EMNLP Tutorial 2016

Methods and Theories for Large-scale Structured Prediction

Xu SUN

Peking University

xusun@pku.edu.cn

Yansong FENG

Peking University

fengyansong@pku.edu.cn

Content And Lecturer

Models & Regularization

- Conventional Model
- Latent Model
- Neural Model

Xu SUN

Structures & Applications

- Sequence Structure
- Tree/Graph Structure

Yansong FENG

Content And Lecturer

Models & Regularization

- Conventional Model
- Latent Model
- Neural Model

Xu SUN

Structures & Applications

- Sequence Structure
- Tree/Graph Structure

Yansong FENG

Models and Regularization for Large-scale Structured Prediction

Xu SUN

Peking University

xusun@pku.edu.cn

Outline

Introduction

Model

Conventional Model

Latent Model

Neural Model

Regularization

How-tos

Outline

Introduction

Model

Conventional Model

Latent Model

Neural Model

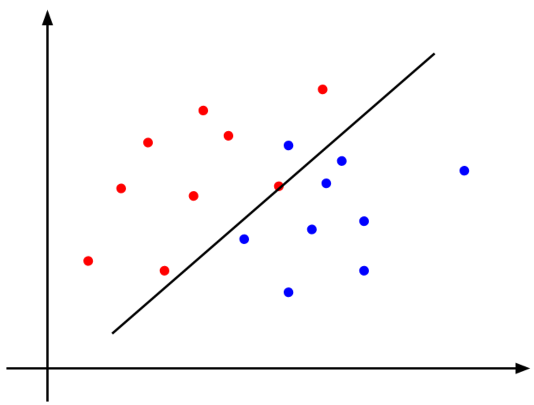
Regularization

How-tos

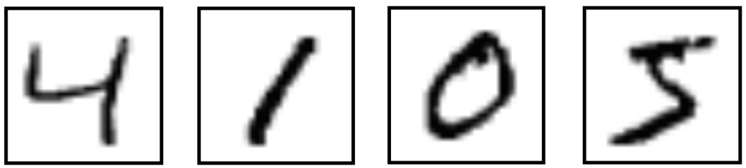
What Is Structured Prediction/Classification?

- **Types of Classification**

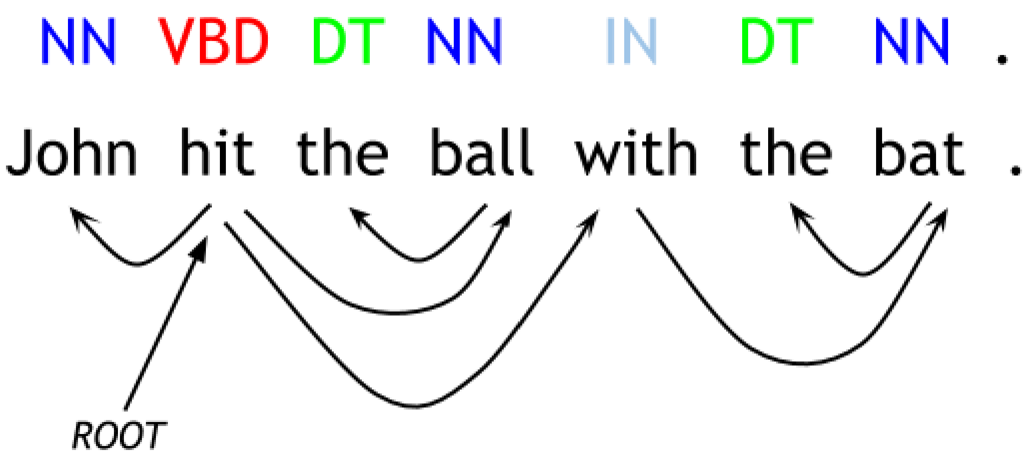
Binary Classification



Multiclass Classification



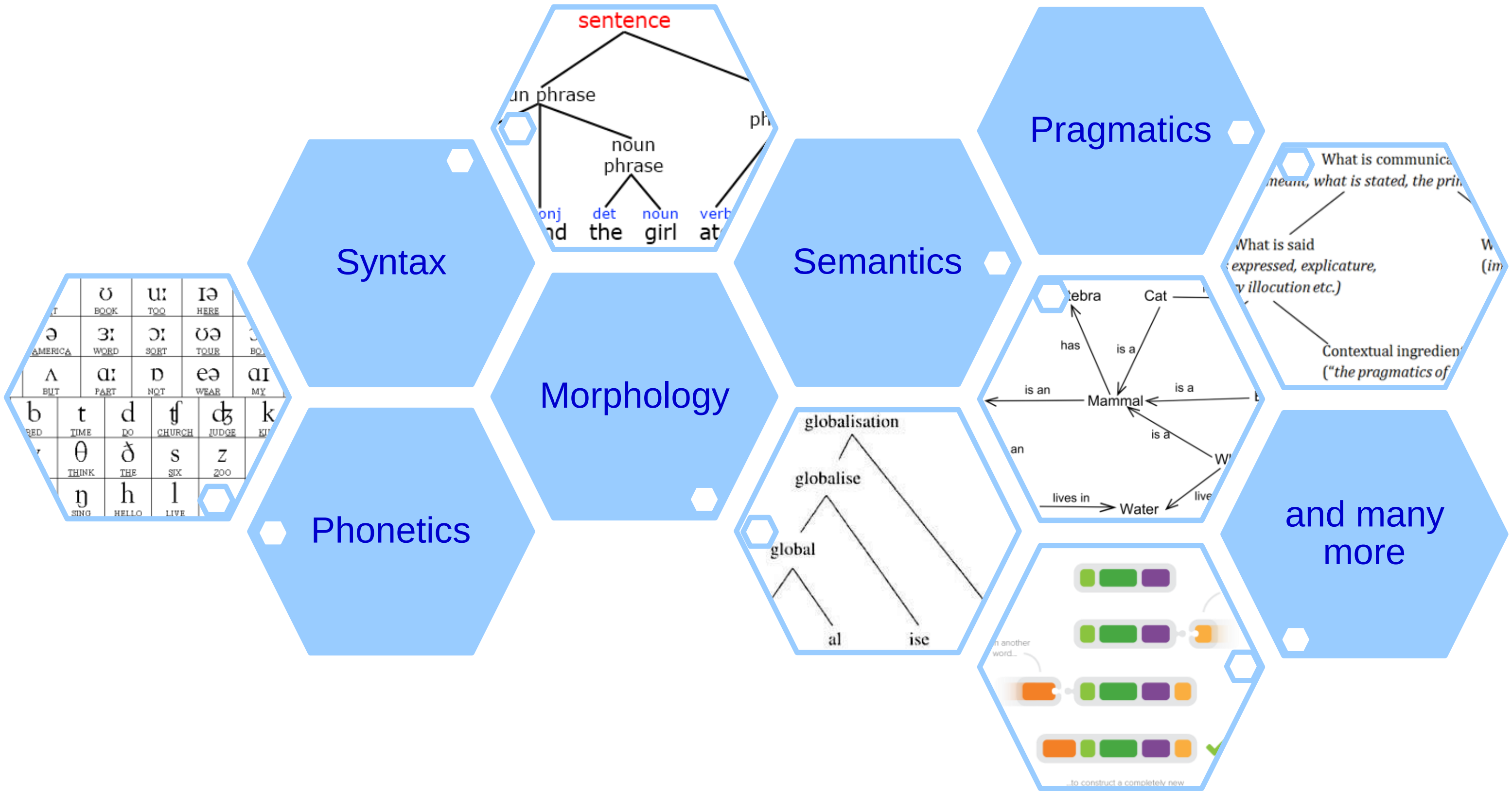
Structured Classification



Why Structured Prediction?

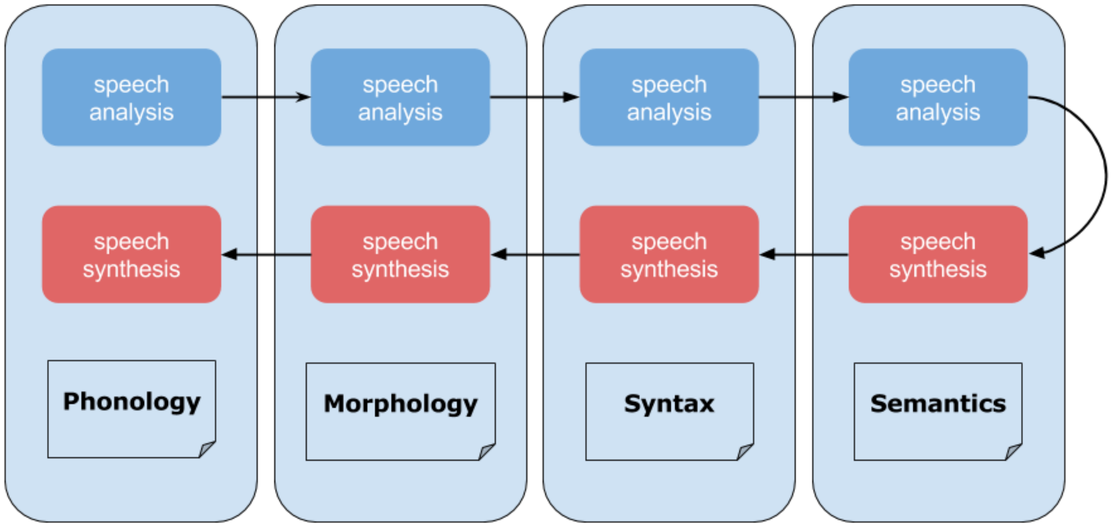
- **Structures are important in natural language processing**

  Linguists also attempt to understand the rules regarding language structures



Why Structured Prediction?

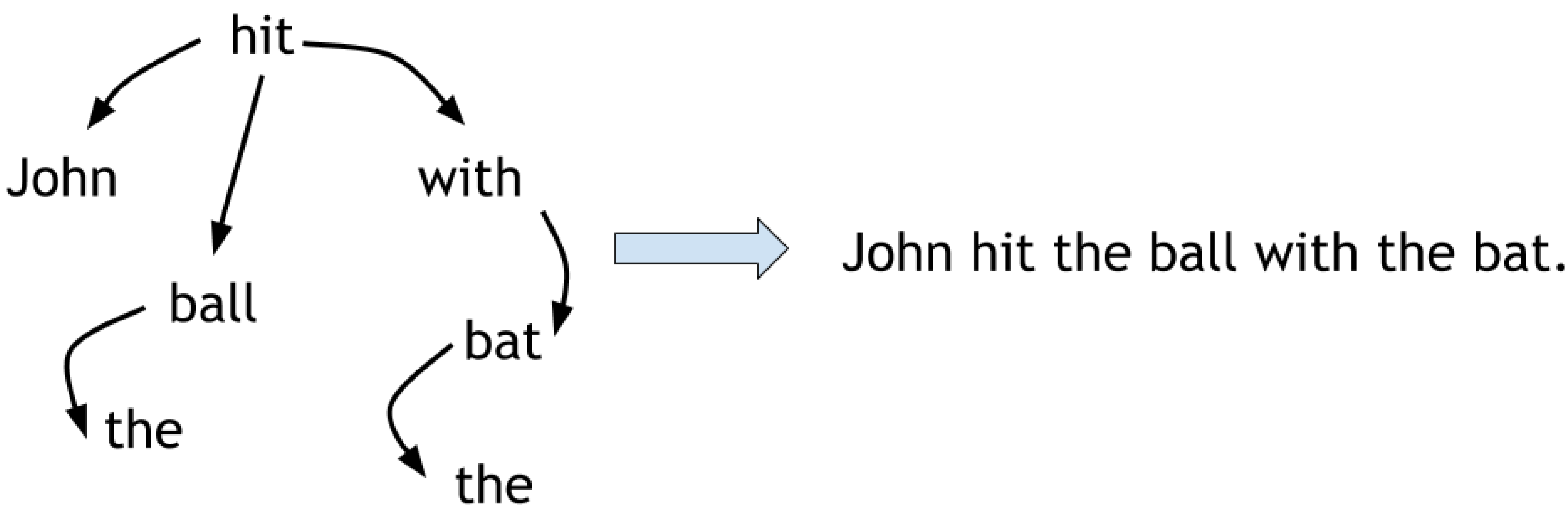
- Challenges in NLP involve Understanding and Generation
- Understanding the structures in natural languages is an essential step towards the goal



M Bates. Models of natural language understanding.
D Jurafsky and JH Martin. Speech and language processing.
CD Manning and H Schütze. Foundations of Statistical Natural Language Processing.

Why Structured Prediction?

- However, most of the time, structure prediction is **not straight-forward**

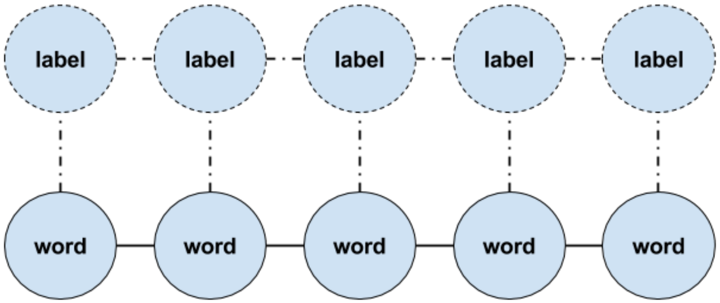


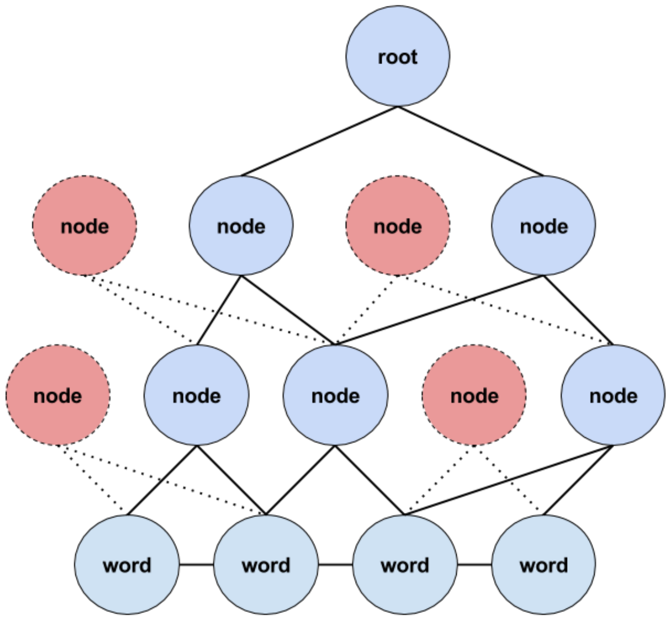
Why it is hard? --Natural languages encode all of the structures in a linear form

N Chomsky. Language and mind.
N Chomsky. Reflections on language.
MAK Halliday. Language structure and language function.
N Chomsky. Knowledge of language: Its nature, origin, and use.
D Biber, S Conrad, R Reppen. Corpus linguistics: Investigating language structure and use.

Why Structured Prediction?

- **Structured prediction helps to recover the structures in natural languages**





POS Tagging (Collins, ACL 2002; Gimenez & Marquez, LREC 2004.; Shen et al., ACL 2007; Søegaard, ACL-HLT 2011; Sun, NIPS 2014; Collobert et al., JMLR 2011; Huang et al., 2015)

Chunking (Kudo & Matsumoto, NAACL 2001; Collins, ACL 2002; McDonald et al., HLT-EMNLP 2005; Sun et al., COLING 2008; Collobert et al, JMLR 2011; Huang et al., 2015)

NER (Florian et al., HLT-NAACL 2003; Chieu, CoNLL 2003; Ando & Zhang, JMLR 2005; Collobert et al., JMLR 2011; Passos et al., CoNLL 2014; Huang et al., 2015)

Parsing (Earley, 1970; Collins, EACL 1997; Klein & Manning, ACL 2003; Sha & Pereira, HLT-NAACL 2003; Nivre, IWPT 2003; Collins, ACL 2004; Nivre & Scholz, COLING 2004; McDonald, HLT-EMNLP 2005; Zhang & Clark, EMNLP 2008; Zhang & Nivre, ACL-HLT 2011;Socher, et at., EMNLP 2013; Chen & Manning, EMNLP 2014)

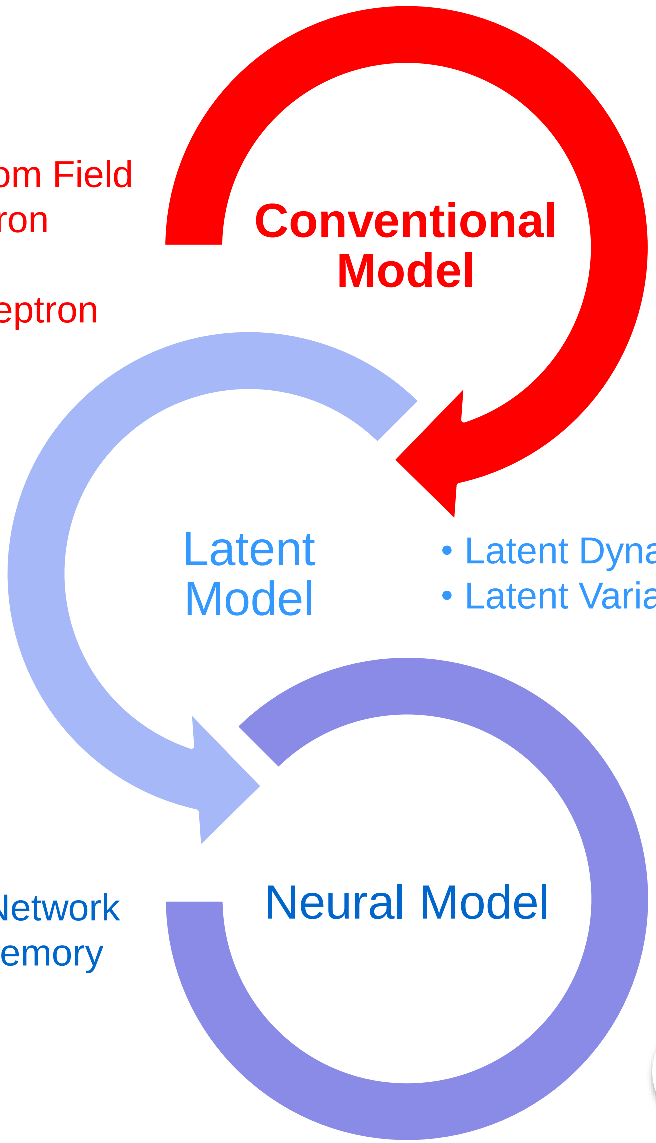
Word Segmentation, Summarization, Machine Translation...

Outline

- Introduction
- Model
  - Conventional Model
  - Latent Model
  - Neural Model
- Regularization
- How-tos

Outline

- Conditional Random Field
- Structure Perceptron
- MIRA
- Probabilistic Perceptron

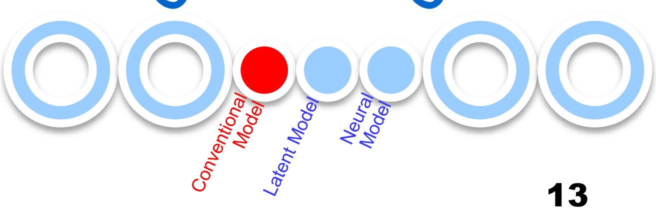


- Latent Dynamic CRF
- Latent Variable Perceptron

Reduce Annotation Engineering

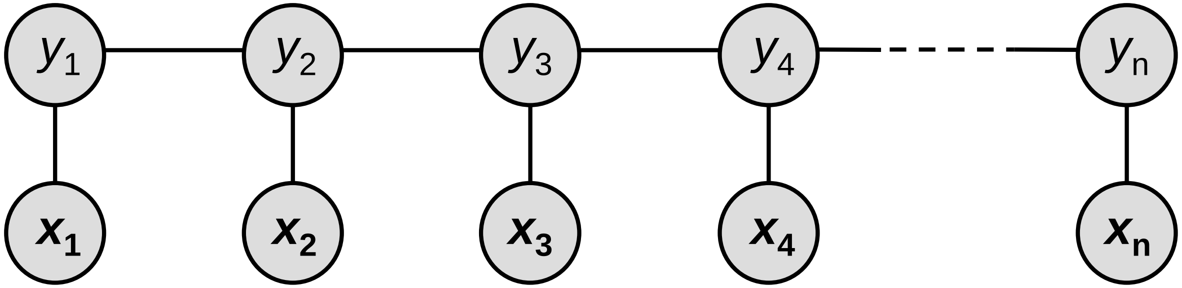
- Recurrent Neural Network
- Long Short-term Memory
- LSTM-CRF

Reduce Feature Engineering



Conditional Random Fields (CRFs)

- **Proposed by** Lafferty et al. (2001)
- **Maximize a conditional probability**
  - $p(y|x,\theta) = \frac{1}{z(x,\theta)} \exp(\sum_k \theta_k f_k(y,x))$
  - $Z(x,\theta) = \sum_{y'} \exp(\sum_k \theta_k f_k(y',x))$
- **Global model**
  - Predict global structure, not local classifier
  - Training: globally normalized objective
  - Decode: Viterbi algorithm



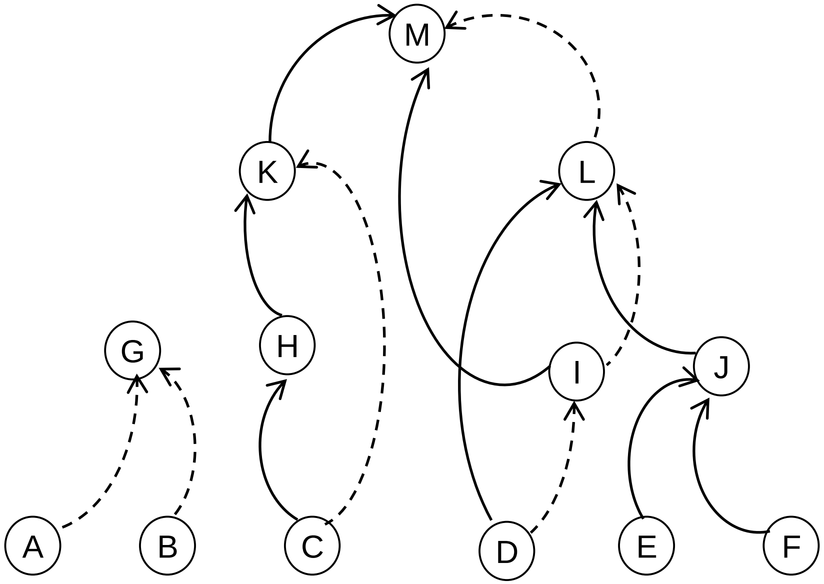
Conditional Random Fields (CRFs)

- **Proposed by** Lafferty et al. (2001)
- **Maximize a conditional probability**
  - $p(y|x,\theta) = \frac{1}{z(x,\theta)} \exp(\sum_k \theta_k f_k(y,x))$
  - $Z(x,\theta) = \sum_{y'} \exp(\sum_k \theta_k f_k(y',x))$
- **Global model**
  - Predict global structure, not local classifier
  - Training: globally normalized objective
  - Decode: Viterbi algorithm

But the training speed is quite slow…

Structured Perceptron

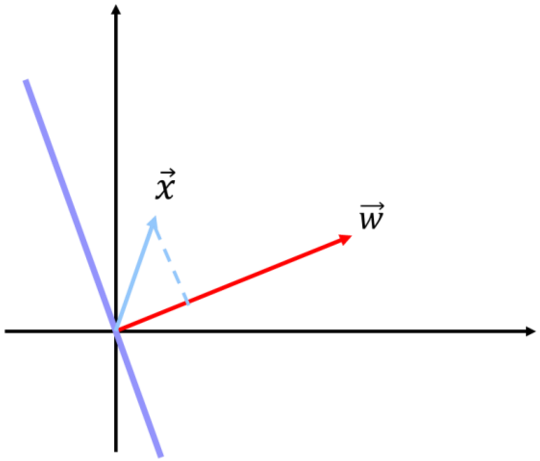
- **Proposed by** Collins (2002)
- **Simple and fast**
  - No gradient computation
  - Update only on error
  - Viterbi decode
- **Theoretical guarantee**
  - Converge if data is separable



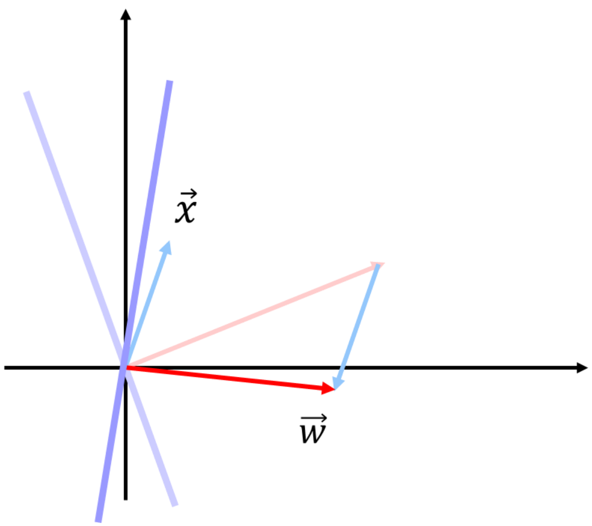
input: Training Examples $\{(x_i, y_i)\}_{i=1}^{n}$
initialize: $\alpha = 0$
repeat
    Get a random example (x_i, y_i)
    $y^* = argmax_{z \in GEN(x)} \Phi(x, y) \cdot \alpha$
    if $(y^* \neq y)$ then $\alpha = \alpha + \Phi(x, y) - \Phi(x, y^*)$
until Convergence
return parameter α

Structured Perceptron

- **Proposed by** Collins (2002)
- **Simple and fast**
  - No gradient computation
  - Update only on error
  - Viterbi decode
- **Theoretical guarantee**
  - Converge if data is separable



input: Training Examples $\{(x_i, y_i)\}_{i=1}^{n}$
initialize: $\alpha = 0$
repeat
    Get a random example (x_i, y_i)
    $y^* = argmax_{z \in GEN(x)} \Phi(x, y) \cdot \alpha$
    if $(y^* \neq y)$ then $\alpha = \alpha + \Phi(x, y) - \Phi(x, y^*)$
until Convergence
return parameter α



Structured Perceptron

- **Inexact search**
  - Greedy search
  - Beam search
- **Parameter update**
  - Early update (Collins & Roark, ACL 2004)
  - Margin Infused Relaxed Algorithm (MIRA) (Crammer et al. 2006)
  - Max-violation (Huang et al., NAACL 2012)

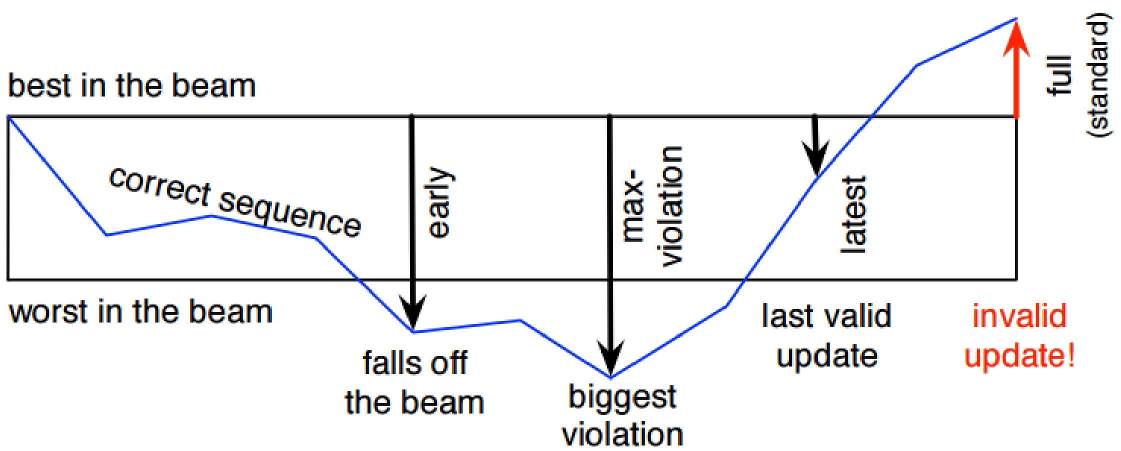
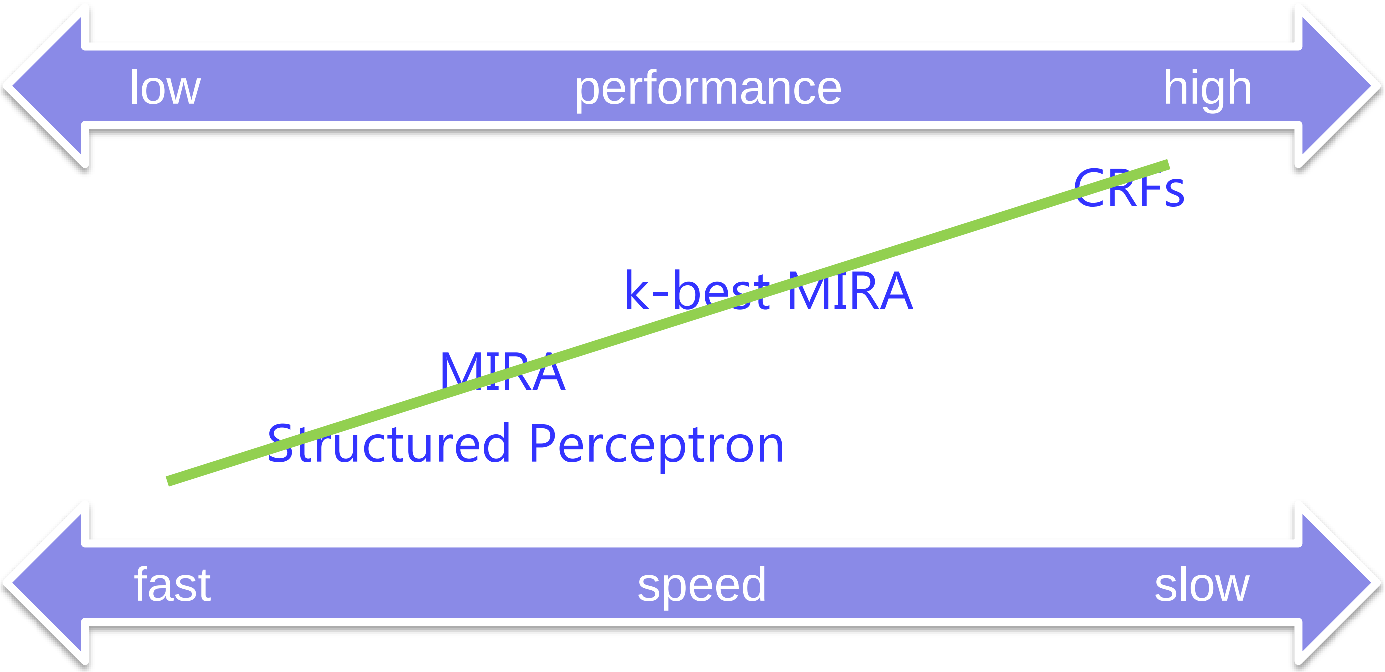


Figure from Huang et al. (2012)

For Large-scale Structured Prediction

- **Current structured prediction methods are not ideal**
  - A trade-off between accuracy and speed...



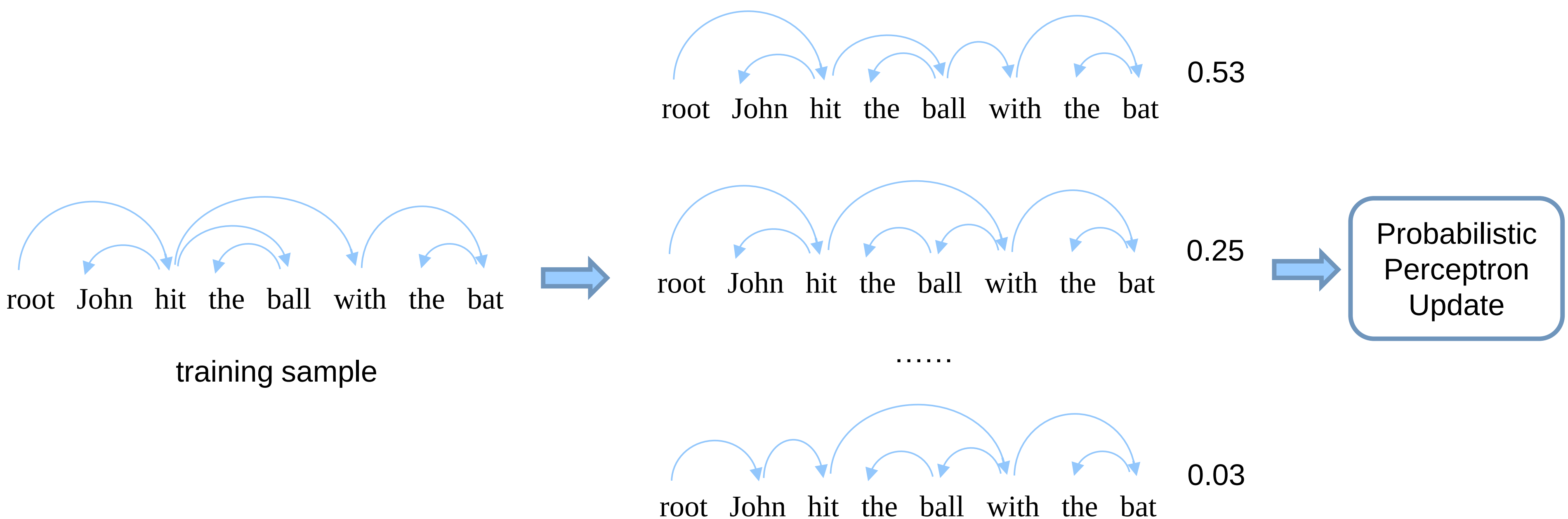
For Large-scale Structured Prediction

- **A solution works well in practice for large-scale structured prediction problems**
  - Introducing probabilistic information into perceptrons
  - This goes to Probabilistic Perceptron (SAPO) (Sun 2015)

X. Sun. Towards shockingly easy structured classification: A search-based probabilistic online learning framework. 2015

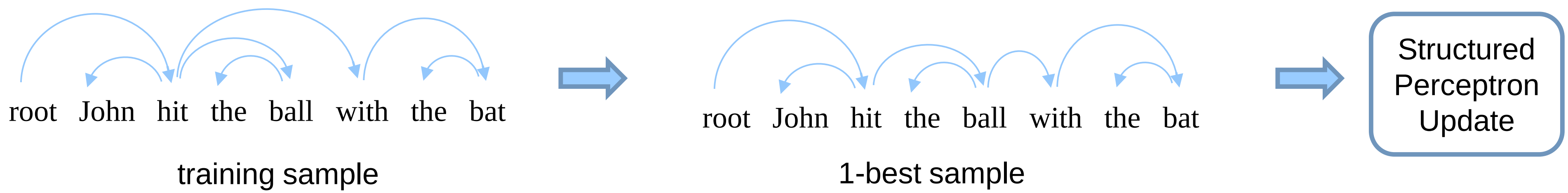
Probabilistic Perceptron (SAPO)

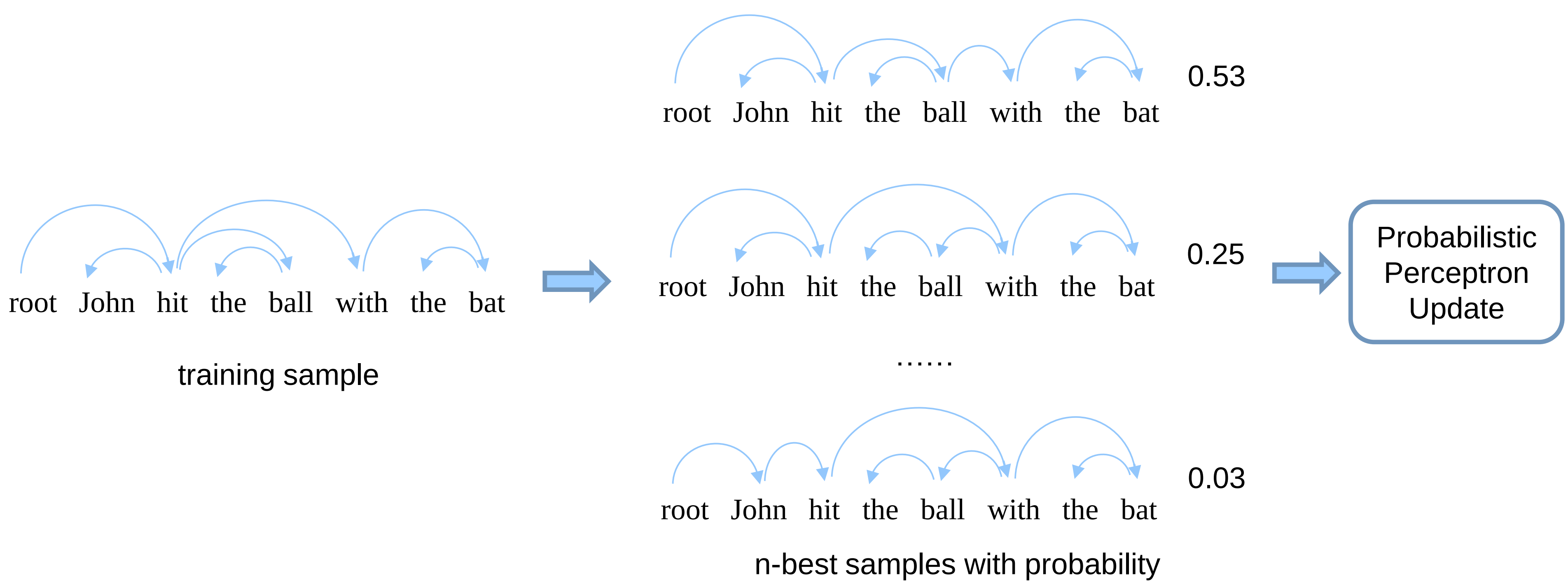
- **Proposed by** Sun (2015)
- **Same (or even higher) accuracy like CRF**
- **Fast training speed like perceptron**



X. Sun. Towards shockingly easy structured classification: A search-based probabilistic online learning framework. 2015

- **Proposed by** Sun (2015)





Probabilistic Perceptron (SAPO)

Theoretical guarantee of convergence

Theorem 1 (Optimum, convergence, and rate) *With the conditions (16), (17), (18), (19), let $\epsilon > 0$ be a target degree of convergence. Let τ be an approximation-based bound from $\boldsymbol{s}(\boldsymbol{w})$ to $\nabla f(\boldsymbol{w})$ such that*

Probabilistic perceptron converges! (20)

where $\boldsymbol{w}$ is a historical weight vector that updated during SAPO training, and $\boldsymbol{s}(\boldsymbol{w})$ is expected $\boldsymbol{s}_{\boldsymbol{z}}(\boldsymbol{w})$ over $\boldsymbol{z}$ such that $\boldsymbol{s}(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{s}_{\boldsymbol{z}}(\boldsymbol{w})]$. Since $\boldsymbol{s}(\boldsymbol{w})$ can be arbitrary-close to $\nabla f(\boldsymbol{w})$ by increasing n, SAPO can use the smallest n as far as the following holds:

$$\tau \leq \frac{c\epsilon}{2q} \tag{21}$$

Let γ be a learning rate as

$$\gamma = \frac{c\epsilon - 2\tau q}{\beta q \kappa^2} \tag{22}$$

where we can set β as any value as far as $\beta \geq 1$. Let t be the smallest integer satisfying

$$t \geq \frac{\beta q \kappa^2 \log(q a_0/\epsilon)}{c(c\epsilon - 2\tau q)} \tag{23}$$

where a_0 is the initial distance such that $a_0 = ||\boldsymbol{w}_0 - \boldsymbol{w}^*||^2$. Then, after t updates of $\boldsymbol{w}$, SAPO converges towards the optimum such that

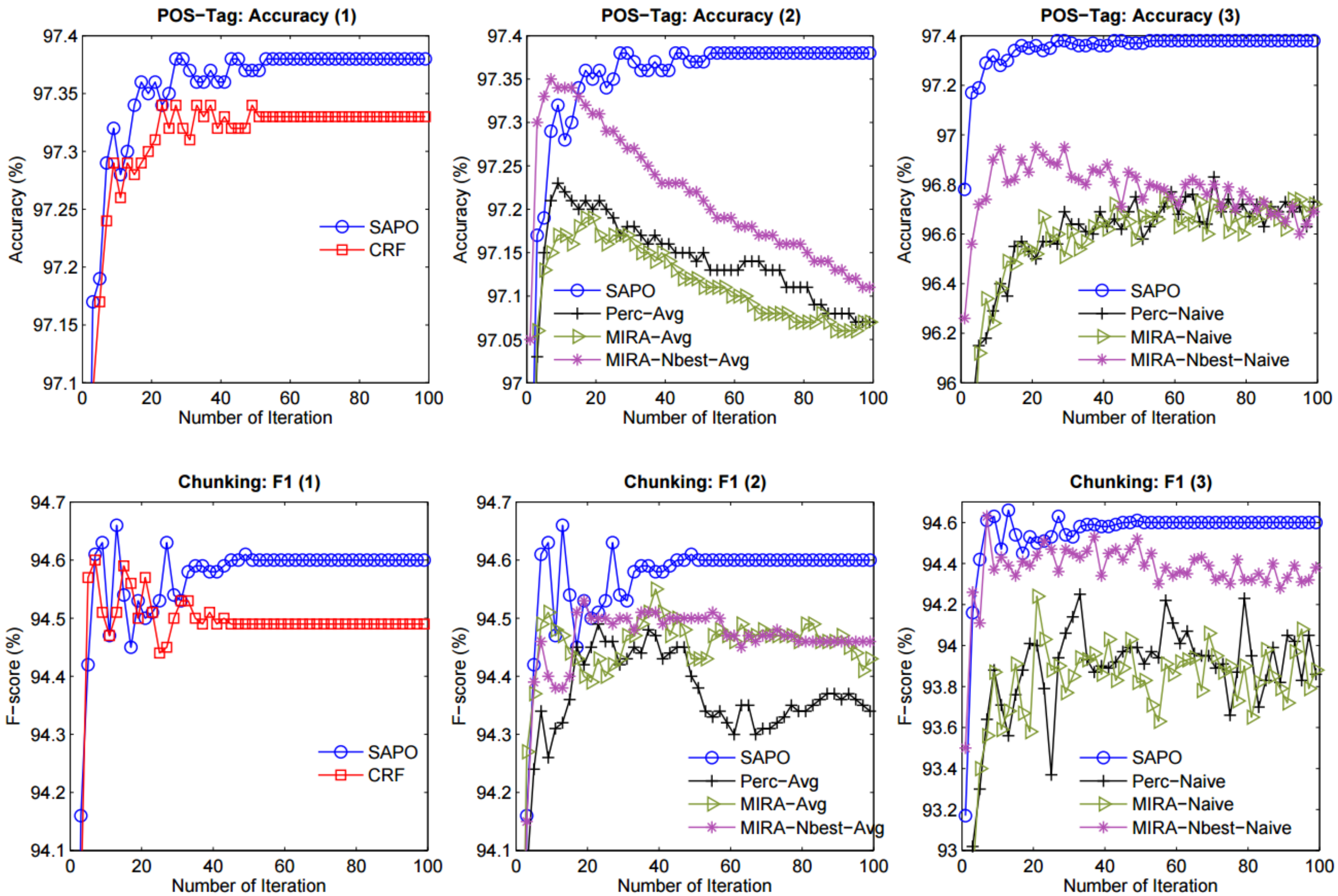
$$\boxed{\mathbb{E}[f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)] \leq \epsilon} \tag{24}$$

Probabilistic Perceptron (SAPO)

Experiment results

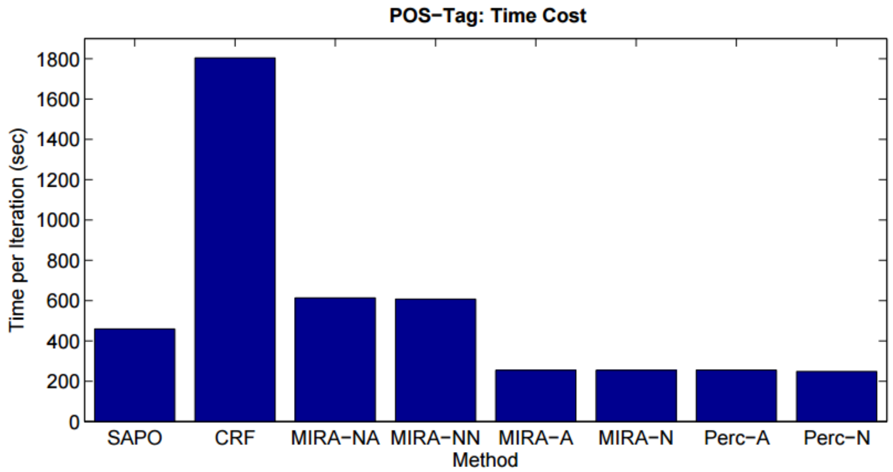
It indicates the number of samples is not the larger the better, why?

