Sum over all paths, on both sides of each y_i

$$\alpha(i, y_i) = p(x_1 \ldots x_i, y_i) = \sum_{y_1 \ldots y_{i-1}} p(x_1 \ldots x_i, y_1 \ldots y_i)$$

$$= \sum_{y_{i-1}} e(x_i | y_i) q(y_i | y_{i-1}) \alpha(i-1, y_{i-1})$$

$$\beta(i, y_i) = p(x_{i+1} \ldots x_n | y_i) = \sum_{y_{i+1} \ldots y_n} p(x_{i+1} \ldots x_n, y_{i+1} \ldots y_n)$$

$$= \sum_{y_{i+1}} e(x_{i+1} | y_{i+1}) q(y_{i+1} | y_i) \beta(i+1, y_{i+1})$$

EM for HMMs

- For t = 1..T

  1) [E-step] calculate posteriors (soft completions) for each training example i:

$$c^t(y) = \sum_{j:y_j=y} p^{t-1}(x_1 \ldots x_m, y_j) \qquad c^t(y,x) = \sum_{j:x=x_j,y=y_j} p^{t-1}(y_j|x_1 \ldots x_m)$$

$$c^t(y,y') = \sum_{j:y'=y_j,y=y_{j-1}} p^{t-1}(y_j, y_{j-1}|x_1 \ldots x_m)$$

  2) [M-step] compute maximum likelihood estimates, given counts

$$q^t(y_i|y_{i-1}) = \frac{c^t(y_{i-1},y_i)}{c^t(y_{i-1})} \qquad e^t(x|y) = \frac{c^t(y,x)}{c^t(y)}$$

  where there is a different HMM for each iteration t:

$$p^t(x_1 \ldots x_n, y_1 \ldots y_n) = q^t(STOP|y_n) \prod_{i=1}^{n} q^t(y_i|y_{i-1}) e^t(x_i|y_i)$$

Unsupervised Learning Results

- EM for HMM
  - POS Accuracy: 74.7%
- Bayesian HMM Learning [Goldwater, Griffiths 07]
  - Significant effort in specifying prior distriubtions
  - Integrate our parameters e(x|y) and t(y'|y)
  - POS Accuracy: 86.8%
- Unsupervised, feature rich models [Smith, Eisner 05]
  - Challenge: represent p(x,y) as a log-linear model, which requires normalizing over all possible sentences x
  - Smith presents a very clever approximation, based on local neighborhoods of x
  - POS Accuracy: 90.1%
- Newer, feature rich methods do better, not near supervised SOTA

Semi-supervised Learning

- AKA: boot strapping, self training, etc.
- Task: learn from two types of data
  - Tagged Sentences
  - Raw / unlabeled sentences
- Output: a complete POS tagger
- What should we do?
  - Use labeled data to initialize EM?
  - Sum the counts (real and expected) together?
  - Something fancier?

Merialdo: Setup

- Some initial results [Merialdo 94]
- Setup
  - You know the set of possible tags for each word
  - You have k fully labeled training examples
    - Estimate e(x|y) and t(y'|y) on this data
  - Use the supervised model to initialize the EM algorithms, and run it on all of the data

- Question: Will this work?

Merialdo: Results

Number of tagged sentences used for the initial model							
	0	100	2000	5000	10000	20000	all
Iter	Correct tags (% words) after ML on 1M words						
0	77.0	90.0	95.4	96.2	96.6	96.9	97.0
1	80.5	92.6	95.8	96.3	96.6	96.7	96.8
2	81.8	93.0	95.7	96.1	96.3	96.4	96.4
3	83.0	93.1	95.4	95.8	96.1	96.2	96.2
4	84.0	93.0	95.2	95.5	95.8	96.0	96.0
5	84.8	92.9	95.1	95.4	95.6	95.8	95.8
6	85.3	92.8	94.9	95.2	95.5	95.6	95.7
7	85.8	92.8	94.7	95.1	95.3	95.5	95.5
8	86.1	92.7	94.6	95.0	95.2	95.4	95.4
9	86.3	92.6	94.5	94.9	95.1	95.3	95.3
10	86.6	92.6	94.4	94.8	95.0	95.2	95.2

Co-Training / Self-Training

- Simple approach, often (but not always) works…
- Repeat
  - Learn N independent classifiers on supervised data
  - Use each classifier to tag new, unlabeled data
  - Select subset of unlabeled data (where models agree and are most confident) and add to labeled data (with automatically label tags)
- N=1: Self-training
- N>1: Co-Training [Blum and Mitchell, 1998]
  - assumed independent features sets, same learner
    - Proved bounds on when this will work well, see paper!
  - for POS, can do different models with the same features

English POS Self/Co-Training

- Two POS Taggers [Clark, Curran, Osbourne, 2003]

