

# Self-training Models with Latent Variables for Natural Language Processing

Zhongqiang Huang

Department of Computer Science  
University of Maryland  
College Park, MD

# Overview

- Self-training has helped for some NLP tasks
  - but seems to fail for tagging and parsing with moderate amounts of training data
- Models with latent variables
  - flexible at learning dependencies in different granularities
  - benefit much more from self-training

# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?



$t_1^n$ : VBZ DT NN VB NN .

# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?



$t_1^n$ : VBZ DT NN VB NN .

- Bigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}) P(w_i | t_i)$$



# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?

$\Downarrow$

$t_1^n$ : VBZ DT (NN)  $\rightarrow$  (VB) NN .

- Bigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}) P(w_i | t_i)$$

# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?  
↓  
 $t_1^n$ : VBZ DT NN **VB** NN .



- Bigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}) P(w_i | t_i)$$

# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?

$\Downarrow$

$t_1^n$ : VBZ DT NN VB NN .

- Bigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}) P(w_i | t_i)$$

- Trigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}, t_{i-2}) P(w_i | t_i, t_{i-1})$$

# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?

$\Downarrow$

$t_1^n$ : VBZ (DT NN)  $\rightarrow$  (VB) NN .

- Bigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}) P(w_i | t_i)$$

- Trigram HMM tagger


$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}, t_{i-2}) P(w_i | t_i, t_{i-1})$$



# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?  
↓  
 $t_1^n$ : VBZ DT NN VB NN .



- Bigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}) P(w_i | t_i)$$

- Trigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}, t_{i-2}) P(w_i | t_i, t_{i-1})$$

# Part-of-Speech Tagging

- Assigns Part-of-Speech tags to words in a sentence:

$w_1^n$ : Does that flight serve dinner ?



$t_1^n$ : VBZ DT NN VB NN .

- Bigram HMM tagger

$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}) P(w_i | t_i)$$

- Trigram HMM tagger

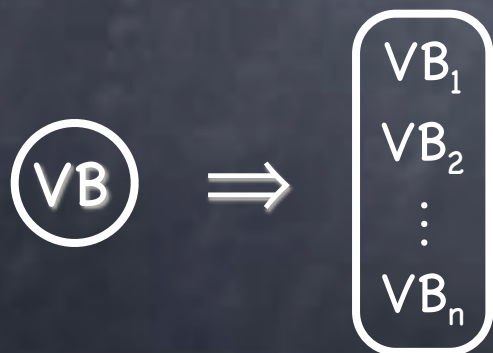
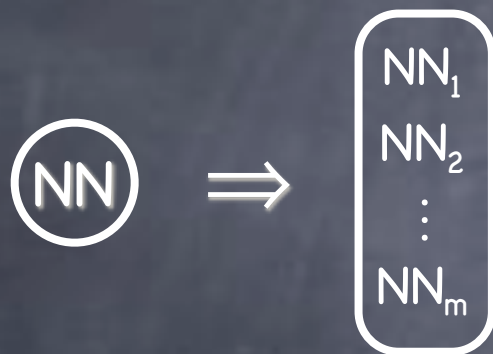
$$P(w_1^n, t_1^n) \approx \prod_i P(t_i | t_{i-1}, t_{i-2}) P(w_i | t_i, t_{i-1})$$

## PROBLEMS

- Bigram tagger:
  - Strong independence assumption
  - Poor performance
- Trigram tagger:
  - Some trigrams too sparse, need sophisticated smoothing
  - Some trigrams are abundant, could use higher n-grams
- Higher order taggers?
  - Data sparseness, harder to smooth

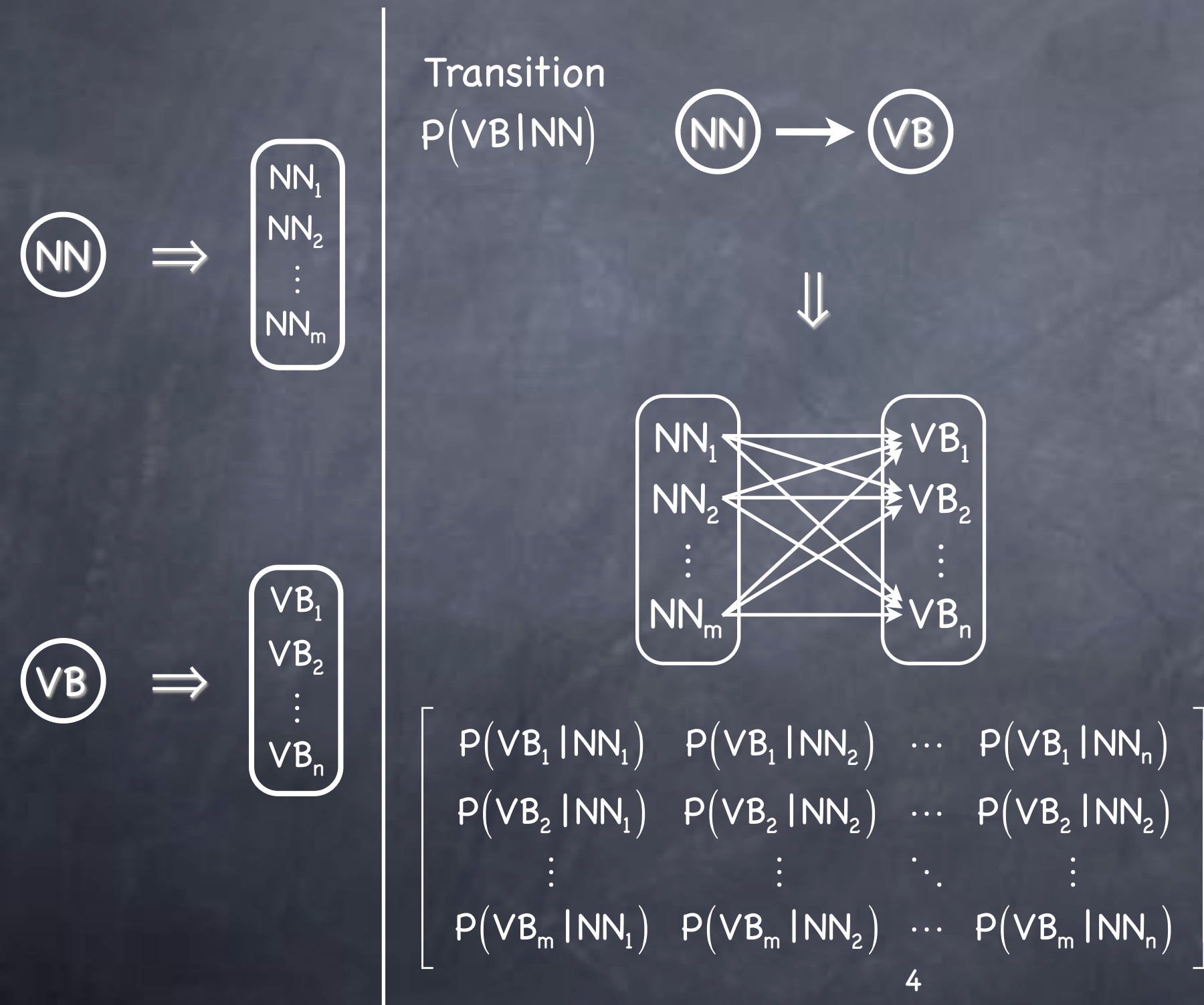
# Tagging with Latent Variables

[Huang et al. 2009]



# Tagging with Latent Variables

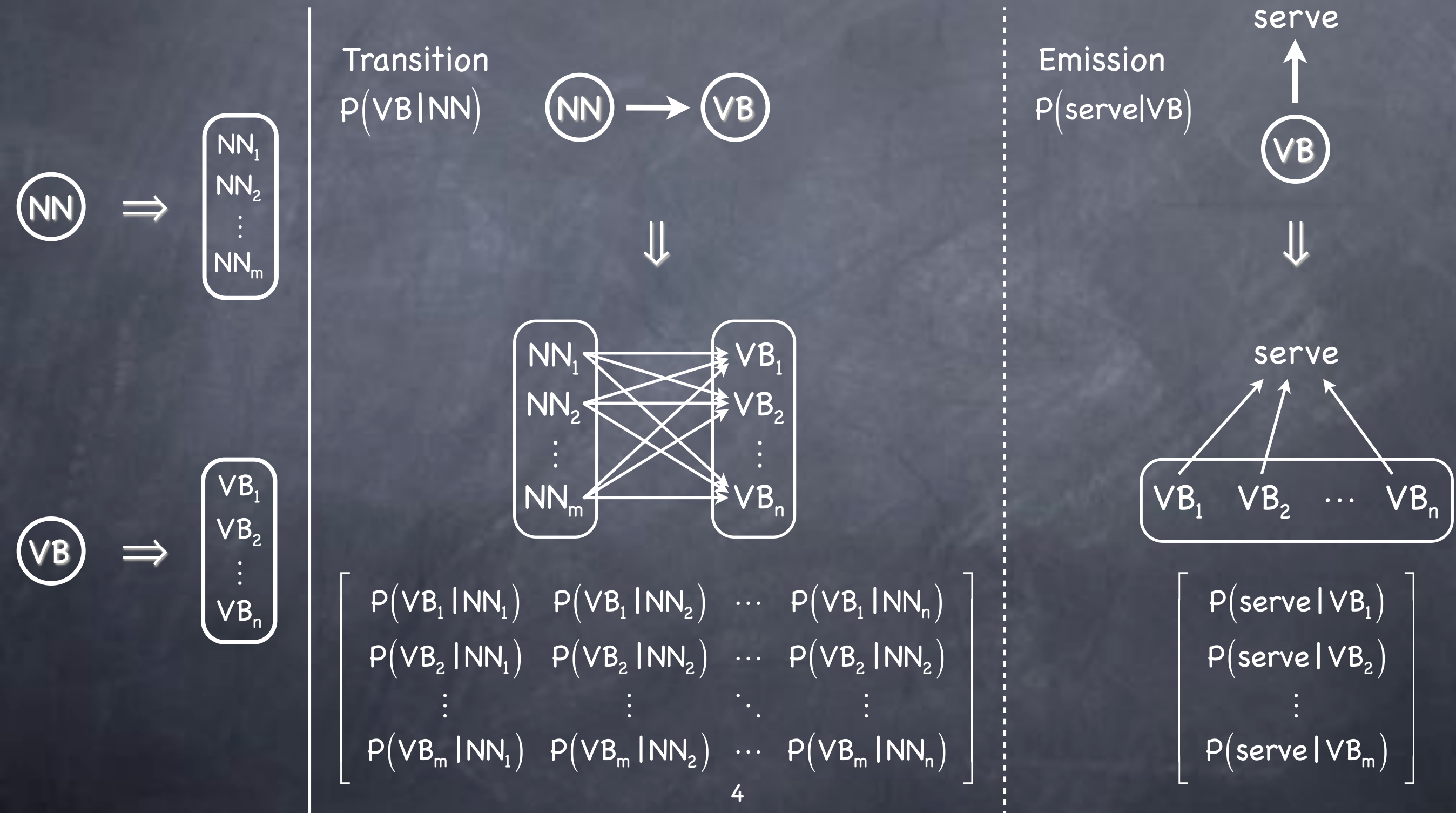
[Huang et al. 2009]