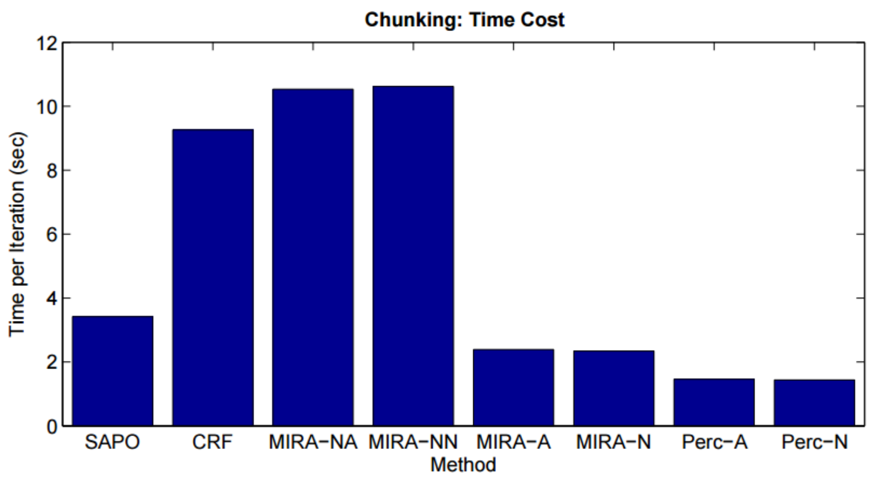
- Similar or even higher accuracy compared with CRFs, perceptron and MIRA



Probabilistic Perceptron (SAPO)

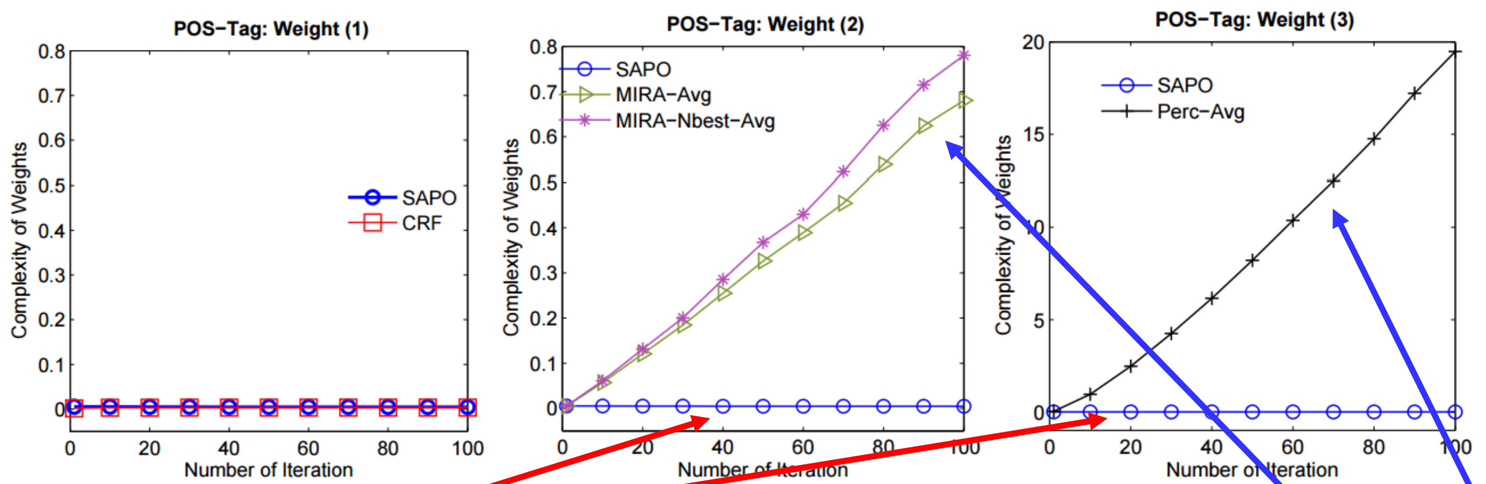
- **Experiment results**
  - Much faster than CRFs
  - Nearly as fast as perceptrons



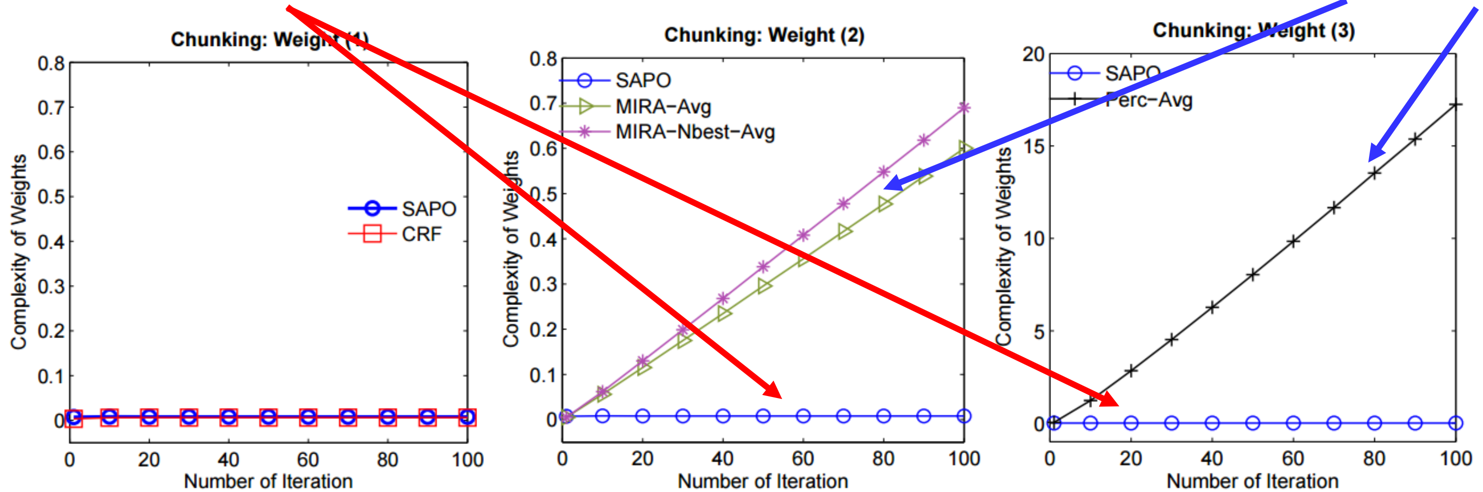


Probabilistic Perceptron (SAPO)

Experiment result

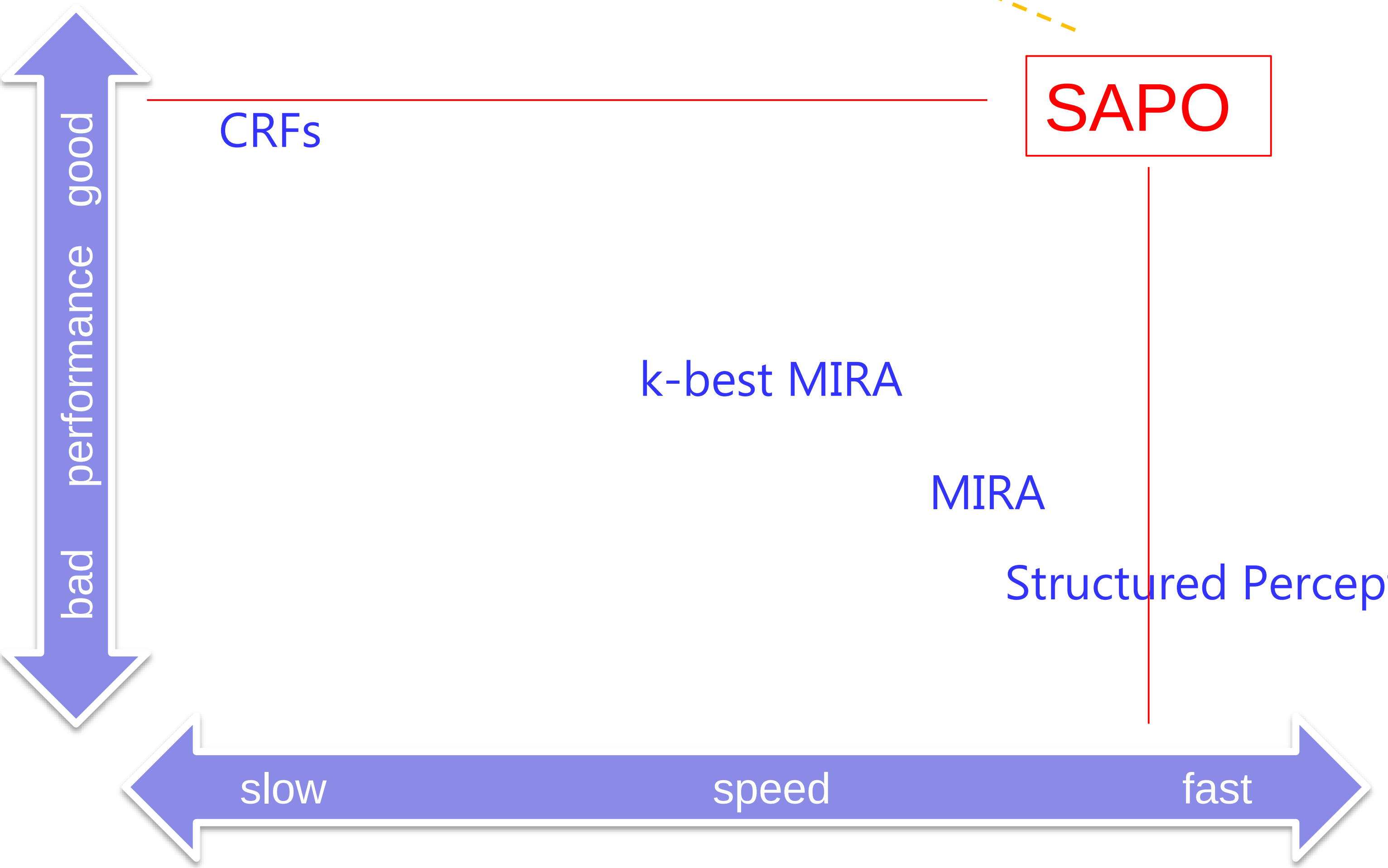


No rising complexity of weights compared to MIRA or Perc.



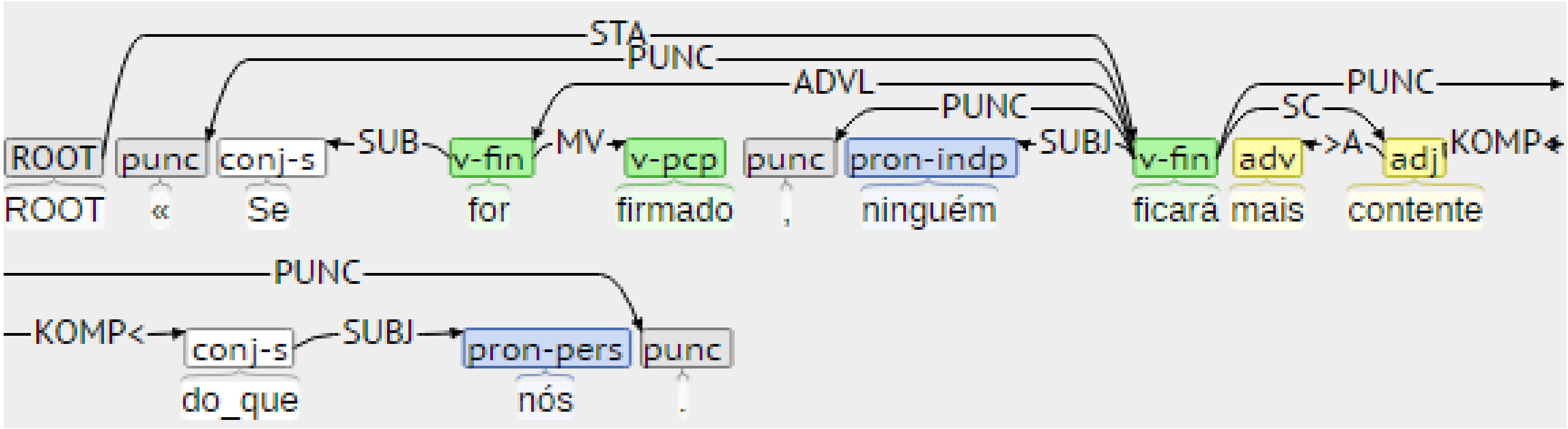
For Large-scale Structured Prediction

- **Probabilistic Perceptron** Sun (2015)



Outline

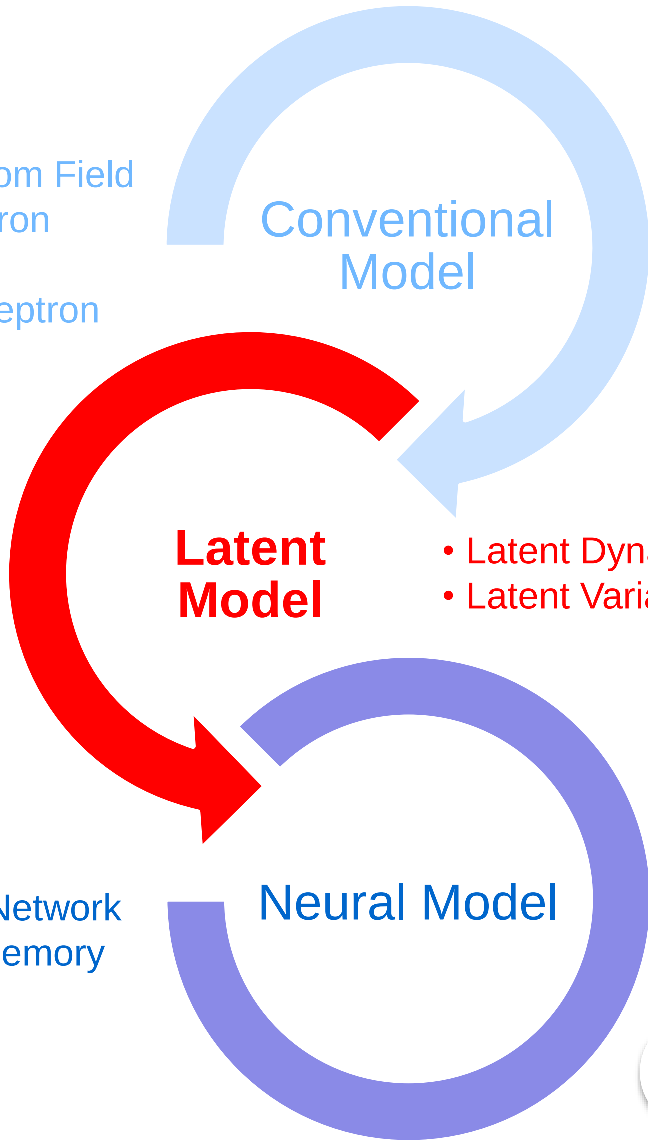
- **Typical methods need large-scale annotations**
- **Problem in reality**
  - Lack of annotations
  - Inaccurate annotations
- **Latent Model**
  - Reduce Annotation Engineering



Outline

- Conditional Random Field
- Structure Perceptron
- MIRA
- Probabilistic Perceptron

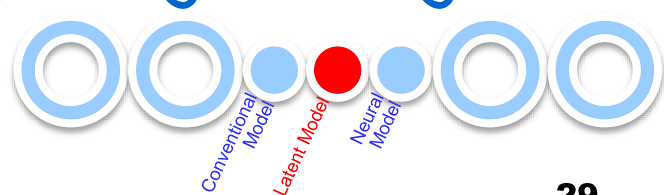
Reduce Annotation Engineering



- Latent Dynamic CRF
- Latent Variable Perceptron

- Recurrent Neural Network
- Long Short-term Memory
- LSTM-CRF

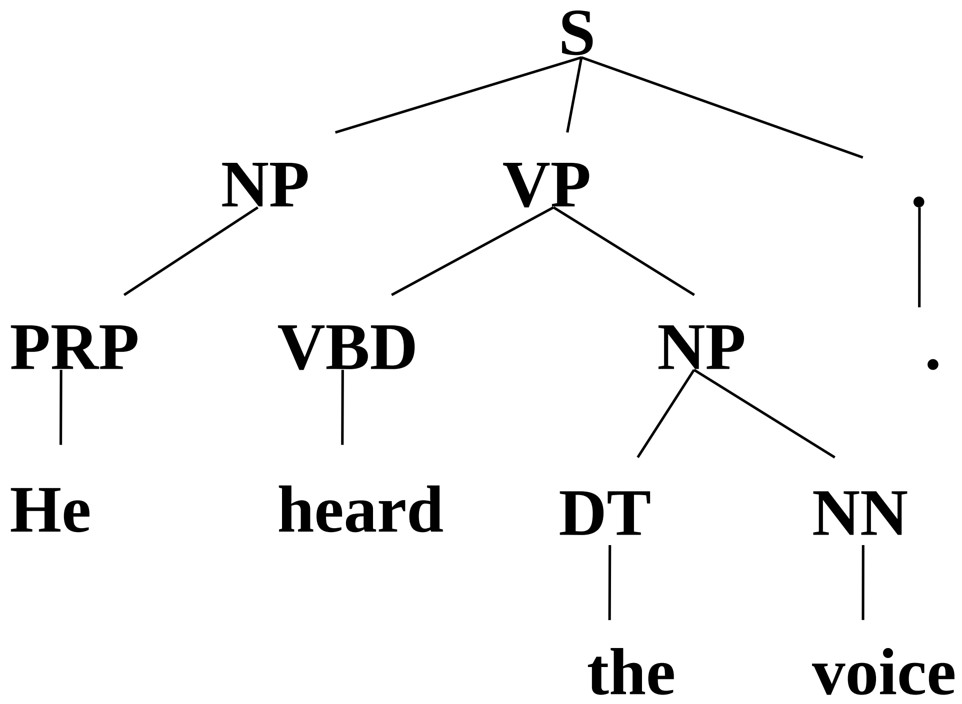
Reduce Feature Engineering



Motivation

- **Latent-structures (hidden info) are important in natural language processing** (Matsuzaki et al., ACL 2005; Petrov & Klein, NIPS 2008)

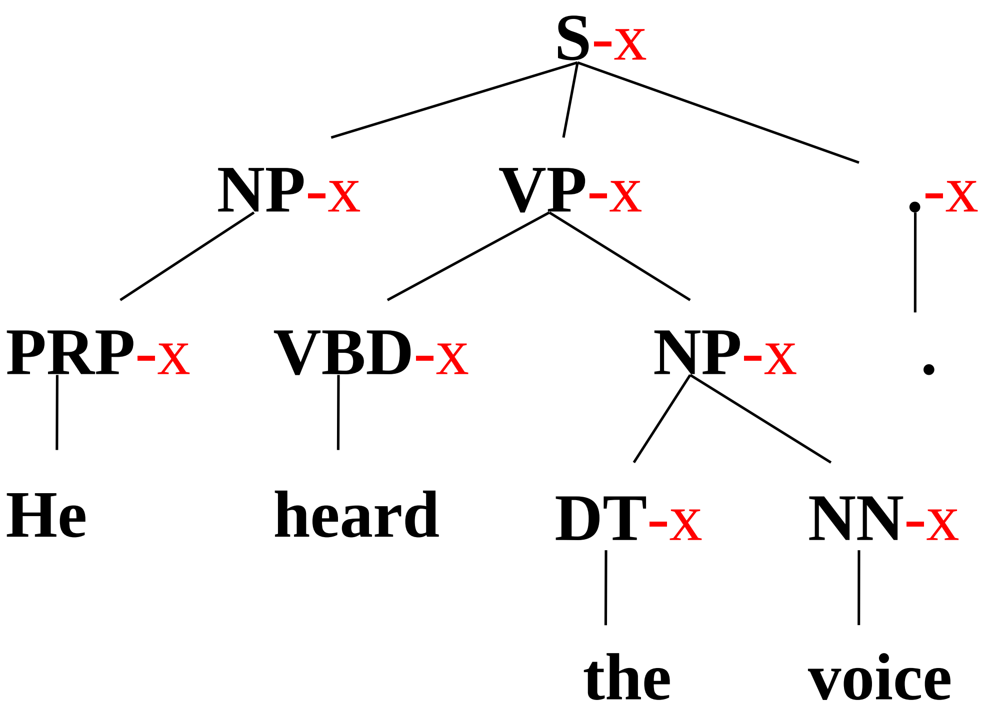
Parsing: Learn refined grammars with latent info



Motivation

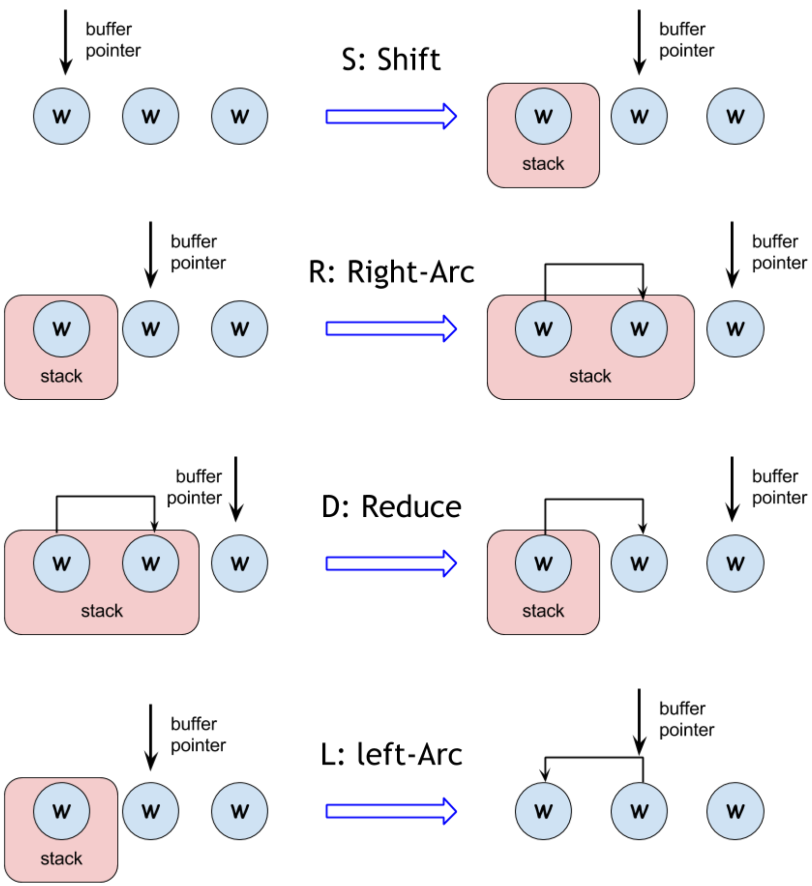
- **Latent-structures (hidden info) are important in natural language processing** (Matsuzaki et al., ACL 2005; Petrov & Klein, NIPS 2008)

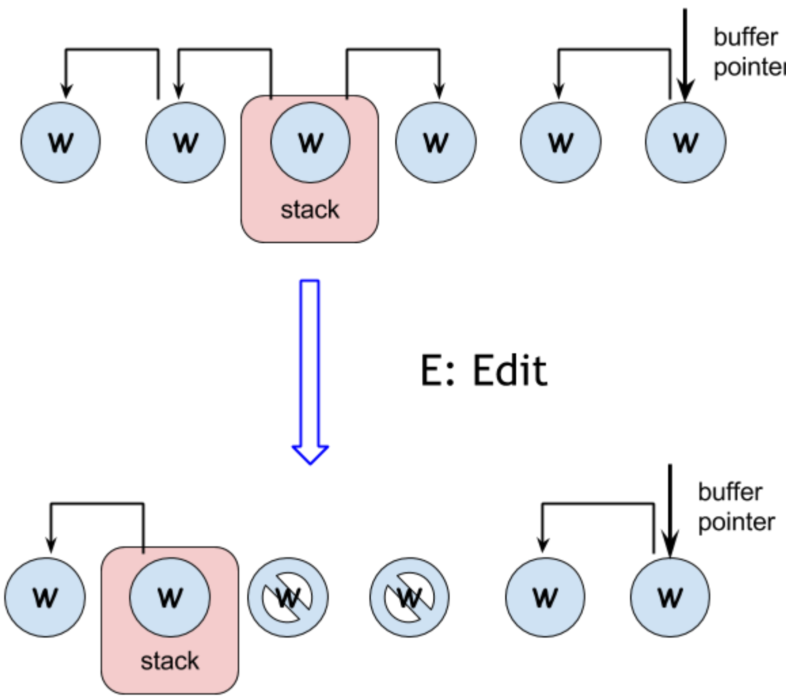
Parsing: Learn refined grammars with latent info



Motivation

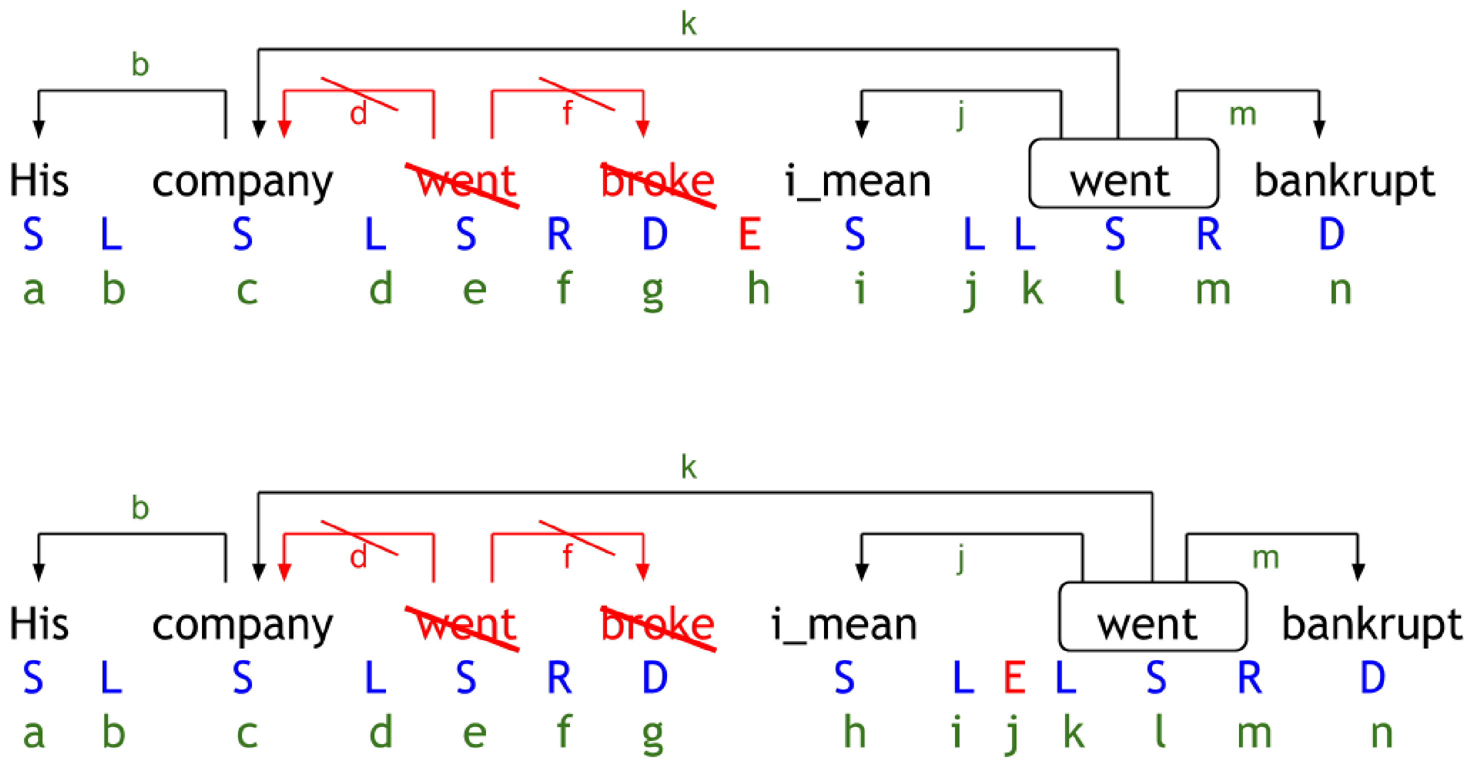
- **Latent-structures (hidden info) are important in dependency parsing** (Honnibal & Johnson, TACL 2014)
  - A transition system (arc-eager) extended with an EDIT transition





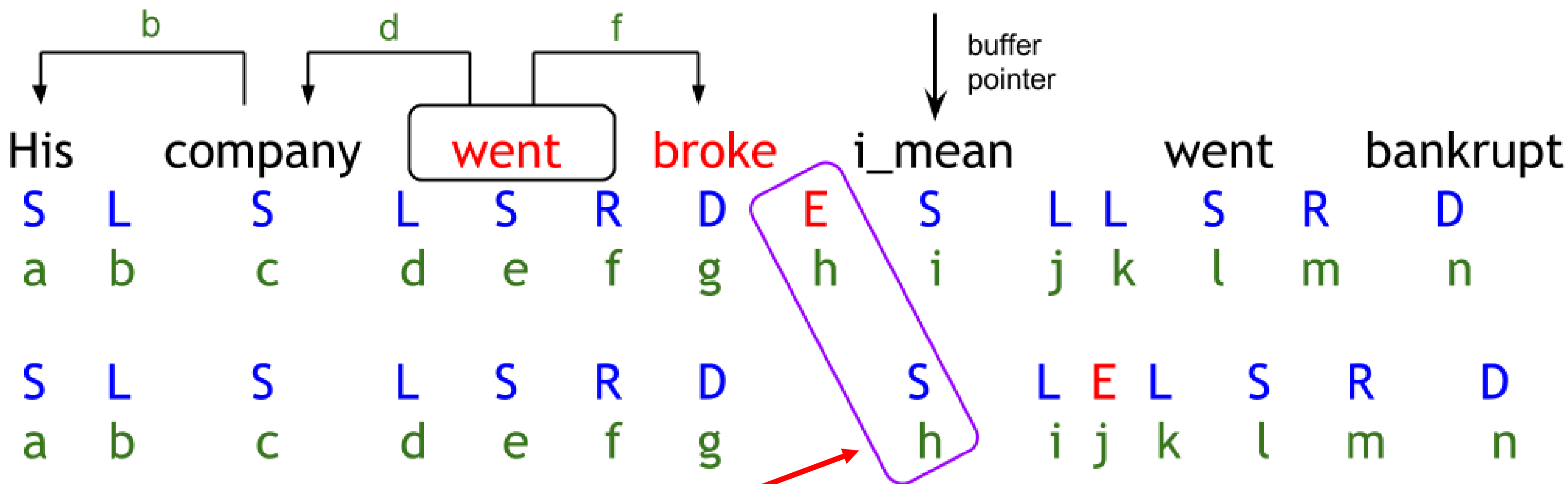
Motivation

- **Latent-structures (hidden info) are important in dependency parsing** (Honnibal & Johnson, TACL 2014)
  - Different transition sequences to the same gold-standard tree



Motivation

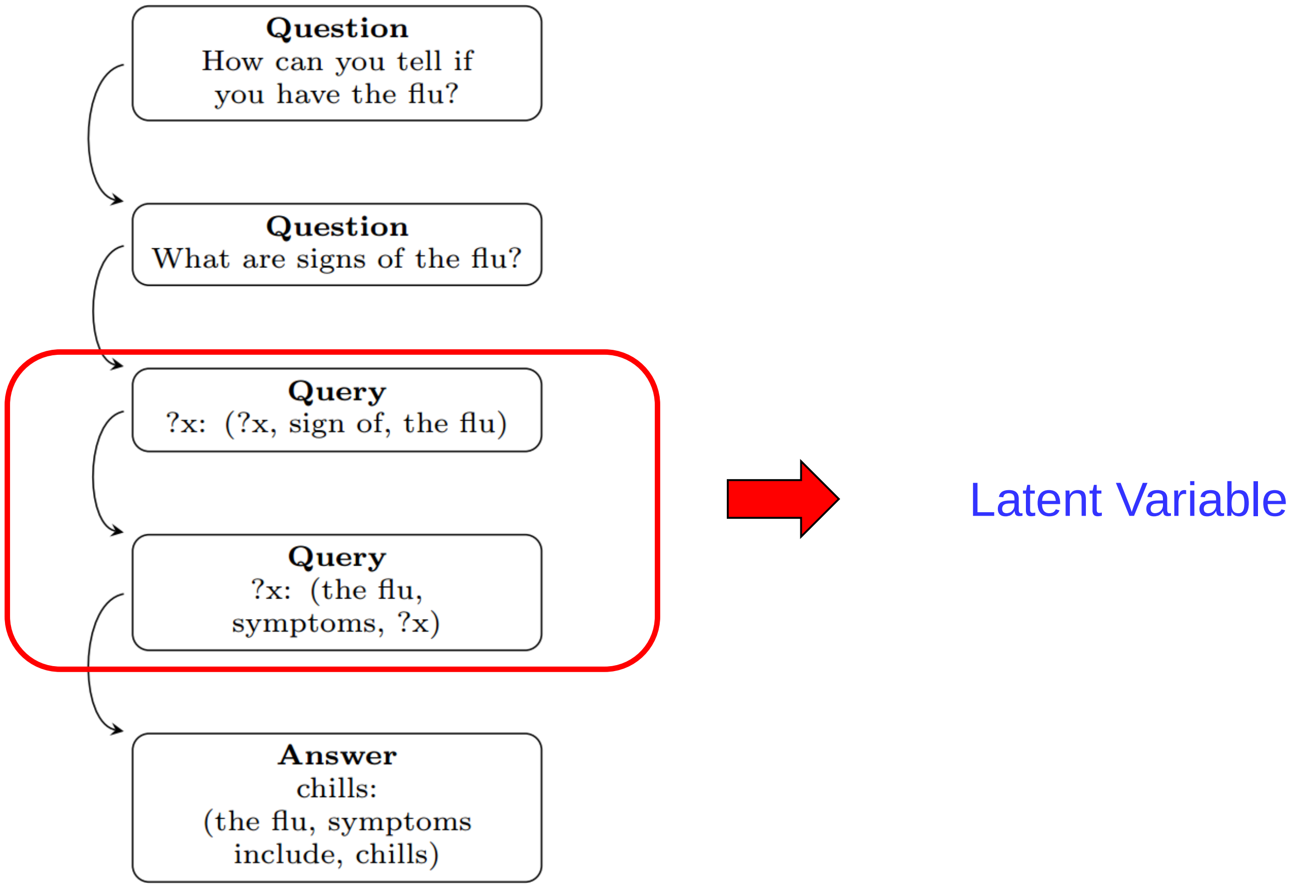
- **Latent-structures (hidden info) are important in dependency parsing** (Honnibal & Johnson, TACL 2014)



dynamic oracle
map a configuration to a set of transitions
partially annotated
latent variable

Motivation

- **Latent-structures** **(hidden info) are important in** **question answering** (Fader et al., SIGKDD 2014)



Motivation

- **Latent-structures (hidden info) are important in multi-intent speech recognition** (Xu & Sarikaya, INTERSPEECH 2013)