Self Training

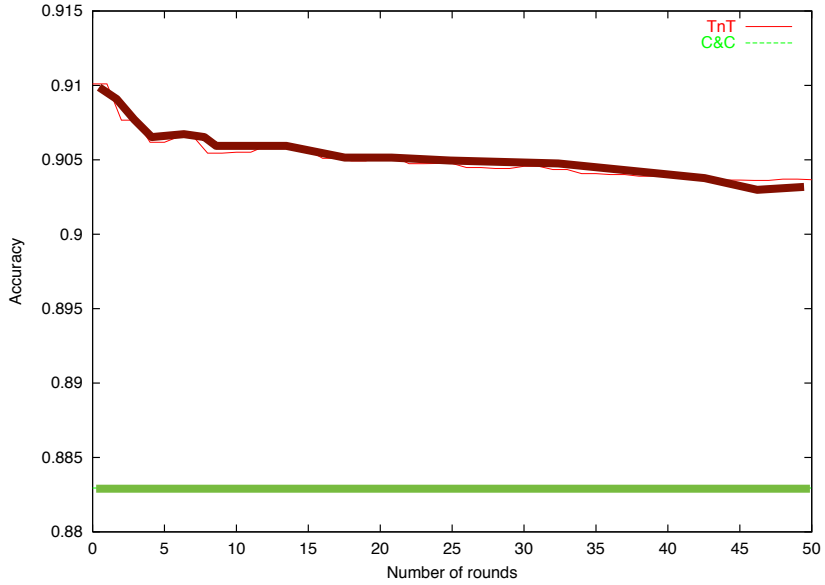


Figure 5: Self-training TNT and C&C (500 seed sentences). The upper curve is for TNT; the lower curve is for C&C.

500 seeds

Co-Training

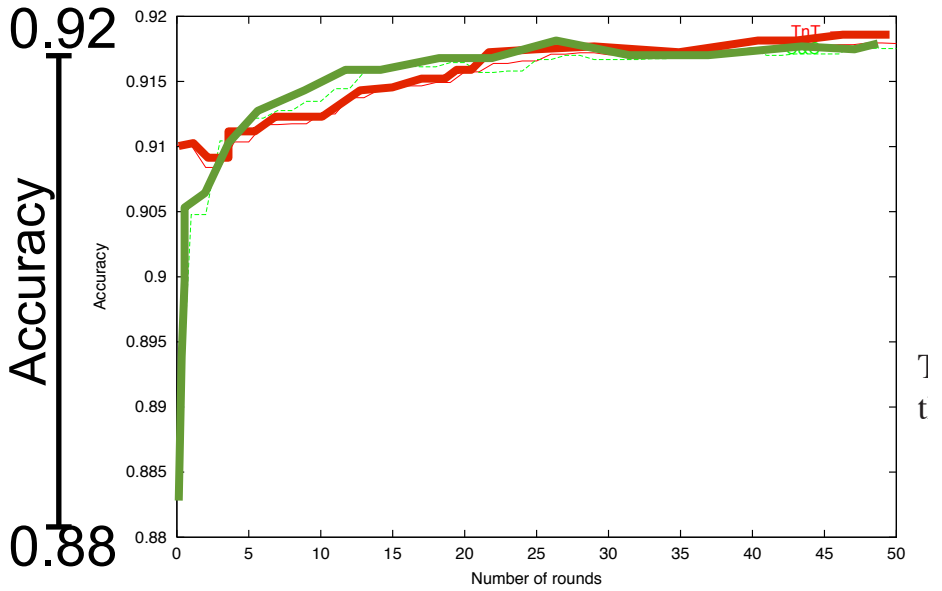


Figure 6: Agreement-based co-training between TNT and C&C (500 seed sentences). The curve that starts at a higher value is for TNT.

500 seeds

Mandarin POS Self/Co-Training

- Two POS Taggers

  [Wang, Huang, Harper, 2007]
  - HMM
  - MEMM
- CTB: Chinese Penn Tree Bank

Table 3. Comparison of the tagging accuracy (%) of the HMM tagger and ME tagger when trained on the entire CTB corpus and the additional Mandarin BN seed corpus and tested on the Mandarin BN POS-eval test set. Known word, unknown word, and overall accuracies are included.

Tagger		Known	Unknown	Overall
HMM	CTB	80.0	69.2	79.0
	CTB+seed	90.5	75.1	89.6
ME	CTB	79.2	66.8	78.5
	CTB+seed	89.2	74.0	88.1

Table 4. Overall POS tagging accuracy (%) on the Mandarin BN POS-eval test set after applying self-training and co-training.

Training Condition		Tagger	
		HMM	ME
Initial (i.e., CTB+seed)		89.6	88.1
self-training		90.8	90.2
co-training	naive	91.9	91.8
	agreement-based	94.1	94.1
	max-score	93.2	93.1
	max-t-min-s	94.1	93.9