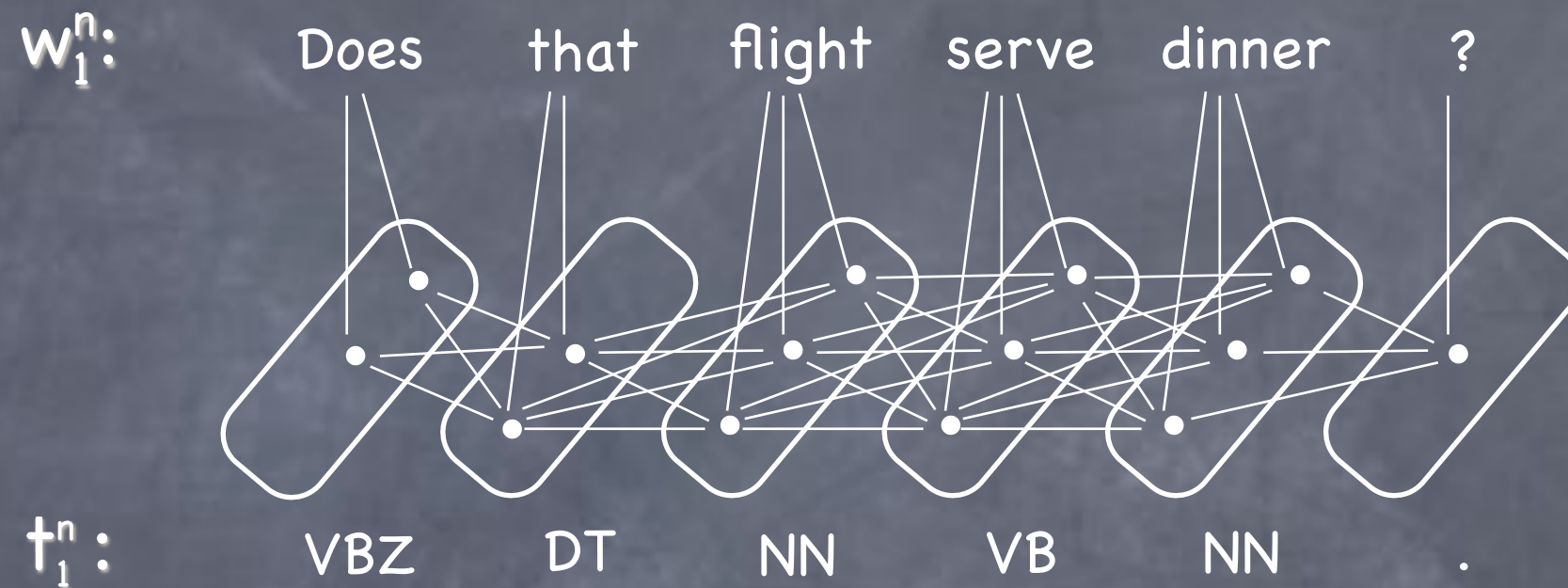


# Tagging with Latent Variables

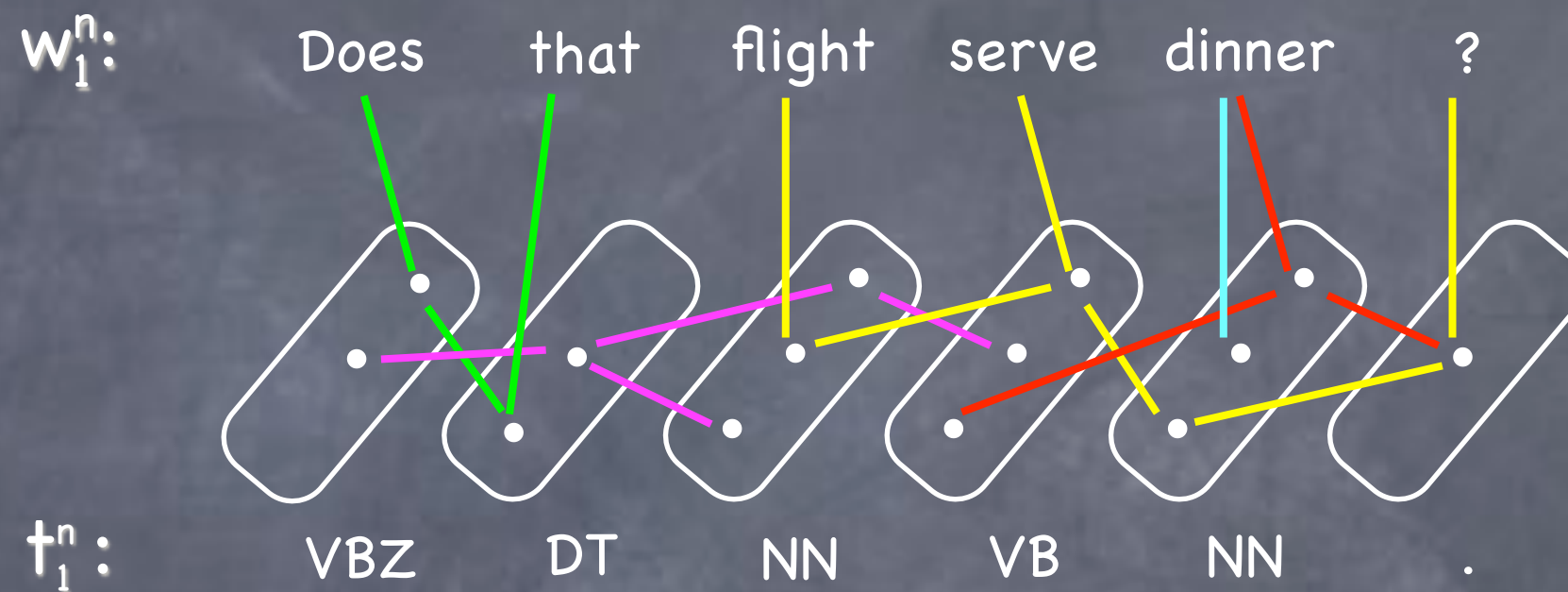
[Huang et al. 2009]



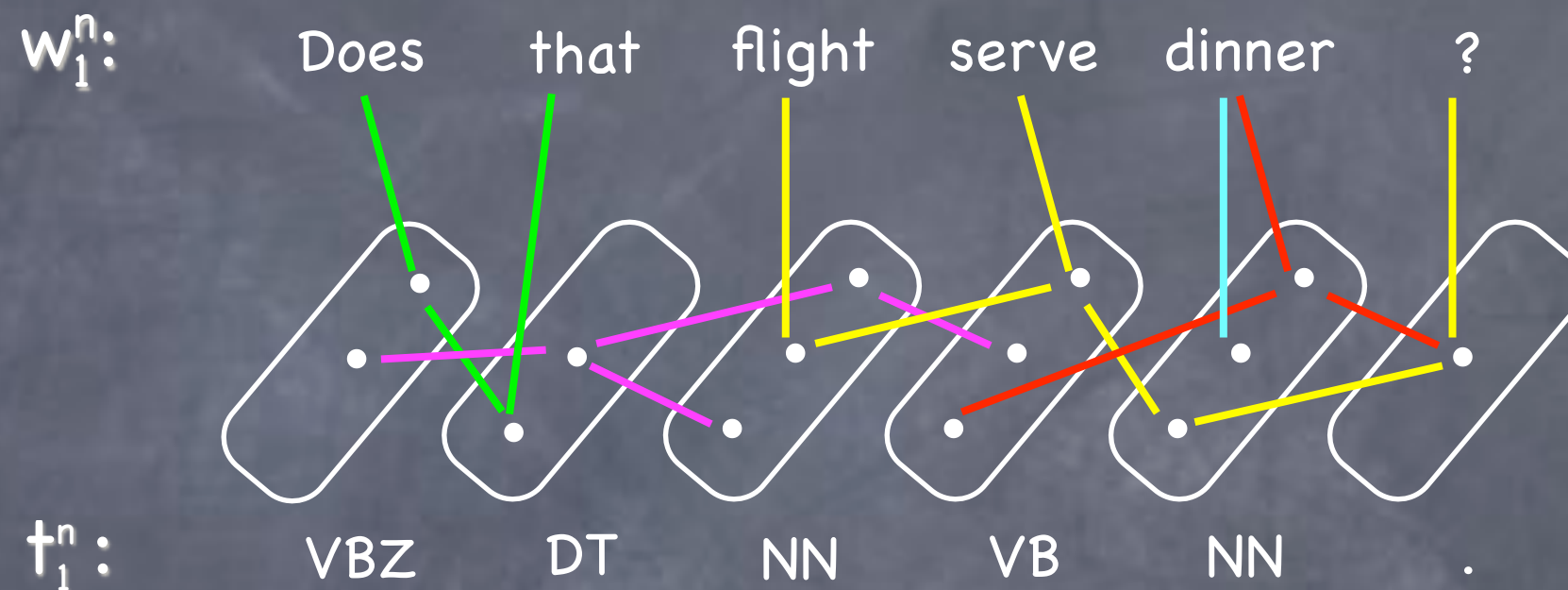
# Tagging with Latent Variables (cont.)



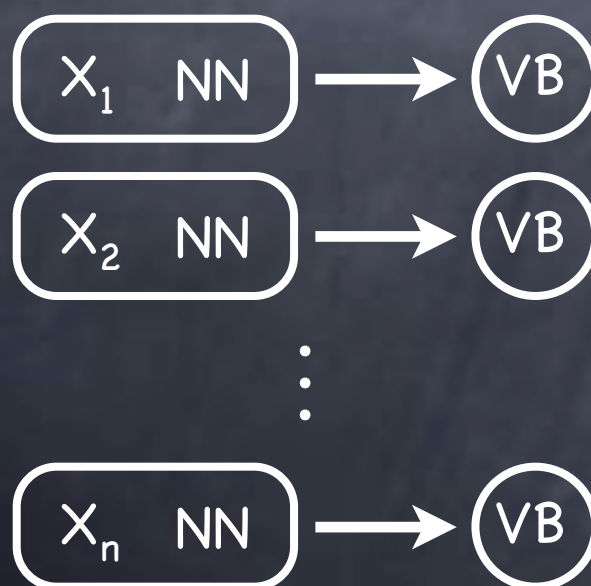
# Tagging with Latent Variables (cont.)