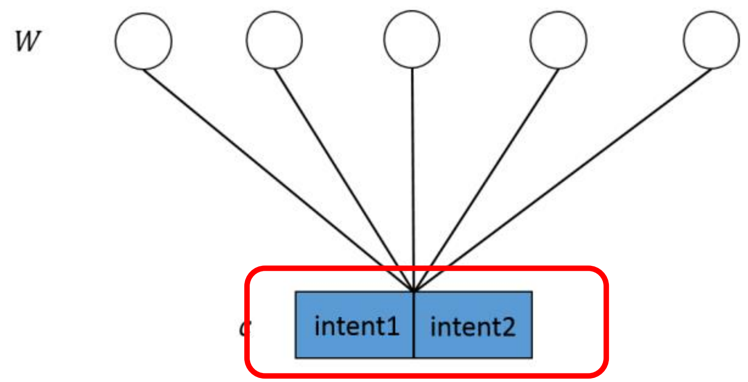


Figure 1: *Graphical model representation of the baseline log-linear classification model.*

Latent Variable

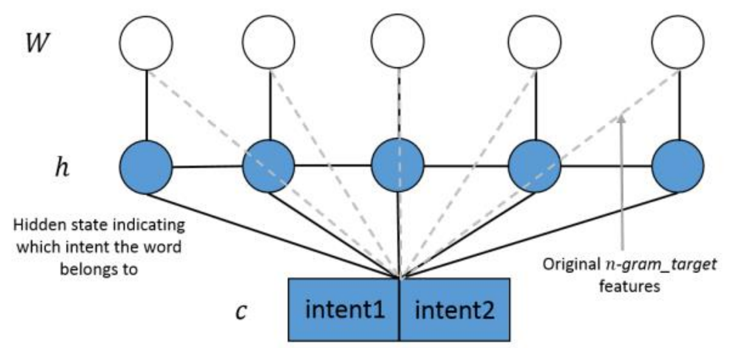


Figure 2: *Graphical model representation of the hidden variable model.*

Challenges

- **Latent-structures (hidden info) are important in multi-intent speech recognition** (Xu & Sarikaya, INTERSPEECH 2013)

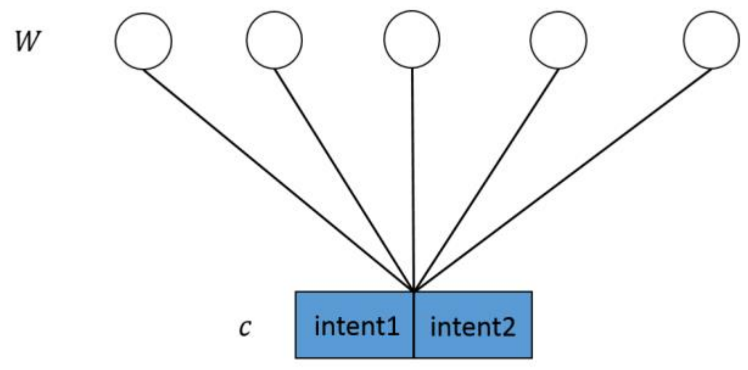


Figure 1: *Graphical model representation of the baseline log-linear classification model.*

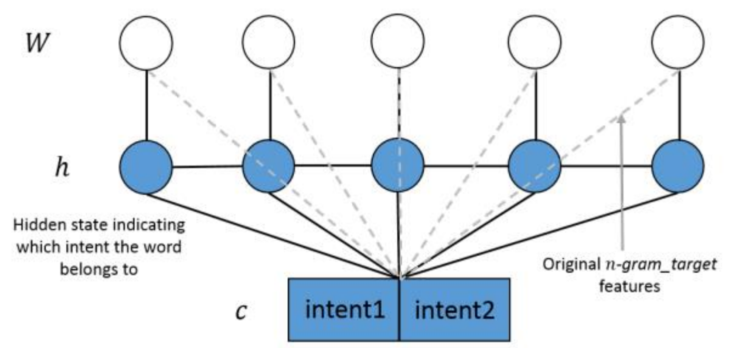


Figure 2: *Graphical model representation of the hidden variable model.*

Problem 1:
Annotating latent info requires much more tags and human time
→ Costly to annotate

Problem 2:
Different tasks have different latent info.
→ Hard to annotate

Latent-dynamic CRFs

- **A solution without additional annotation**

Latent-dynamic CRFs (LDCRF)

[Morency et al., CVPR 2007; Sun et al., COLING 2008]

* No need to annotate latent info

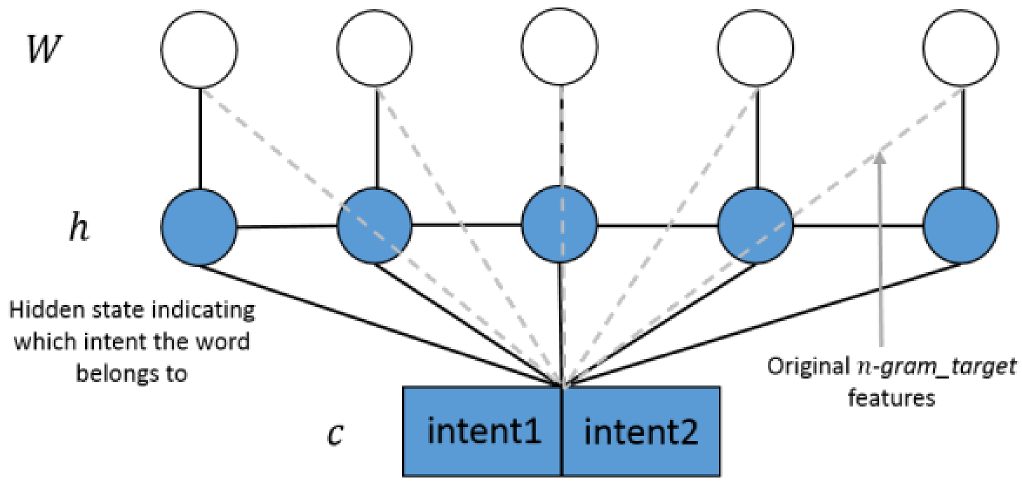
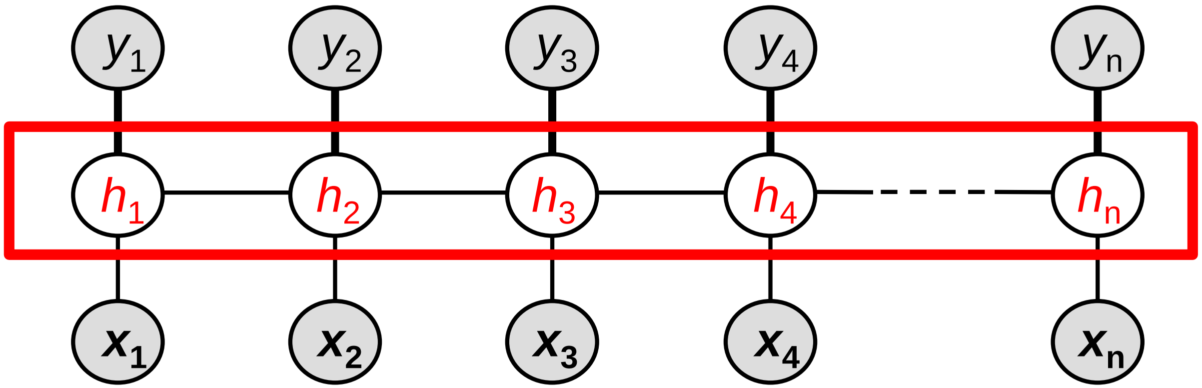


Figure from Xu & Sarikaya, INTERSPEECH 2013.

Latent-dynamic CRFs

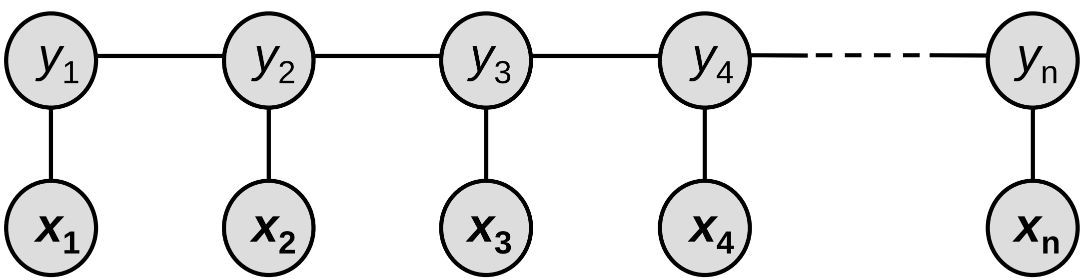
Latent-dynamic CRFs (LDCRF)

(Morency et al., CVPR 2007; Sun et al., COLING 2008)



LDCRF

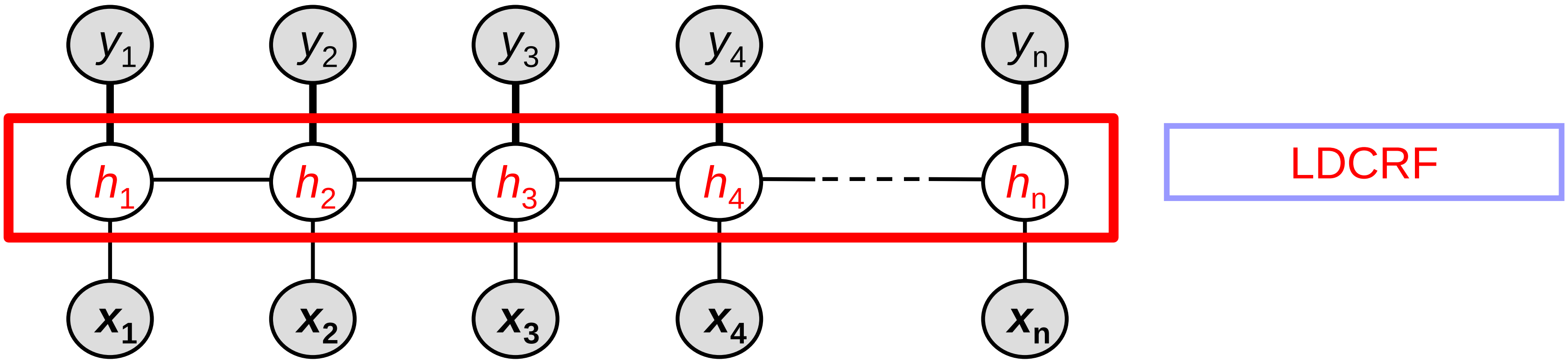
We can think (informally) it as
“CRF + unsup. learning on latent info”



Conditional random fields

Latent-dynamic CRFs

- **Latent-dynamic CRFs (LDCRF)**
  (Morency et al., CVPR 2007; Sun et al., COLING 2008)



$$P(\mathbf{y} \mid \mathbf{x}, \theta) = \sum_{\mathbf{h}: \forall h_j \in \mathcal{H}_{y_j}} P(\mathbf{h} \mid \mathbf{x}, \theta) \quad = \sum_{\mathbf{h}: \forall h_j \in \mathcal{H}_{y_j}} \frac{1}{\mathcal{Z}(x, \theta)} \exp\left(\sum_k \theta_k \mathbf{F}_k(\boldsymbol{h}, \boldsymbol{x}) \right)$$

For Large-scale Structured Prediction

- **Latent-dynamic CRFs (LDCRF)**
  - Training is slow
    - May need week-level time

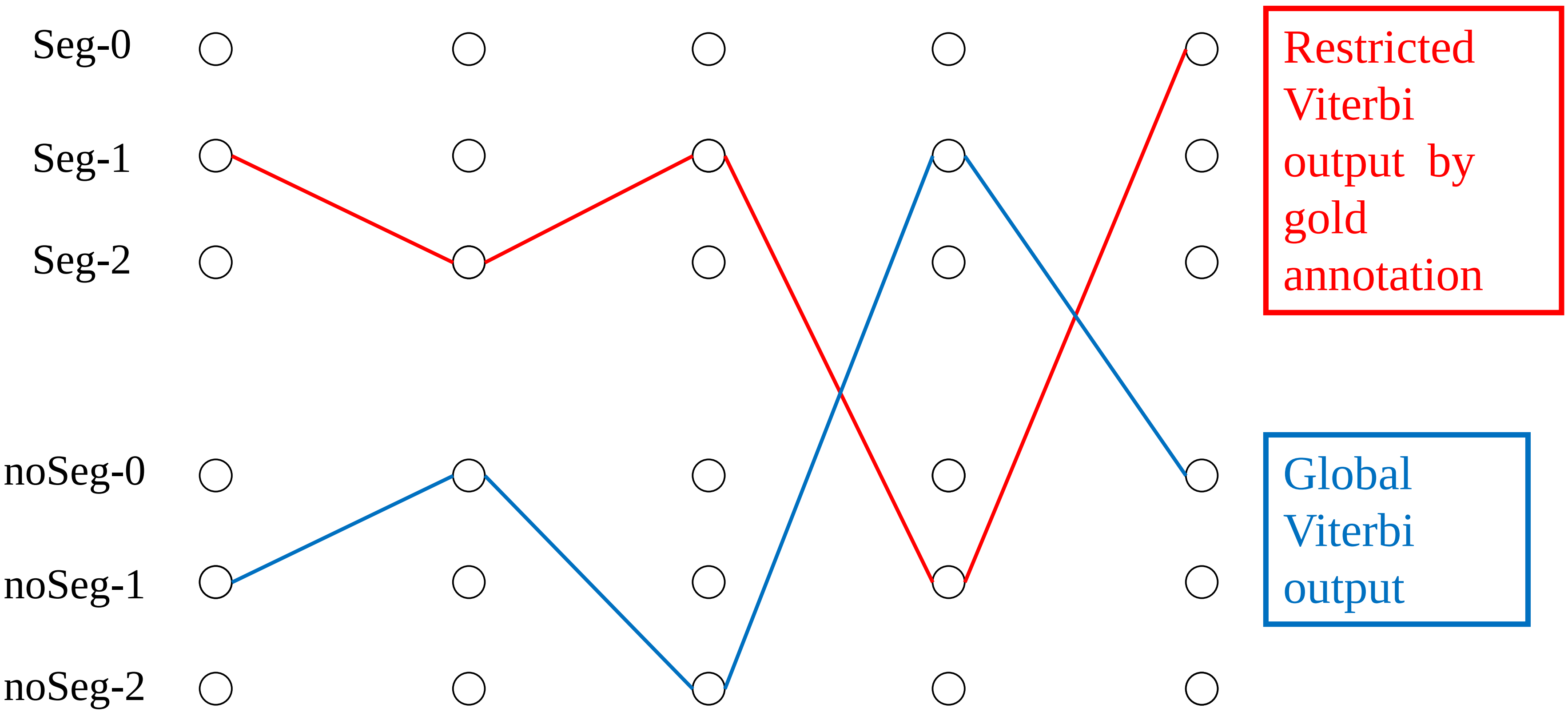
- **Solution**
  - Perceptron is much faster than CRF
  - Latent CRF -> **Latent perceptron**

Sun et al. Latent variable perceptron algorithm for structured classification. IJCAI 2009.

Sun et al. Latent structured perceptrons for large-scale learning with hidden information. TKDE 2013.

Latent Variable Perceptron

- **For fast training of latent variable models**
  (Sun et al., IJCAI 2009; Sun et al., TKDE 2013)



A related work on machine translation: Liang et al., 2006

Latent Variable Perceptron

- **For fast training of latent variable models**
  (Sun et al., IJCAI 2009; Sun et al., TKDE 2013)

LDCRF:

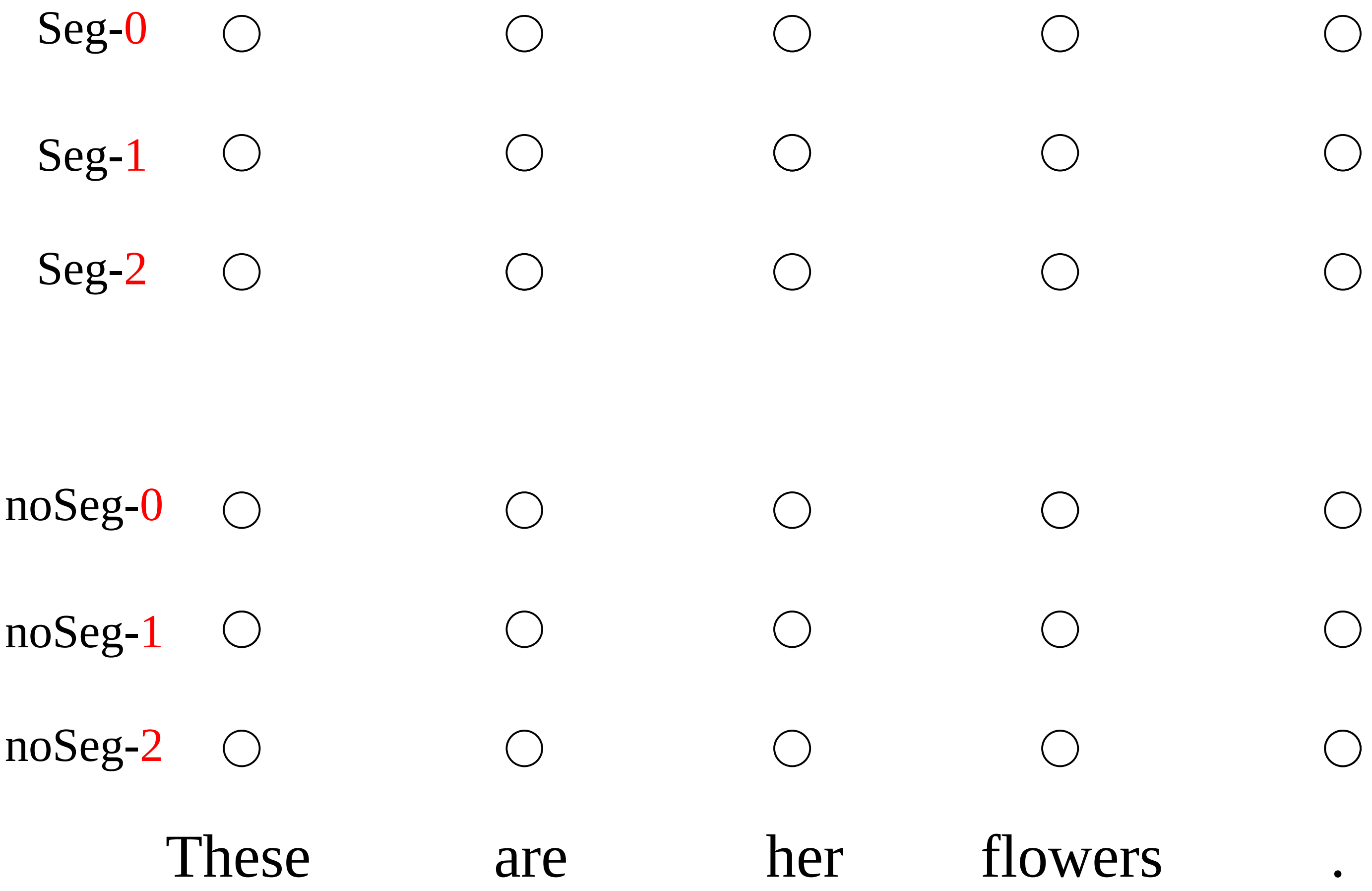
$$\begin{cases} \boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \cancel{\sum_{\boldsymbol{h}:\mathrm{Proj}(\boldsymbol{h})=\boldsymbol{y}}} P(\boldsymbol{h} \mid \boldsymbol{x}, \boldsymbol{\theta}) \\ \text{Normally, batch training} \\ \text{(do weight update after go over all samples)} \end{cases}$$

Latent variable perceptron (*Sun et al., 2009*) :

$$\begin{cases} \boldsymbol{h}^* = \arg\max_{\boldsymbol{h}} P'(\boldsymbol{h} \mid \boldsymbol{x}, \boldsymbol{\theta}) \\ \text{Online training} \\ \text{(do weight update on each sample)} \end{cases}$$

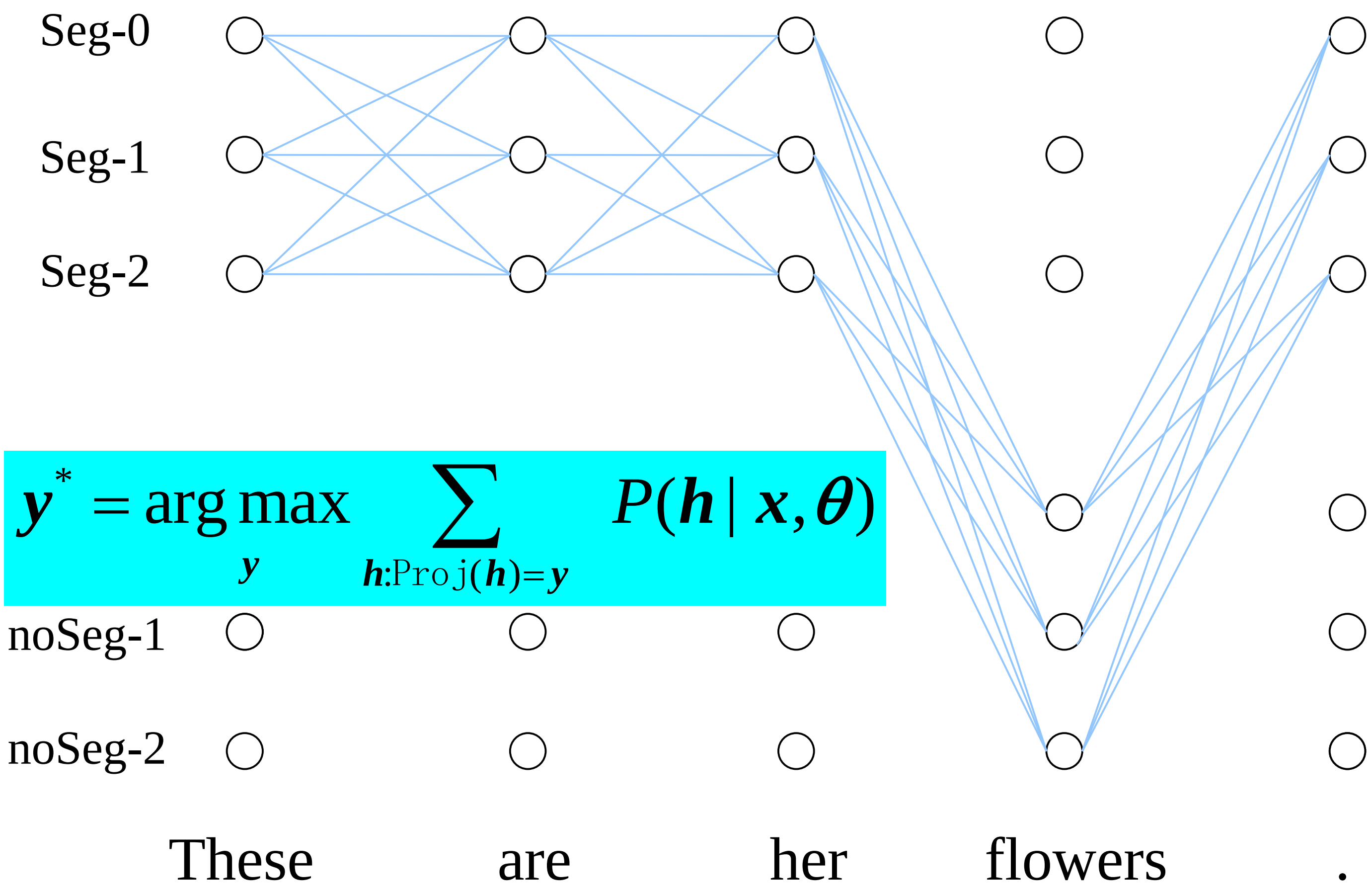
Latent Variable Perceptron

LDCRF training method



Latent Variable Perceptron

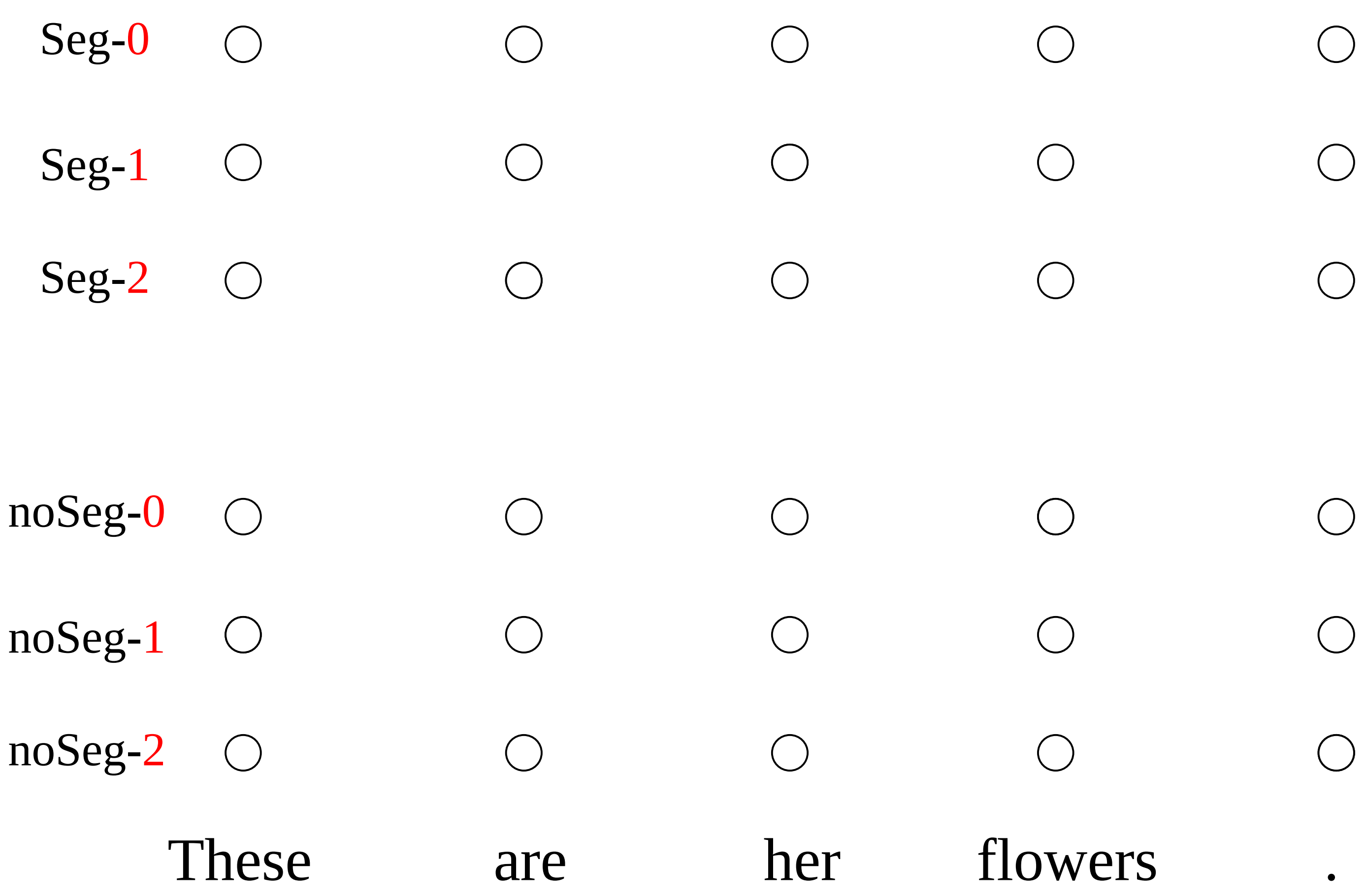
LDCRF training method



$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{h}:\mathrm{Proj}(\boldsymbol{h})=\boldsymbol{y}} P(\boldsymbol{h} \mid \boldsymbol{x}, \boldsymbol{\theta})$$

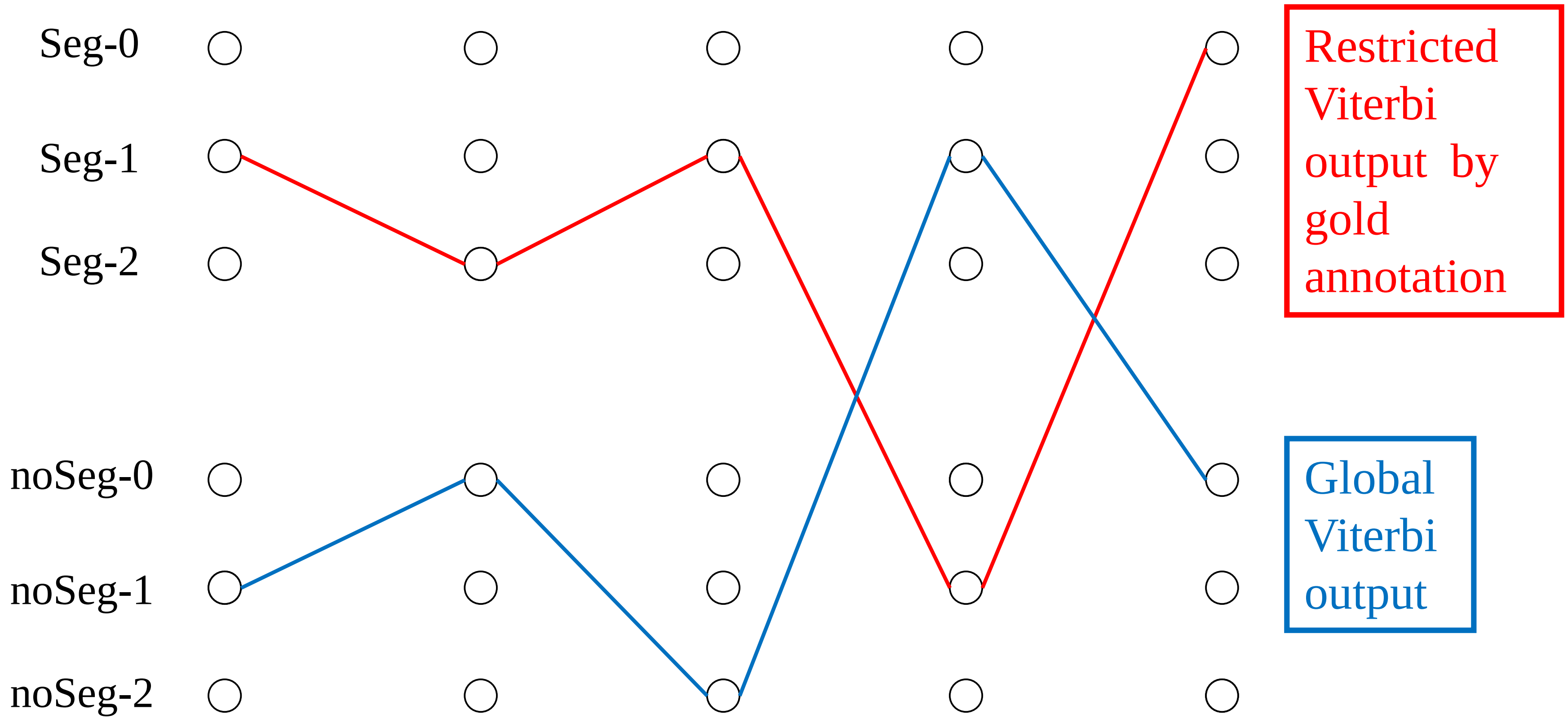
Latent Variable Perceptron

- **Latent perceptron training**



Latent Variable Perceptron

- **Latent perceptron training**



$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} + \boxed{\mathbf{f}[\arg\max_{\mathbf{h}} F(\mathbf{h} \mid \mathbf{y}_i^{*}, \mathbf{x}_i, \boldsymbol{\theta}^{i}), \mathbf{x}_i]} - \boxed{\mathbf{f}[\arg\max_{\mathbf{h}} F(\mathbf{h} \mid \mathbf{x}_i, \boldsymbol{\theta}^{i}), \mathbf{x}_i]}$$

Latent Variable Perceptron

- **Theoretical guarantee of convergence**
  - **As far as traditional perceptron is separable, latent structured perceptron is also separable.**

Theorem 1. *Given the latent feature mapping* $\mathbf{m} = (m_1, \ldots, m_n)$, *for any sequence of training examples* $(\mathbf{x}_i, \mathbf{y}_i^*)$ *which is separable with margin* δ *by a vector* $\mathbf{U}$ *represented by* $(\alpha_1, \ldots, \alpha_n)$ *with* $\sum_{i=1}^{n} {\alpha_i}^2 = 1$, *the examples then will also be latently separable with margin* $\bar{\delta}$, *and* $\bar{\delta}$ *is bounded below by*

$$\boxed{\bar{\delta} \geq \delta / T,}$$

where $T = (\sum_{i=1}^{n} m_i {\alpha_i}^2)^{1/2}$.

Latent Variable Perceptron

- Theoretical guarantee of convergence
  - Latent perceptron still converges

Theorem 2. *For any sequence of training examples* $(\mathbf{x}_i, \mathbf{y}_i^*)$ *which is separable with margin* δ*, the number of mistakes of the latent perceptron algorithm in Figure 1 is bounded above by*

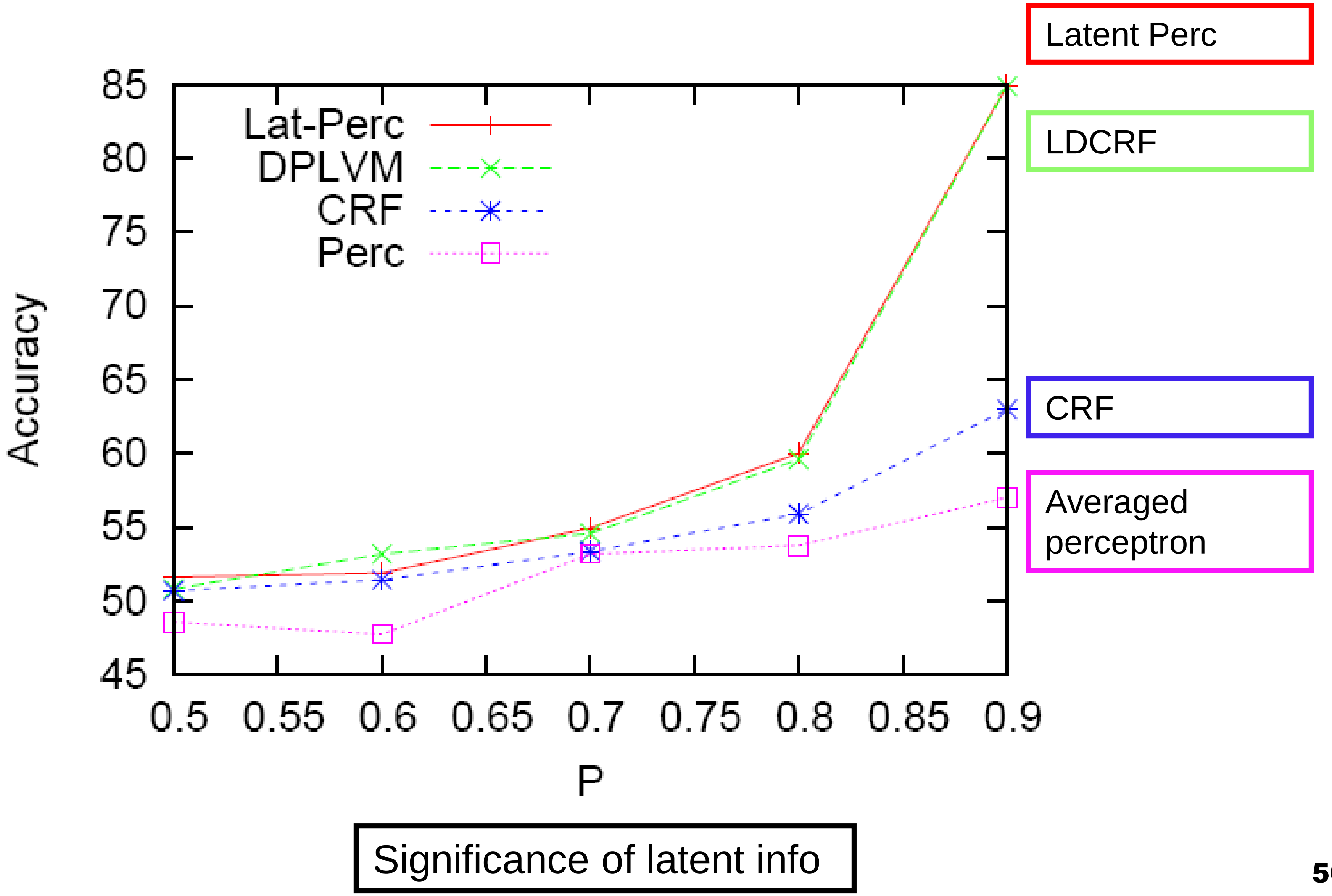
$$\boxed{number\ of\ mistakes \le 2T^2M^2/\delta^2}$$

Theorem 3. *For any training sequence* $(\mathbf{x}_i, \mathbf{y}_i^*)$*, the number of mistakes made by the latent perceptron training algorithm is bounded above by*

$$\boxed{number\ of\ mistakes \le \min_{\overline{\mathbf{U}}, \overline{\delta}} (\sqrt{2}M + D_{\overline{\mathbf{U}}, \overline{\delta}})^2 / \overline{\delta}^2}$$

Latent Variable Perceptron

- **Experiment on synthetic data:**
  Much better accuracy than CRF & Perceptron



Latent Variable Perceptron

- **Good performance in question answering** (Fader et al. SIGKDD2014)
  - Use query with partial anotation as latent variable

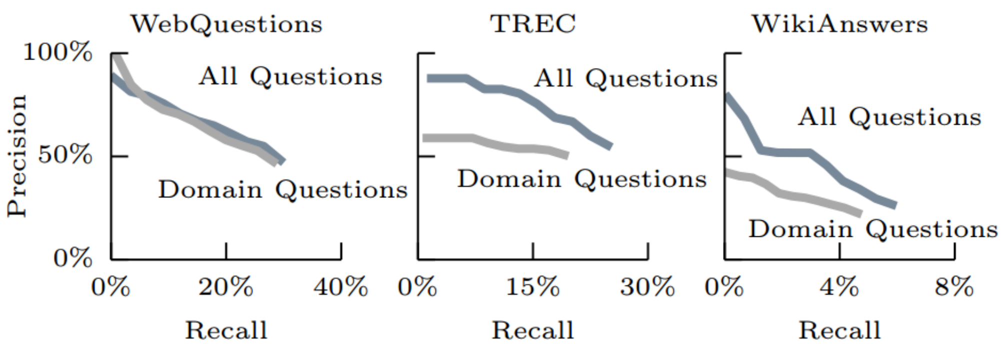


Figure 5: Training OQA on questions from all question sets leads to greater precision and recall than training on domain questions only.

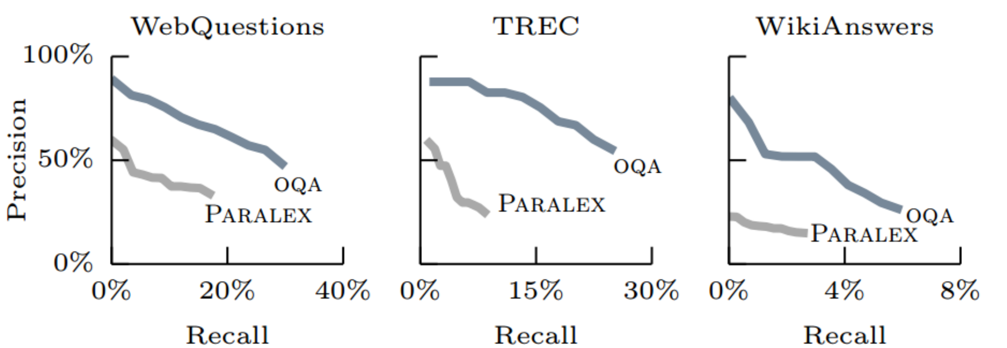


Figure 6: OQA has higher precision and recall than the Open QA system Paralex.