# Tagging with Latent Variables (cont.)

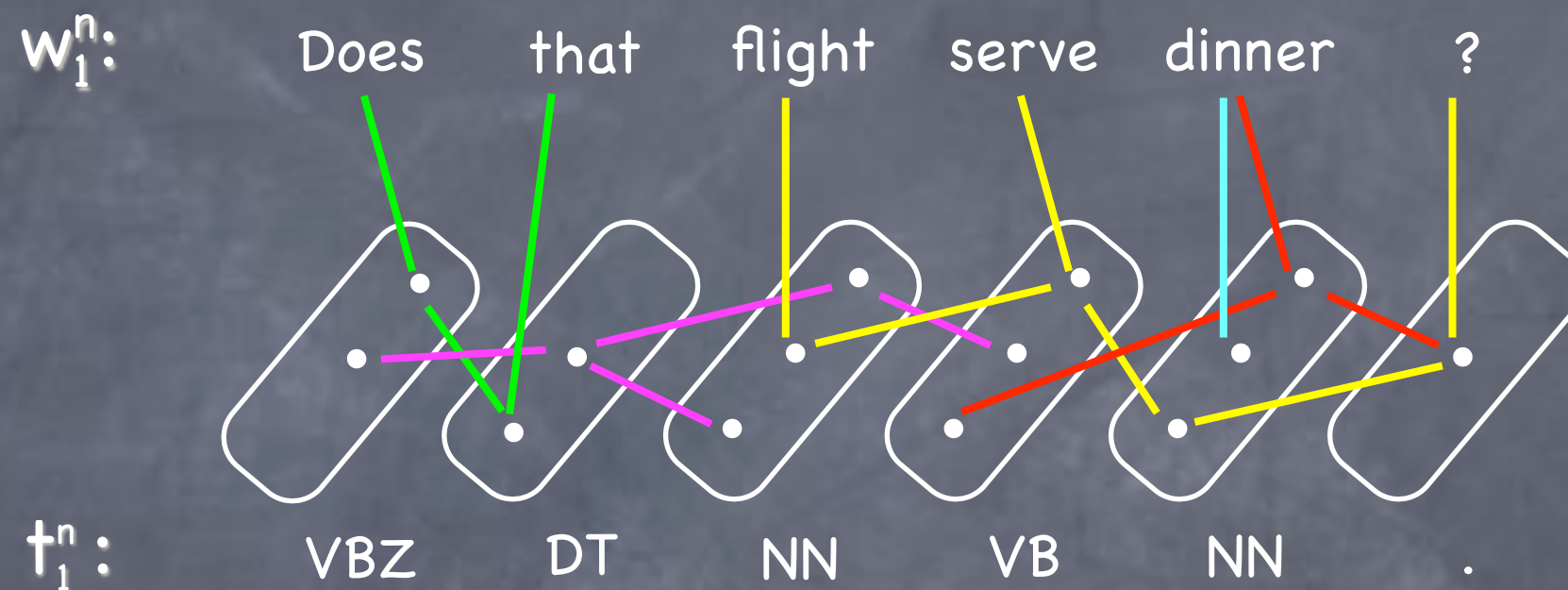


Trigram tagger

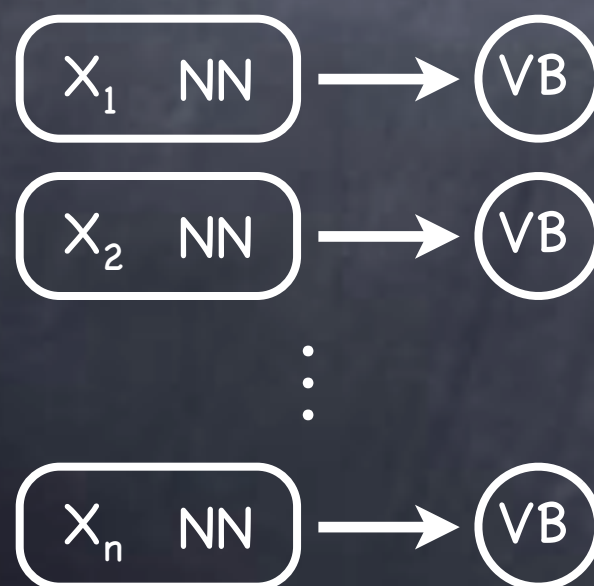




# Tagging with Latent Variables (cont.)



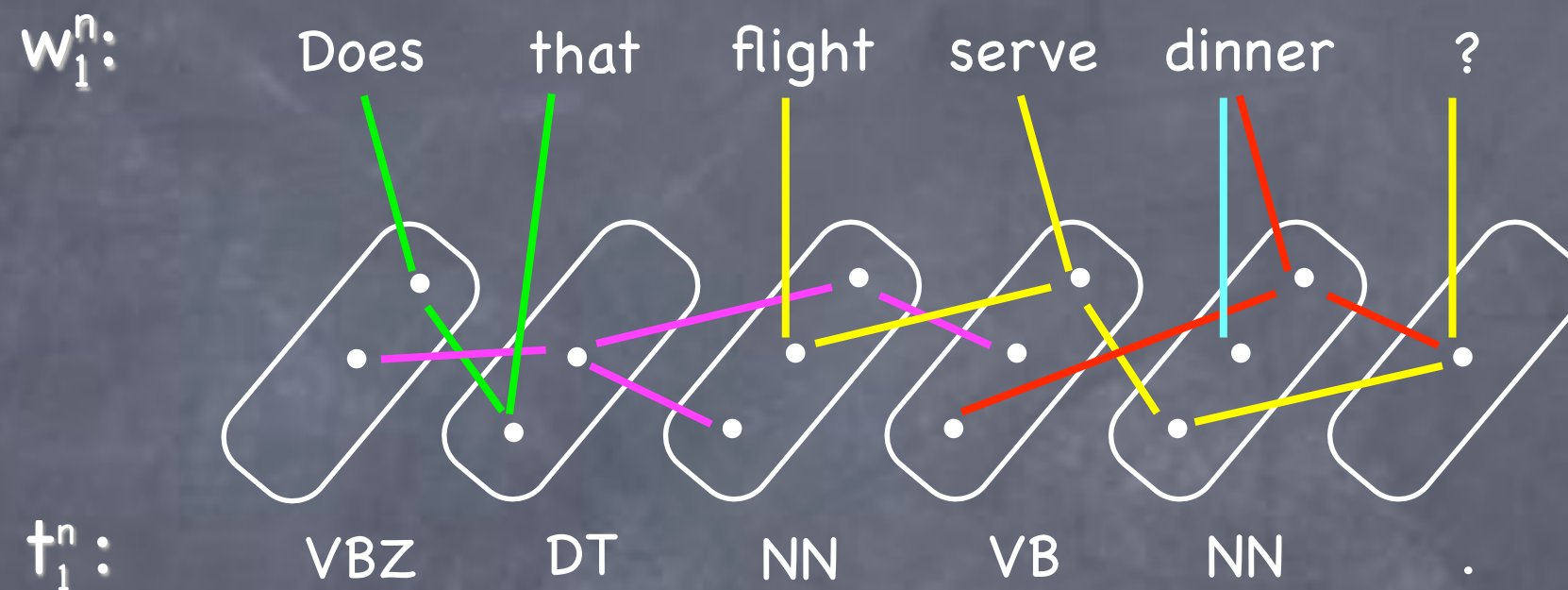
Trigram tagger



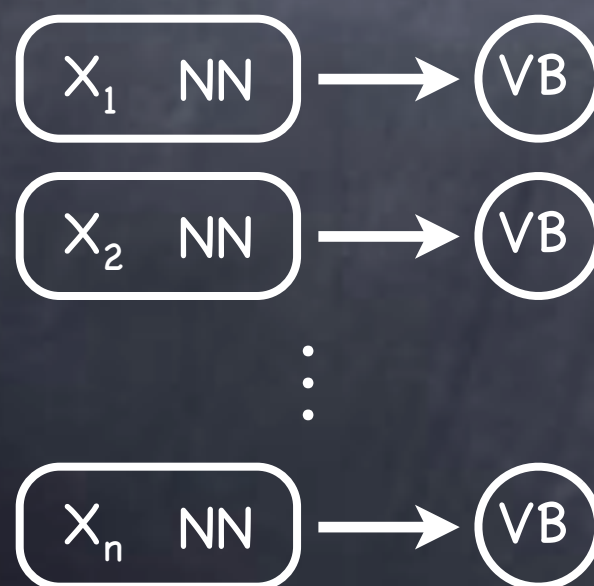
$\Rightarrow$

for  $1 \leq i \leq n$ ,  
 $X_i \text{ NN} \Rightarrow \text{NN}_i$   
 and  
 $\text{NN VB} \Rightarrow \text{VB}_k$

# Tagging with Latent Variables (cont.)

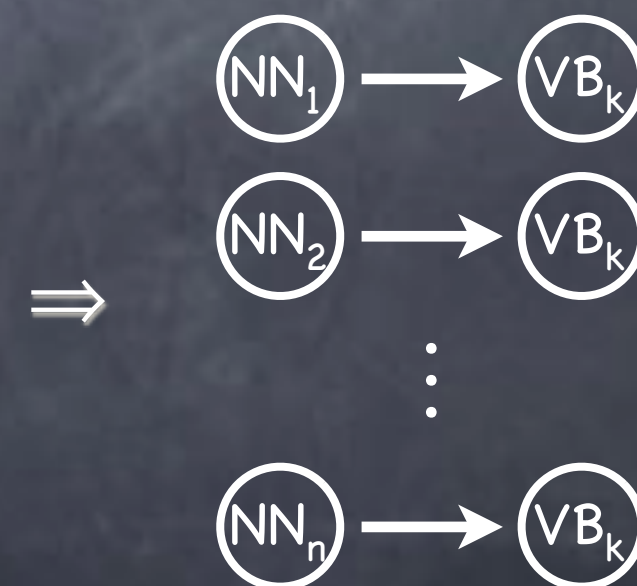


Trigram tagger



Bigram tagger with latent variables

for  $1 \leq i \leq n$ ,  
 $X_i \text{ NN} \Rightarrow \text{NN}_i$   
 and  
 $\text{NN VB} \Rightarrow \text{VB}_k$



# Performance of Chinese POS Taggers

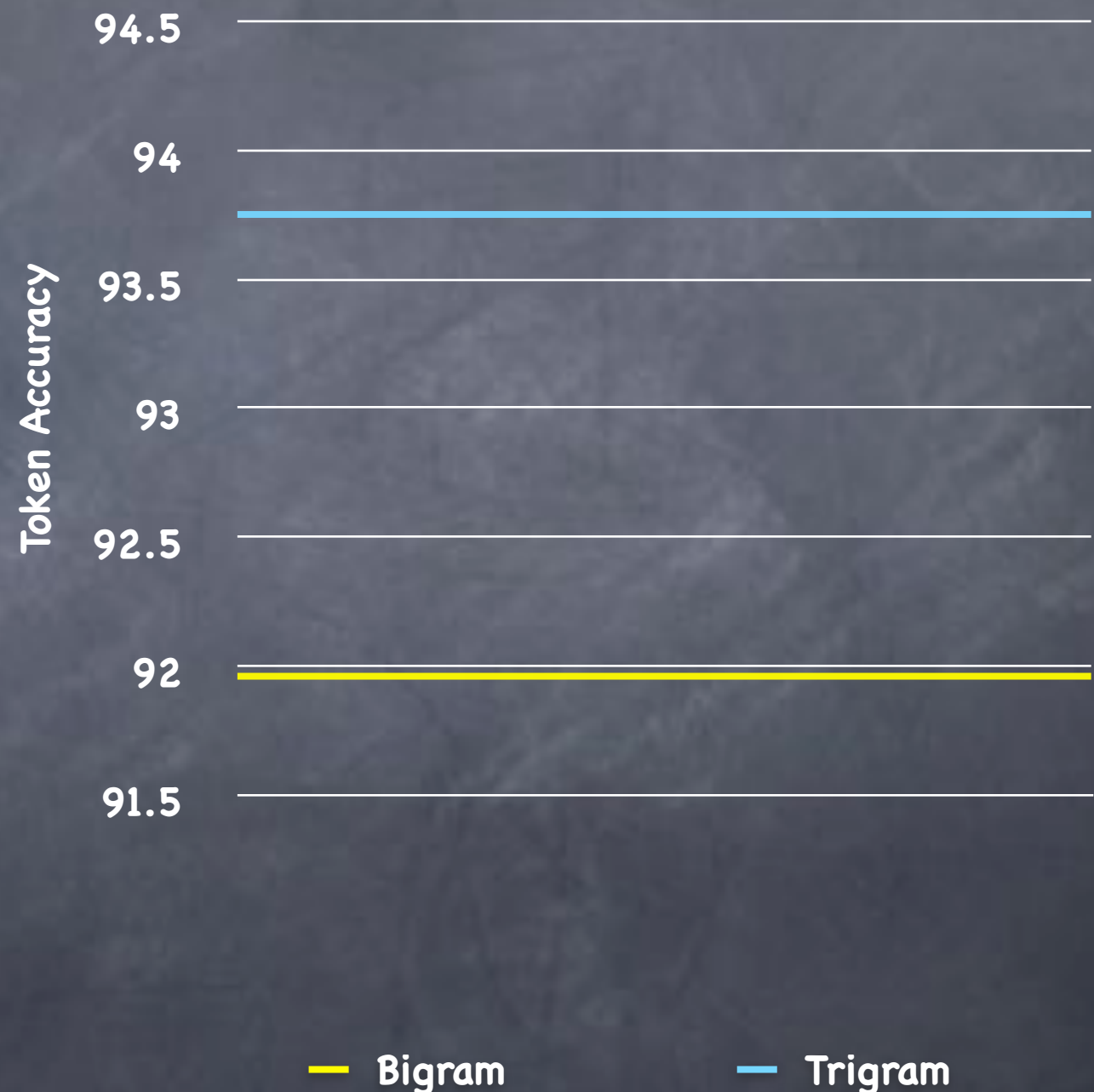
## EXPERIMENT SETUP

- The traditional bigram and trigram HMM taggers
- The bigram HMM tagger with latent variables: with increasing amounts of latent tags
- Penn Chinese Treebank 6.0 (CTB6)
- 28k labeled sentences in total
- 80% for training, 10% for dev, 10% for test

# Performance of Chinese POS Taggers

## EXPERIMENT SETUP

- The traditional bigram and trigram HMM taggers
- The bigram HMM tagger with latent variables: with increasing amounts of latent tags
- Penn Chinese Treebank 6.0 (CTB6)
- 28k labeled sentences in total
- 80% for training, 10% for dev, 10% for test

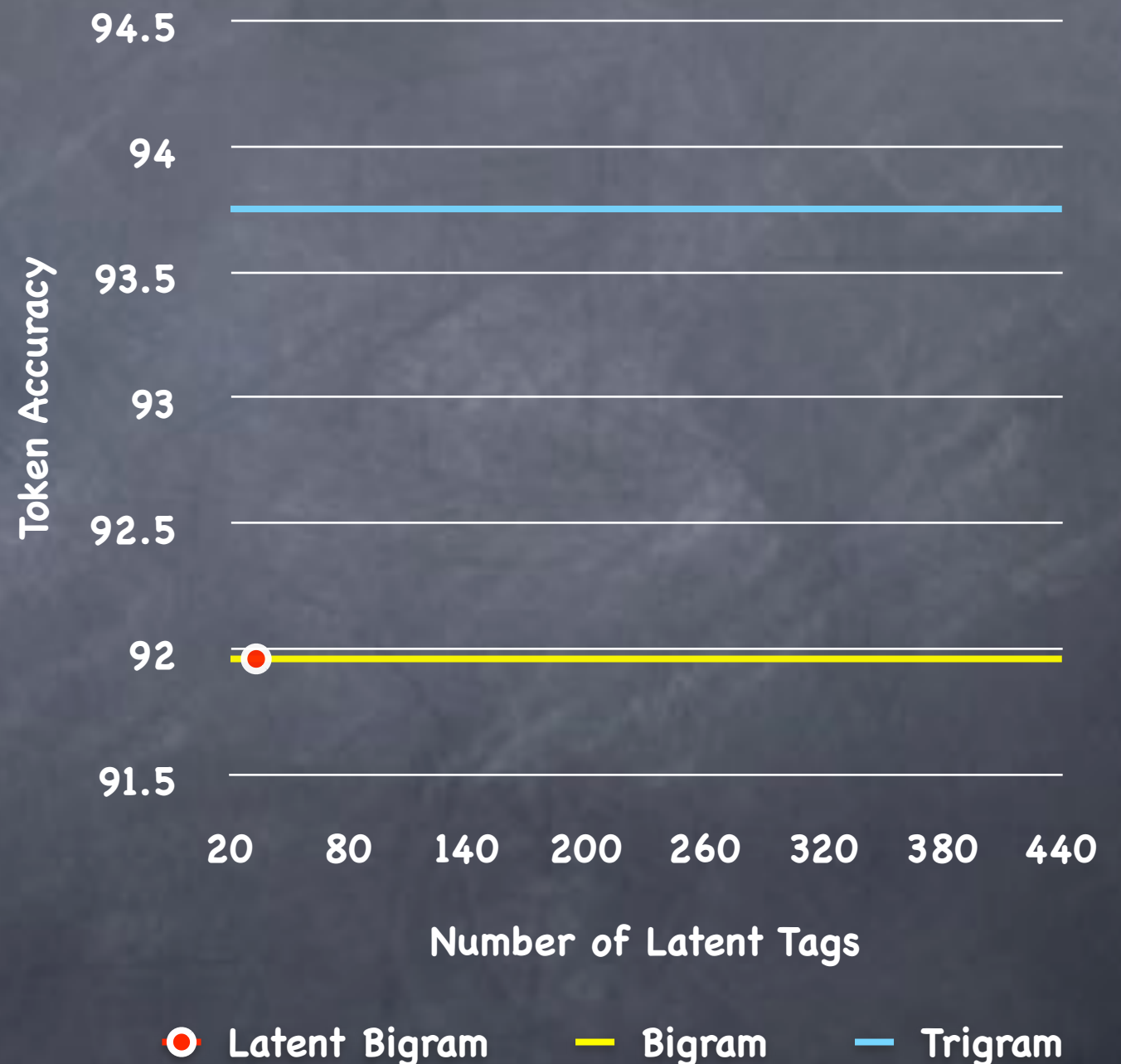




# Performance of Chinese POS Taggers

## EXPERIMENT SETUP

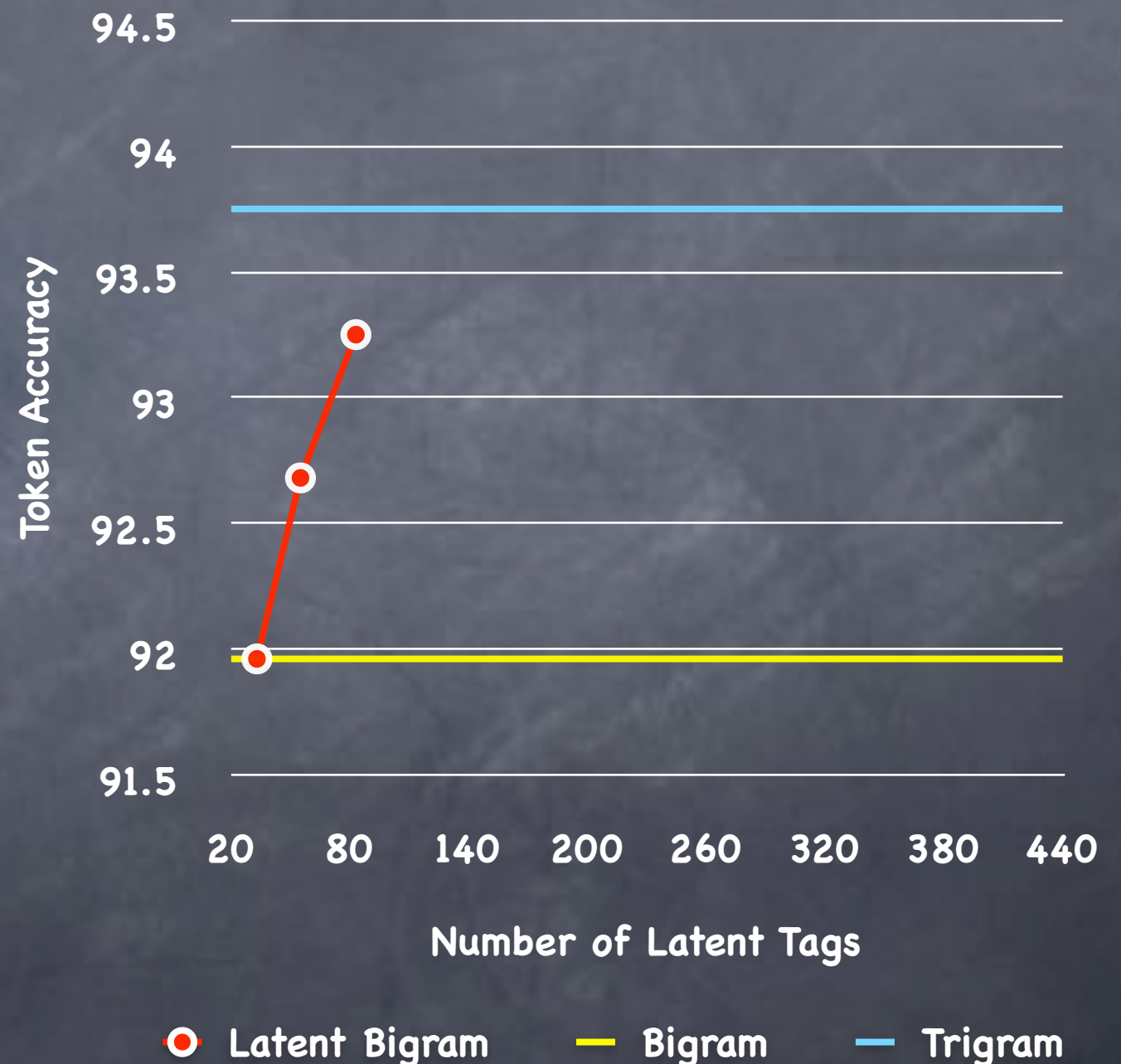
- The traditional bigram and trigram HMM taggers
- The bigram HMM tagger with latent variables: with increasing amounts of latent tags
- Penn Chinese Treebank 6.0 (CTB6)
- 28k labeled sentences in total
- 80% for training, 10% for dev, 10% for test



# Performance of Chinese POS Taggers

## EXPERIMENT SETUP

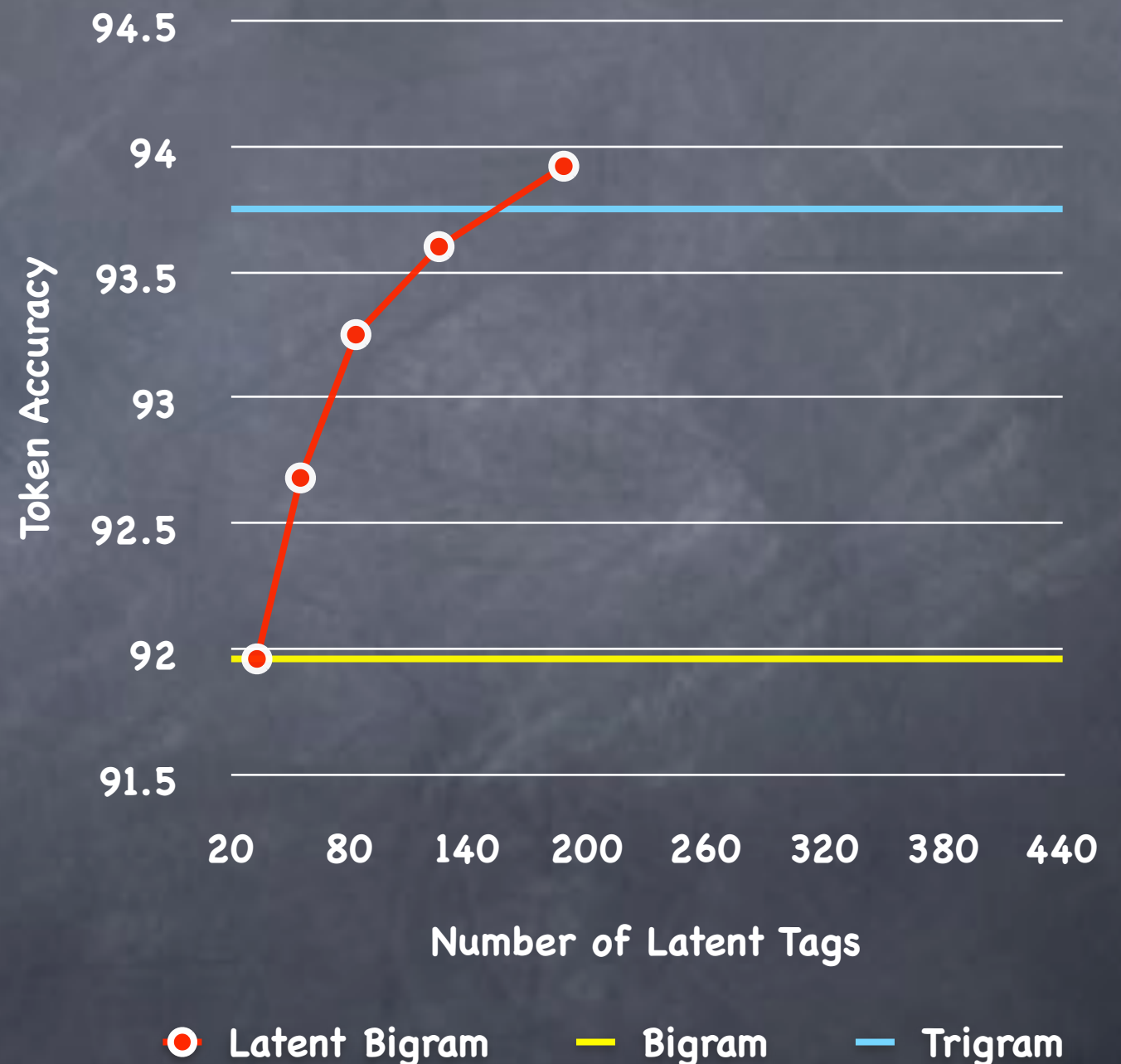
- The traditional bigram and trigram HMM taggers
- The bigram HMM tagger with latent variables: with increasing amounts of latent tags
- Penn Chinese Treebank 6.0 (CTB6)
- 28k labeled sentences in total
- 80% for training, 10% for dev, 10% for test



# Performance of Chinese POS Taggers

## EXPERIMENT SETUP

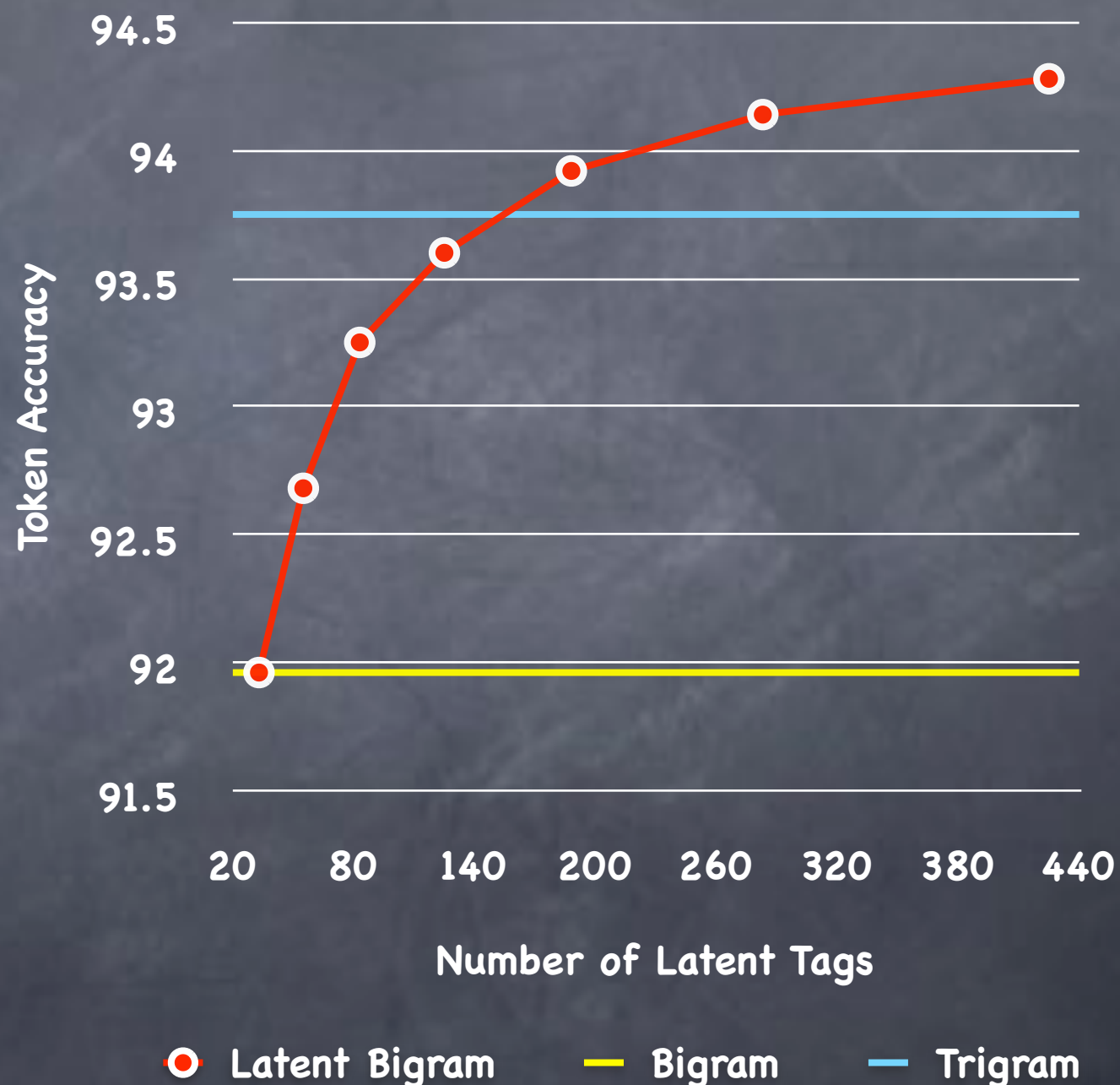
- The traditional bigram and trigram HMM taggers
- The bigram HMM tagger with latent variables: with increasing amounts of latent tags
- Penn Chinese Treebank 6.0 (CTB6)
- 28k labeled sentences in total
- 80% for training, 10% for dev, 10% for test