OQA is based on latent variable perceptron

(Sun et al., IJCAI 2009;
Sun et al., TKDE 2013)

Latent Variable Perceptron

- **Good performance in speech sentence classification**
  (Xu & Sarikaya, INTERSPEECH 2013)
  - Split multi-intent using latent variable
  - Based on latent variable perceptron (Sun et al., IJCAI 2009; Sun et al., TKDE 2013)

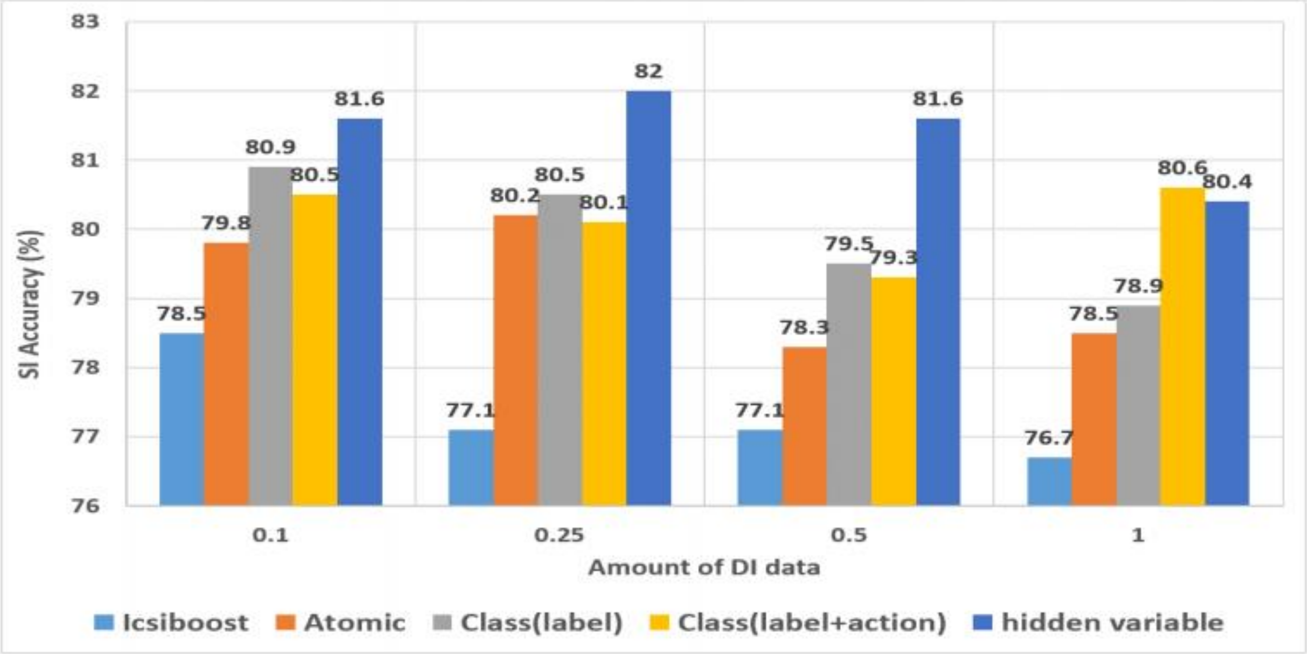


Figure 4: *Intent detection accuracy (%) on SI test set with models trained on SI data plus various amounts of DI data.*

Latent Variable Perceptron

- **Good performance in semantic parsing** (Zhou et al., IJCAI 2013)
  - Hybrid tree as latent structure variable
  - Based on latent variable perceptron (Sun et al., IJCAI 2009; Sun et al., TKDE 2013)

Better Performance

	10-fold cross-val			600 train/280 test		
	Prec.	Rec.	F	Prec.	Rec.	F.
WASP	87.2	74.8	80.5	85.6	71.1	77.7
LU	89.3	81.5	85.2	85.7	76.8	81.0
tsVB-hand	–	–	–	79.3	79.3	79.3
LVP+EXT	**90.9**	**90.9**	**90.9**	**87.5**	**87.5**	**87.5**

Table 4: Performance comparison with other directly comparable systems over English corpus.

	German			Greek			Thai		
	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
WASP	**87.1**	65.7	74.9	**88.5**	70.7	78.6	79.0	71.4	75.0
LU	76.4	62.1	68.5	80.8	69.3	74.6	80.1	73.6	76.7
tsVB-hand	74.6	74.6	74.6	75.4	75.4	75.4	78.2	**78.2**	**78.2**
LVP+EXT	78.6	**78.6**	**78.6**	80.3	**80.3**	**80.3**	76.4	76.4	76.4

Table 5: Performance comparison among models on the multilingual section of GeoQuery.

Latent Variable Perceptron

- **Good performance in dependency parsing**
  (Honnibal & Johnson, TACL 2014)
  - Dynamic oracle/transitions as latent variable
  - Train with latent variable perceptron (Sun et al., IJCAI 2009; Sun et al., TKDE 2013)

	P	R	F	UAS	LAS	w/s
Baseline joint	79.4	70.1	74.5	89.9	86.9	711
+Features	86.0	77.2	81.3	90.5	87.5	539
+Edit transition	92.2	80.2	85.8	90.9	87.9	555
Oracle pipeline	100	100	100	91.7	88.6	782

Table 1: Development results for the joint models. For the baseline model, disfluencies reduce parse accuracy by 1.7% Unlabelled Attachment Score (UAS). Our features and Edit transition reduce the gap to 0.7%, and improve disfluency detection by 11.3% F-measure.

	Disfl. F	UAS
Johnson et al pipeline	82.1	90.3
Qian and Liu pipeline	83.9	90.1
Baseline joint parser	73.9	89.4
Final joint parser	84.1	**90.5**

Much better compared with baseline model

Table 2: Test-set parse and disfluency accuracies. The joint parser is improved by the features and Edit transition, and is better than pre-processing the text with state-of-the-art disfluency detectors.

Latent Variable Perceptron

- **Good performance in coreference resolution**
  (Fernandes et al., CL 2012)
  - Use structures of coreference trees as latent variable
  - Based on latent variable perceptron (Sun et al., IJCAI 2009; Sun et al., TKDE 2013)

Language	Parse / Mentions	MUC			B^3			$CEAF_e$			Mean
		R	P	F_1	R	P	F_1	R	P	F_1	
Arabic	Auto / GB	45.18	47.39	46.26	64.56	69.44	66.91	49.73	47.39	48.53	**53.90**
	Auto / GM	57.25	76.48	65.48	60.27	79.81	68.68	72.61	46.00	56.32	**63.49**
	Golden / Auto	46.38	51.78	48.93	63.53	72.37	67.66	52.57	46.88	49.56	**55.38**
	Golden / GB	46.38	51.78	48.93	63.53	72.37	67.66	52.57	46.88	49.56	**55.38**
	Golden / GM	56.89	76.27	65.17	60.07	80.02	68.62	72.24	45.58	55.90	**63.23**
Chinese	Auto / GB	58.76	71.46	64.49	66.62	79.88	72.65	54.09	42.02	47.29	**61.48**
	Auto / GM	61.64	90.81	73.43	63.55	89.43	74.30	72.78	39.68	51.36	**66.36**
	Golden / Auto	59.35	74.49	66.07	66.31	81.43	73.10	55.97	41.50	47.66	**62.28**
	Golden / GB	59.35	74.49	66.07	66.31	81.43	73.10	55.97	41.50	47.66	**62.28**
	Golden / GM	61.70	91.45	73.69	63.57	89.76	74.43	72.84	39.49	51.21	**66.44**
English	Auto / GB	64.92	77.53	70.67	64.25	78.95	70.85	56.48	41.69	47.97	**63.16**
	Auto / GM	70.69	91.21	79.65	65.46	85.61	74.19	74.71	42.55	54.22	**69.35**
	Golden / Auto	67.73	77.25	72.18	66.42	78.01	71.75	56.16	44.51	49.66	**64.53**
	Golden / GB	65.65	78.26	71.40	64.36	79.09	70.97	57.36	42.23	48.65	**63.67**
	Golden / GM	71.18	91.24	79.97	65.81	85.51	74.38	74.93	43.09	54.72	**69.69**

Table 4: Supplementary results on the test sets alternating parse quality and mention candidates. Parse quality can be automatic or golden; and mention candidates can be automatically identified (Auto), golden mention boundaries (GB) or golden mentions (GM).

For Large-scale Structured Prediction

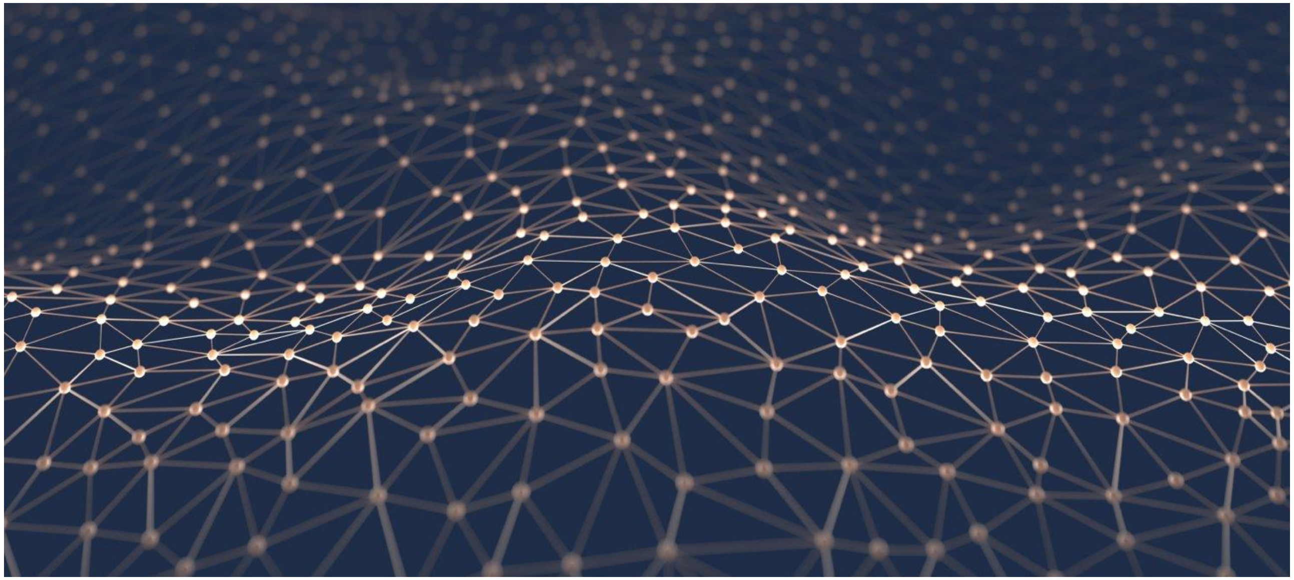
- **Latent Variable Perceptron**
  - Proposed by Sun (2009, 2013)
- **Fast and accurate**
  - Accuracy equal or even better than CRF
  - Almost as fast as perceptron
  - Suitable for large-scale structured prediction problems

Sun et al. Latent variable perceptron algorithm for structured classification. IJCAI 2009.

Sun et al. Latent structured perceptrons for large-scale learning with hidden information. TKDE 2013.

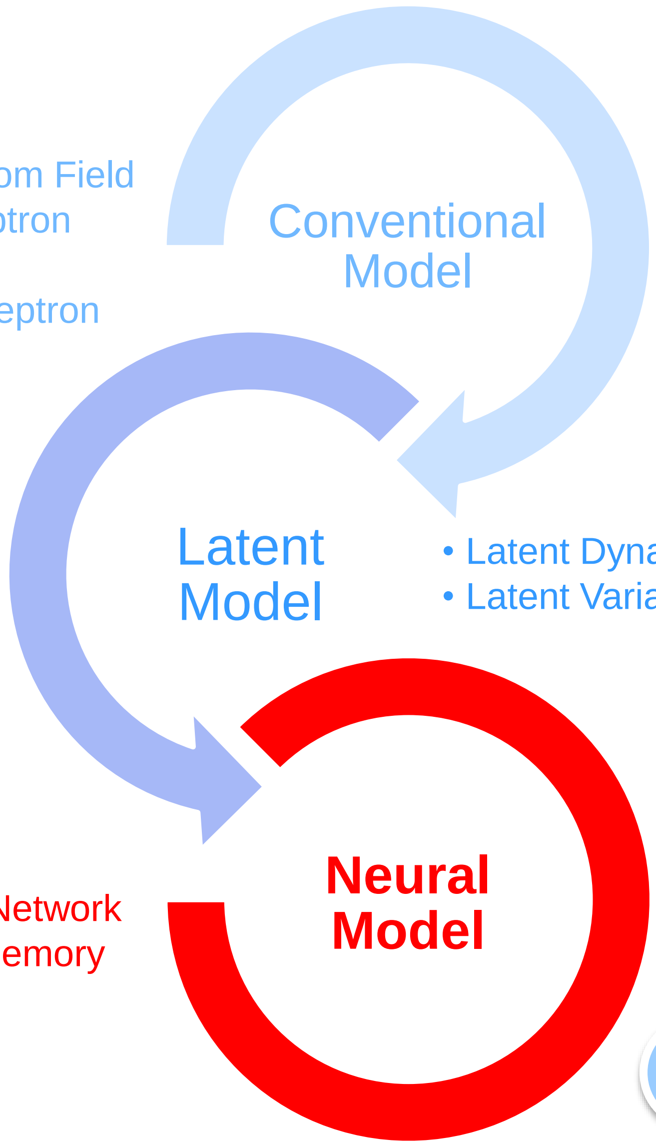
Outline

- Latent Model can reduce annotation engineering
- Furthermore, how to reduce the cost of feature engineering?
- Neural Model
  - Automatically extract features



Outline

- Conditional Random Field
- Structured Perceptron
- MIRA
- Probabilistic Perceptron



Reduce Annotation Engineering

- Latent Dynamic CRF
- Latent Variable Perceptron

- Recurrent Neural Network
- Long Short-term Memory
- LSTM-CRF

Reduce Feature Engineering

Motivation

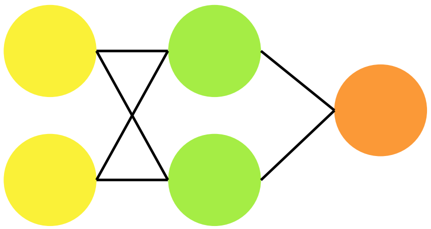
- **Problem in feature engineering**
  - Require linguistics knowledge
  - A lot of feature templates
  - Ad-hoc, some features are not very reasonable
  - Task-sensitive

Feature Type	Description
Prior	$\forall k \; x_i = k$
Transition	$\forall k,k' \; x_i = k$ and $x_{i+1}=k'$
Word	$\forall k,w \; x_i = k$ and $o_i=w$ $\forall k,w \; x_i = k$ and $o_{i-1}=w$ $\forall k,w \; x_i = k$ and $o_{i+1}=w$ $\forall k,w,w' \; x_i = k$ and $o_i=w$ and $o_{i-1}=w'$ $\forall k,w,w' \; x_i = k$ and $o_i=w$ and $o_{i+1}=w'$
Orthography: Suffix	$\forall s$ in {"ing","ed","ogy","s","ly","ion","tion", "ity", ...} and $\forall k \; x_i=k$ and o_i ends with s
Orthography: Punctuation	$\forall k \; x_i = k$ and o_i is capitalized $\forall k \; x_i = k$ and o_i is hyphenated $\forall k \; x_i = k$ and o_i contains a period $\forall k \; x_i = k$ and o_i is ALL CAPS $\forall k \; x_i = k$ and o_i contains a digit (0-9) ...

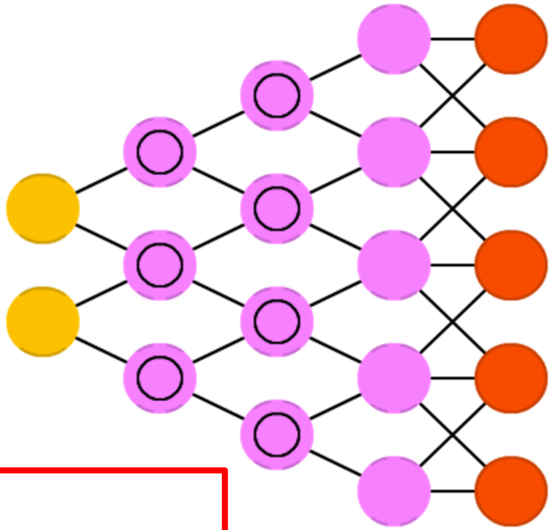
- **Neural networks**
  - Automatically learn features in hidden layers

Neural Network

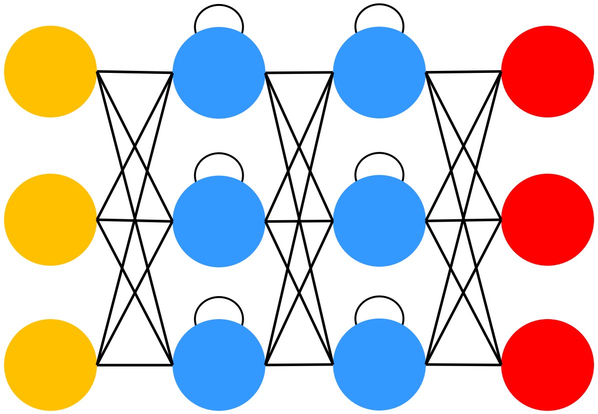
- **Many kinds**



  - Feed Forward NN
    - logistic regression



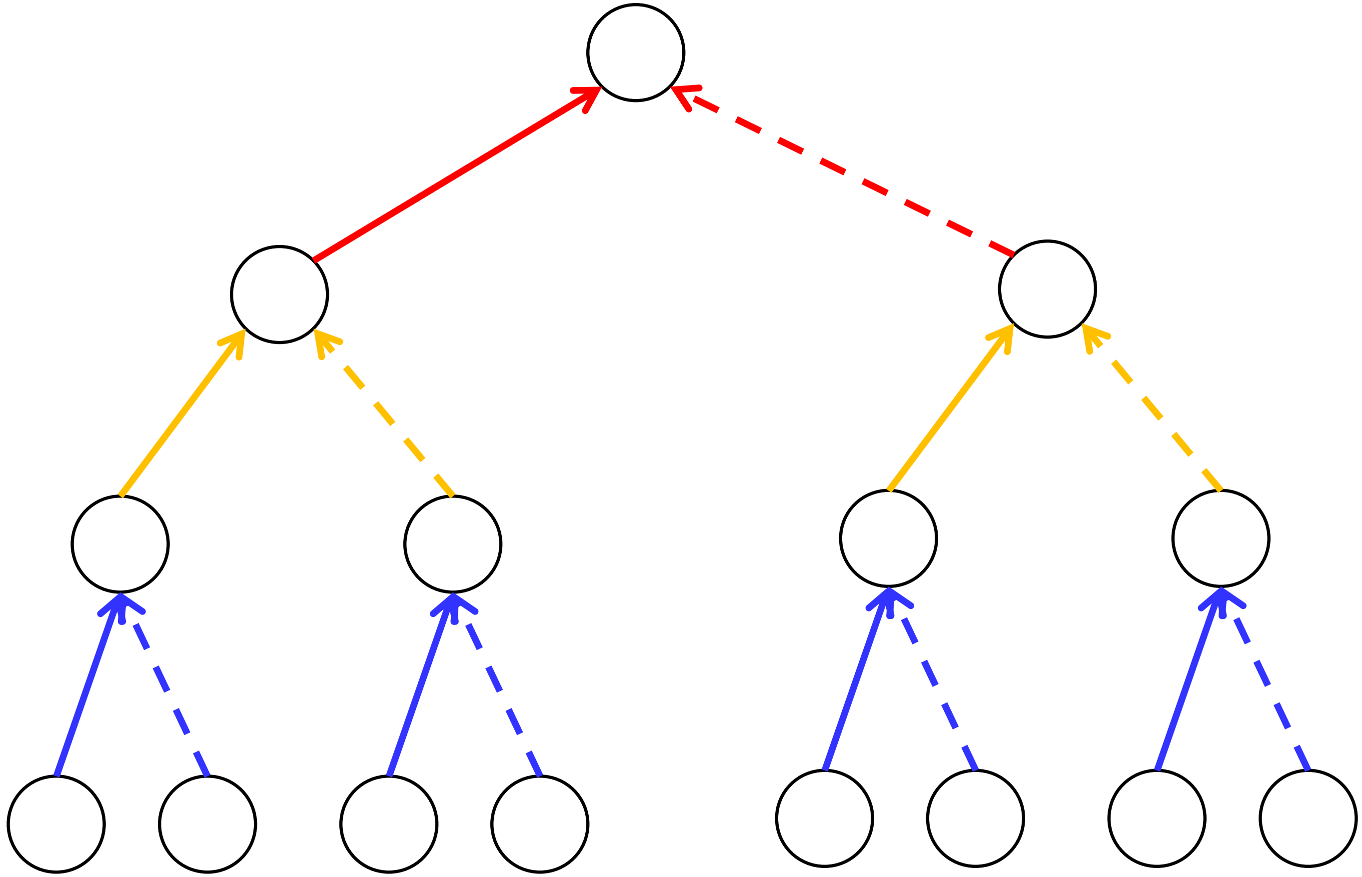
  - Convolutional NN
    - image processing



  - Recursive/Recurrent NN
    - structured prediction

Recursive Neural Network

- **Recursive neural network** (Socher et al., ICML 2011)
  - Model hierarchical structures
  - Condition on each sub-structure independently

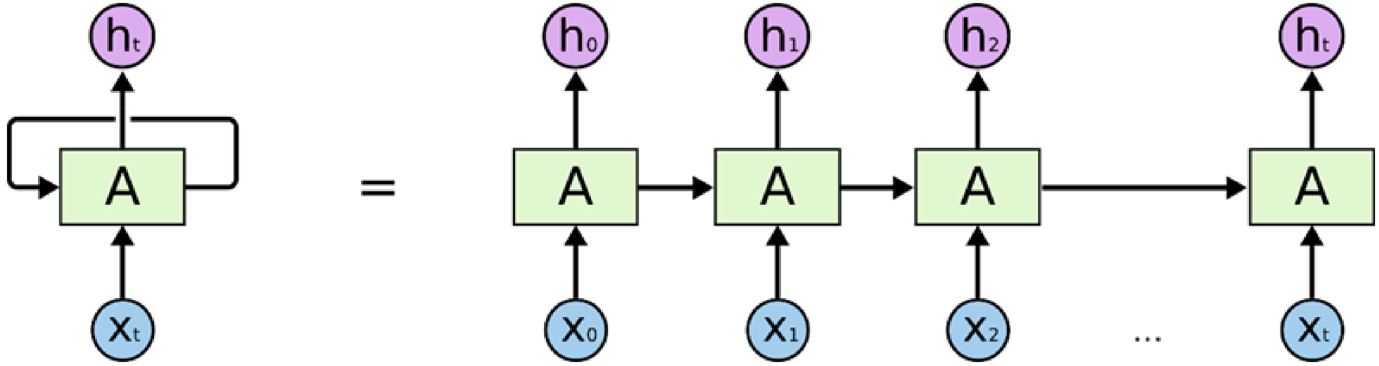


Recurrent Neural Network (RNN)

- **Recurrent neural network** (Elman, Cognitive Science 1990)
  - Model time series
  - Predict linear-chain structures
  - Conditioned on all previous input

$$h_t = f(Uh_{t-1} + Wx_t)$$

$$\hat{y}_t = softmax(W^{(s)}h_t)$$



Picture from Christopher Olah

Recurrent Neural Network (RNN)

- **Problems**
  - Gradient Exploding/Vanishing (Pascanu et al., ICML 2013)
  - Hard to capture long-term dependencies

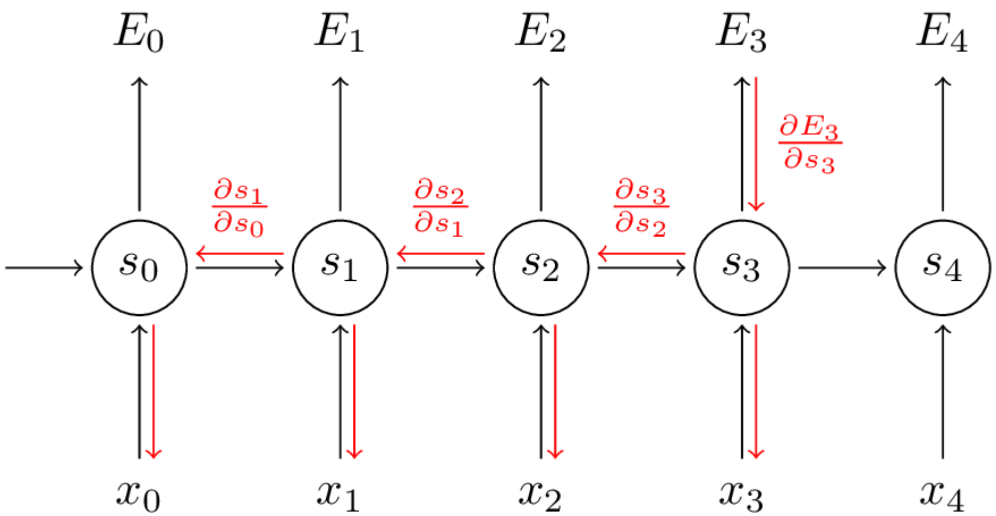
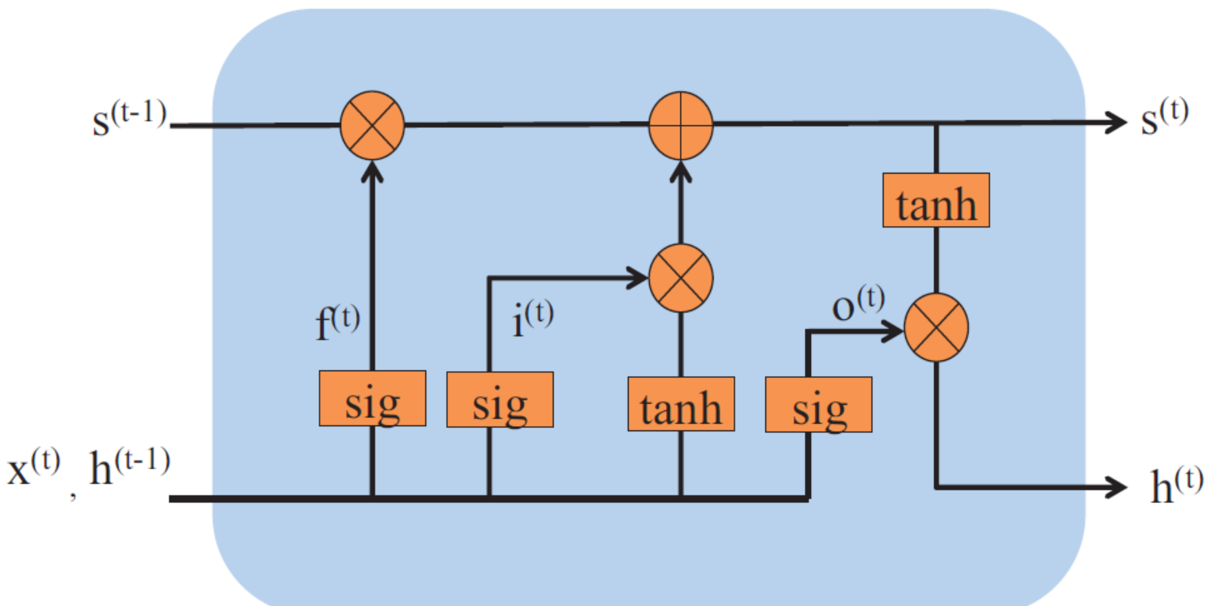


Figure from Denny Britz

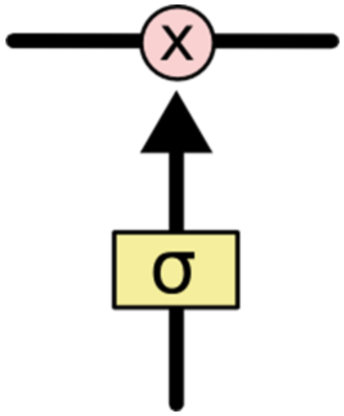
Long Short-term Memory (LSTM)

- **Long short-term memory** (Hochreiter and Schmidhuber 1997)
  - A lasting linear memory
    - Capture long distance dependency
  - Three gates: input, forget and output gates
    - Control modification to the memory

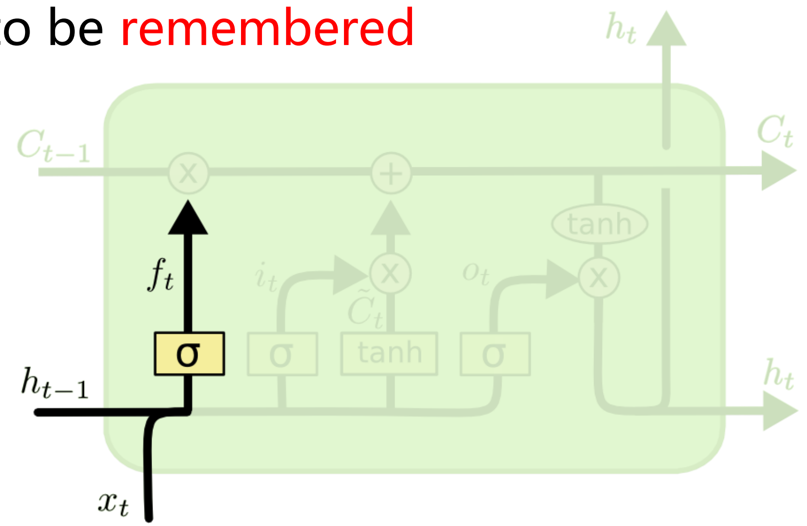


Introduction of Gates

- **Gate**
  - Sigmoid-activated layer
    - Output value from 0 to 1
  - Member-wise multiplication
    - How much to flow through



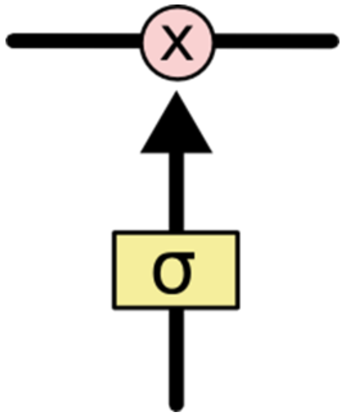
- **Gates in LSTM**
  - Forget gate
    - How much old memory needs to be remembered



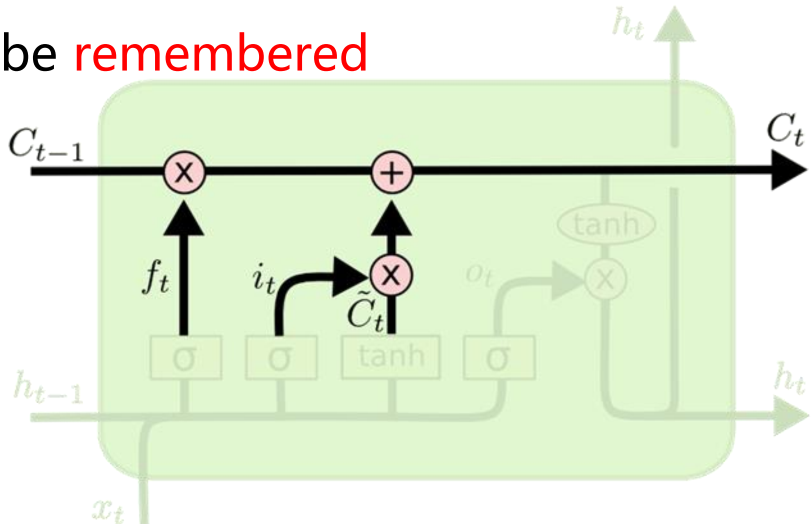
Some Picture from Christopher Olah

Introduction of Gates

- **Gate**
  - Sigmoid-activated layer
    - Output value from 0 to 1
  - Member-wise multiplication
    - How much to flow through

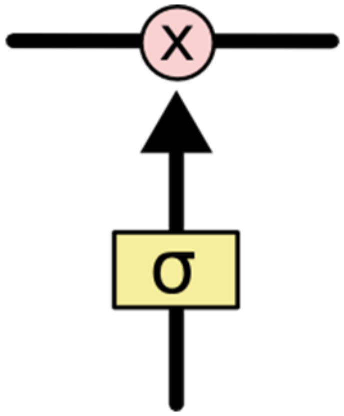


- **Gates in LSTM**
  - Forget gate
    - How much old memory needs to be remembered
  - Input gate

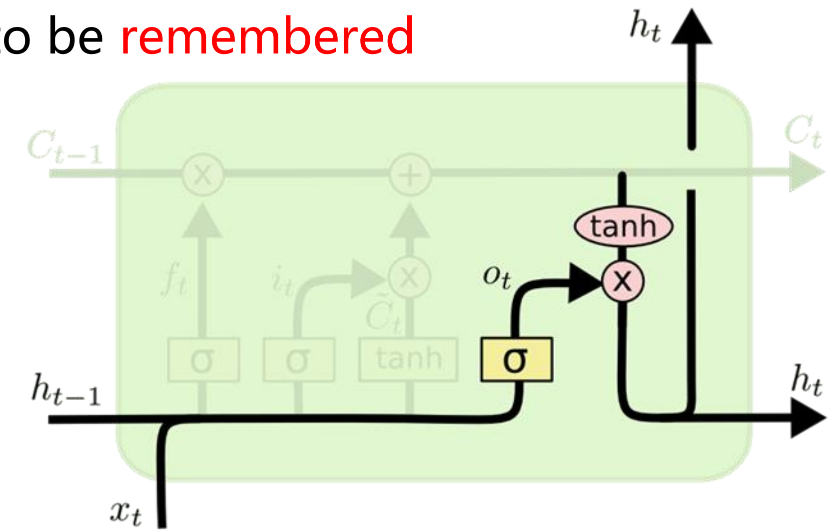


Introduction of Gates

- **Gate**
  - Sigmoid-activated layer
    - Output value from 0 to 1
  - Member-wise multiplication
    - How much to flow through



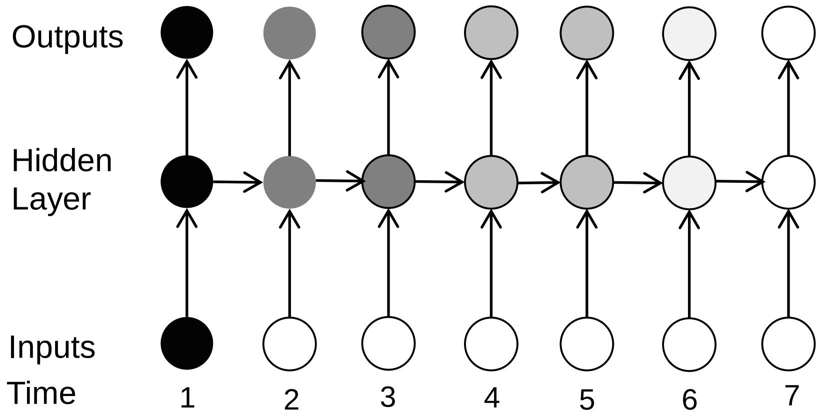
- **Gates in LSTM**
  - Forget gate
    - How much old memory needs to be remembered
  - Input gate
  - Output gate



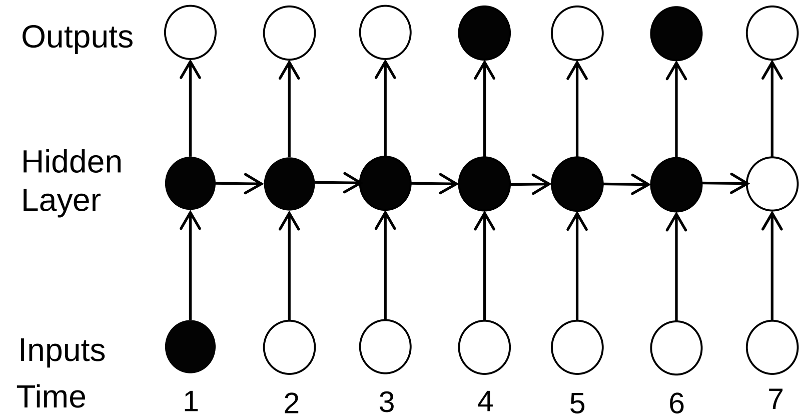
Some Picture from Christopher Olah

Introduction of Gates

- RNN
  - Vanishing influence

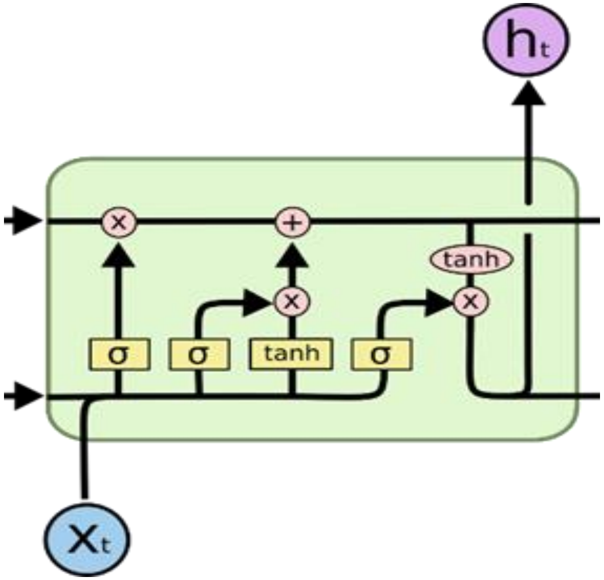


- LSTM
  - Situation-aware



Problem of LSTM

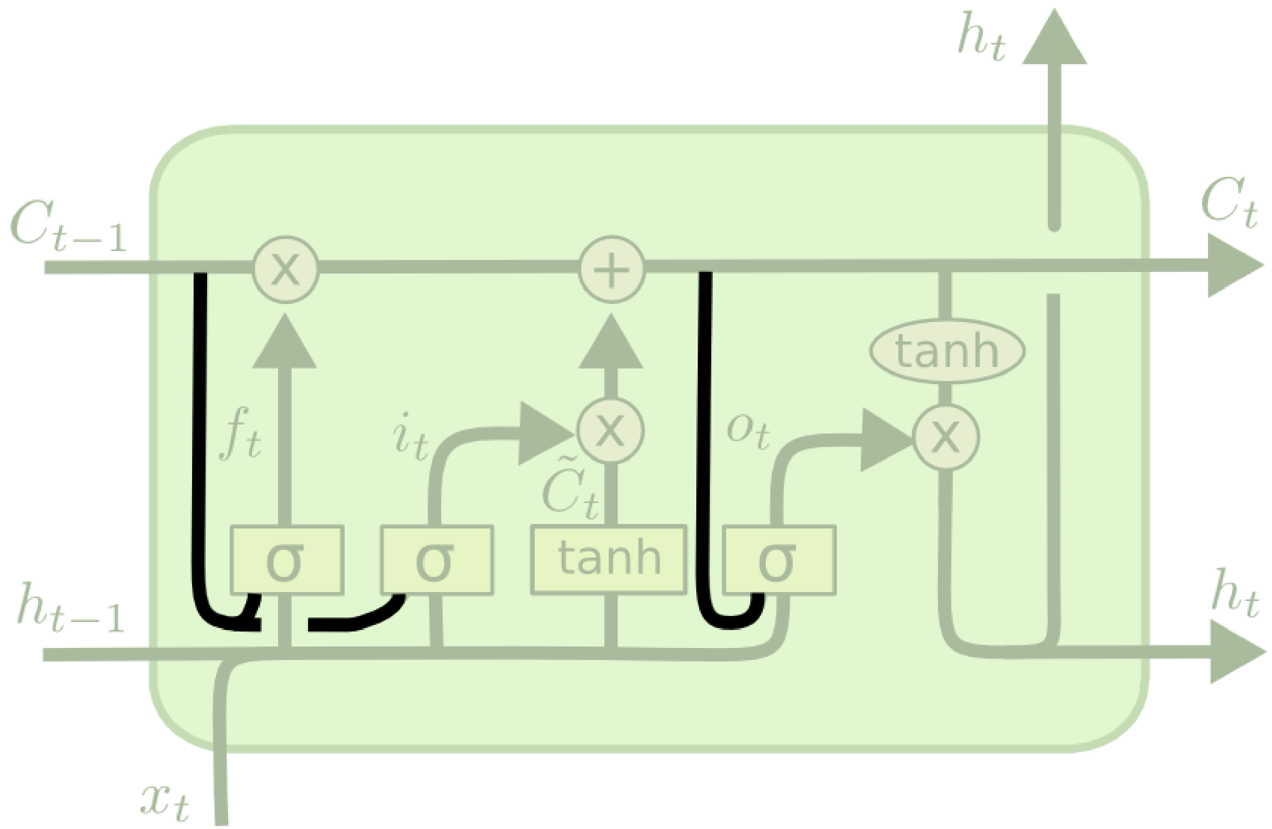
- Can be slow for large models (hard to train)
  - Structure complexity is higher
  - Structure redundancy?
  - Too much parameters