# Performance of Chinese POS Taggers

## EXPERIMENT SETUP

- The traditional bigram and trigram HMM taggers
- The bigram HMM tagger with latent variables: with increasing amounts of latent tags
- Penn Chinese Treebank 6.0 (CTB6)
- 28k labeled sentences in total
- 80% for training, 10% for dev, 10% for test





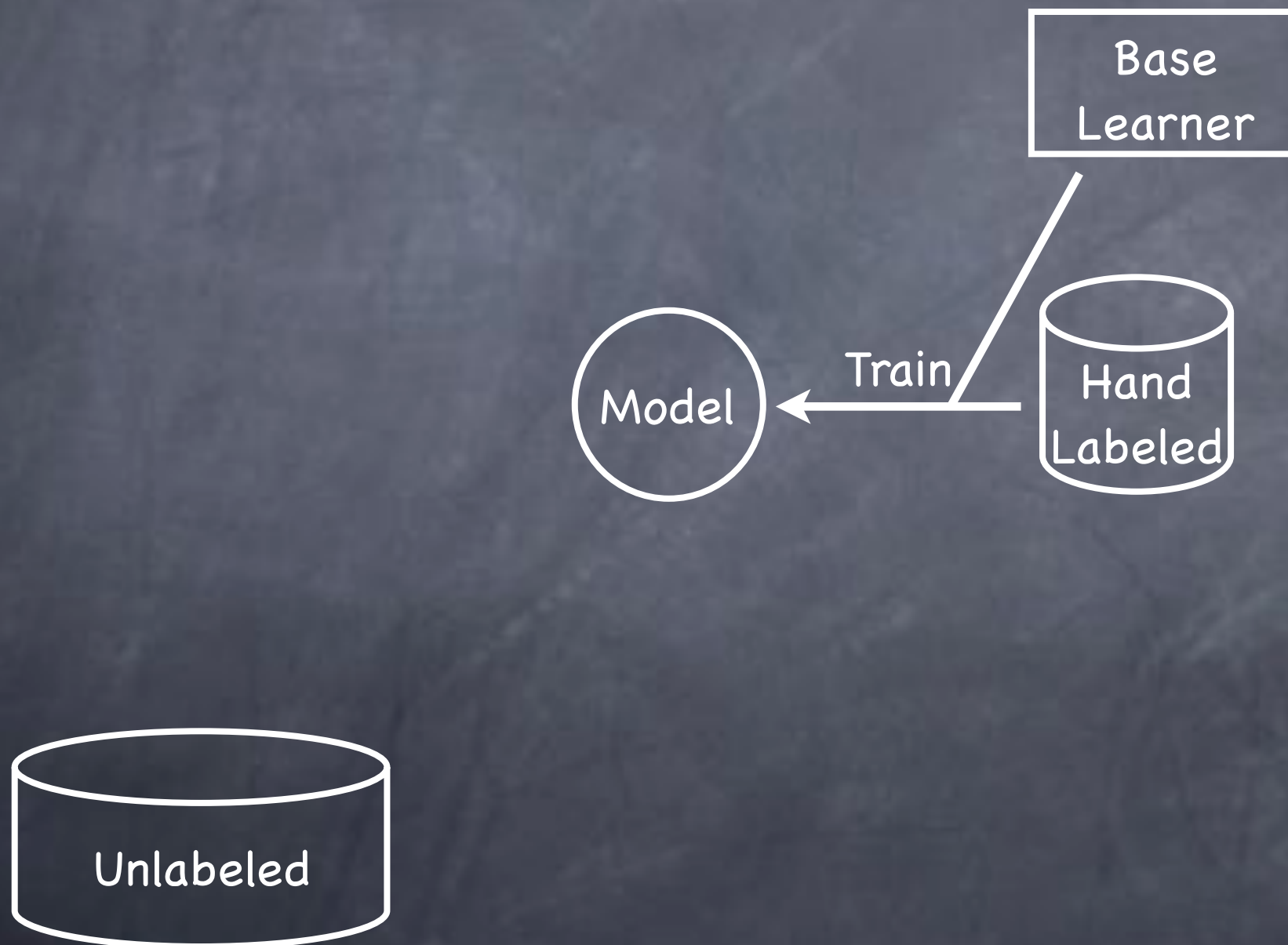
# Self-training (ST)

- A simple semi-supervised learning method



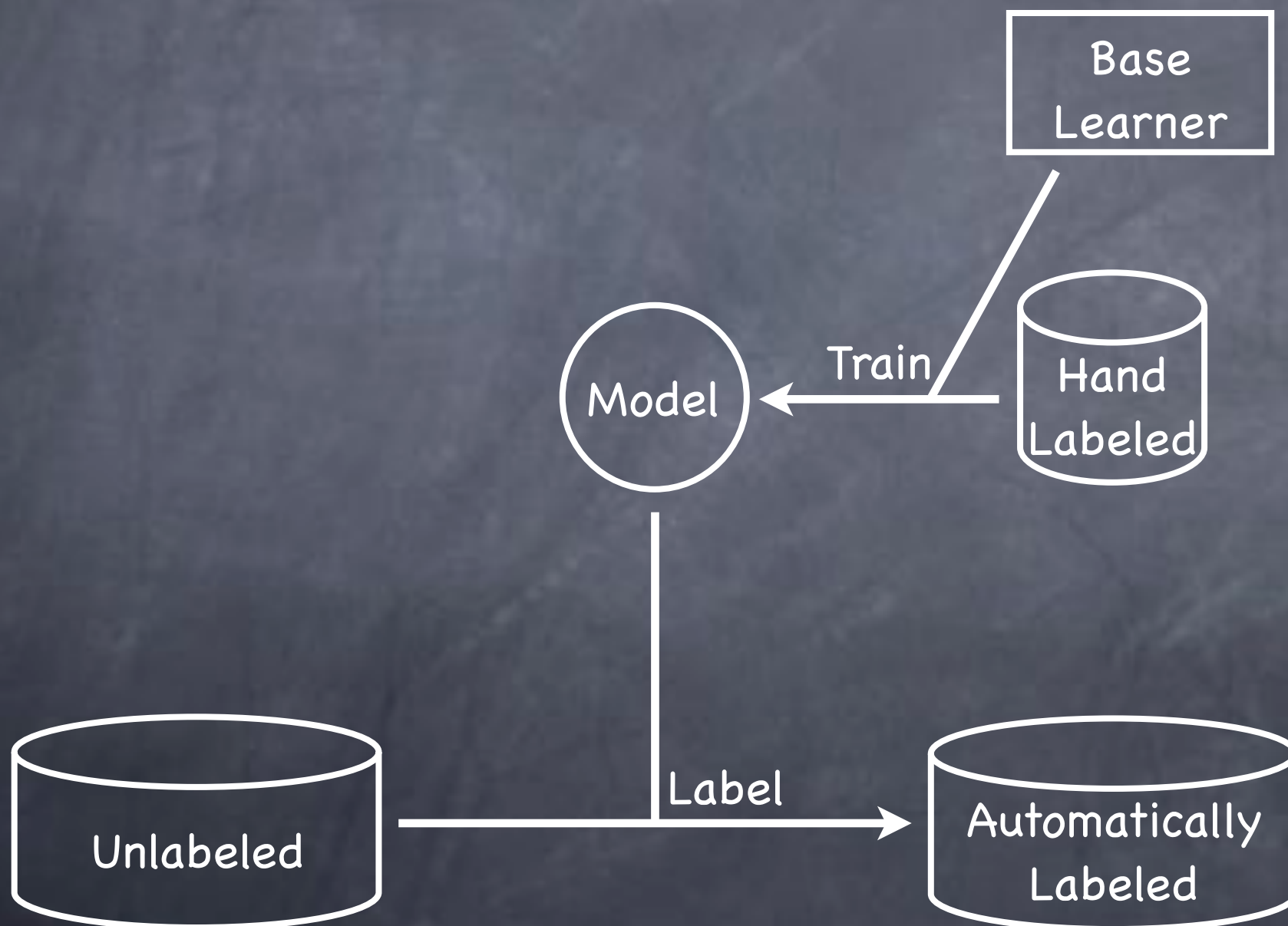
# Self-training (ST)

- A simple semi-supervised learning method



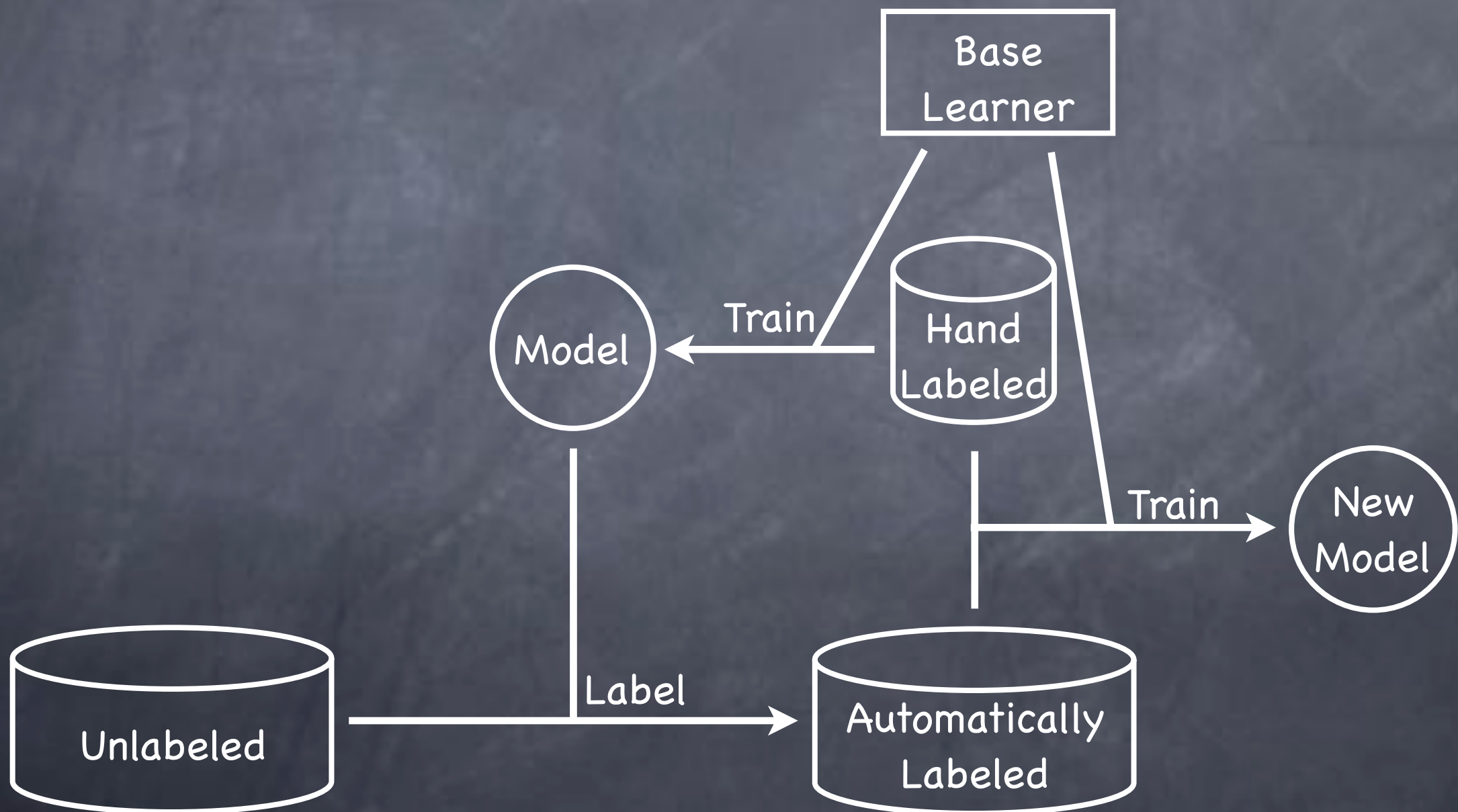
# Self-training (ST)

- A simple semi-supervised learning method



# Self-training (ST)

- A simple semi-supervised learning method





# Self-training Chinese POS Taggers

[Huang et al. 2009]

## EXPERIMENT SETUP

- Penn Chinese Treebank 6: 28k labeled sentences in total; 80% for training, 10% for dev, 10% for test
- 210k unlabeled Chinese newswire sentences for self-training
- Use 10%, 20%, 40%, 60%, and 80% of the hand labeled training data
- Always use the full unlabeled training set (210k sentences) for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model

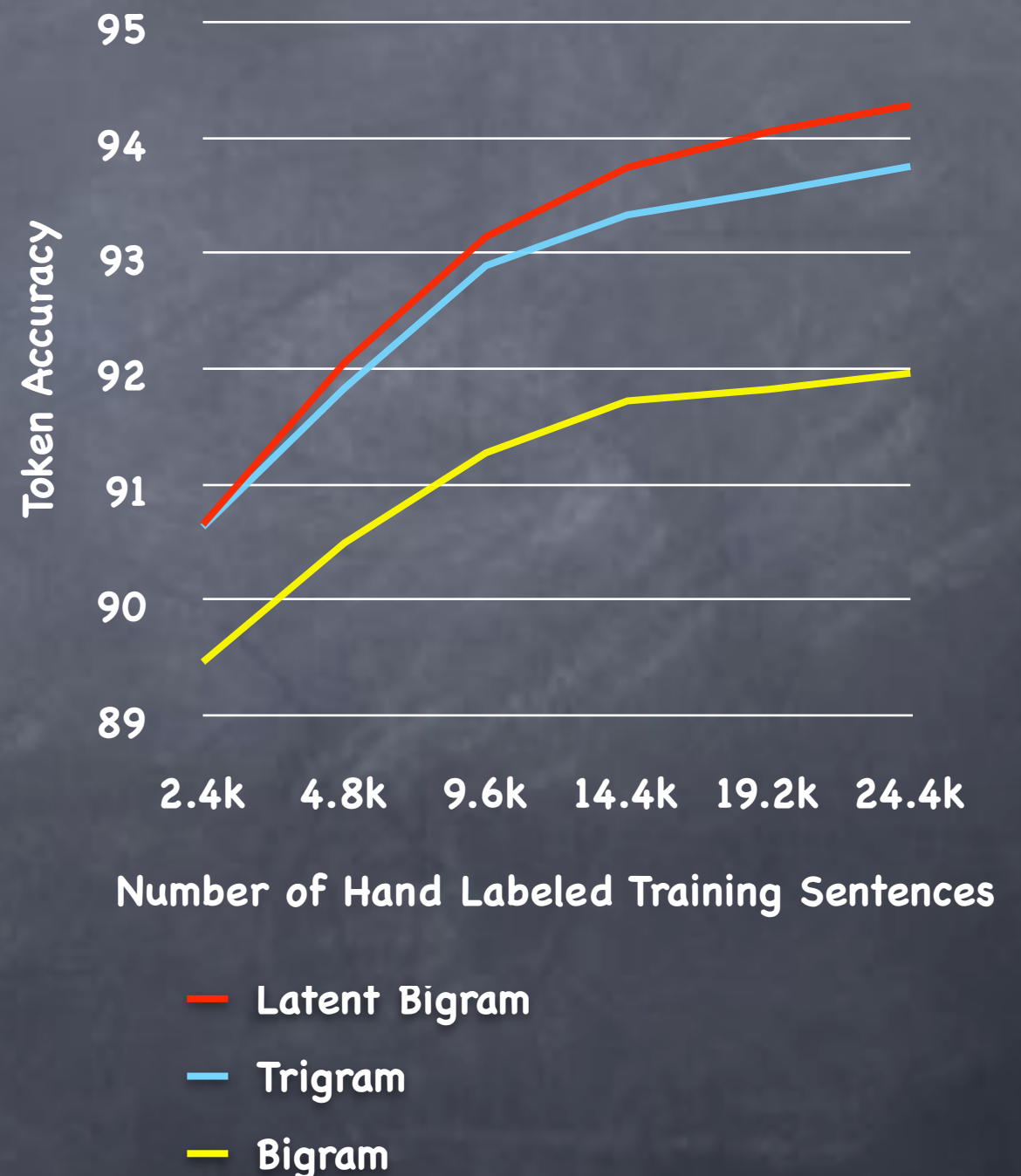
# Self-training Chinese POS Taggers

[Huang et al. 2009]

## EXPERIMENT SETUP

- Penn Chinese Treebank 6: 28k labeled sentences in total; 80% for training, 10% for dev, 10% for test
- 210k unlabeled Chinese newswire sentences for self-training
- Use 10%, 20%, 40%, 60%, and 80% of the hand labeled training data
- Always use the full unlabeled training set (210k sentences) for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model

POS Taggers w/ and w/o Self-Training (ST)



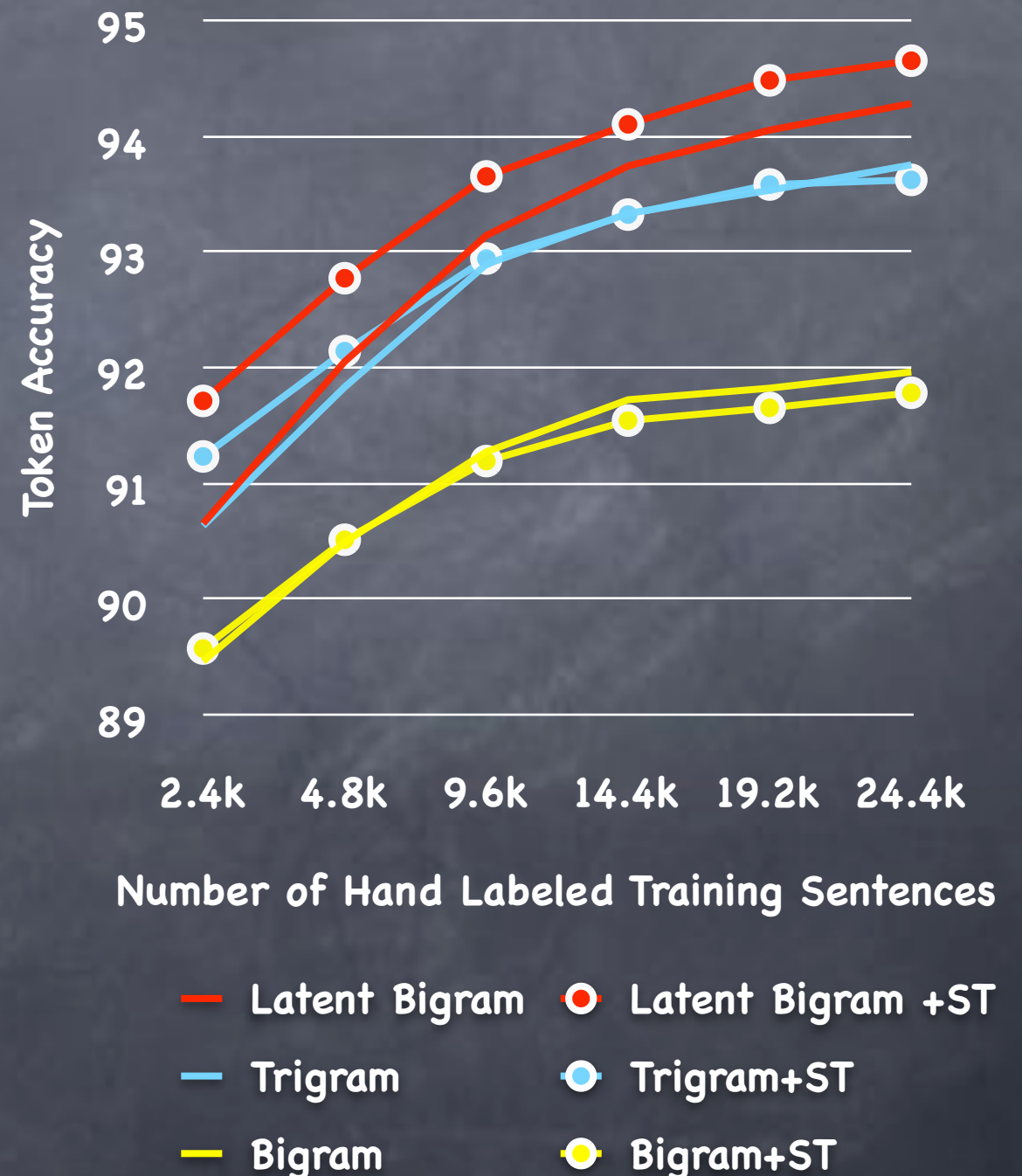
# Self-training Chinese POS Taggers

[Huang et al. 2009]

## EXPERIMENT SETUP

- Penn Chinese Treebank 6: 28k labeled sentences in total; 80% for training, 10% for dev, 10% for test
- 210k unlabeled Chinese newswire sentences for self-training
- Use 10%, 20%, 40%, 60%, and 80% of the hand labeled training data
- Always use the full unlabeled training set (210k sentences) for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model

POS Taggers w/ and w/o Self-Training (ST)



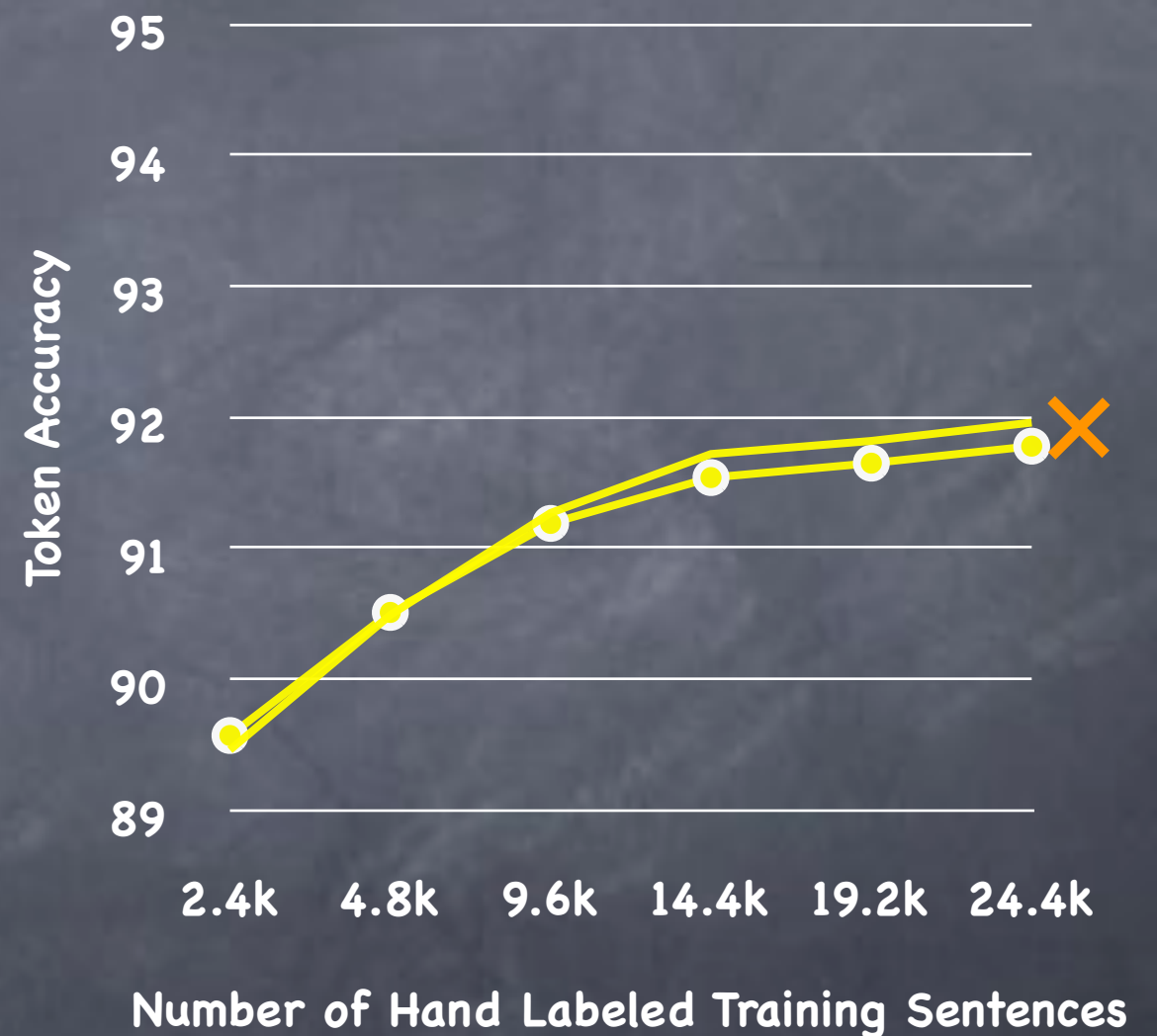
# Self-training Chinese POS Taggers

[Huang et al. 2009]

## EXPERIMENT SETUP

- Penn Chinese Treebank 6: 28k labeled sentences in total; 80% for training, 10% for dev, 10% for test
- 210k unlabeled Chinese newswire sentences for self-training
- Use 10%, 20%, 40%, 60%, and 80% of the hand labeled training data
- Always use the full unlabeled training set (210k sentences) for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model

POS Taggers w/ and w/o Self-Training (ST)