Picture from Christopher Olah

Refinement of LSTM

- **Peepholes**
  - Introduced by Gers (2000)
  - Include current memory when compute gates

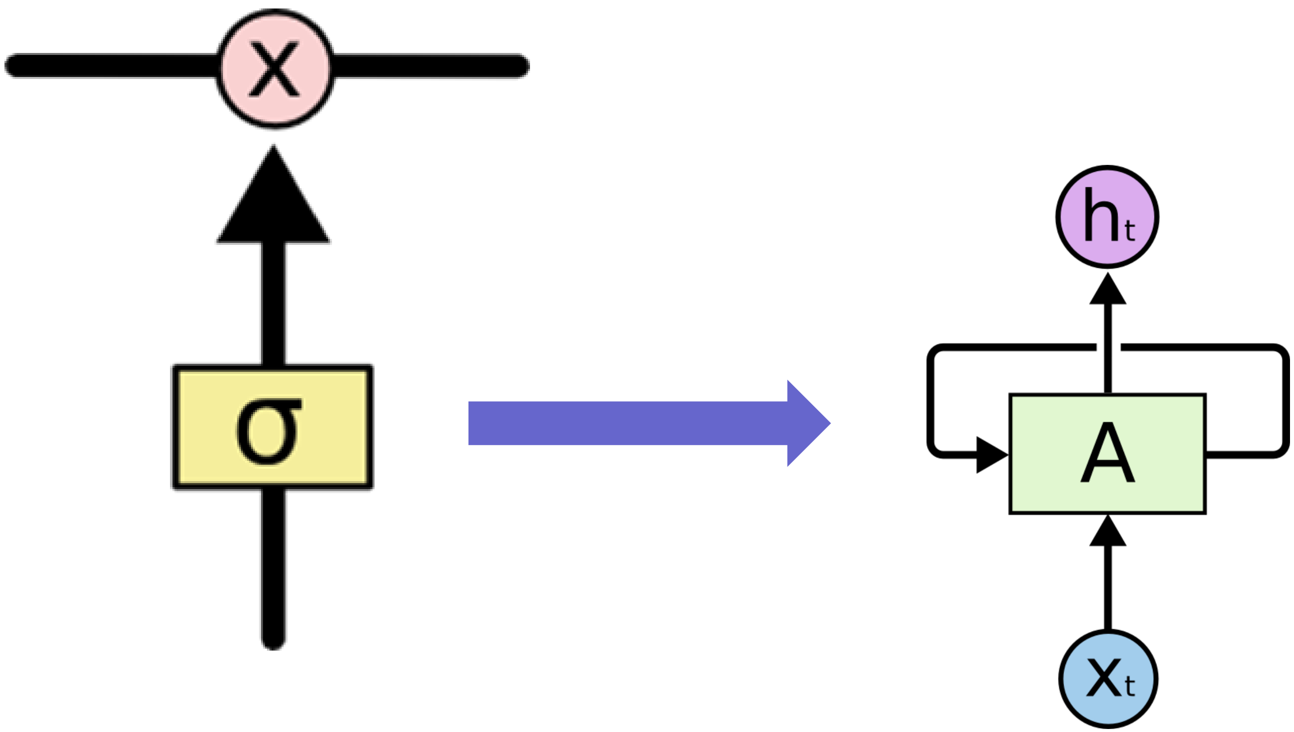


Picture from Christopher Olah

Refinement of LSTM

Fully connected gates

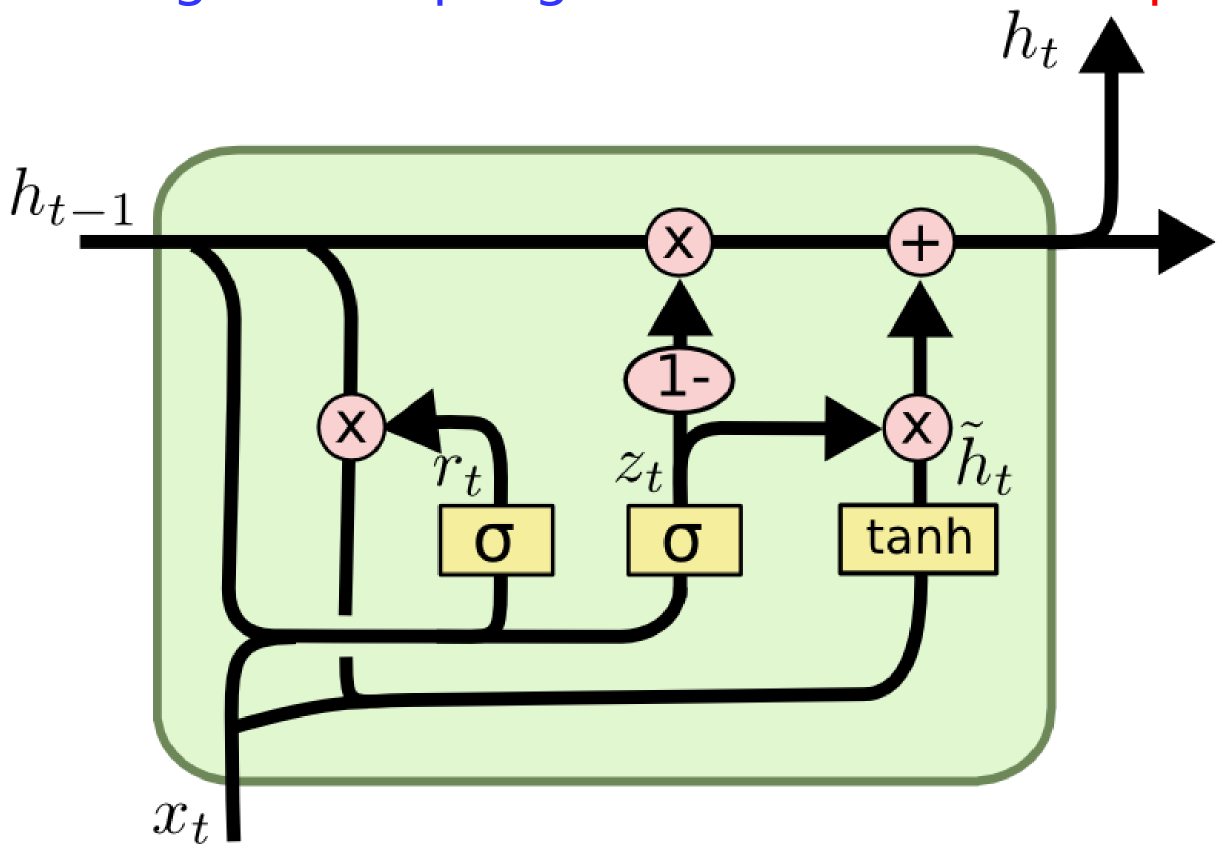
- Included in Hochreiter and Schmidhuber (1997)
- Gates are also recurrent



Picture from Christopher Olah

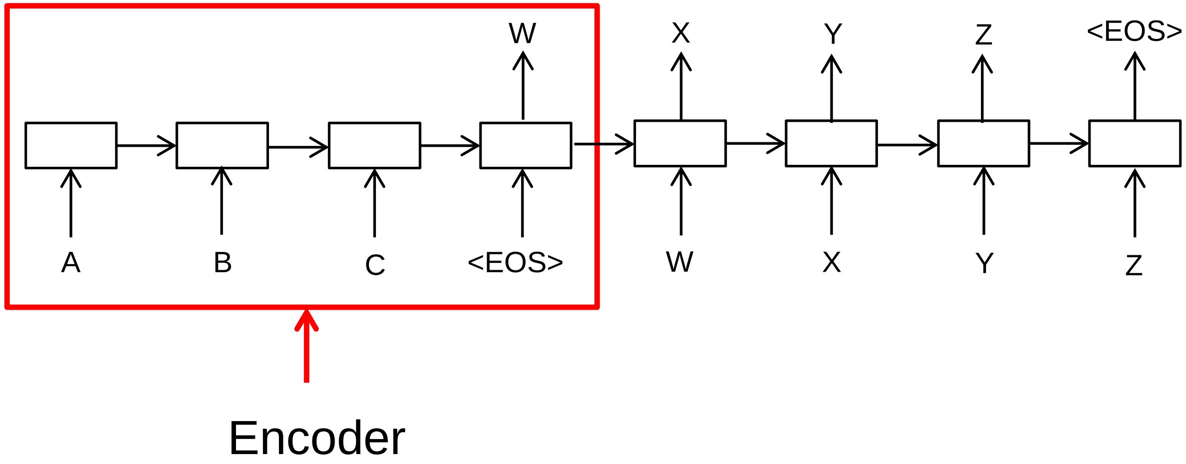
Refinement of LSTM

- **Gated recurrent unit (GRU)**
  - Introduced by Chung et al. (2014)
  - Coupled forget and input gates with structure simplification



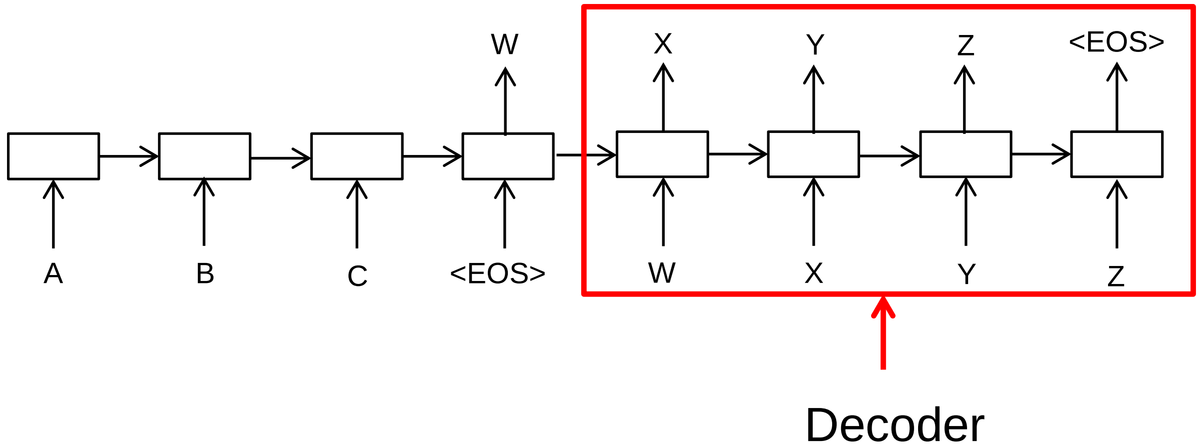
More recent development

- **Sequence to sequence neural network** (Sutskever et al., NIPS 2014)
  - Encoder & Decoder
  - The encoder information is stored in a fixed-length vector



More recent development

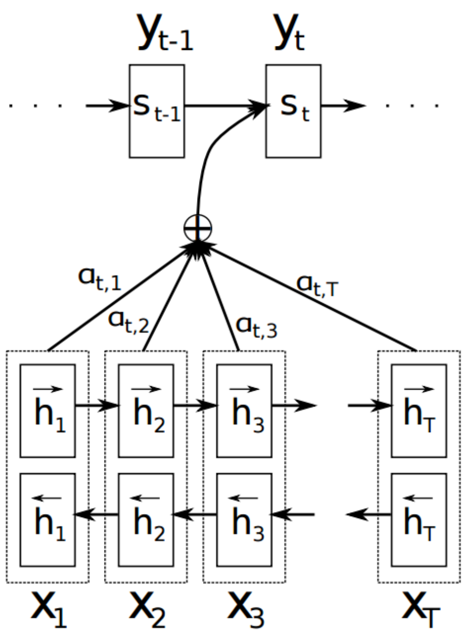
- **Sequence to sequence neural network** (Sutskever et al., NIPS 2014)
  - Encoder & Decoder
  - The encoder information is stored in a fixed-length vector

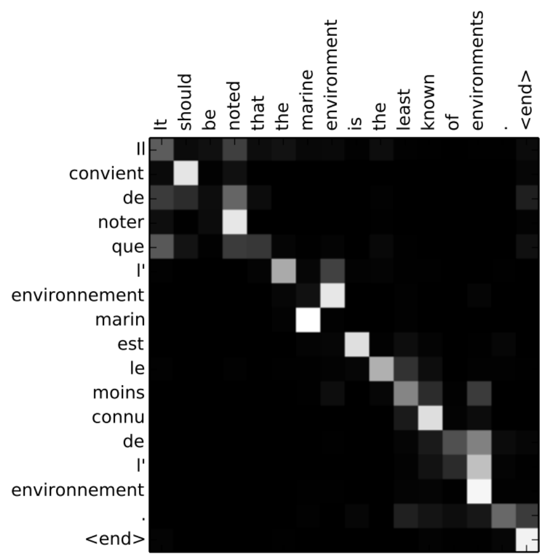


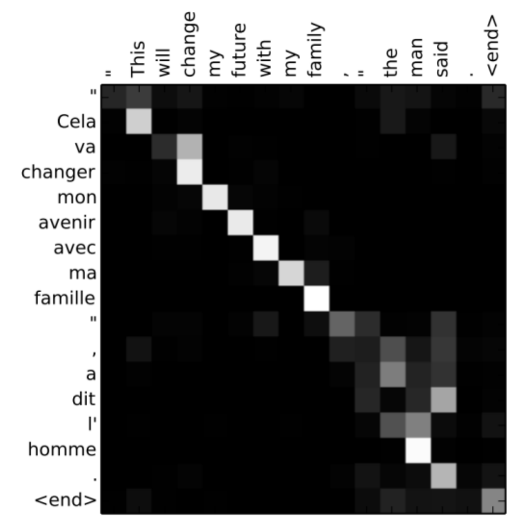
- Learn to align
- Neural machine translation

Structured Neural Network in Machine Translation

- **Attention-based neural network** (Bahdanau et al. 2014; Luong et al. 2015)
  - Each hidden state has an unique weight/attention/importance







For Large-Scale Structured Prediction

- **Training large-scale neural models is costly**
  - Numerous parameters
  - Very slow
  - A NMT model may take weeks (even months) to train

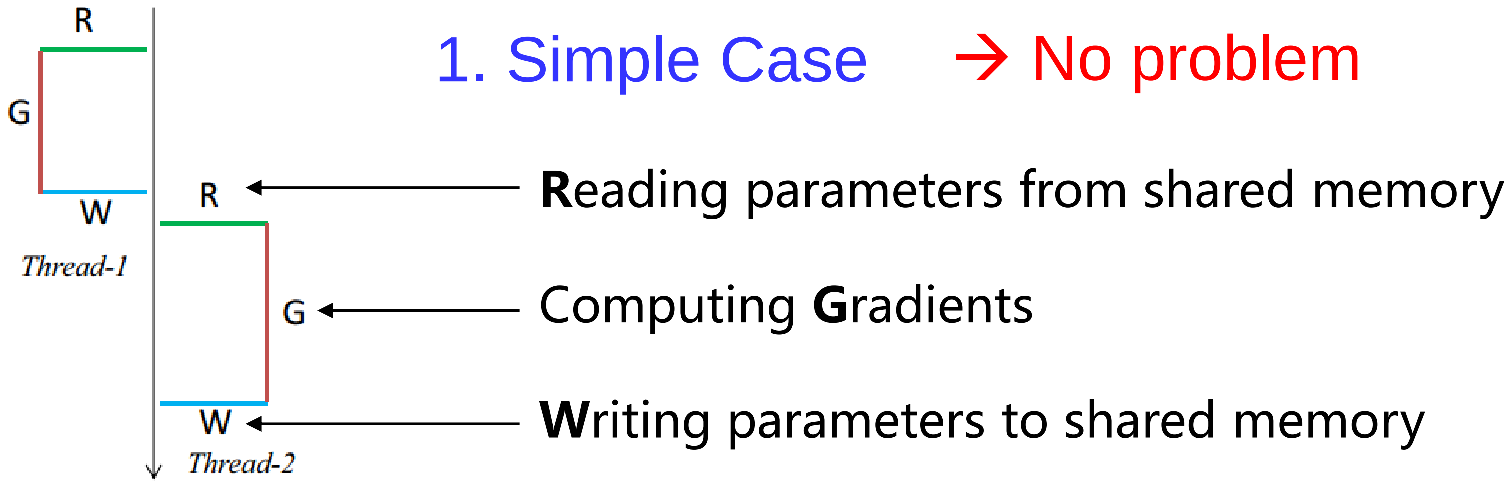
- **How to accelerate training speed?**
  - Parallel training
  - Especially, asynchronous (lock-free) parallel training

Asynchronous Parallel Learning

- **Motivation**
  - Asynchronous parallel learning is very popular for traditional sparse feature models
    - E.g., *HogWild!* (Niu et al. NIPS 2011)
  - However, previous asynchronous parallel learning methods do not suit neural networks
    - Because NN is dense feature model
    - Previous parallel learning for dense feature models is mostly synchronous, e.g., mini-batch parallel learning, GPU parallel learning

Asynchronous Parallel Learning

- **Motivation**
  - Asynchronous parallel learning is very popular for traditional sparse feature models
    - E.g., *HogWild!* (Niu et al. NIPS 2011)

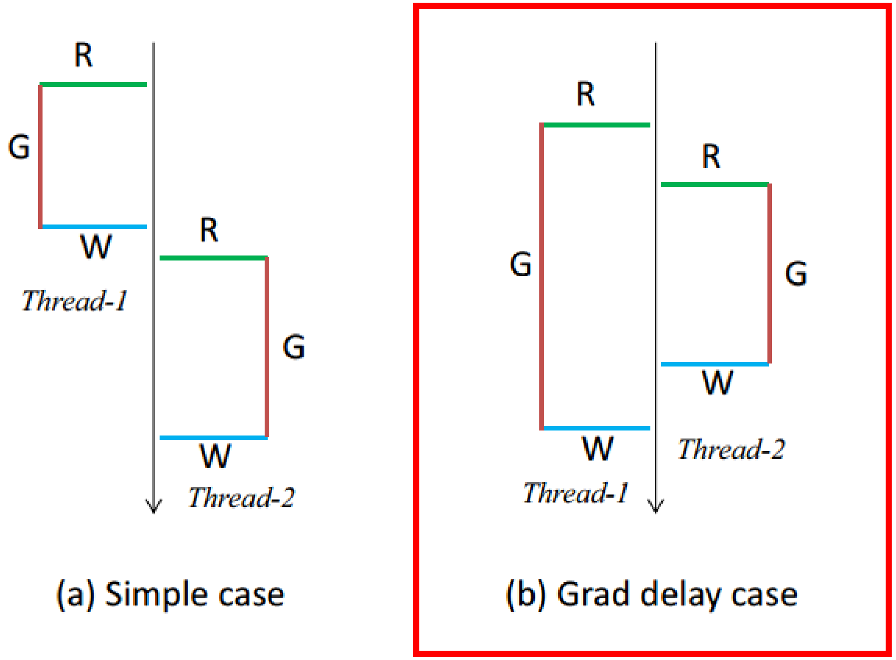


(a) Simple case

1. Simple Case → No problem

Asynchronous Parallel Learning

- **Motivation**
  - Asynchronous parallel learning is very popular for traditional sparse feature models
    - E.g., *HogWild!* (Niu et al. NIPS 2011)



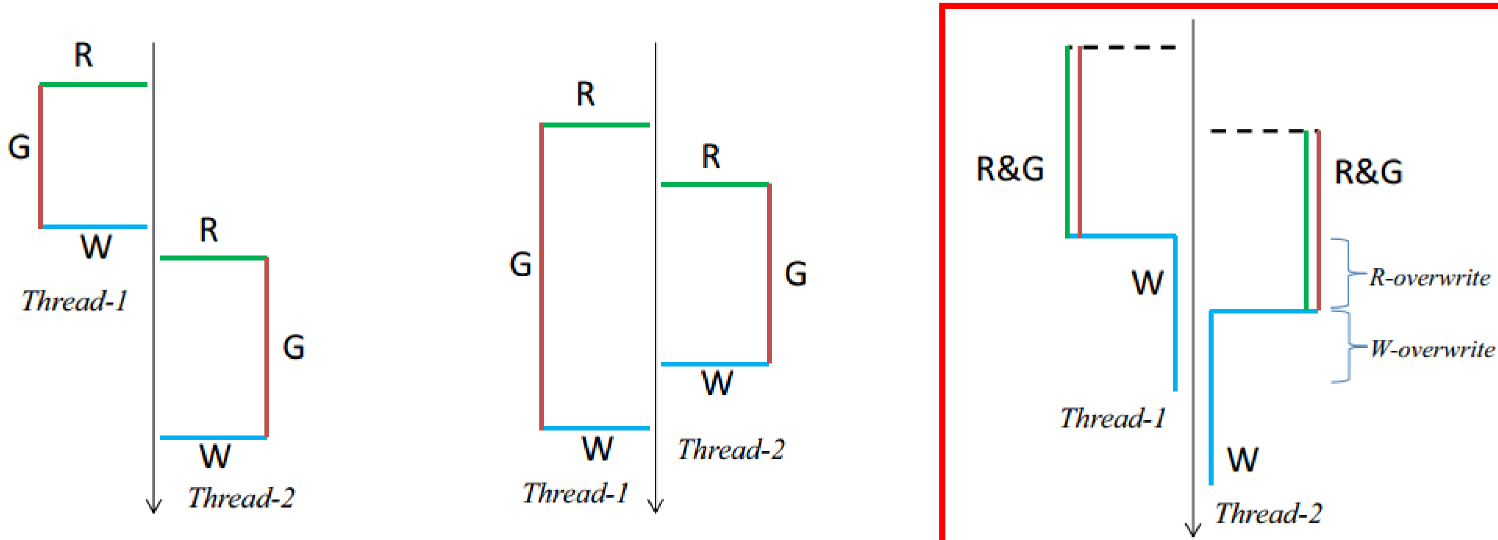
2. This case is called Gradient Delay case

→ More complicated, but problem solved for sparse feature models (Niu et al. NIPS 2011)

Asynchronous Parallel Learning

Motivation

- Asynchronous parallel learning is very popular for traditional sparse feature models
  - E.g., *HogWild!* (Niu et al. NIPS 2011)

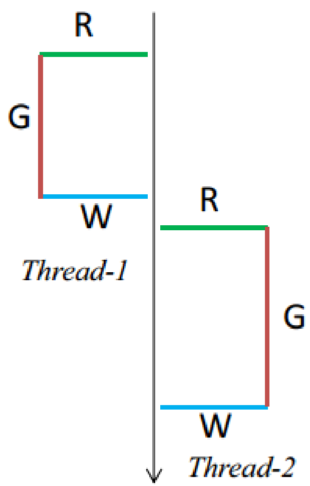


(a) Simple case (b) Grad delay case (c) Grad error case

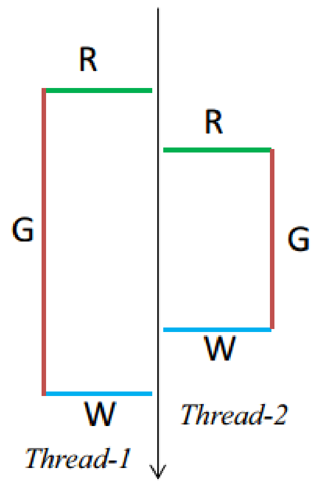
Asynchronous Parallel Learning

- **3. Even more difficult case: Gradient Error Case**
  - Happens for dense feature models, like neural networks
    - Actions (R, G & W) are time-consuming
  - Read-overwrite and write-overwrite problems

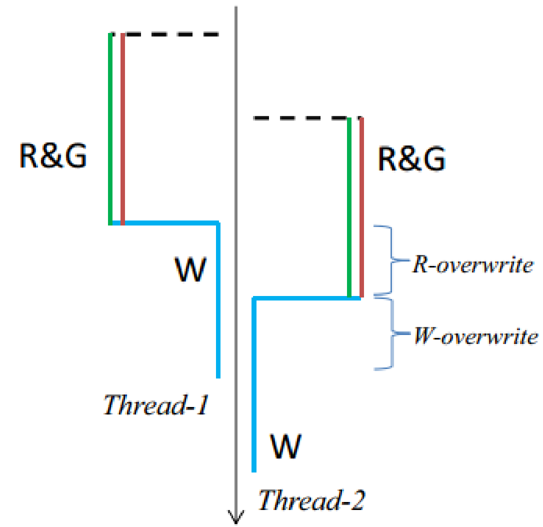
→Not well studied before, how to deal with this problem?



(a) Simple case



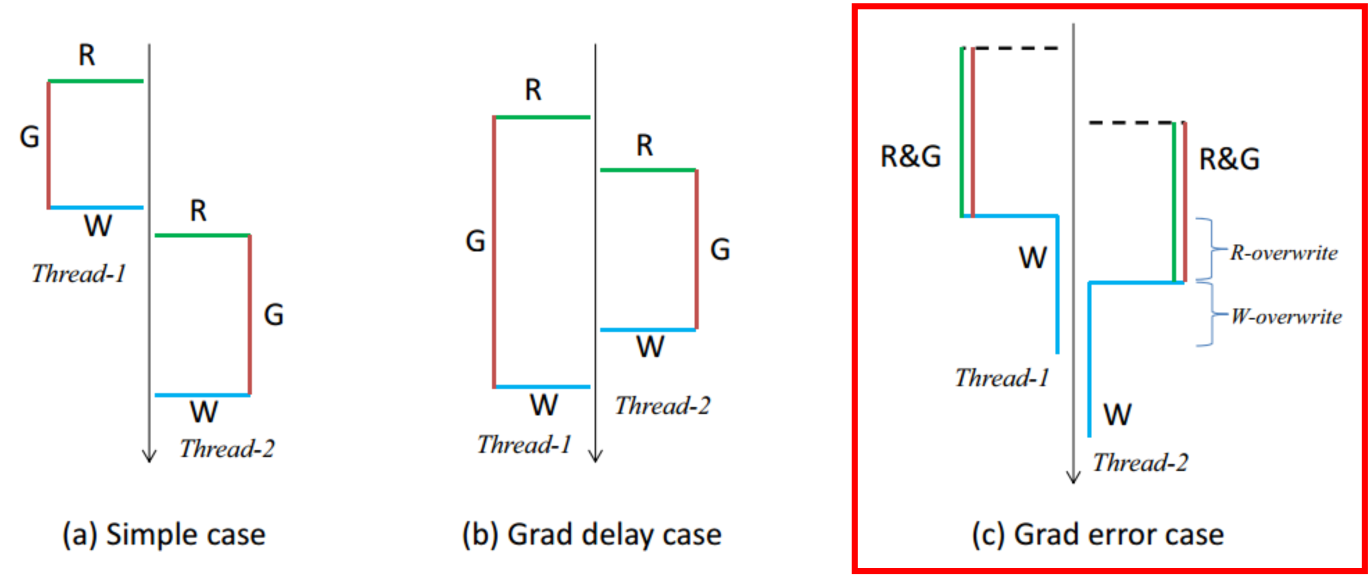
(b) Grad delay case

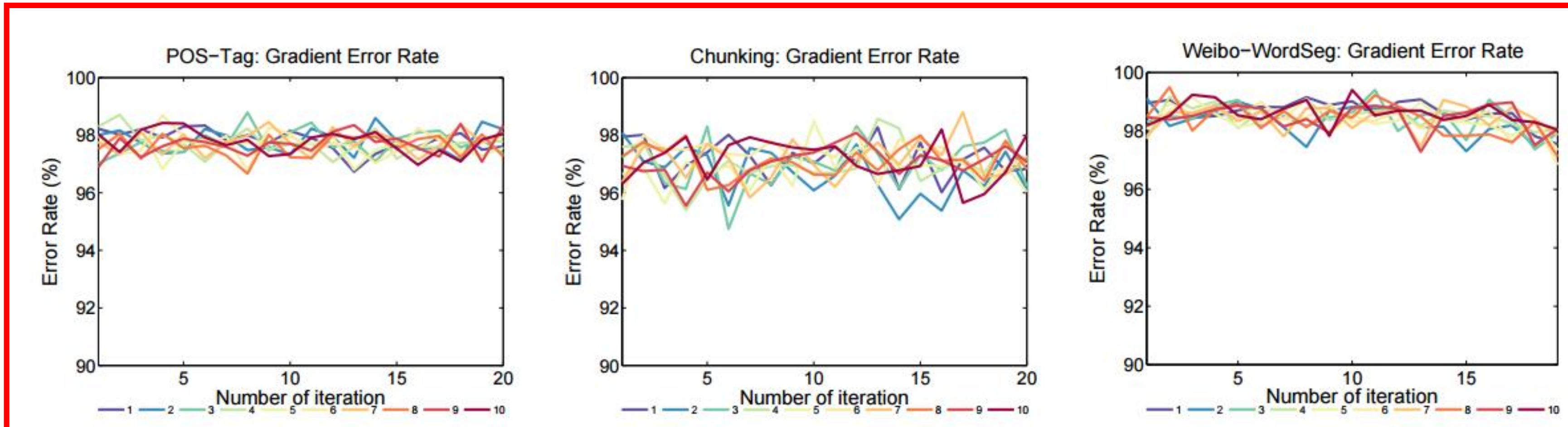


(c) Grad error case

Experimental observations

- **Gradient error is inevitable in asynchronous training of neural networks** in real-world tasks



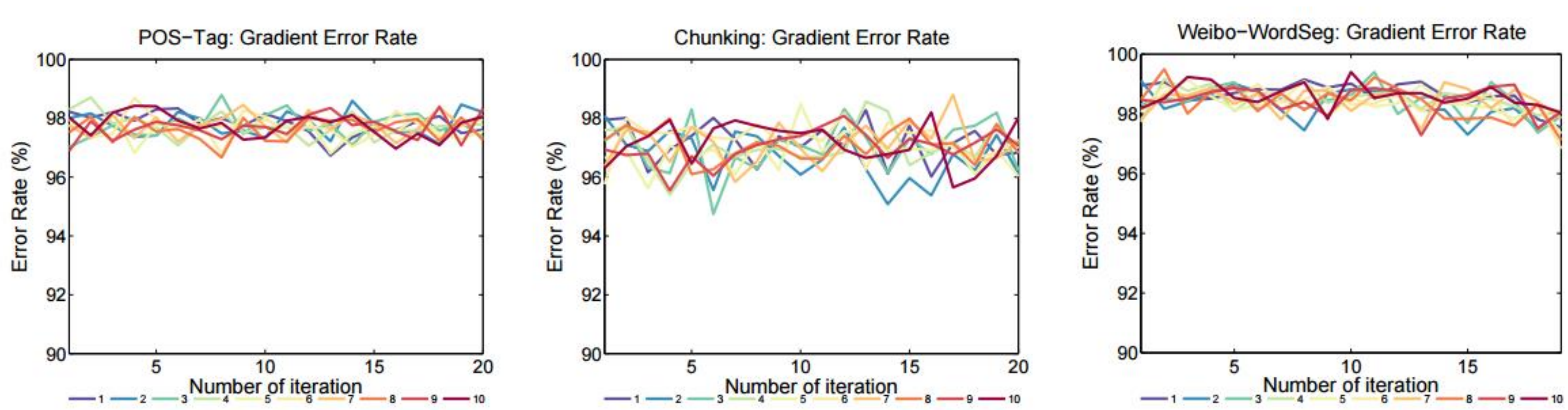


Experimental observations

- **Gradient error is inevitable in asynchronous training of neural networks** in real-world tasks

So asynchronous parallel learning is doomed for neural networks?

→No, still this problem can be solved



A Recent Solution

- **An asynchronous parallel learning solution for fast training of neural networks** Proposed by Sun (COLING 2016)
  - Asynchronous Parallel Learning with Gradient Error (AsynGrad)
- **Algorithm**

Algorithm 1 *AsynGrad*: Asynchronous Parallel Learning with Gradient Error

Input: model weights $\boldsymbol{w}$, training set S of m samples
Run k threads in parallel with share memory, and procedure of each thread is as follows:
repeat
  Get a sample $\boldsymbol{z}$ uniformly at random from S
  Get the update term $\boldsymbol{s_z}(\boldsymbol{w})$, which is computed as $\nabla f_{\boldsymbol{z}}(\boldsymbol{w})$ but usually contains error
  Update $\boldsymbol{w}$ such that $\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma \boldsymbol{s_z}(\boldsymbol{w})$
until Convergence
return $\boldsymbol{w}$

X. Sun. Asynchronous Parallel Learning for Neural Networks and Structured Models with Dense Features. COLING 2016.

Theoretical Analysis

- Can AsynGrad still converge with gradient errors?

Theorem 1 (*AsynGrad* convergence and convergence rate). *With the conditions (4), (5), (6), (7), let $\epsilon > 0$ be a targe... imum). Let τ denote ...*

(8)

where $\boldsymbol{w}$ ... z such that $\boldsymbol{s}(\boldsymbol{w}) = \mathbb{E}$ ...

(9)

where we ...

Even though there are gradient errors, AsynGrad does not diverge… it still converges near the optimum with a small distance, when the errors are bounded.

→ The assumptions usually hold in the final convergence region

→ Confirmed by real-world experiments

$$t \doteq \frac{\beta q \kappa^2 \log(q a_0/\epsilon)}{c(c\epsilon - 2\tau q)} \tag{10}$$

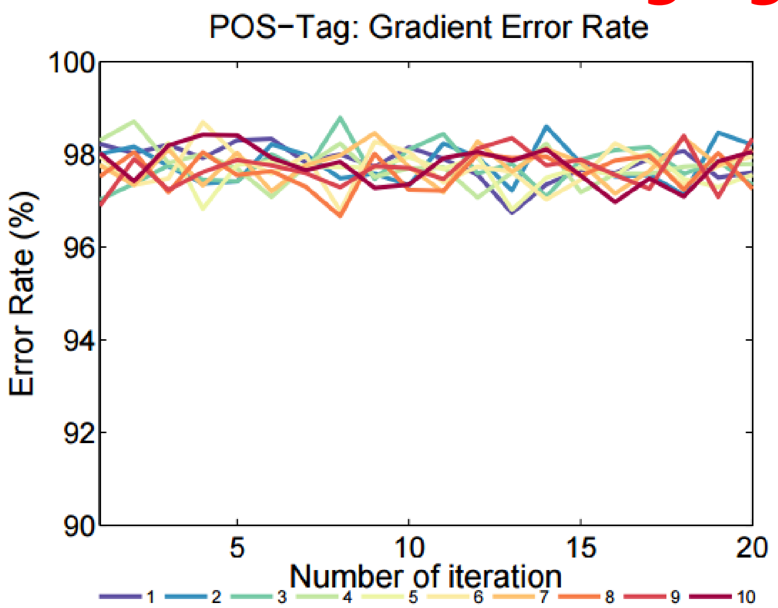
where $\doteq$ means ceil-rounding of a real value to an integer, and a_0 is the initial distance such that $a_0 = ||\boldsymbol{w}_0 - \boldsymbol{w}^*||^2$. Then, after t updates of $\boldsymbol{w}$, AsynGrad *converges towards the optimum such that $\mathbb{E}[f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)] \leq \epsilon$, as far as the gradient errors are bounded such that*

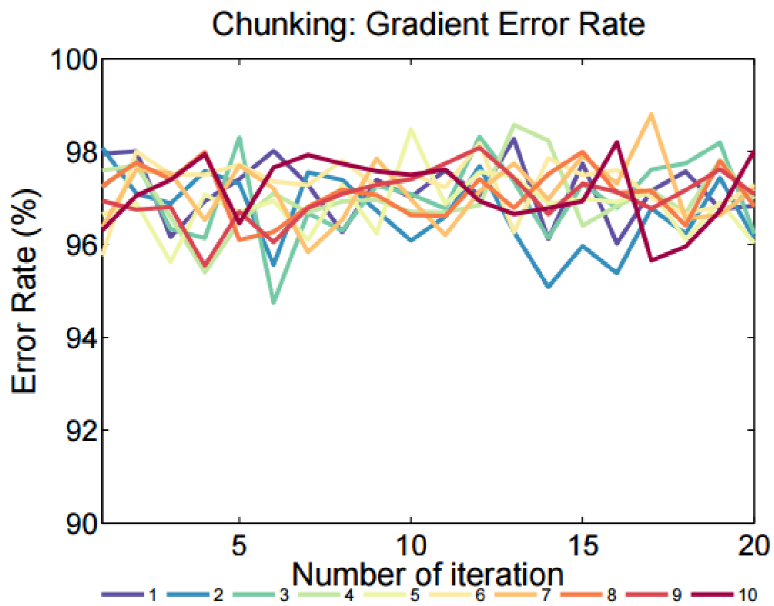
bounded gradient errors

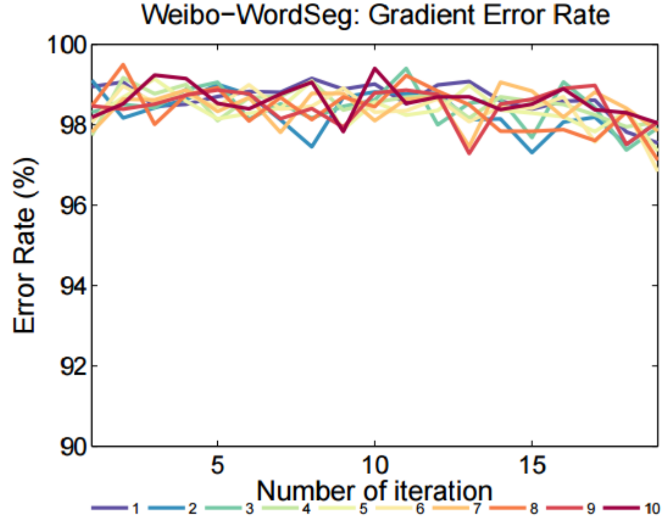
$$\boxed{\tau \leq \frac{c\epsilon}{2q}} \tag{11}$$

Experiments on LSTM

Experiments show that AsynGrad still converge even with a high gradient error rate





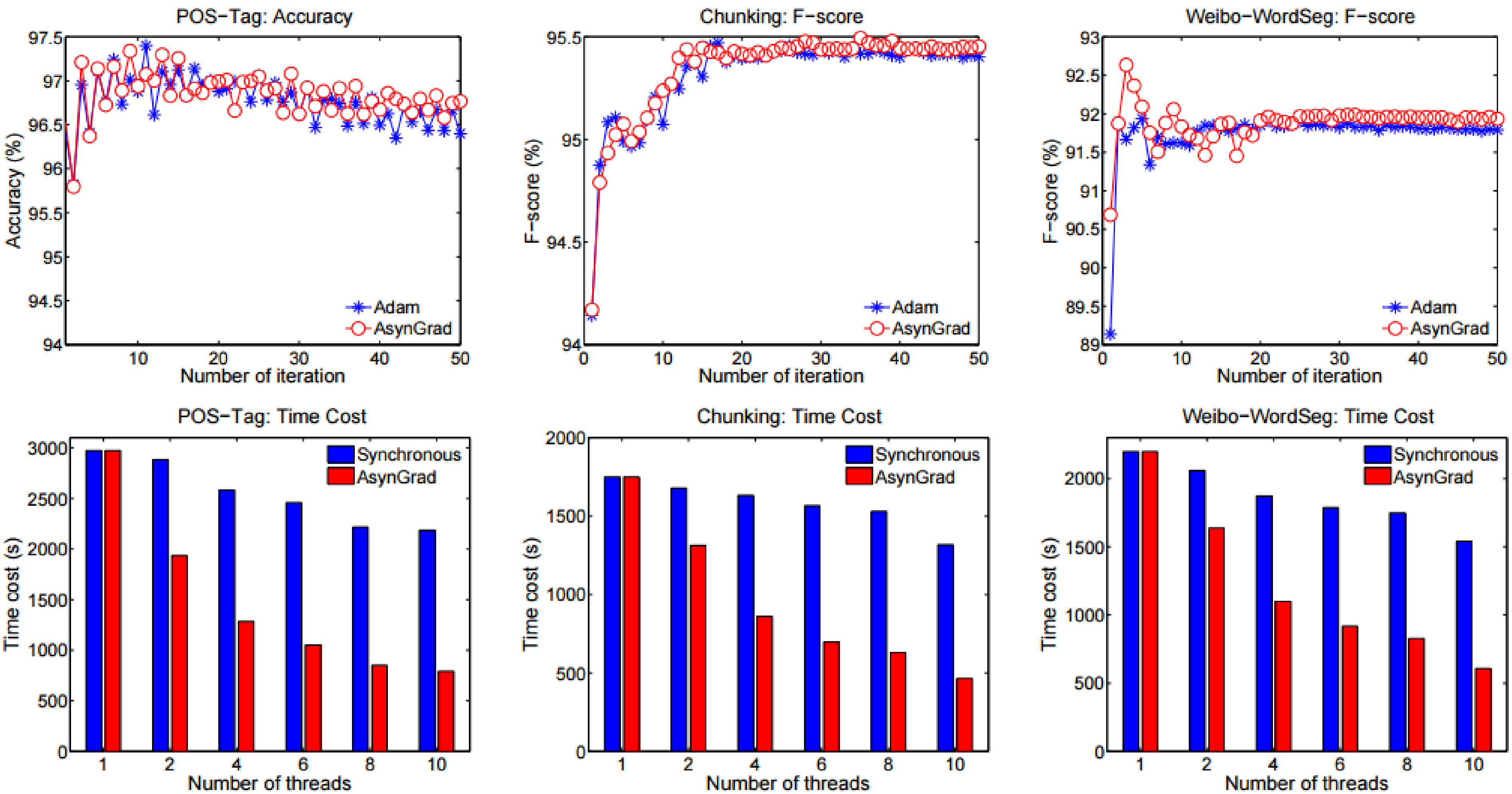


Experiments on LSTM

- No loss on accuracy/F-score

AsynGrad

- With substantially faster training speed



AsynGrad

- **Gradient errors are common and inevitable in asynchronous training of dense feature models**
- **AsynGrad tolerates gradient errors**
  - For dense feature models, such as neural networks and dense-CRF
  - With faster speed and no loss on accuracy
- **An alternative learning approach for large-scale structured predictions using neural networks**

Thanks!