— Bigram    ● Bigram+ST

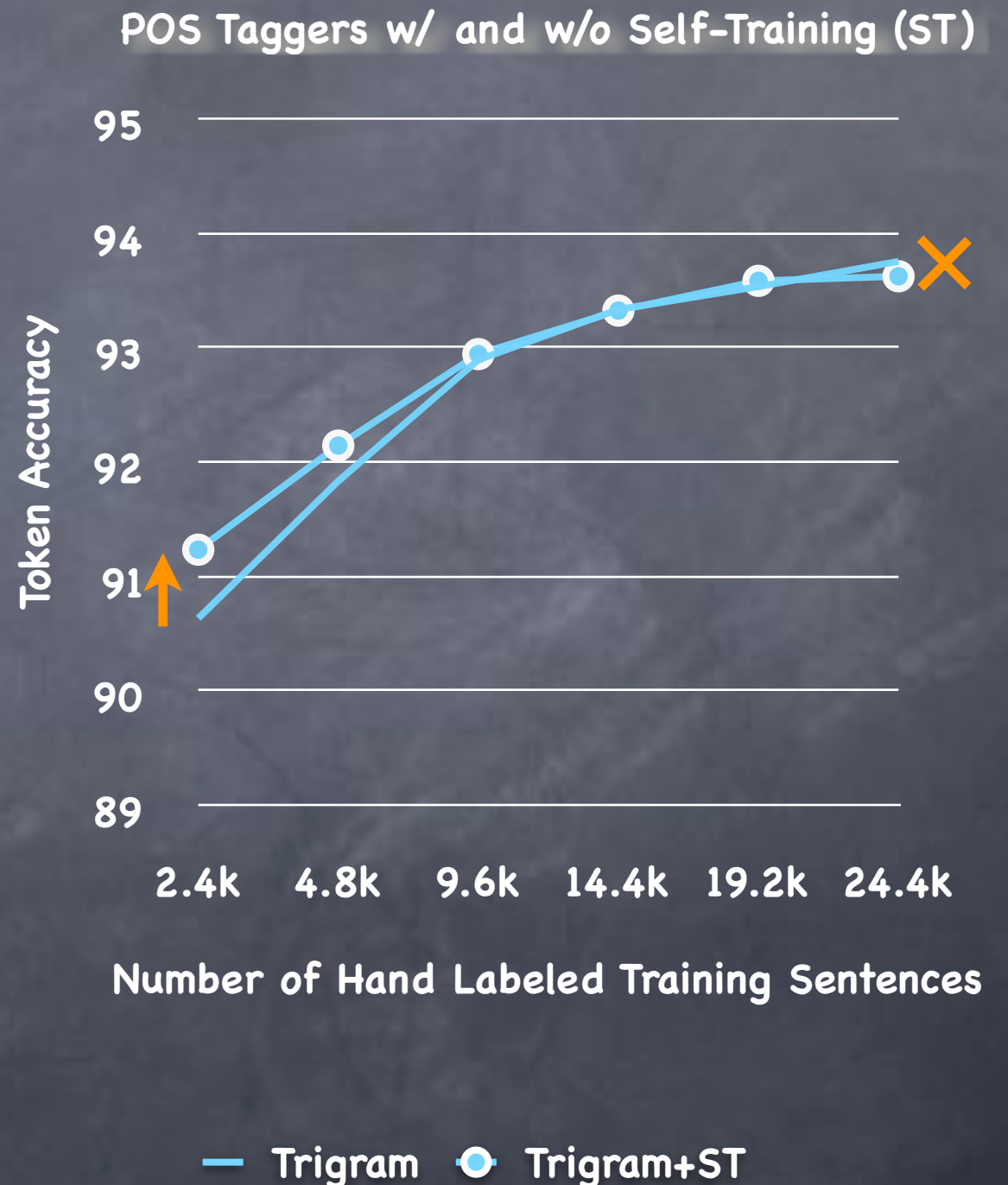


# Self-training Chinese POS Taggers

[Huang et al. 2009]

## EXPERIMENT SETUP

- Penn Chinese Treebank 6: 28k labeled sentences in total; 80% for training, 10% for dev, 10% for test
- 210k unlabeled Chinese newswire sentences for self-training
- Use 10%, 20%, 40%, 60%, and 80% of the hand labeled training data
- Always use the full unlabeled training set (210k sentences) for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model



# Self-training Chinese POS Taggers

[Huang et al. 2009]

## EXPERIMENT SETUP

- Penn Chinese Treebank 6: 28k labeled sentences in total; 80% for training, 10% for dev, 10% for test
- 210k unlabeled Chinese newswire sentences for self-training
- Use 10%, 20%, 40%, 60%, and 80% of the hand labeled training data
- Always use the full unlabeled training set (210k sentences) for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model



# Self-training English Syntactic Parsers

[Huang and Harper 2009]

## EXPERIMENT SETUP

- PCFG parser without latent variables:  
Charniak's lexicalized PCFG parser
- PCFG parser with latent variables: a  
modified Berkeley Parser
- Penn Treebank: sections 2-19 for training,  
22 for dev, 23 for test.
- 210k unlabeled English newswire  
sentences for self-training
- Use 20%, 40%, 60%, and 80% of the  
labeled training data
- Always use the full unlabeled training set  
for self-training
- Combine the hand labeled and  
automatically unlabeled data in a  
weighted manner, so that each  
contributes 50% to the final model

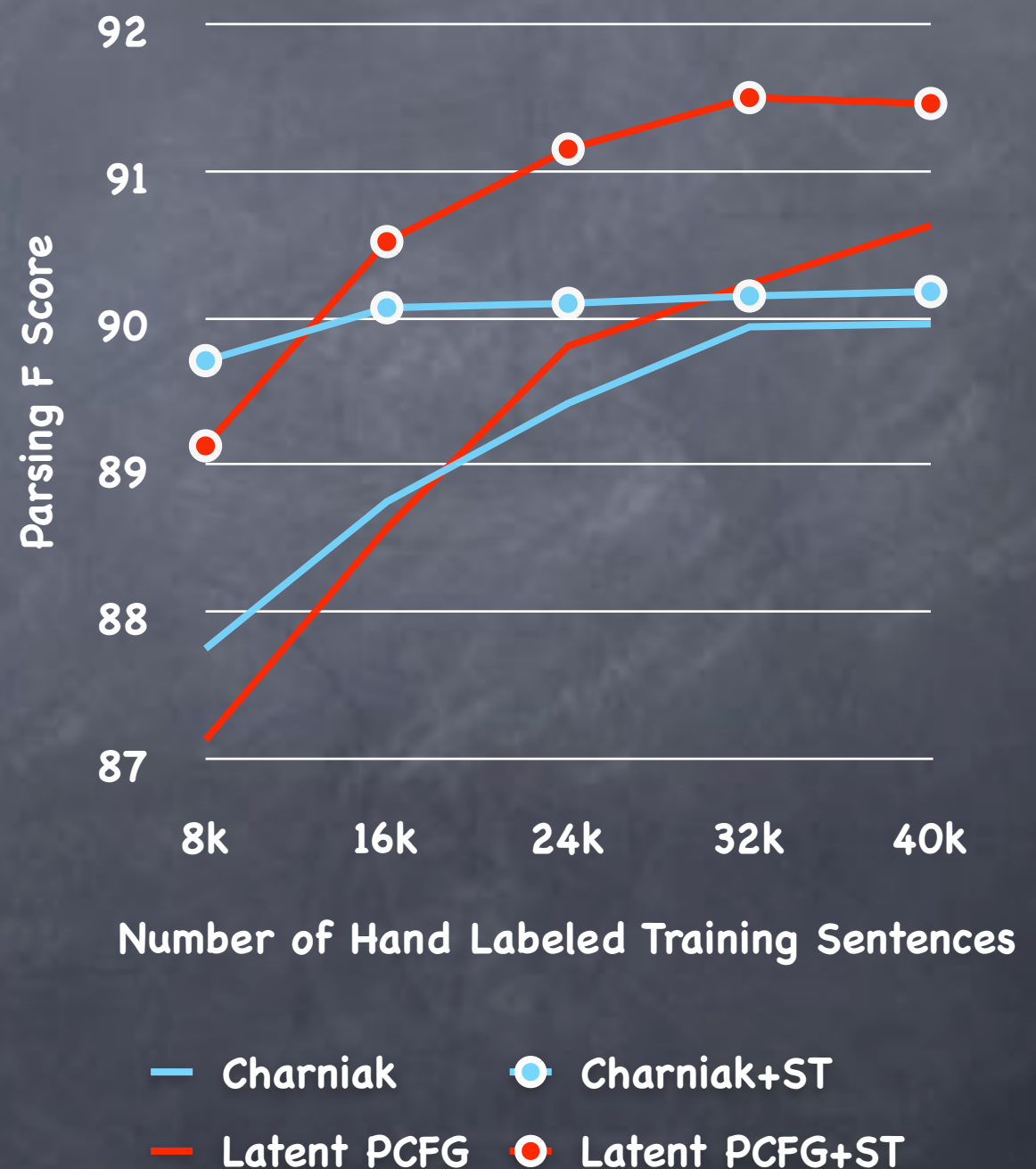
# Self-training English Syntactic Parsers

[Huang and Harper 2009]

## EXPERIMENT SETUP

- PCFG parser without latent variables: Charniak's lexicalized PCFG parser
- PCFG parser with latent variables: a modified Berkeley Parser
- Penn Treebank: sections 2-19 for training, 22 for dev, 23 for test.
- 210k unlabeled English newswire sentences for self-training
- Use 20%, 40%, 60%, and 80% of the labeled training data
- Always use the full unlabeled training set for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model

Syntactic Parsers w/ and w/o Self-Training (ST)





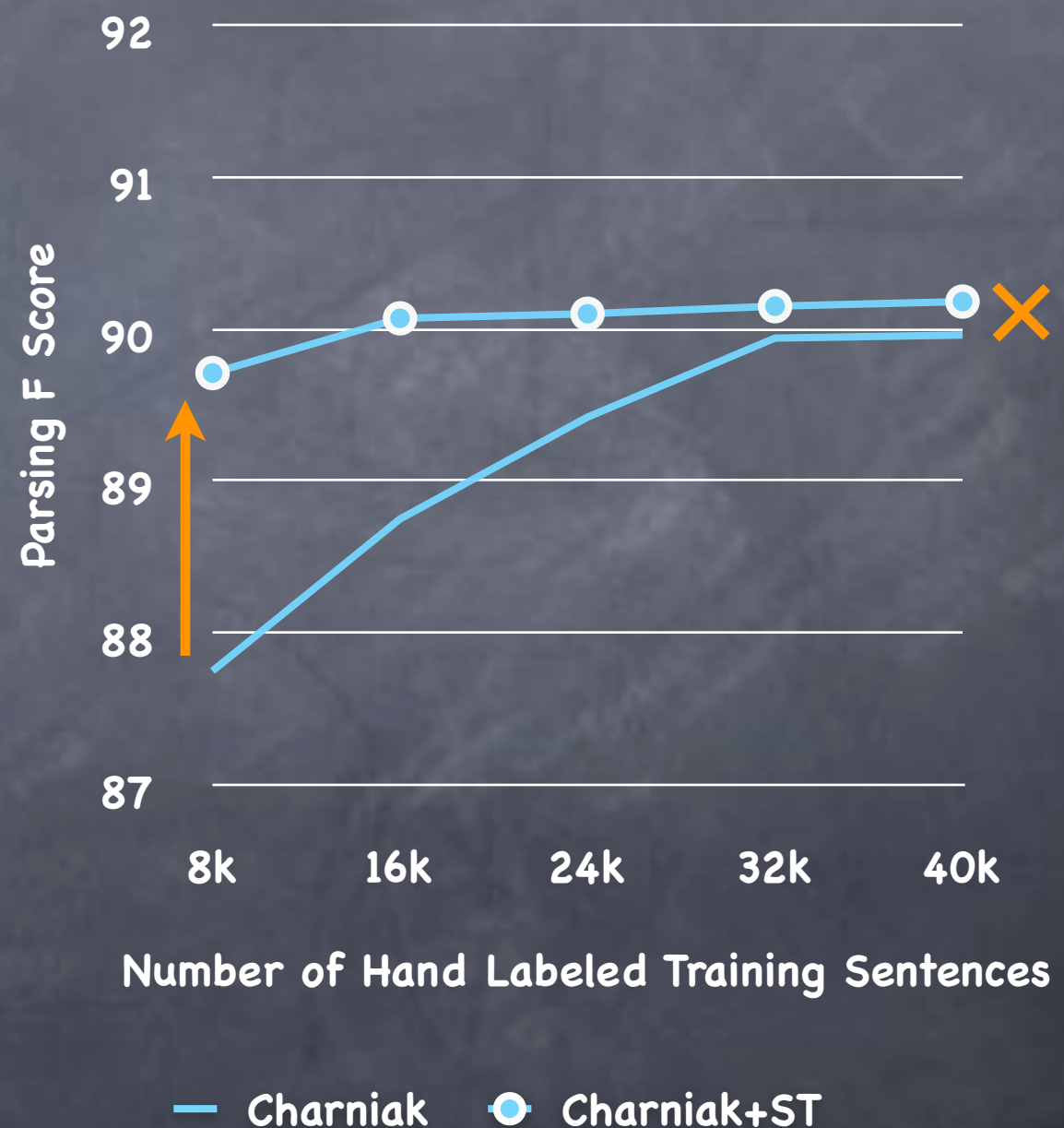
# Self-training English Syntactic Parsers

[Huang and Harper 2009]

## EXPERIMENT SETUP

- PCFG parser without latent variables: Charniak's lexicalized PCFG parser
- PCFG parser with latent variables: a modified Berkeley Parser
- Penn Treebank: sections 2-19 for training, 22 for dev, 23 for test.
- 210k unlabeled English newswire sentences for self-training
- Use 20%, 40%, 60%, and 80% of the labeled training data
- Always use the full unlabeled training set for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model

Syntactic Parsers w/ and w/o Self-Training (ST)