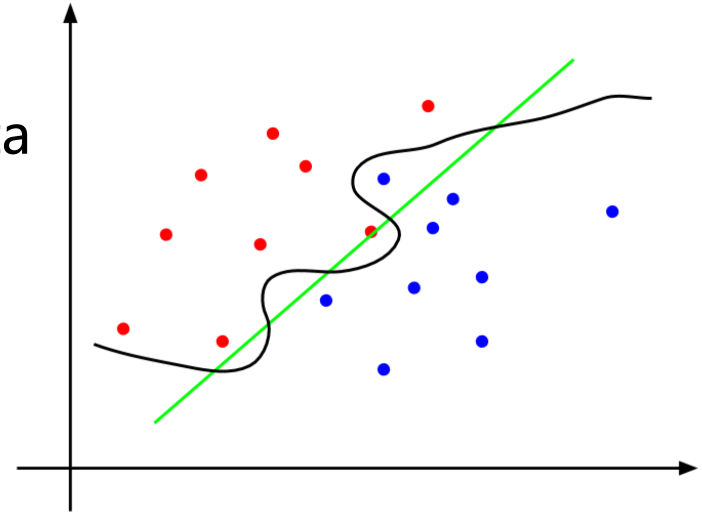
Any questions until now?

Outline

- Introduction
- Model
  - Conventional Model
  - Latent Model
  - Neural Model
- Regularization
- How-tos

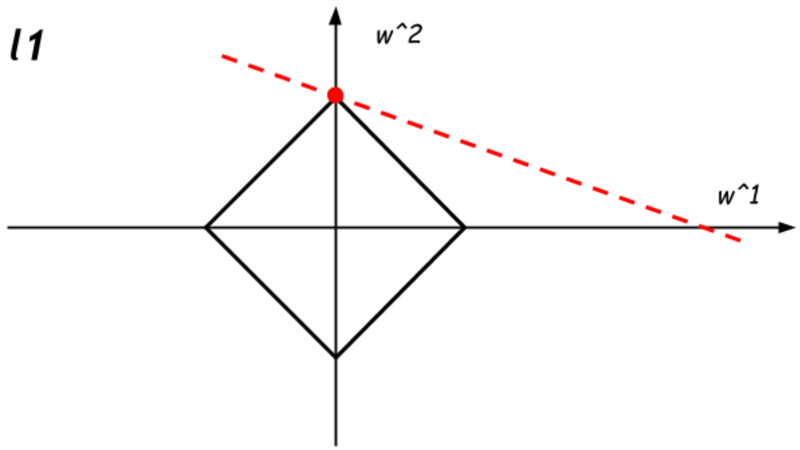
Overfitting

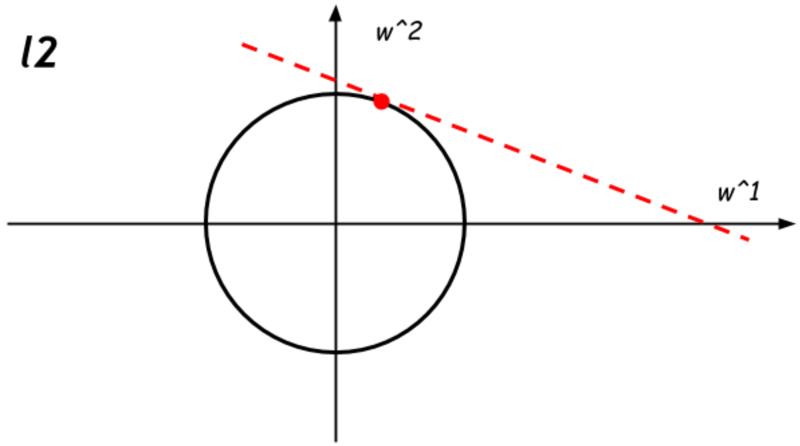
- **Models for large-scale structured prediction often suffer from overfitting**
- **Overfitting**
  - Low error rate in training set
  - High error rate in test set
- **Why overfitting?**
  - Complex model
  - Too many parameters, too little data
- **How to deal with?**
  - Penalty
  - Reduce complexity



Weight Regularization

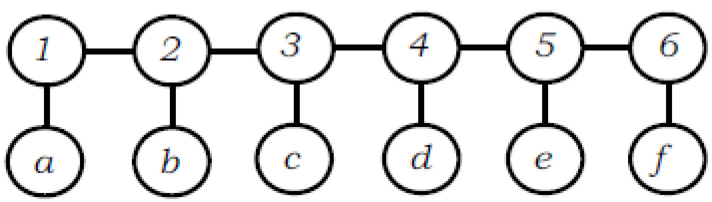
- **Penalty parameters in loss function**
  - $\min_{w} loss(x, y, w) + \lambda regularizer(w)$
- **L1 regularizer**
  - $regularizer = \lambda\|w\|$
  - $\frac{d}{dw_j} regularizer = \lambda sign(w_j)$
- **L2 regularizer**
  - $regularizer = \frac{\lambda}{2}\|w\|^2$
  - $\frac{d}{dw_j} regularizer = \lambda w_j$

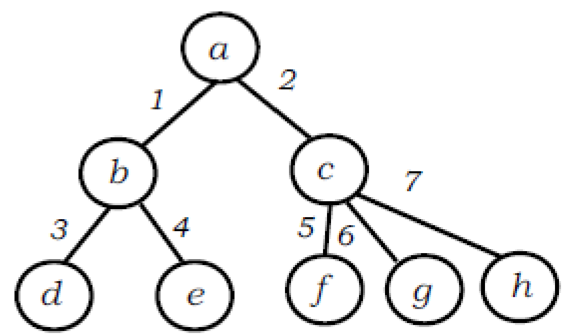




Structure Regularization

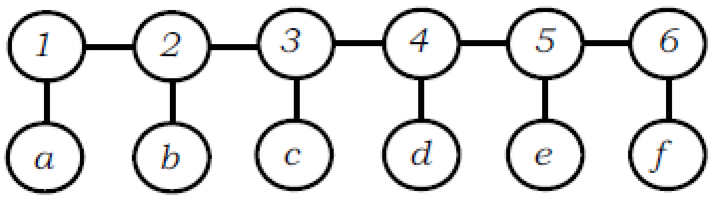
- **Motivation**
  - Reduce complexity of model structure
- **Structure regularization** (Sun. NIPS 2014)
  - Complex structure -> Simple structure
  - Faster
  - Easy to implement
  - Theoretical guarantee

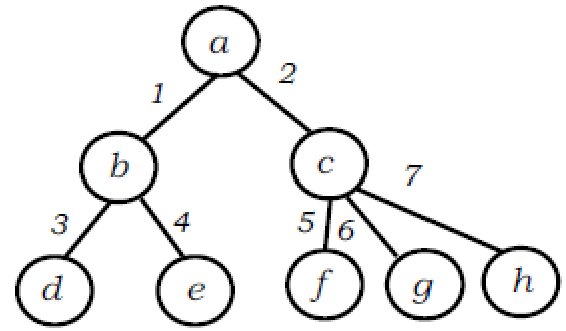




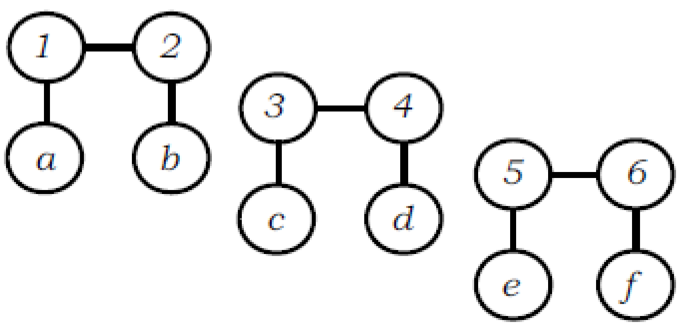
Illustration

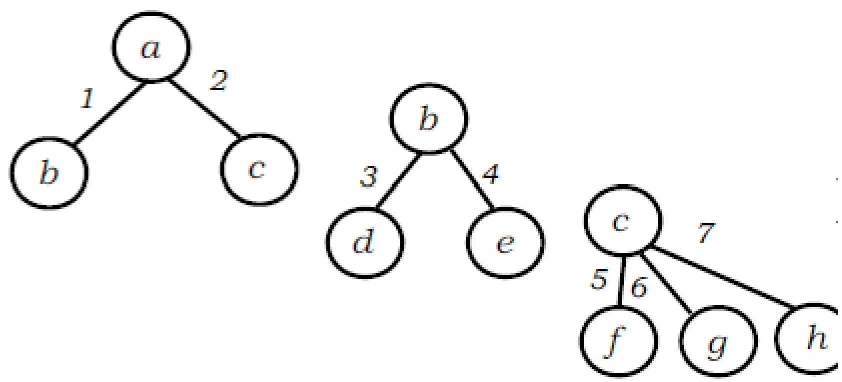
- **Complex structures (high complexity)**





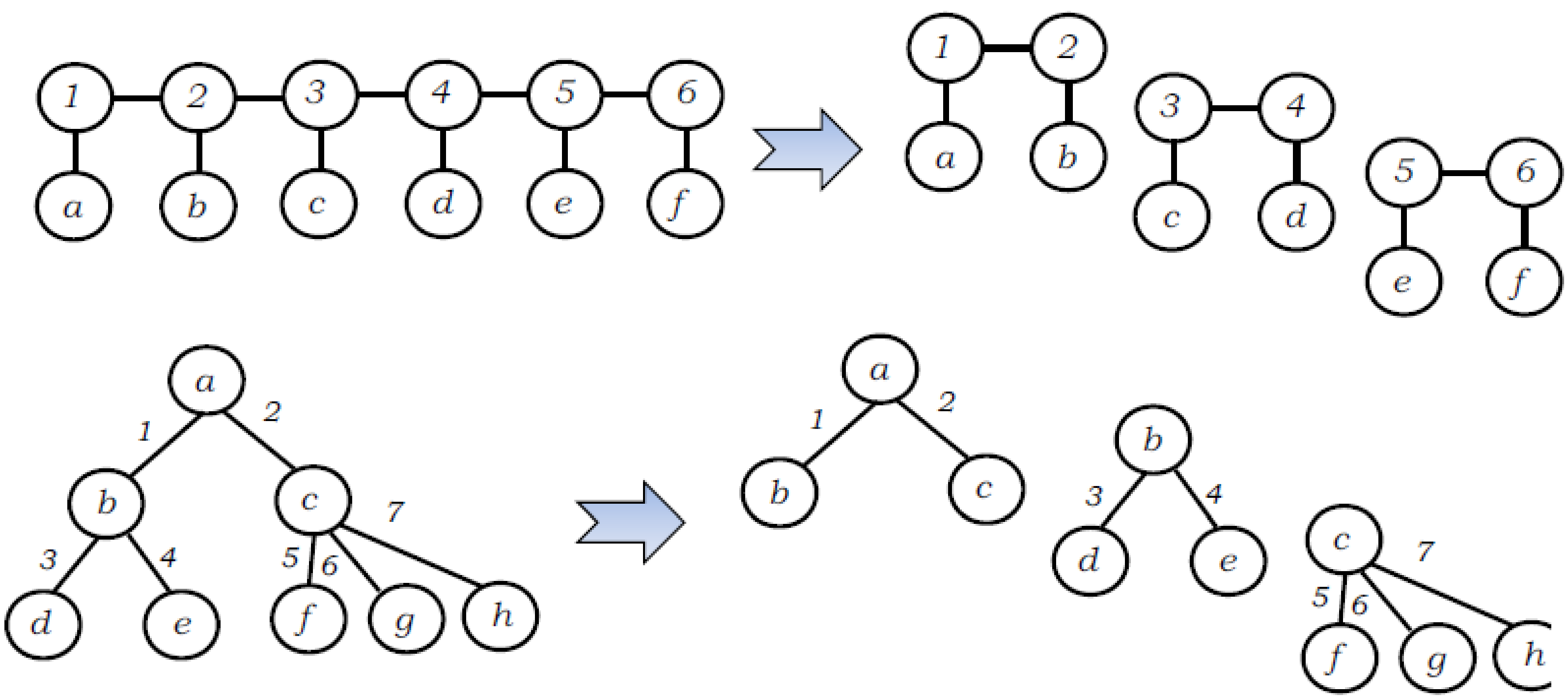
- **Simple structures (low complexity)**



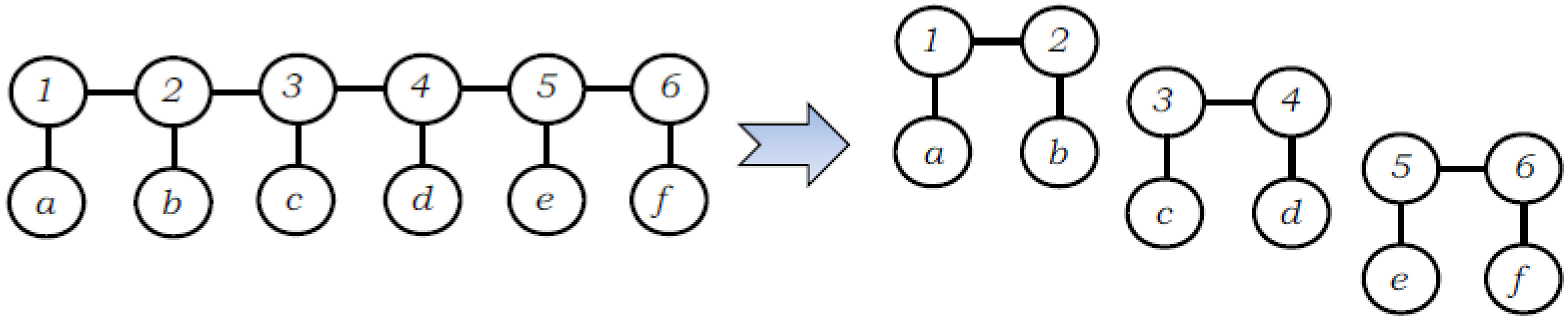


Structure Regularization

- **Structure regularization (SR) can find good complexity**
  - Simply split the structures!
  - Can (almost) be seen as a preprocessing step of the training data



Structure Regularization

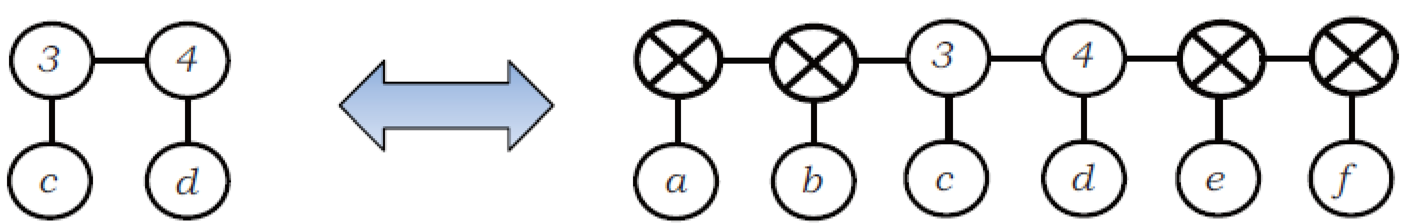


- **Will the split causes feature loss? – loss of long distance features?**

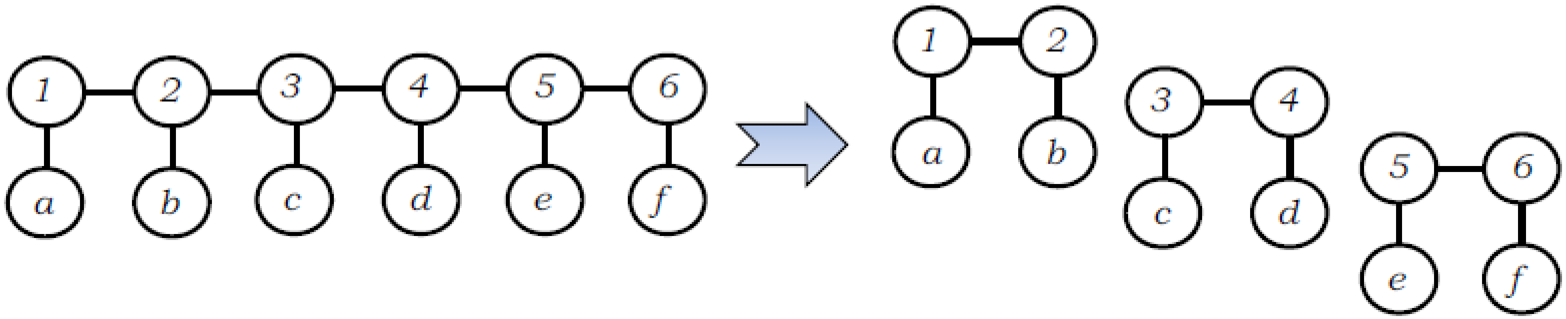
No loss of any (long distance) features

→ We can first extract features, then split the structures

→ Or, by simply copying observations to mini-samples, i.e., the split is only on tag-structures, like this:



Structure Regularization



- Is structure regularization also required for test data?

No, no use of SR for testing data (in current implementation & experiments)
→Like other regularization methods, SR is only for the training
→i.e., No SR on the test stage (no decomposition of test samples)!

Structure Regularization

Structure & weight regularization

$$R_{\alpha,\lambda}(G_S) \triangleq R_\alpha(G_S) + N_\lambda(G_S)$$

Algorithm 1 Training with structure regularization

Input: model weights $\boldsymbol{w}$, training set S, structure regularization strength α
repeat
  $S' \leftarrow \emptyset$
  for $i = 1 \rightarrow m$ **do**
    Randomly decompose $\boldsymbol{z}_i \in S$ into mini-samples $N_\alpha(\boldsymbol{z}_i) = \{\boldsymbol{z}_{(i,1)}, \ldots, \boldsymbol{z}_{(i,\alpha)}\}$
    $S' \leftarrow S' \cup N_\alpha(\boldsymbol{z}_i)$
  end for
  for $i = 1 \rightarrow |S'|$ **do**
    Sample $\boldsymbol{z}'$ uniformly at random from S', with gradient $\nabla g_{\boldsymbol{z}'}(\boldsymbol{w})$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla g_{\boldsymbol{z}'}(\boldsymbol{w})$
  end for
until Convergence
return $\boldsymbol{w}$

The implementation is very simple

Theoretical Analysis: Overfitting Risk

Theorem 4 (Generalization vs. structure regularization) *Let the structured prediction objective function of G be penalized by structure regularization with factor $\alpha \in [1, n]$ and L_2 weight regularization with factor λ, and the penalized function has a minimizer f:*

$$f = \underset{g \in \mathcal{F}}{\operatorname{argmin}} R_{\alpha,\lambda}(g) = \underset{g \in \mathcal{F}}{\operatorname{argmin}} \Big(\frac{1}{mn} \sum_{j=1}^{m\alpha} \mathcal{L}_\tau(g, \boldsymbol{z}'_j) + \frac{\lambda}{2} ||g||_2^2 \Big) \tag{8}$$

Assume the point-wise loss ℓ_τ is convex and differentiable, and is bounded by $\ell_\tau(f, \boldsymbol{z}, k) \leq \gamma$. Assume $f(\boldsymbol{x}, k)$ is ρ-admissible. Let a local feature value be bounded by v such that $\boldsymbol{x}_{(k,q)} \leq v$ for $q \in \{1, \ldots, d\}$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draw of the training set S, the generalization risk $R(f)$ is bounded by

$$R(f) \leq R_e(f) + \boxed{\frac{2d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \Big(\frac{(4m-2)d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \gamma \Big) \sqrt{\frac{\ln \delta^{-1}}{2m}}} \tag{9}$$

Expected risk (risk on test data)

Empirical risk (risk on training data)

Overfitting risk (risk of overfitting from training data to test data)

Theoretical Analysis: Overfitting Risk

Theorem 4 (Generalization vs. structure regularization) *Let the structured prediction objective function of G be penalized by structure regularization with factor $\alpha \in [1, n]$ and L_2 weight regularization with factor λ, and the penalized function has a minimizer f:*

$$f = \underset{g \in \mathcal{F}}{\operatorname{argmin}} R_{\alpha,\lambda}(g) = \underset{g \in \mathcal{F}}{\operatorname{argmin}} \Big(\frac{1}{mn} \sum_{j=1}^{m\alpha} \mathcal{L}_\tau(g, \boldsymbol{z}'_j) + \frac{\lambda}{2} ||g||_2^2 \Big) \tag{8}$$

Assume the point-wise loss ℓ_τ is convex and differentiable, and is bounded by $\ell_\tau(f, \boldsymbol{z}, k) \leq \gamma$. Assume $f(\boldsymbol{x}, k)$ is ρ-admissible. Let a local feature value be bounded by v such that $\boldsymbol{x}_{(k,q)} \leq v$ for $q \in \{1, \ldots, d\}$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draw of the training set S, the generalization risk $R(f)$ is bounded by

$$\boxed{R(f) \leq R_e(f) + \frac{2d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \Big(\frac{(4m-2)d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \gamma \Big) \sqrt{\frac{\ln \delta^{-1}}{2m}}} \tag{9}$$

Complexity of structure (nodes of a training sample with structured dependencies)

→ Complex structure leads to higher overfitting risk

Theoretical Analysis: Overfitting Risk

Theorem 4 (Generalization vs. structure regularization) *Let the structured prediction objective function of G be penalized by structure regularization with factor $\alpha \in [1, n]$ and L_2 weight regularization with factor λ, and the penalized function has a minimizer f:*

$$f = \operatorname*{argmin}_{g \in \mathcal{F}} R_{\alpha,\lambda}(g) = \operatorname*{argmin}_{g \in \mathcal{F}} \Big(\frac{1}{mn} \sum_{j=1}^{m\alpha} \mathcal{L}_\tau(g, \boldsymbol{z}'_j) + \frac{\lambda}{2} ||g||_2^2 \Big) \tag{8}$$

Assume the point-wise loss ℓ_τ is convex and differentiable, and is bounded by $\ell_\tau(f, \boldsymbol{z}, k) \leq \gamma$. Assume $f(\boldsymbol{x}, k)$ is ρ-admissible. Let a local feature value be bounded by v such that $\boldsymbol{x}_{(k,q)} \leq v$ for $q \in \{1, \ldots, d\}$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draw of the training set S, the generalization risk $R(f)$ is bounded by

$$\boxed{R(f) \leq R_e(f) + \frac{2d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \Big(\frac{(4m-2)d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \gamma \Big) \sqrt{\frac{\ln \delta^{-1}}{2m}}} \tag{9}$$

Strength of structure regularization (strength of decomposition)

→ Stronger SR leads to reduction of overfitting risk

Theoretical Analysis: Overfitting Risk

Theorem 4 (Generalization vs. structure regularization) *Let the structured prediction objective function of G be penalized by structure regularization with factor $\alpha \in [1, n]$ and L_2 weight regularization with factor λ, and the penalized function has a minimizer f:*

$$f = \operatorname*{argmin}_{g \in \mathcal{F}} R_{\alpha,\lambda}(g) = \operatorname*{argmin}_{g \in \mathcal{F}} \Big(\frac{1}{mn} \sum_{j=1}^{m\alpha} \mathcal{L}_\tau(g, \boldsymbol{z}'_j) + \frac{\lambda}{2} ||g||_2^2 \Big) \tag{8}$$

Assume the point-wise loss ℓ_τ is convex and differentiable, and is bounded by $\ell_\tau(f, \boldsymbol{z}, k) \leq \gamma$. Assume $f(\boldsymbol{x}, k)$ is ρ-admissible. Let a local feature value be bounded by v such that $\boldsymbol{x}_{(k,q)} \leq v$ for $q \in \{1, \dots, d\}$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draw of the training set S, the generalization risk $R(f)$ is bounded by

$$\boxed{R(f) \leq R_e(f) + \frac{2d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \Big(\frac{(4m-2)d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \gamma \Big) \sqrt{\frac{\ln \delta^{-1}}{2m}}} \tag{9}$$

Number of training samples

→ More training samples leads to reduction of overfitting risk

Theoretical Analysis: Overfitting Risk

Theorem 4 (Generalization vs. structure regularization) *Let the structured prediction objective function of G be penalized by structure regularization with factor $\alpha \in [1, n]$ and L_2 weight regularization with factor λ, and the penalized function has a minimizer f:*

$$f = \operatorname*{argmin}_{g \in \mathcal{F}} R_{\alpha,\lambda}(g) = \operatorname*{argmin}_{g \in \mathcal{F}} \Big(\frac{1}{mn} \sum_{j=1}^{m\alpha} \mathcal{L}_\tau(g, \boldsymbol{z}'_j) + \frac{\lambda}{2} ||g||_2^2 \Big) \quad (8)$$

Assume the point-wise loss ℓ_τ is convex and differentiable, and is bounded by $\ell_\tau(f, \boldsymbol{z}, k) \leq \gamma$. Assume $f(\boldsymbol{x}, k)$ is ρ-admissible. Let a local feature value be bounded by v such that $\boldsymbol{x}_{(k,q)} \leq v$ for $q \in \{1, \ldots, d\}$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draw of the training set S, the generalization risk $R(f)$ is bounded by

$$R(f) \leq R_e(f) + \frac{2d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \Big(\frac{(4m-2)d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \gamma \Big) \sqrt{\frac{\ln \delta^{-1}}{2m}} \quad (9)$$

✓Conclusions from our analysis:

1. Complex structure → low empirical risk & high overfitting risk
2. Simple structure → high empirical risk & low overfitting risk

3. Need a balanced complexity of structures

Theoretical Analysis: Overfitting Risk

Theorem 4 (Generalization vs. structure regularization) *Let the structured prediction objective function of G be penalized by structure regularization with factor $\alpha \in [1, n]$ and L_2 weight regularization with factor λ, and the penalized function has a minimizer f:*

$$f = \underset{g \in \mathcal{F}}{\operatorname{argmin}} R_{\alpha,\lambda}(g) = \underset{g \in \mathcal{F}}{\operatorname{argmin}} \left(\frac{1}{mn} \sum_{j=1}^{m\alpha} \mathcal{L}_\tau(g, \boldsymbol{z}'_j) + \frac{\lambda}{2} ||g||_2^2 \right) \tag{8}$$

Assume the point-wise loss ℓ_τ is convex and differentiable, and is bounded by $\ell_\tau(f, \boldsymbol{z}, k) \leq \gamma$. Assume $f(\boldsymbol{x}, k)$ is ρ-admissible. Let a local feature value be bounded by v such that $\boldsymbol{x}_{(k,q)} \leq v$ for $q \in \{1, \ldots, d\}$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draw of the training set S, the generalization risk $R(f)$ is bounded by

$$\boxed{R(f) \leq R_e(f) + \frac{2d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \left(\frac{(4m-2)d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \gamma \right) \sqrt{\frac{\ln \delta^{-1}}{2m}}} \tag{9}$$

- In other words, more intuitively:
  1. Too complex structure → high accuracy on training + very easy to overfit → low accuracy on testing
  2. Too simple structure → very low accuracy on training + not easy to overfit → low accuracy on testing

  Proper structure → good accuracy on training + not easy to overfit → high accuracy on testing

Theoretical Analysis: Overfitting Risk

Theorem 4 (Generalization vs. structure regularization) *Let the structured prediction objective function of G be penalized by structure regularization with factor $\alpha \in [1, n]$ and L_2 weight regularization with factor λ, and the penalized function has a minimizer f:*

$$f = \operatorname*{argmin}_{g \in \mathcal{F}} R_{\alpha,\lambda}(g) = \operatorname*{argmin}_{g \in \mathcal{F}} \Big(\frac{1}{mn} \sum_{j=1}^{m\alpha} \mathcal{L}_\tau(g, \boldsymbol{z}'_j) + \frac{\lambda}{2} ||g||_2^2 \Big) \tag{8}$$

Assume the point-wise loss ℓ_τ is convex and differentiable, and is bounded by $\ell_\tau(f, \boldsymbol{z}, k) \leq \gamma$. Assume $f(\boldsymbol{x}, k)$ is ρ-admissible. Let a local feature value be bounded by v such that $\boldsymbol{x}_{(k,q)} \leq v$ for $q \in \{1, \ldots, d\}$. Then, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random draw of the training set S, the generalization risk $R(f)$ is bounded by

$$\boxed{R(f) \leq R_e(f) + \frac{2d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \Big(\frac{(4m-2)d\tau^2\rho^2v^2n^2}{m\lambda\alpha} + \gamma \Big) \sqrt{\frac{\ln \delta^{-1}}{2m}}} \tag{9}$$

1. Simple structure → low overfitting risk & high empirical risk
2. Complex structure → high overfitting risk & low empirical risk
3. **Need a balanced complexity of structures**

Some intuition in the proof (as in the full version paper):
1) The decomposition can improve **stability**
2) Better stability leads to better **generalization** (less overfitting)

Theoretical Analysis: Learning Speed

Proposition 5 (Convergence rates vs. structure regularization) *With the aforementioned assumptions, let the SGD training have a learning rate defined as $\eta = \frac{c\epsilon\beta\alpha^2}{q\kappa^2 n^2}$, where $\epsilon > 0$ is a convergence tolerance value and $\beta \in (0, 1]$. Let t be a integer satisfying*

$$\boxed{t \geq \frac{q\kappa^2 n^2 \log(qa_0/\epsilon)}{\epsilon\beta c^2\alpha^2}} \tag{15}$$

where n and $\alpha \in [1, n]$ is like before, and a_0 is the initial distance which depends on the initialization of the weights $\boldsymbol{w}_0$ and the minimizer $\boldsymbol{w}^*$, i.e., $a_0 = ||\boldsymbol{w}_0 - \boldsymbol{w}^*||^2$. Then, after t updates of $\boldsymbol{w}$ it converges to $\mathbb{E}[g(\boldsymbol{w}_t) - g(\boldsymbol{w}^*)] \leq \epsilon$.

- **SR also with faster speed**

(a by-product of simpler structures)

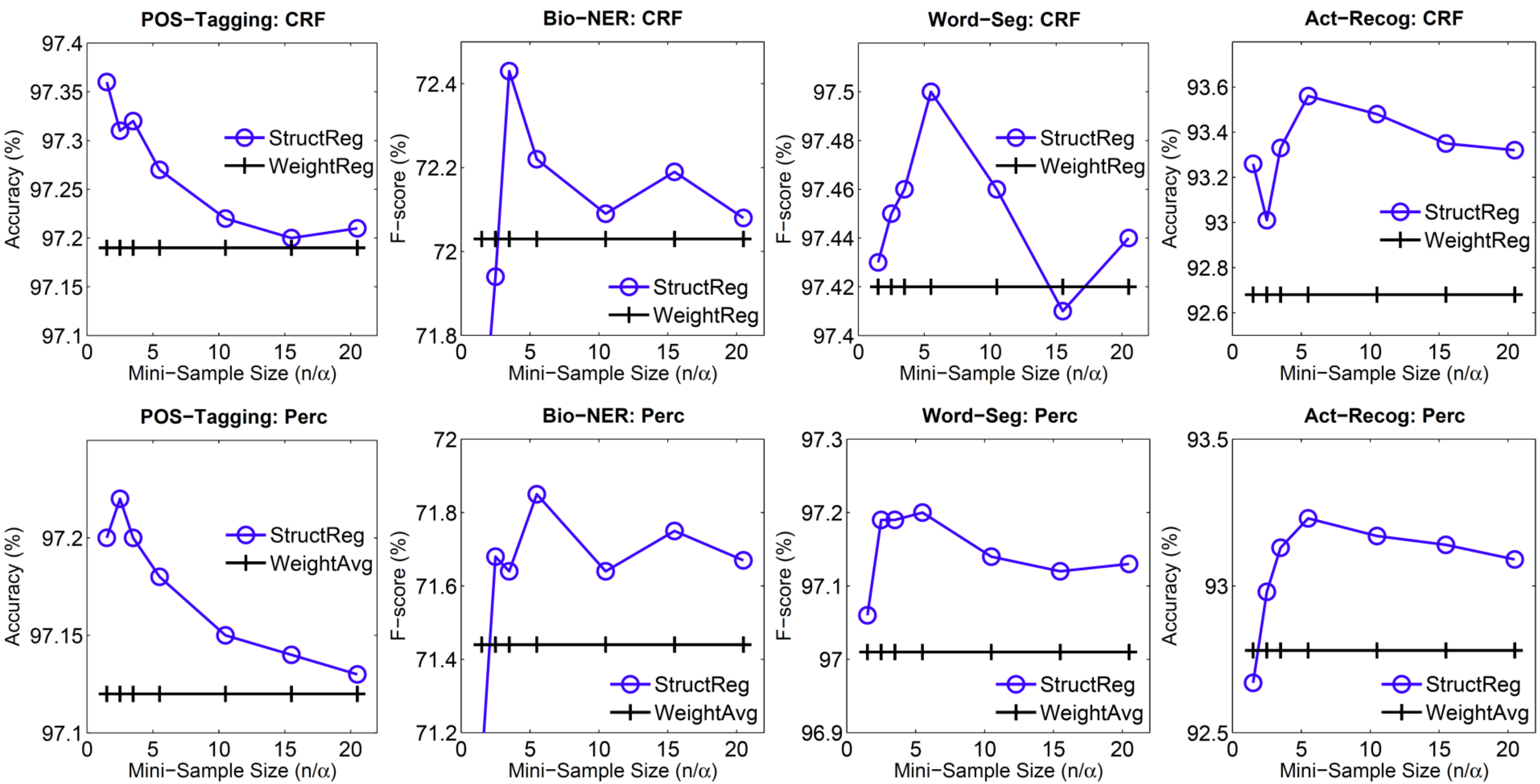
✓ **using structure regularization can quadratically accelerate the convergence rate**

Some Advantages

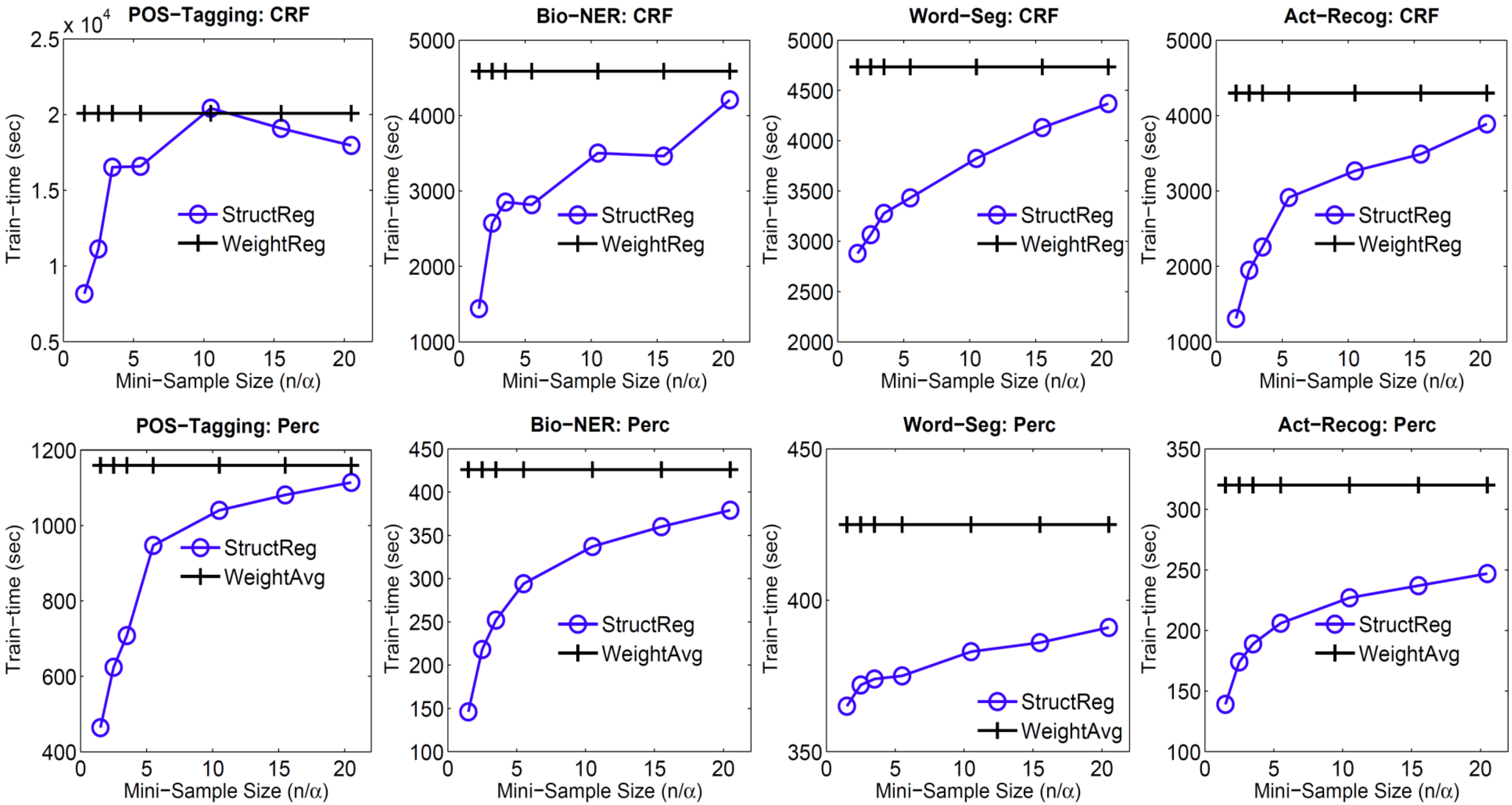
- **If the original obj. function is convex, can still keep the convexity of the objective function**
- **No conflict with the weight regularization**
  - E.g, L2, and/or L1 regularization
- **General purpose and model-independent (because act like a preprocessing step)**
  - E.g., can be used for different types of models, including CRFs, perceptrons, & neural networks

Experiments-1: Accuracy

State-of-the-art scores on competitive tasks



Experiments-2 : Learning Speed



- **Also with faster speed**

(a by-product of simpler structures)

For Large-scale Structured Prediction

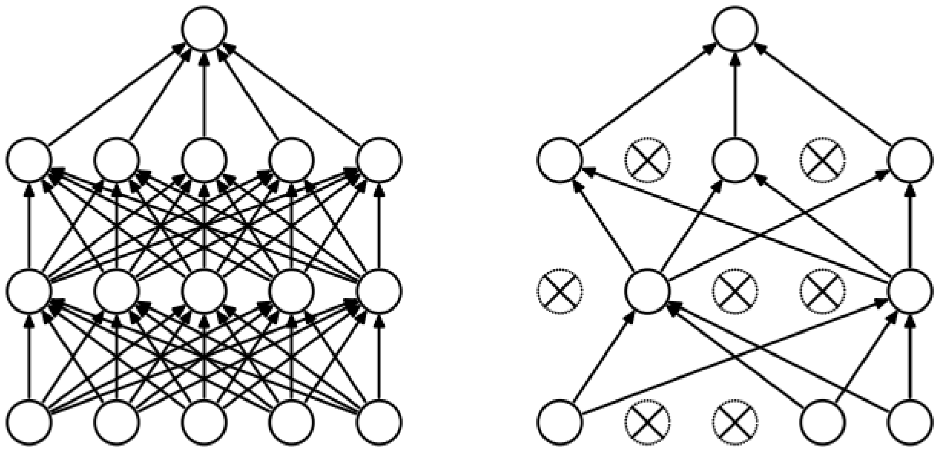
- Question: Is structure complexity matters in structured prediction?
  - Theoretical analysis to the question
    - 1) Yes it matters
    - 2) High complexity of structures → high overfitting risk
    - 3) Low complexity → high empirical risk
    - 4) We need to find an optimal complexity of structures
  - Proposed a solution
    - Split the original structure to find the optimal complexity
    - Better accuracies in real tasks, & faster (a by-product)

This work is published at NIPS 2014:

Xu Sun. Structure Regularization for Structured Prediction. In Advances in Neural Information Processing Systems (NIPS). 2402-2410. 2014

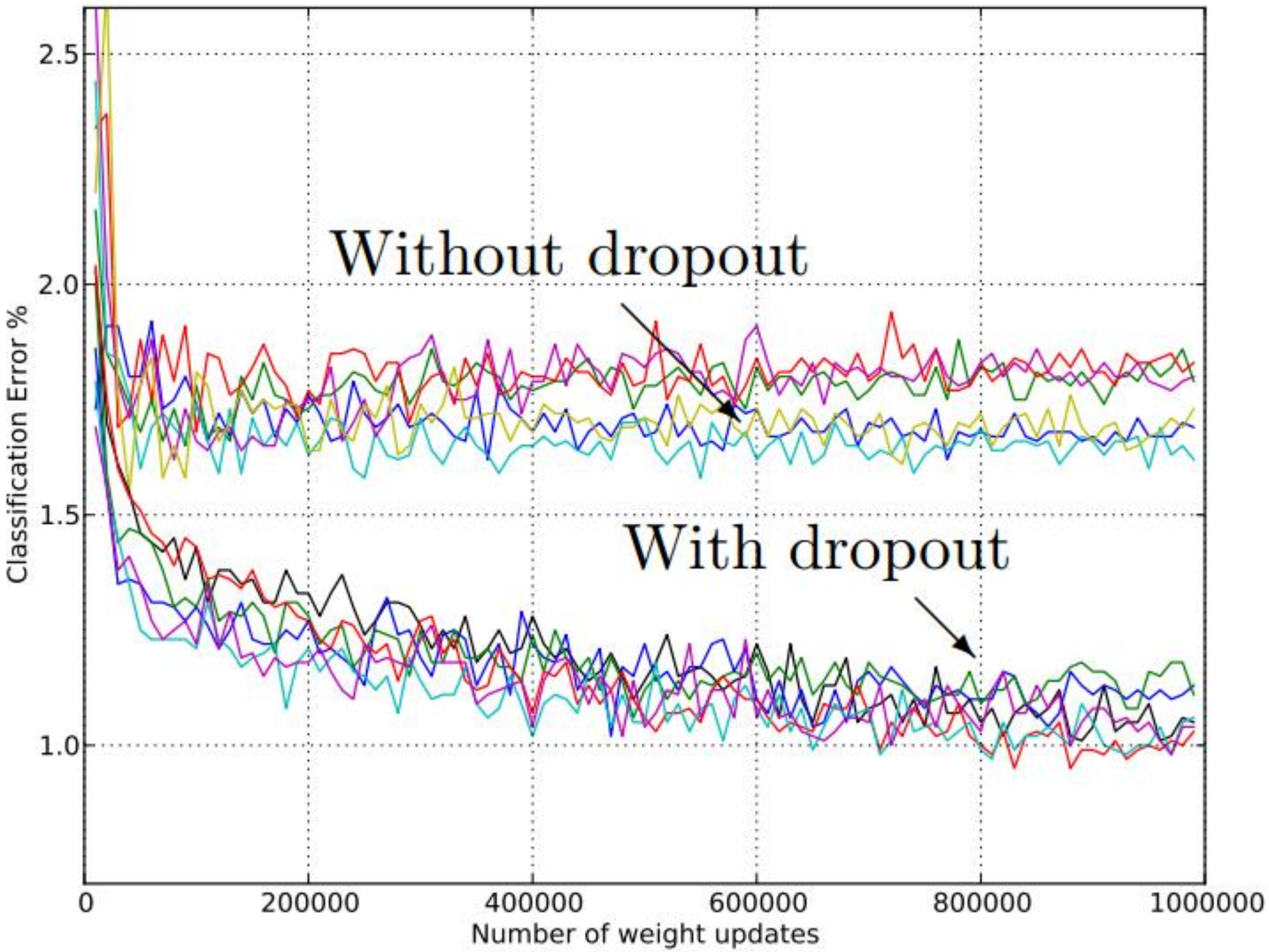
Drop Out

- **Proposed by** Srivastava et al.(2014)
- **Part of neurons do not participate in forward pass and backpropagation**
- **Only use in training**
- **Advantage**
  - Fewer parameters per training sample
  - Reduce training time


(a) Standard Neural Net (b) After applying dropout.