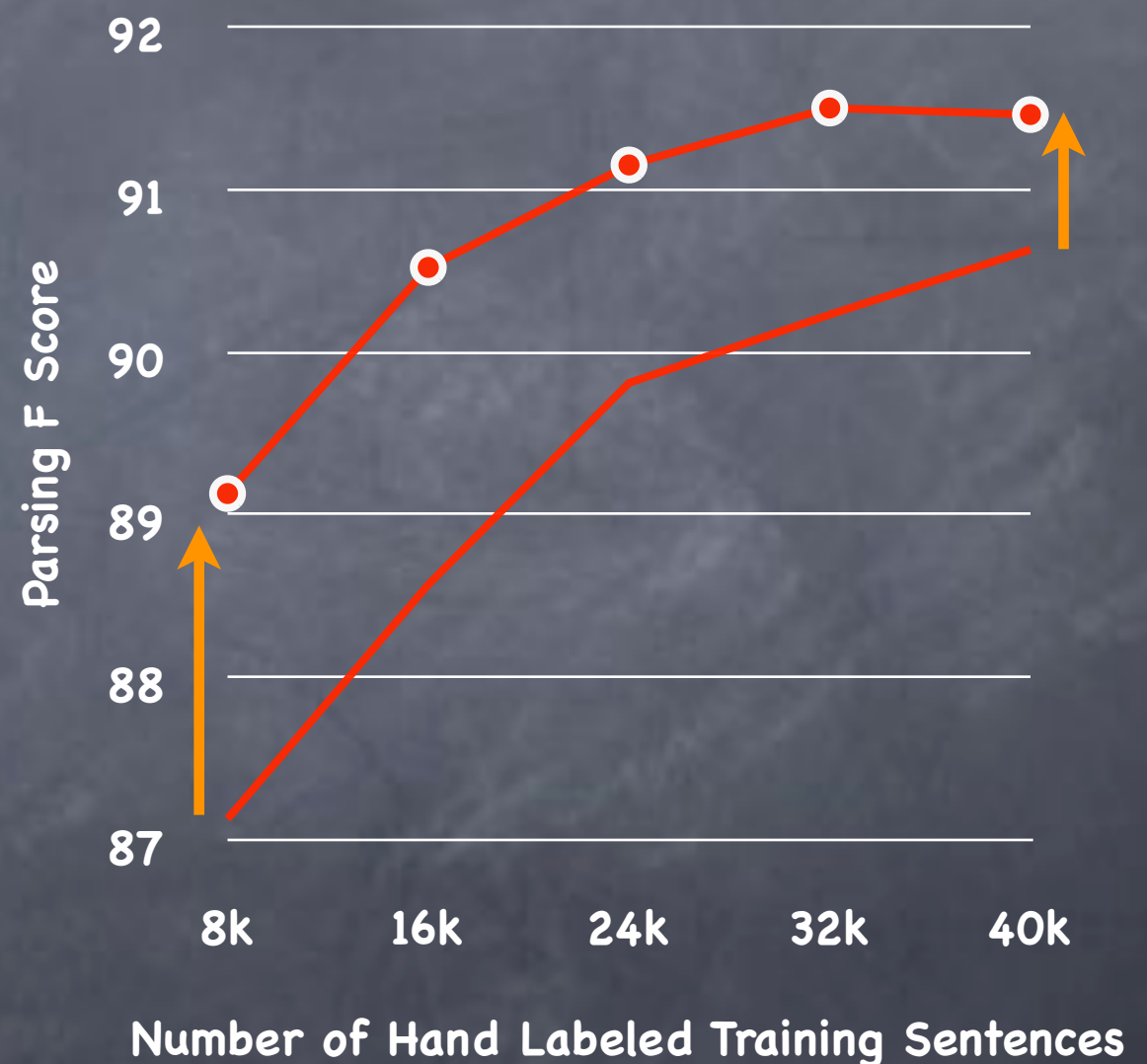
# Self-training English Syntactic Parsers

[Huang and Harper 2009]

## EXPERIMENT SETUP

- PCFG parser without latent variables: Charniak's lexicalized PCFG parser
- PCFG parser with latent variables: a modified Berkeley Parser
- Penn Treebank: sections 2-19 for training, 22 for dev, 23 for test.
- 210k unlabeled English newswire sentences for self-training
- Use 20%, 40%, 60%, and 80% of the labeled training data
- Always use the full unlabeled training set for self-training
- Combine the hand labeled and automatically unlabeled data in a weighted manner, so that each contributes 50% to the final model

Syntactic Parsers w/ and w/o Self-Training (ST)



— Latent PCFG ● Latent PCFG+ST

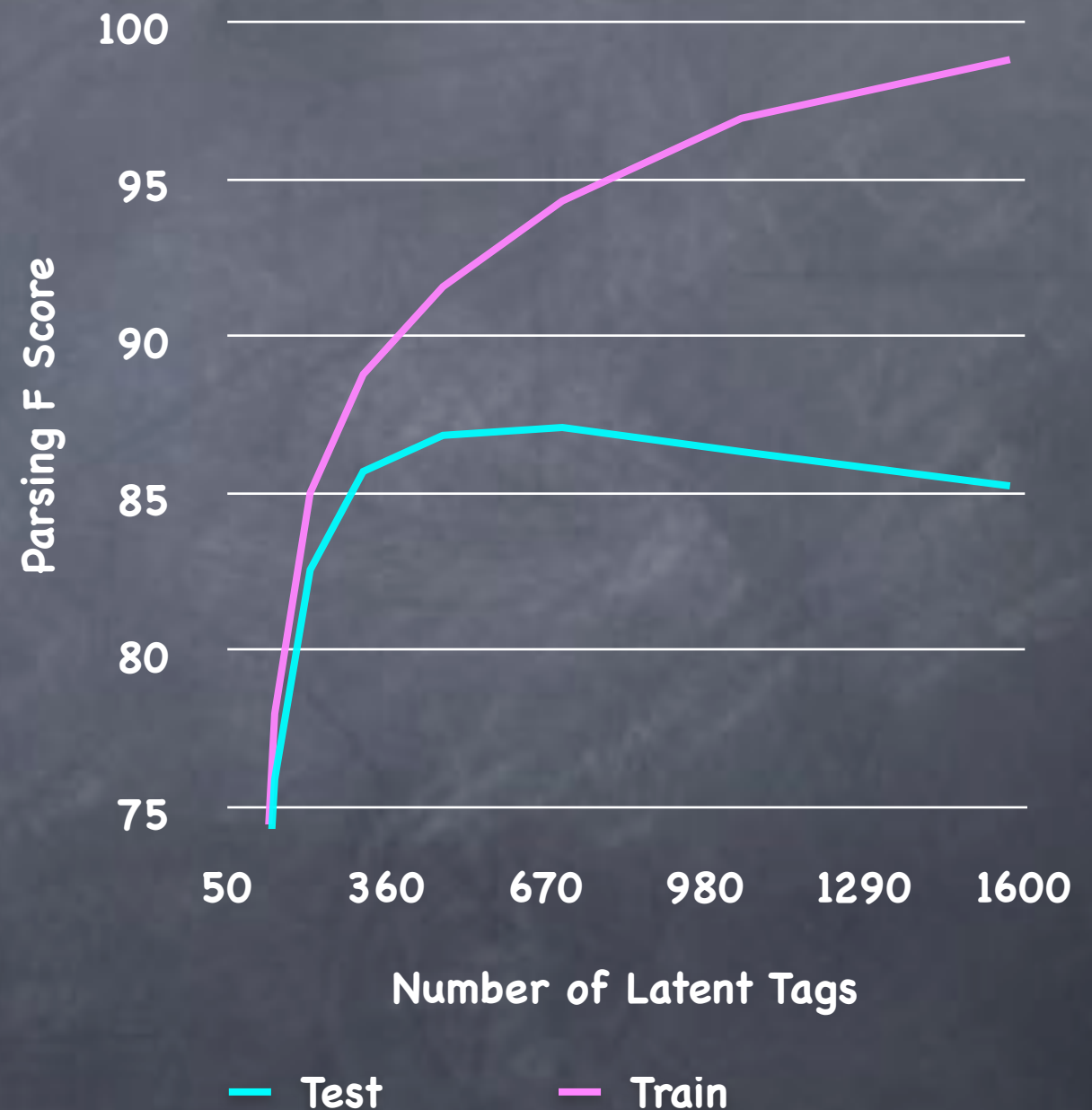
# Why?

- Conventional Models
  - Fixed parameterization
  - Limited benefit from more labeled training data, self-labeled data
- Models with latent variables
  - Flexible parameterization with varying granularities
  - Less data → fewer latent annotations
  - More data → more latent annotations
  - Benefit more from more labeled training data, even imperfect self-labeled data.

# Why?

- Conventional Models
  - Fixed parameterization
  - Limited benefit from more labeled training data, self-labeled data
- Models with latent variables
  - Flexible parameterization with varying granularities
  - Less data → fewer latent annotations
  - More data → more latent annotations
  - Benefit more from more labeled training data, even imperfect self-labeled data.

PCFG Parser with Latent Variables

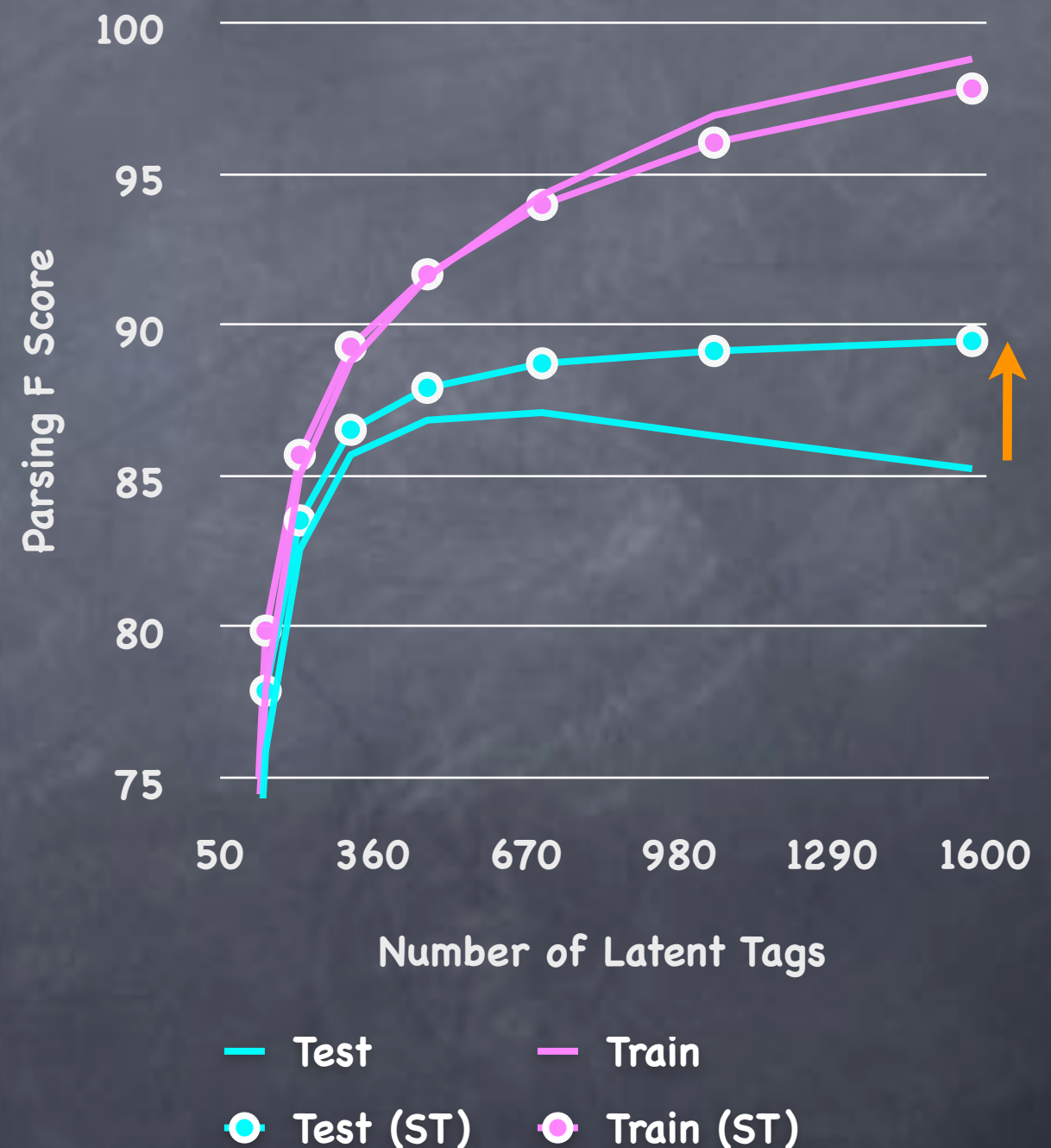




# Why?

- Conventional Models
  - Fixed parameterization
  - Limited benefit from more labeled training data, self-labeled data
- Models with latent variables
  - Flexible parameterization with varying granularities
  - Less data → fewer latent annotations
  - More data → more latent annotations
  - Benefit more from more labeled training data, even imperfect self-labeled data.

PCFG Parser with Latent Variables



# Future Work

- Theoretical understanding of self-training
- Introducing latent variables for other NLP tasks
- Scaling up models to use more unlabeled data
- Automatic data selection
- Domain adaptation