Drop Out

- **Experiment on MINST** (Srivastava, et al. JMLR 2014)



Drop Out

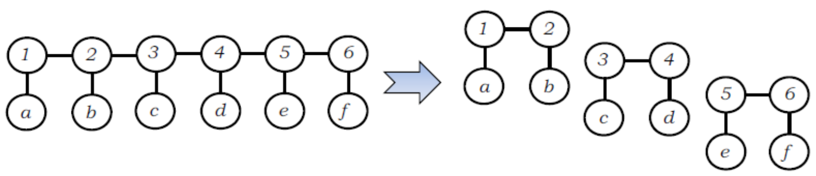
- **Experiment on Penn Tree Bank** (Zaremba, et al. 2015)
  - Language modeling measured by perplexity

Model	Validation set	Test set
A single model		
Pascanu et al. (2013)		107.5
Cheng et al.		100.0
non-regularized LSTM	120.7	114.5
Medium regularized LSTM	86.2	82.7
Large regularized LSTM	82.2	**78.4**
Model averaging		
Mikolov (2012)		83.5
Cheng et al.		80.6
2 non-regularized LSTMs	100.4	96.1
5 non-regularized LSTMs	87.9	84.1
10 non-regularized LSTMs	83.5	80.0
2 medium regularized LSTMs	80.6	77.0
5 medium regularized LSTMs	76.7	73.3
10 medium regularized LSTMs	75.2	72.0
2 large regularized LSTMs	76.9	73.6
10 large regularized LSTMs	72.8	69.5
38 large regularized LSTMs	71.9	**68.7**
Model averaging with dynamic RNNs and n-gram models		
Mikolov & Zweig (2012)		72.9

Discussion of Techniques

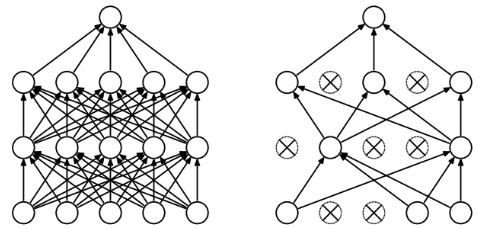
StrutReg

- decomposition of structure
- randomly



Drop Out

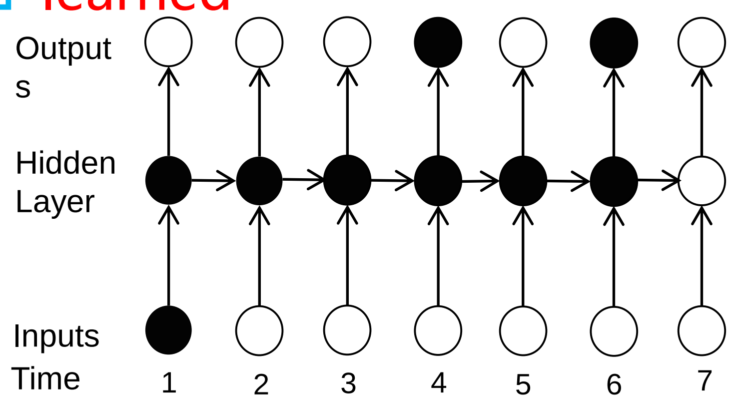
- deactivate neurons
- randomly



(a) Standard Neural Net (b) After applying dropout.

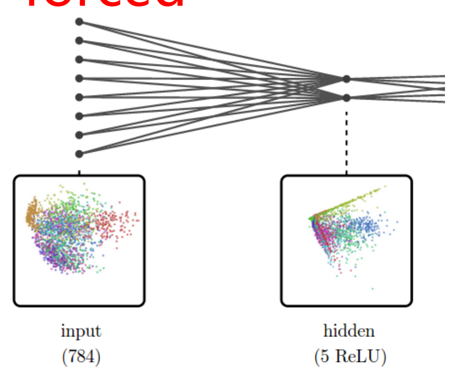
LSTM

- forget useless information
- learned



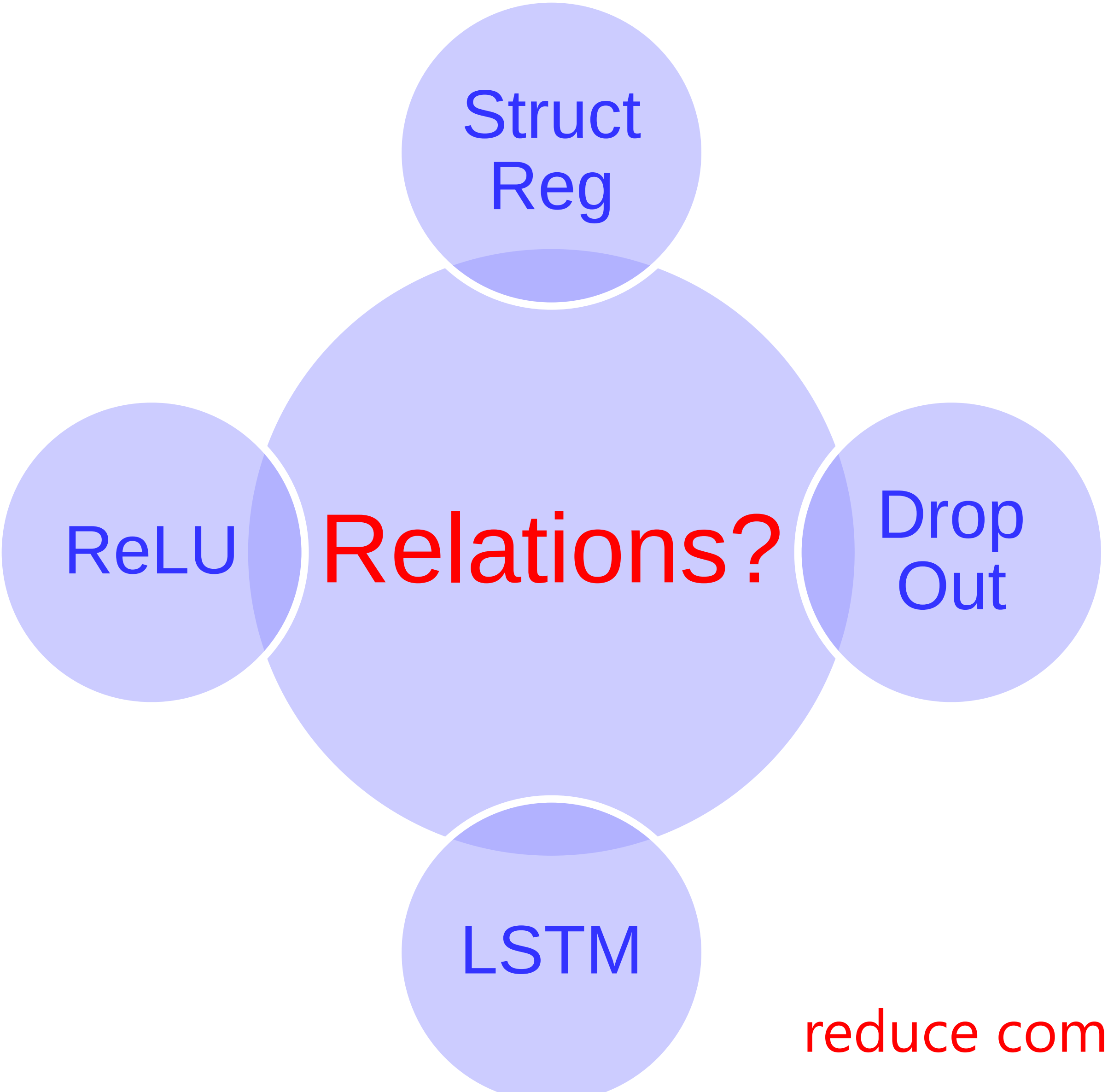
ReLU

- sparse activation
- forced



Picture credit: Christopher Olah

Discussion of Techniques



reduce complexity

Outline

- Introduction
- Model
  - Conventional Model
  - Latent Model
  - Neural Model
- Regularization
- How-tos

How to choose a model

- **Conventional models can achieve the state-of-the-arts results**
  - stable for production setup
- **Latent models are good for tasks short of annotations**
- **Neural models are promising**
  - especially for high-level tasks
    - sentiment analysis
    - summarization, composition, translation
  - many nice frameworks available
    - theano, torch, tensorflow
    - caffe, cntk

How to Choose an Optimizer

- **For models other than neural models**
  - Perceptron

    online, fast convergence
- **For neural models**
  - Mini-batch SGD

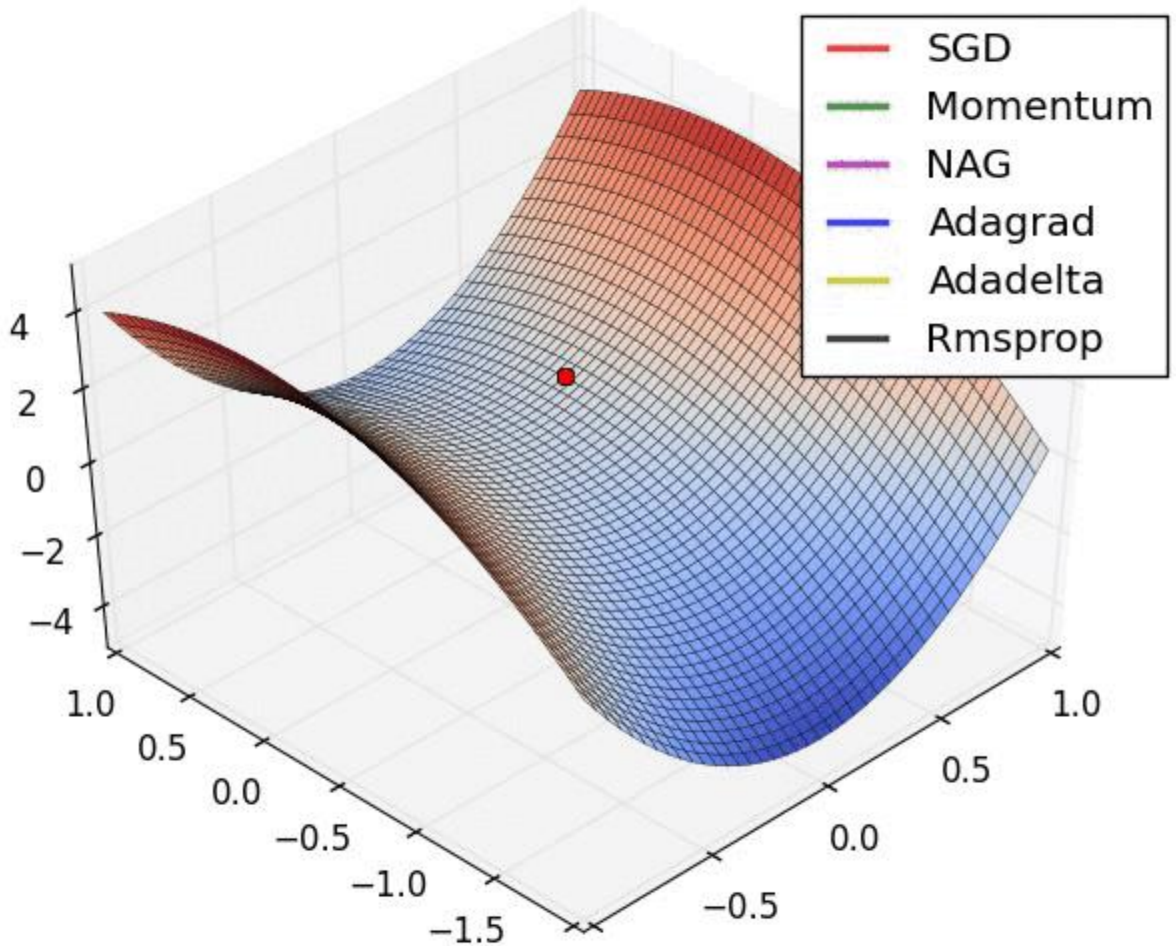
    parameter-scaling
    AdaGrad
    AdaDelta
    Adam
    RMSProp
    less tuning, fast

    momentum-based
    momentum
    Nesterov Accelerated Gradient (NAG)
    better results, more tuning

  - Using GPUs can lead to significant speed-ups, compared to use CPUs only

Visualization of Optimization Algorithms

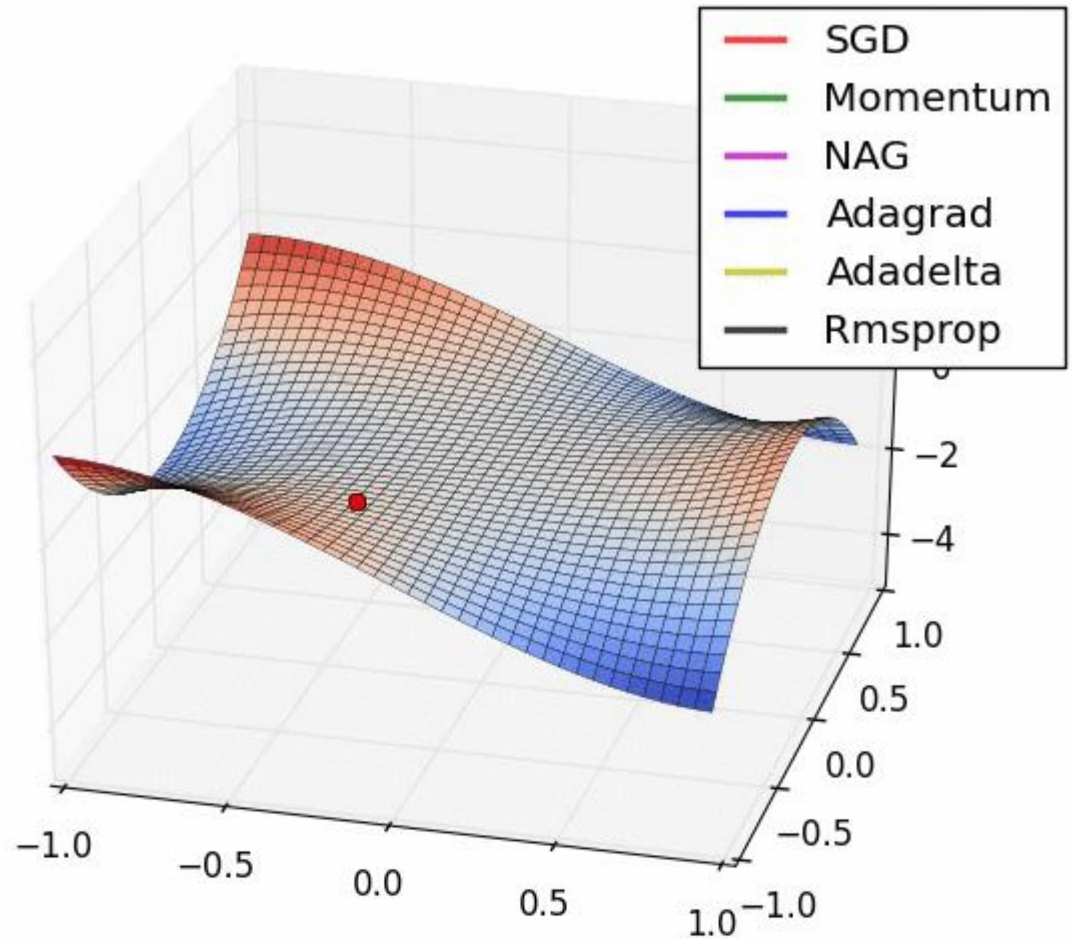
Long Valley



Picture from Alec Radford

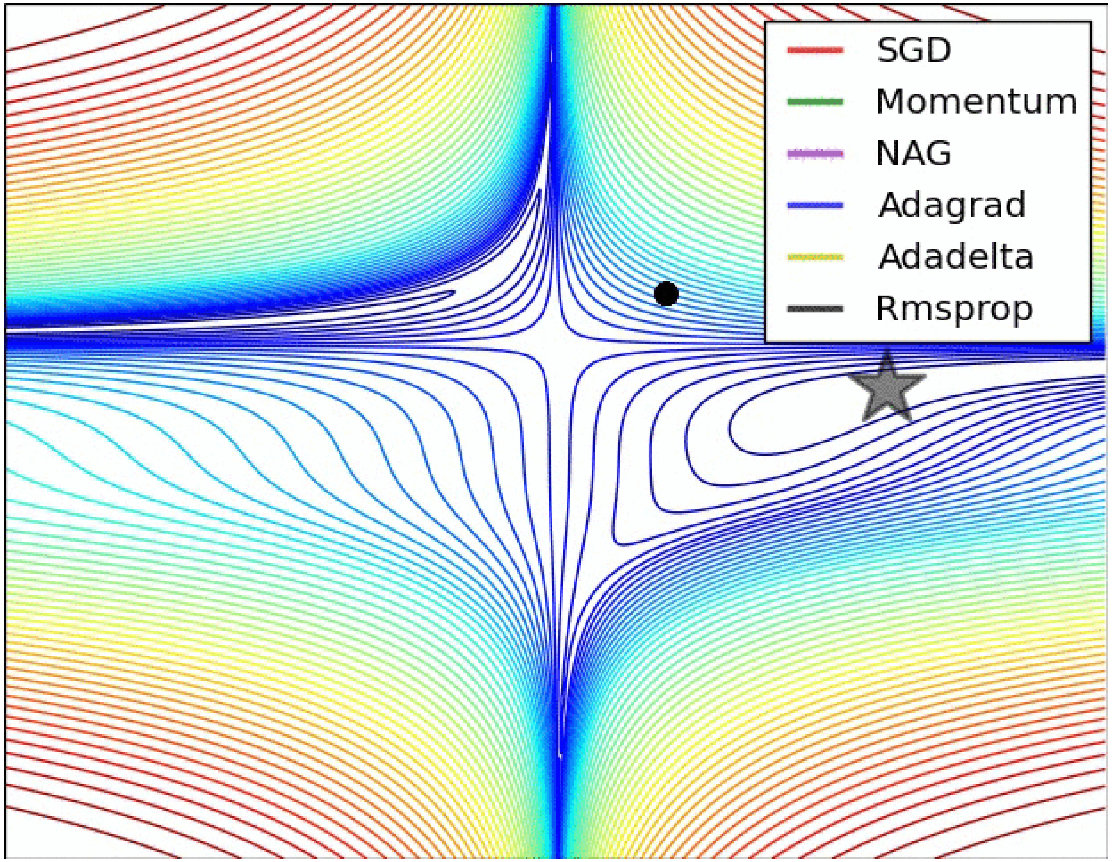
Visualization of Optimization Algorithms

Saddle Point



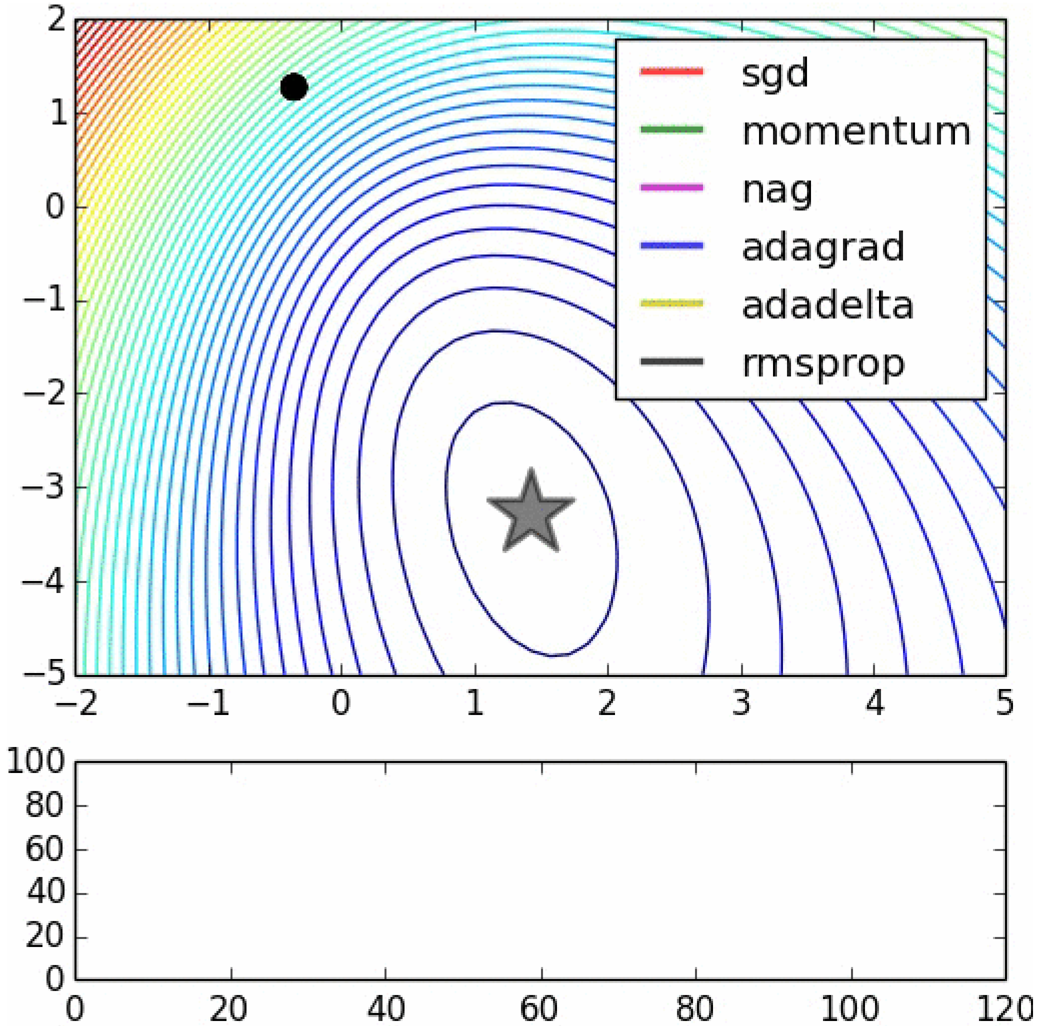
Picture from Alec Radford

Beale's Function



Visualization of Optimization Algorithms

Noisy Data



Picture from Alec Radford

Thanks!

Any Question?

References And Further Reading

1. A. Fader, L. Zettlemoyer, and O. Etzioni. Open question answering over curated and extracted knowledge bases. SIGKDD 2014.
2. A. Graves. Neural Networks. Supervised Sequence Labelling with Recurrent Neural Networks. Springer 2012.
3. A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. NIPS 2012.
4. A. Passos, V. Kumar, and A. McCallum. Lexicon Infused Phrase Embeddings for Named Entity Resolution. CoNLL 2014.
5. A. Radford. http://imgur.com/s25RsOr/. 2014.
6. A. Radford. Visualizing optimization algors. http://imgur.com/a/Hqolp/. 2014.
7. A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. EMNLP 1996.
8. A. Søgaard. Semisupervised condensed nearest neighbor for part-of-speech tagging. ACL-HLT 2011.
9. C. Olah. Understanding LSTM networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. 2015.
10. C. Olah. Visualizing representations: Deep learning and human beings. http://colah.github.io/posts/2015-01-Visualizing-Representations/. 2015.
11. C.D. Manning and H. Schütze. Foundations of Statistical Natural Language Processing. MIT Press 1999.
12. C.D. Manning and H. Schütze. Foundations of Statistical Natural Language Processing. MIT Press 1999.
13. C.D. Manning. Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? CICLing 2011.

References And Further Reading

14. D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. Globally normalized transition-based neural networks. arXiv:1603.06042.
15. D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.
16. D. Biber, S. Conrad, and R. Reppen. Corpus linguistics: Investigating language structure and use. Cambridge University Press 1998.
17. D. Britz. Recurrent neural networks tutorial, part 3 - Backpropagation through time and vanishing gradients. http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/. 2015.
18. D. Chen and C.D. Manning. A Fast and Accurate Dependency Parser using Neural Networks. EMNLP 2014.
19. D. Jurafsky and J.H. Martin. Speech and language processing. Prentice Hall 2008.
20. D. Klein and C.D. Manning. Accurate unlexicalized parsing. ACL 2003.
21. D. McClosky, E. Charniak, and M. Johnson. Effective Self-Training for Parsing. HLT-NAACL 2006.
22. E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. ACL 2005.
23. E. Fernandes, C. dos Santos, and R. Milidiú. Latent trees for coreference resolution. CL 2014.
24. F. Gers. Long short-term memory in recurrent neural networks. PhD diss., Universität Hannover 2001.
25. F. Sha and F. Pereira. Shallow parsing with conditional random fields. HLT-NAACL 2003.
26. F.V. Veen. The neural network zoo. http://www.asimovinstitute.org/neural-network-zoo/. 2016.

References And Further Reading

27. H. Zhou, Y. Zhang, S. Huang, and J. Chen. A neural probabilistic structured-prediction model for transition-based dependency parsing. ACL 2015.
28. H.L. Chieu. Named entity recognition with a maximum entropy approach. CoNLL 2003.
29. I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. NIPS 2014.
30. J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555.
31. J. Earley. An efficient context-free parsing algorithm. Communications of the ACM 1970.
32. J. Giménez and L. Marquez. SVMTool: A general POS tagger generator based on Support Vector Machines. LREC 2004.
33. J. Koutník, K. Greff, F. Gomez, and J. Schmidhuber. A clockwork rnn. ICML 2014.
34. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. ICML. 2001.
35. J. Martens. Deep learning via Hessian-free optimization. ICML 2010.
36. J. Nivre and M. Scholz. Deterministic dependency parsing of English text. COLING 2004.
37. J. Nivre. An efficient algorithm for projective dependency parsing. IWPT 2003.
38. J. Zhou, J. Xu, and W. Qu. Efficient Latent Structural Perceptron with Hybrid Trees for Semantic Parsing. IJCAI 2013.
39. J.L Elman. Finding structure in time. Cognitive science 1990.

References And Further Reading

40. K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. JMLR 2006.
41. L. Huang, S. Fayong, and Y. Guo. Structured perceptron with inexact search. HLT-NAACL 2012.
42. L. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. CVPR 2007.
43. L. Shen, G. Satta, and A. Joshi. Guided learning for bidirectional sequence classification. ACL 2007.
44. M. Bates. Models of natural language understanding. NAS 1995.
45. M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. ACL 2004.
46. M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. ACL 2002.
47. M. Collins. Head-driven Statistical Models for Natural Language Parsing. PhD Thesis 1999.
48. M. Collins. Incremental parsing with the perceptron algorithm. ACL 2004.
49. M. Collins. Three generative, lexicalised models for statistical parsing. EACL 1997.
50. M. Honnibal and M. Johnson. Joint incremental disfluency detection and dependency parsing. TACL 2014.
51. M.A.K Halliday. Language structure and language function. New horizons in linguistics 1970.
52. M.T. Luong, H. Pham, and C.D. Manning. Effective approaches to attention-based neural machine translation. EMNLP 2015.

References And Further Reading

53. N. Chomsky. Knowledge of language: Its nature, origin, and use. Greenwood Publishing Group 1986.
54. N. Chomsky. Language and mind. Cambridge University Press 2006.
55. N. Chomsky. Reflections on language. New York 1975.
56. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. JMLR 2014.
57. P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar. An End-to-End Discriminative Approach to Machine Translation. ACL 2006.
58. P. Xu and R. Sarikaya. Exploiting shared information for multi-intent natural language sentence classification. INTERSPEECH 2013.
59. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. JMLR 2011.
60. R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. HLT-NAACL 2003.
61. R. McDonald and F. Pereira. Online learning of approximate dependency parsing algorithms. EACL 2006.
62. R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. HLT-EMNLP 2005.
63. R. McDonald, K. Crammer, and F. Pereira. Flexible text segmentation with structured multilabel classification. HLT-EMNLP 2005.
64. R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. ICML 2013.

References And Further Reading

65. R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. EMNLP 2013.
66. R. Socher, C. Lin, A. Ng, and C.D. Manning. Parsing natural scenes and natural language with recursive neural networks. ICML 2011.
67. R. Wardhaugh. Introduction to Linguistics. Mc-Graw-Hill Book Company. 1972.
68. R.K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. JMLR 2005.
69. S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation 1997.
70. S. Petrov and D. Klein. Improved inference for unlexicalized parsing. NAACL 2007.
71. S. Petrov and D. Klein. Sparse multi-scale grammars for discriminative latent variable parsing. EMNLP 2008.
72. T. Kudo T and Y. Matsumoto. Chunking with support vector machines. NAACL 2001.
73. T.Matsuzaki, Y. Miyan, and J. Tsujii. Probailistic CFG with latent annotations. ACL 2005.
74. T. Robertshaw. Introduction to machine learning with naive bayes. http://tomrobertshaw.net/2015/12/introduction-to-machine-learning-with-naive-bayes/. 2015.
75. V. Nair and G.E. Hinton. Rectified linear units improve restricted Boltzmann machines. ICML 2010.
76. X. Sun. Asynchronous Parallel Learning for Neural Networks and Structured Models with Dense Features. COLING 2016.

References And Further Reading

77. X. Sun, L. Morency, D. Okanohara, Y. Tsuruoka, and J. Tsujii. Modeling latent-dynamic in shallow parsing: a latent conditional model with improved inference. COLING 2008.
78. X. Sun, T. Matsuzaki, and W. Li. Latent structured perceptrons for large-scale learning with hidden information. TKDE 2013.
79. X. Sun, T. Matsuzaki, D. Okanohara, and J. Tsujii. Latent variable perceptron algorithm for structured classification. IJCAI 2009.
80. X. Sun. Structure regularization for structured prediction. NIPS 2014.
81. X. Sun. Towards shockingly easy structured classification: A search-based probabilistic online learning framework. arXiv:1503.08381.
82. Y. Zhang and J. Nivre. Transition-based dependency parsing with rich non-local features. ACL-HLT 2011.
83. Y. Zhang and S. Clark. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. EMNLP 2008.
84. Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv:1508.01991.

Content And Lecturer

Models & Regularization

- Conventional Model
- Latent Model
- Neural Model

Xu SUN

Structures & Applications

- Sequence Structure
- Tree/Graph Structure

Yansong FENG

Type of Structures in Application

Yansong FENG

Peking University

fengyansong@pku.edu.cn

Outline

Sequence Tagging

Tree/Graph Structure

Take-home Msg

Outline

- **Sequence Tagging**
  - Chinese Word Segmentation
  - POS Tagging/Named Entity Recognition
- **Tree/Graph Structure**
- **Take-home Msg**

Sequence Tagging

- **Sequence**
  - the most basic/simplest structures in NLP
- **Tag each item in the sequence using a given label set**
- **Common to see**
  - World segmentation
  - Part-of-Speech tagging
  - Named entity recognition
  - Chunking
  - Event trigger identification
  - …

Sequence Tagging

- **Conventional models**
  - CRF, Structured perceptron, …
- **Work well, but**
  - Feature engineering
  - Local context / Global information
  - …
- **Neural models**
  - CNN, RNN, LSTM, LSTM-CRF,…
- **Performances**
  - Comparable to state of the arts
  - Combinations may give top performances

Chinese Word Segmentation

- **The Task**

[Xu and Sun, 2016]

地面 积 了 厚厚 的 雪 **(The ground is covered with thick snow)**

这 块 地 面积 还 真 不小 **(This area is really not small)**

- **Challenges**
  - Feature engineering
  - Long distance dependencies
- **Neural Network Practice**
  - CNN/RNN to capture local information, instead of fixed windows
  - RNN to capture long distance dependencies, or sentence-level/global information

Model Local Information

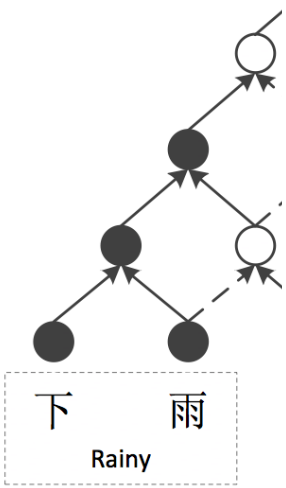
- **NN to model local features**
  - TNN to capture tag/word features and their combinations
  - CNN, RNN, LSTM, BLSTM to capture local features, beyond fixed window size
- **Choice of character/word level**
  - Word level features are still important, but not easy to incorporate in models
  - Explore word level information in a beam-search framework [Zhang et al., 2016, Cai and Zhao, 2016]
  - Word level features give **0.5%**
- **Combinations**
  - Combining neural and discrete features gives top performances

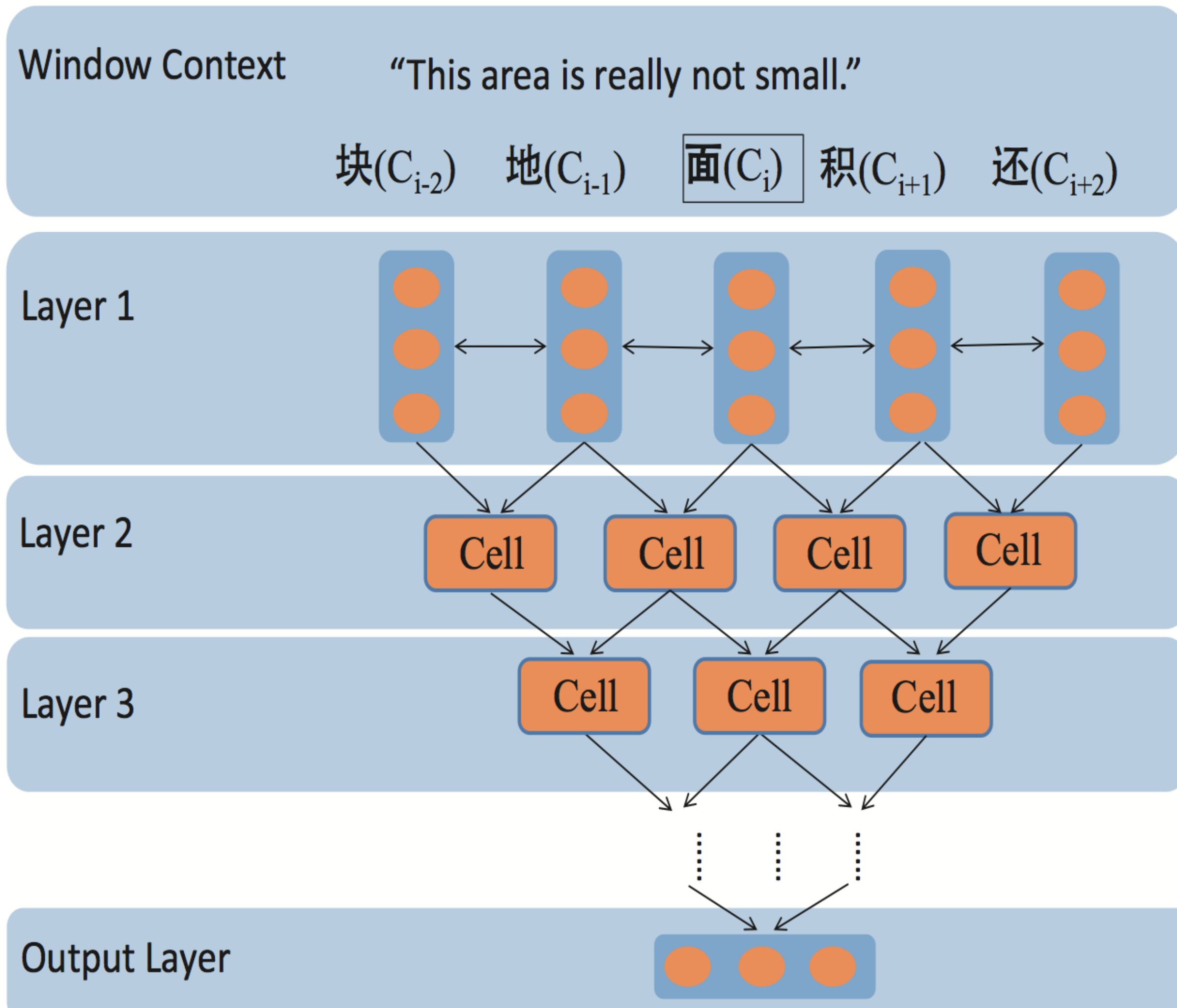
Global View

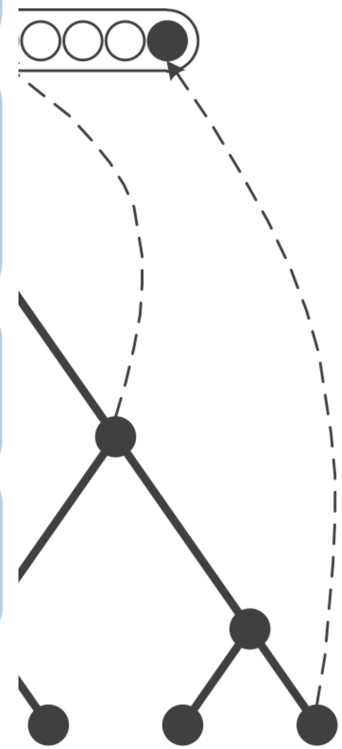
- **Long Distance Dependencies**
  - Gated Recursive NN [Chen et al., 2015, Xu and Sun, 2016]
  - LSTM [Chen et al., 2015b, Zhang et al., 2016, Cai and Zhao, 2016]
- **Search Strategy**
  - CRF framework
    - Viterbi
  - Beam-search style
    - Fully explore word level information
    - Transition based [Zhang et al., 2016]
    - Beam search [Cai and Zhao, 2016]

Models

- **Gated Recursive Neural Networks**
- **Dependency based Gated Recursive Neural**



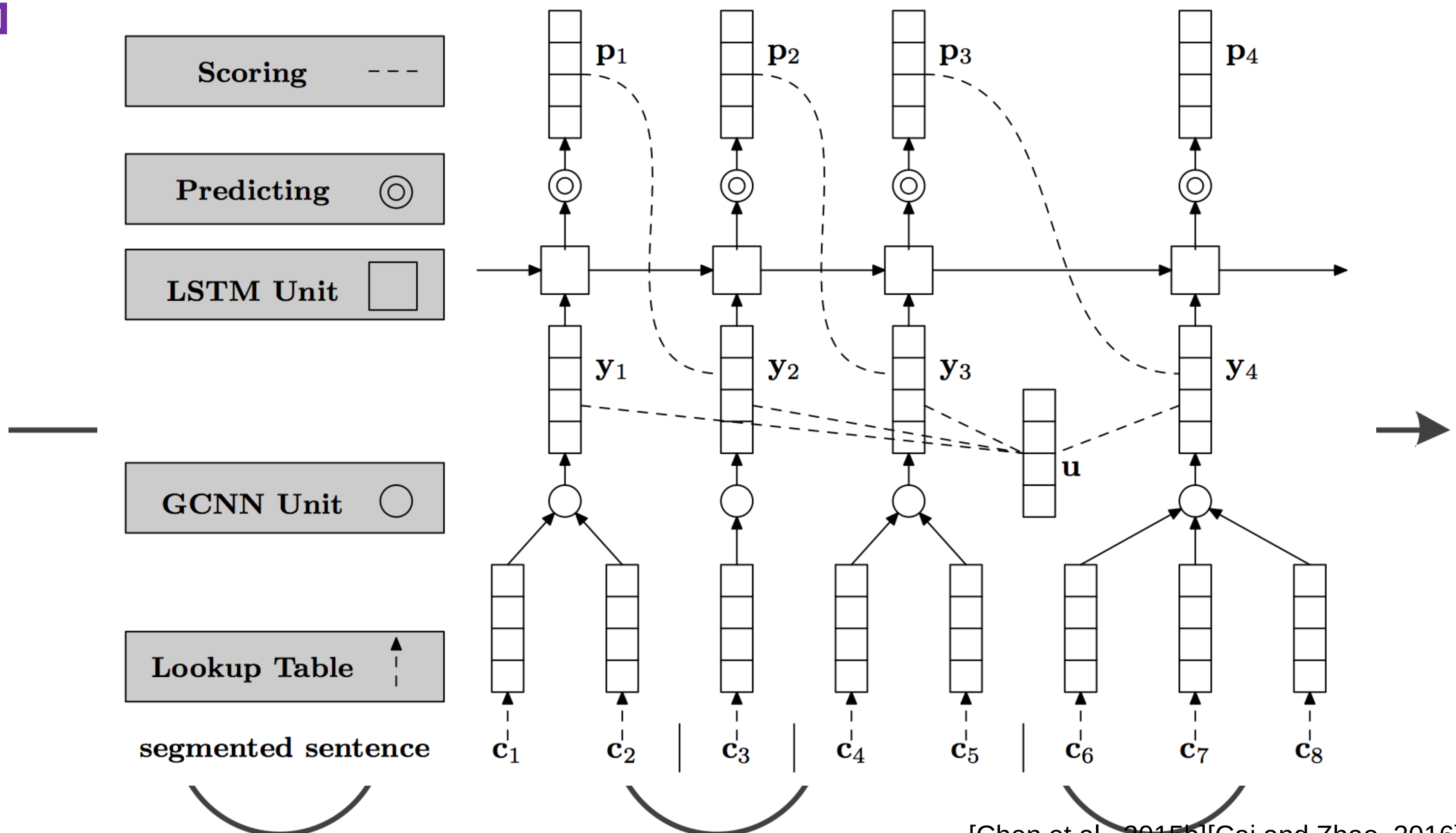




…and Sun, 2016

Models

- LSTM
- LSTM + Gated Combination Neural Networks
- Bi

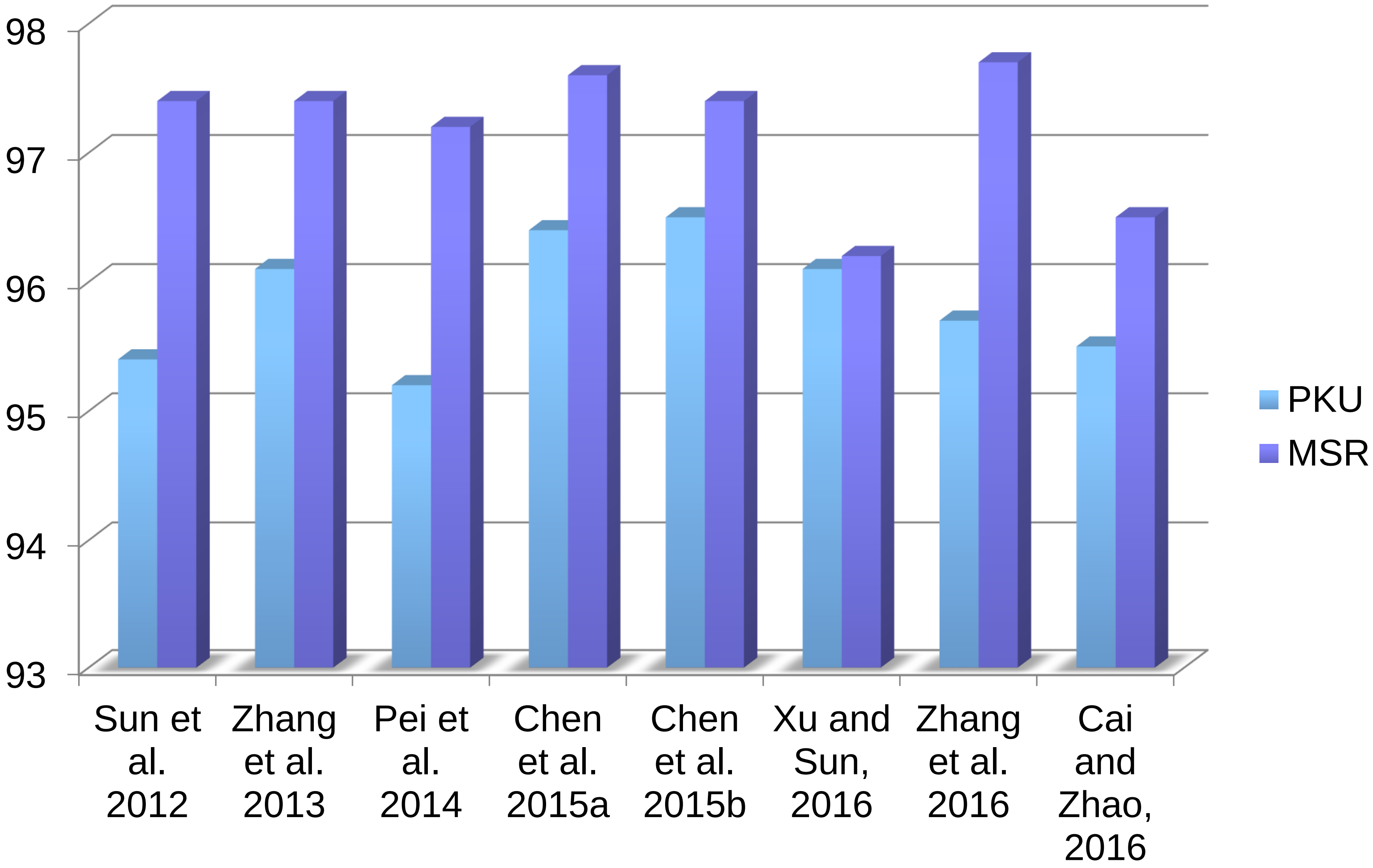


[Chen et al., 2015b][Cai and Zhao, 2016]

Performances on Benchmarks

- **Conventional models are still strong**
- **Neural models can be promising, and sometimes complementary to conventional models**
  - Long distance dependencies
  - Word level features
  - BiLSTM works to capture local context information
  - Various NN models to sentence level/long-distance information
  - CRF is still attractive

On PKU and MSR



POS/NER/Chunking

- **Typical Sequence Labeling tasks**
  - Conventional models have achieved over ~90%
- **Again**
  - Feature engineering
  - Language issues
  - Local/Global
  - Label bias
- **Neural Network Practice**
  - BLSTM/CNN to capture local context, both forward and backward
  - CRF with Viterbi to find the best sequence

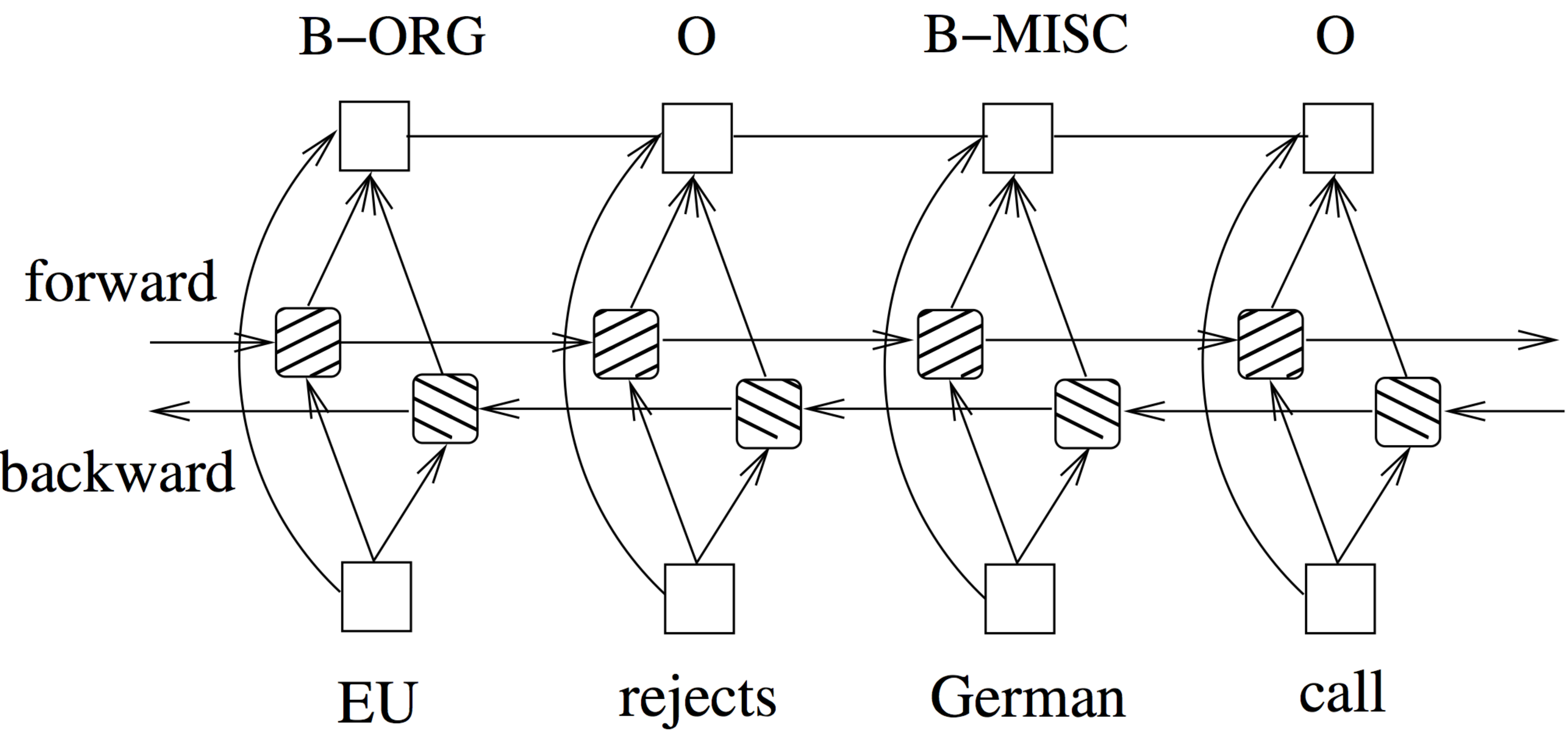
Feature Extraction

- **NN models seem be capable of handling language issues to some extent**
  - CNN
  - BLSTM → Dominating!
  - Character level modeling
- **BLSTM works better than LSTM**
  - Look at both past and future
- **Traditional lexicon features are still there**
  - Extra resources, like dictionary, gazetteers, or Wiki, are always welcome

Models

- **BLSTM-CRF with feature concatenation**
- **work nice for POS, NER, Chunking**

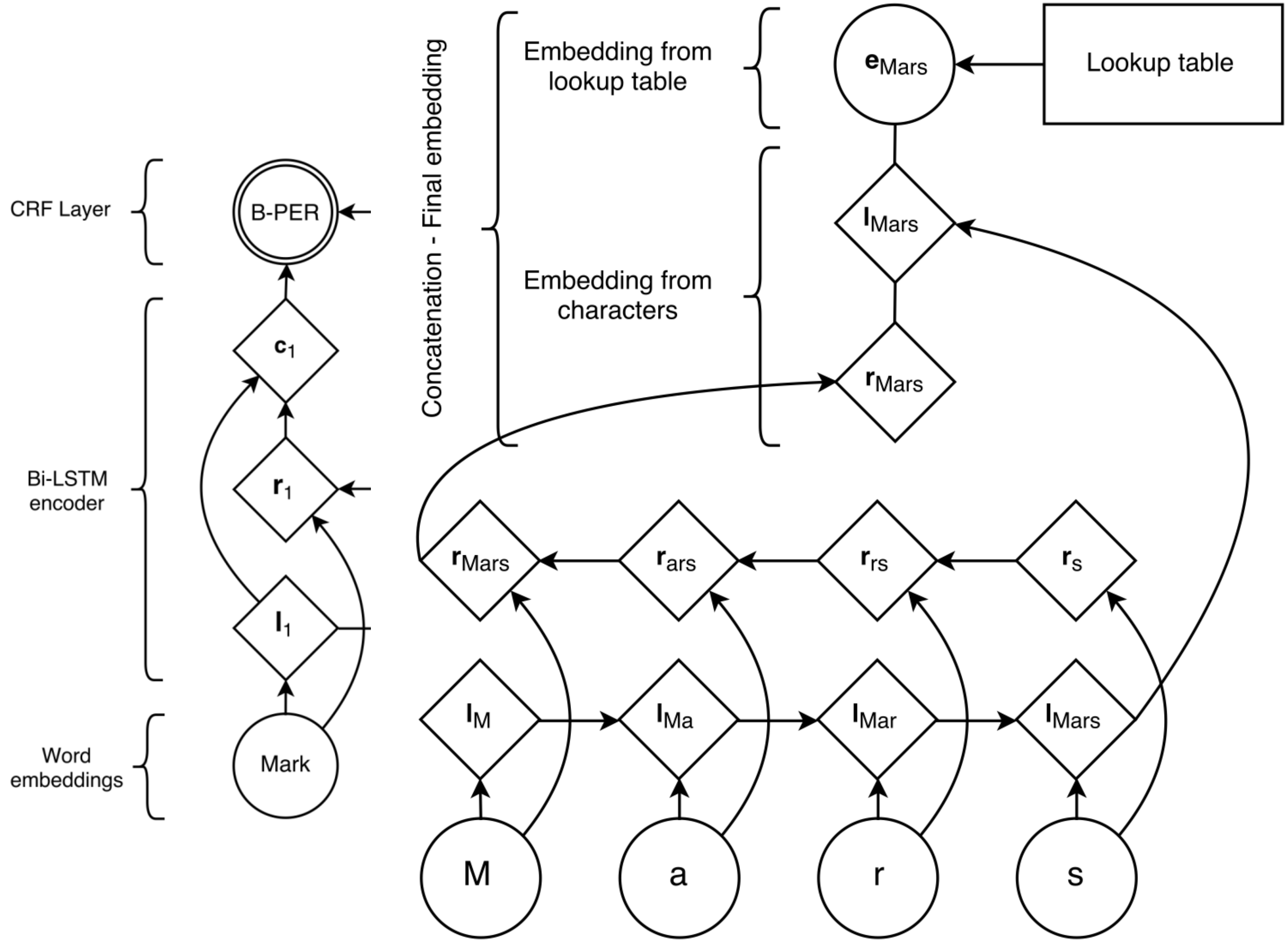
[Huang et al., 2015]

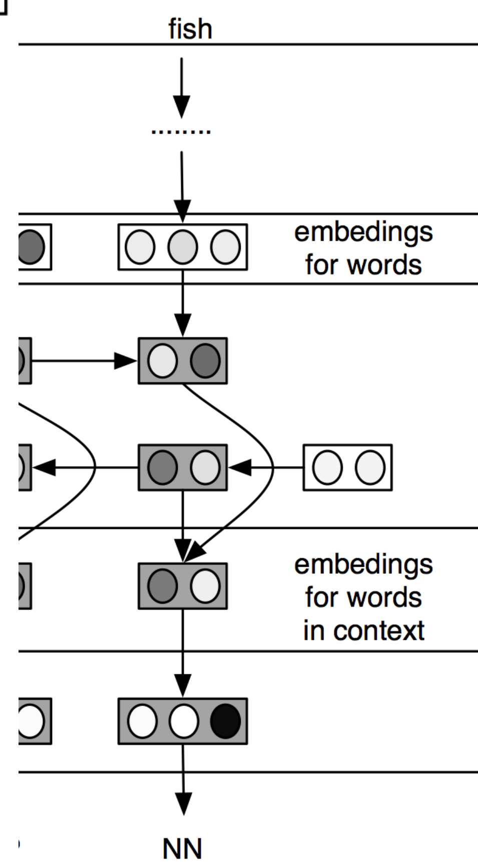


Models

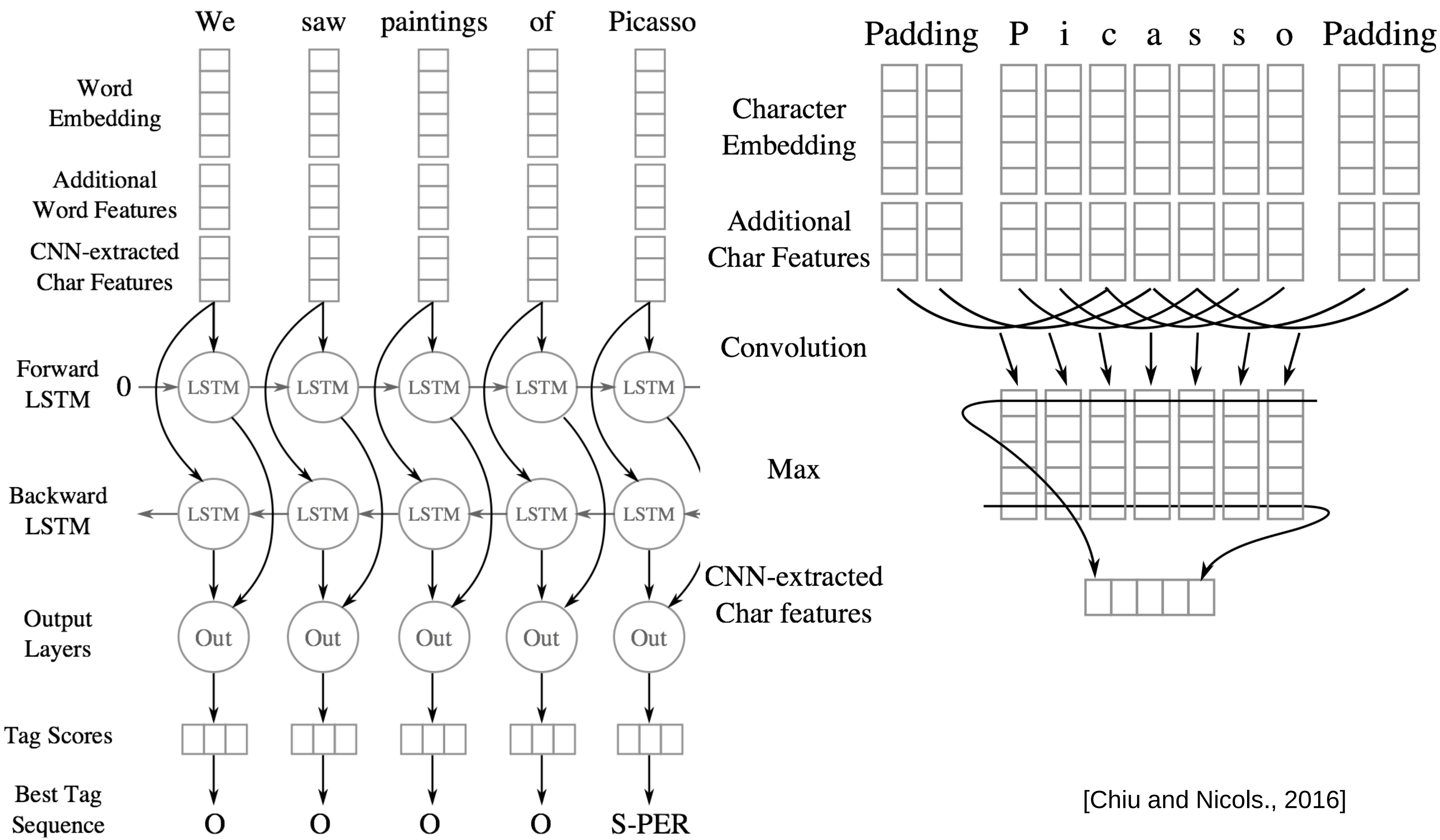
- **Typical BLSTM-CRF for POS / NER**
- **Word / Character Level**

[Ling et al., 2015]
[Lample et al., 2016]





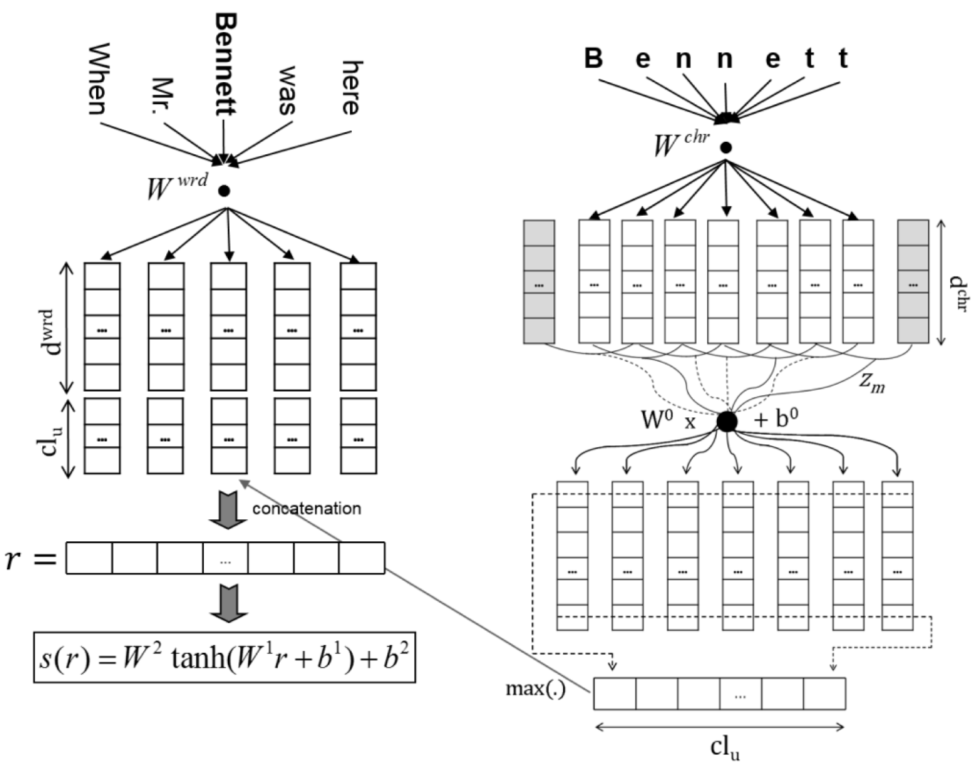
BLSTM+CNN for NER

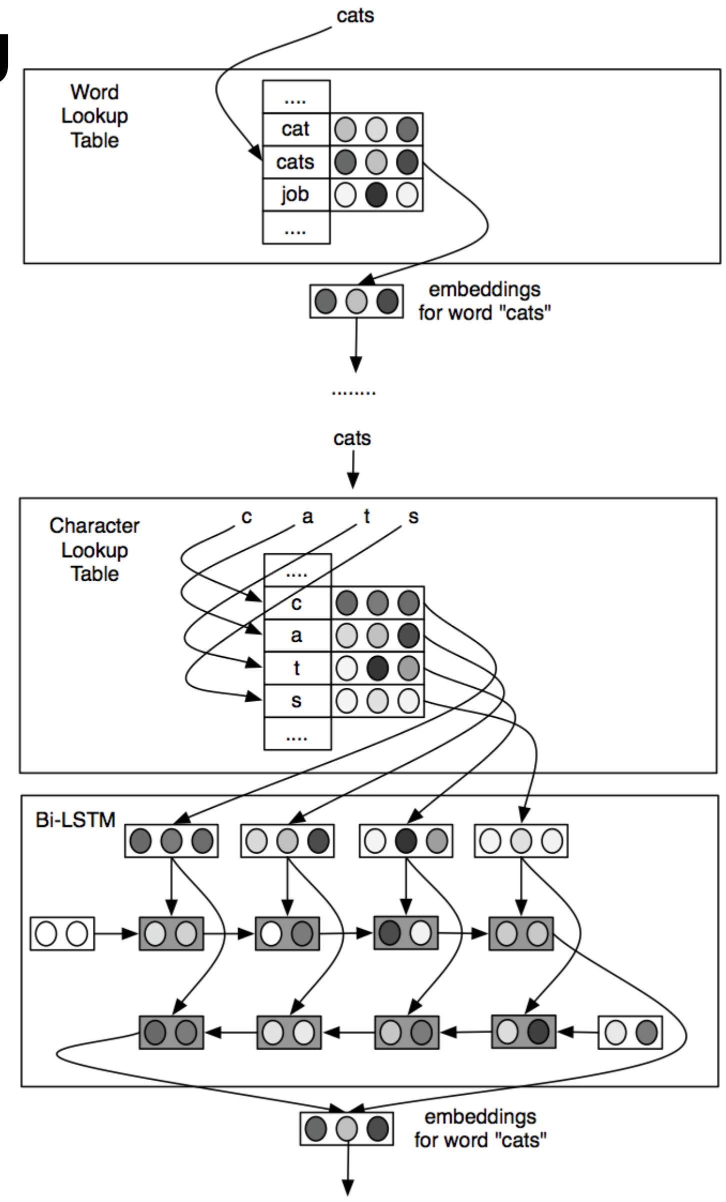


[Chiu and Nicols., 2016]

Language Issues

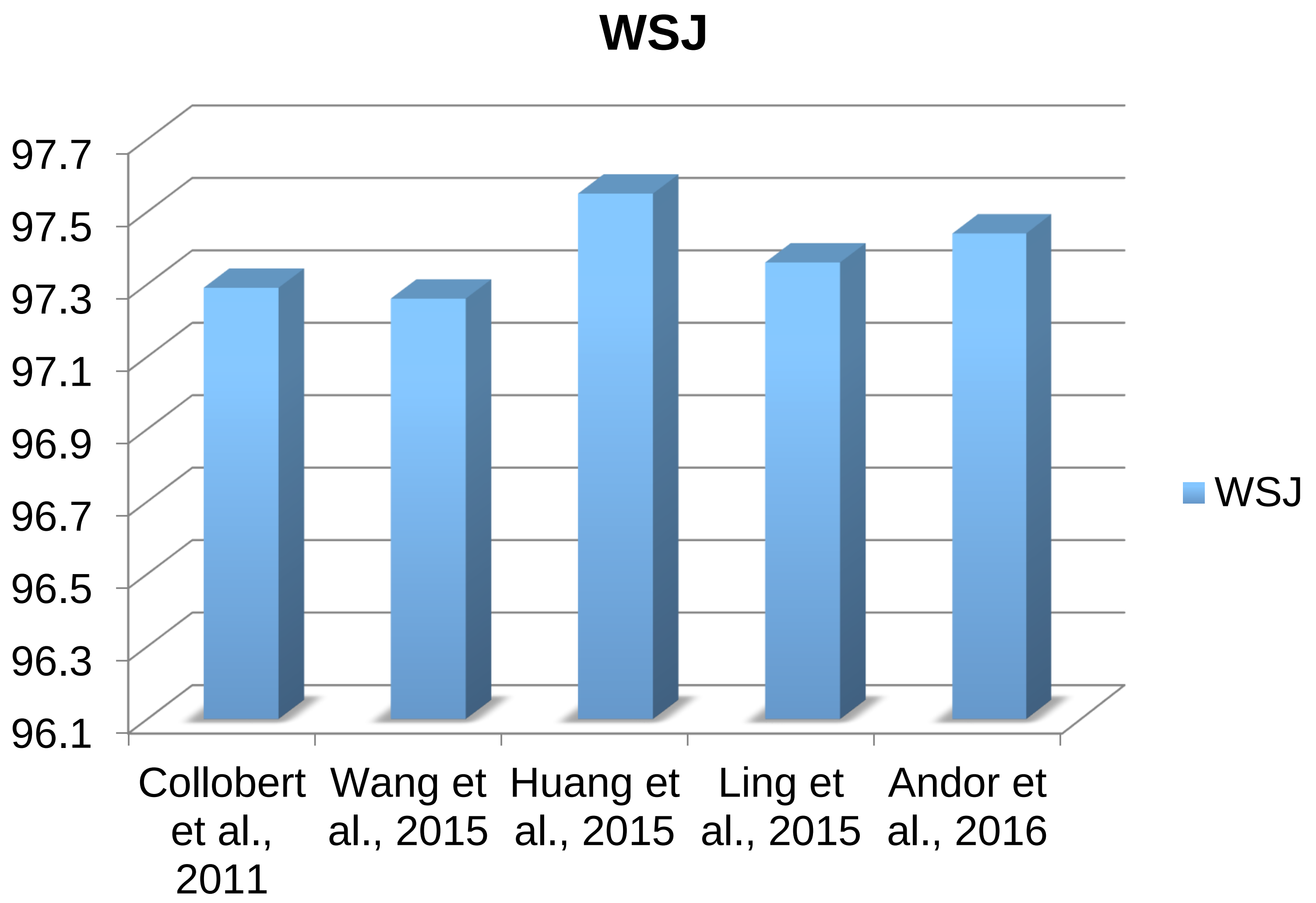
- Handling with different languag
- Handling with OOV
  - using CNN
  - character-level modeling



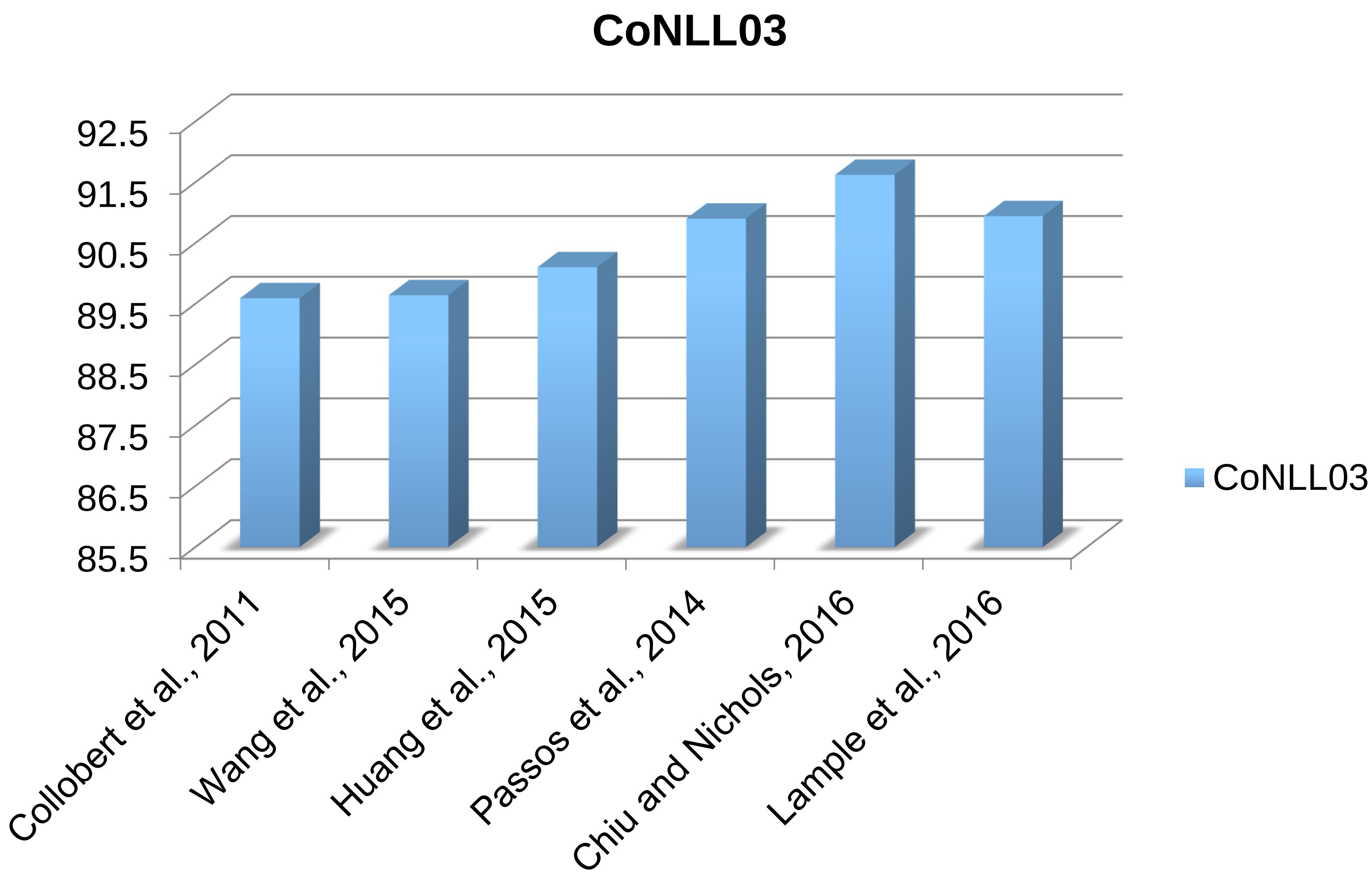


[dos Santos and Guimaraes, 2015] [Ling et al., 2015]

POS Tagging On WSJ



NER On CoNLL


CoNLL03
92.5
91.5
90.5
89.5
88.5
87.5
86.5
85.5
Collobert et al., 2011
Wang et al., 2015
Huang et al., 2015
Passos et al., 2014
Chiu and Nichols, 2016
Lample et al., 2016
CoNLL03

Misc

- **Event Trigger Identification**

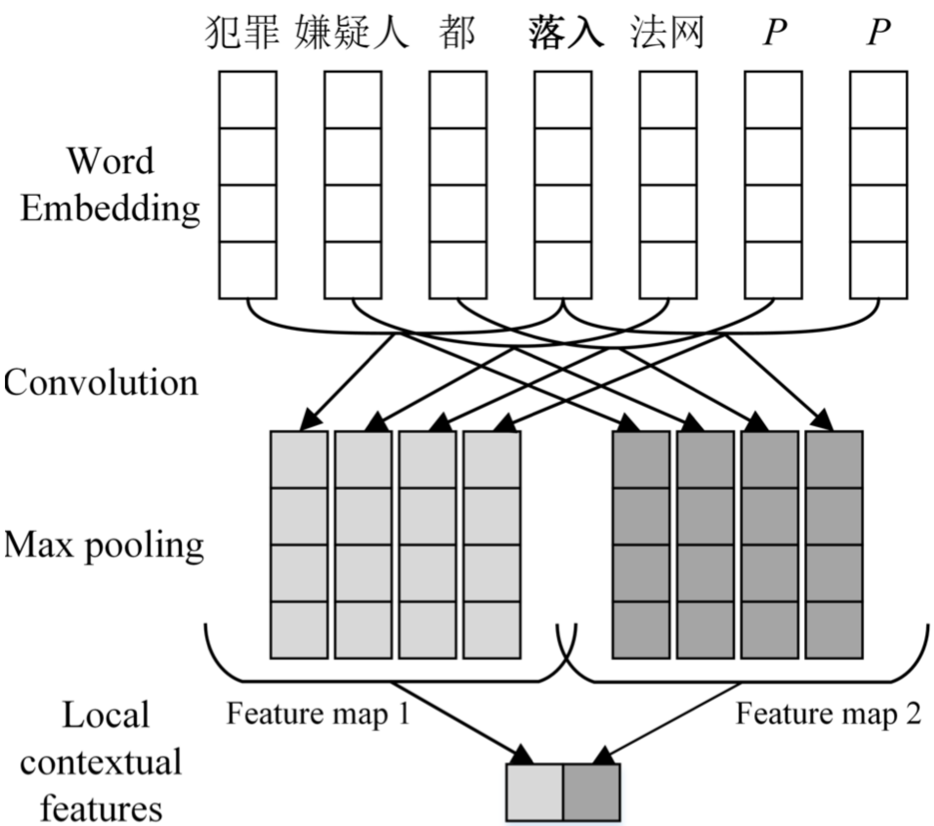
  犯 罪 嫌 疑 人 都 落 入 法 网

  The suspects were arrested [*arrest_jail*]

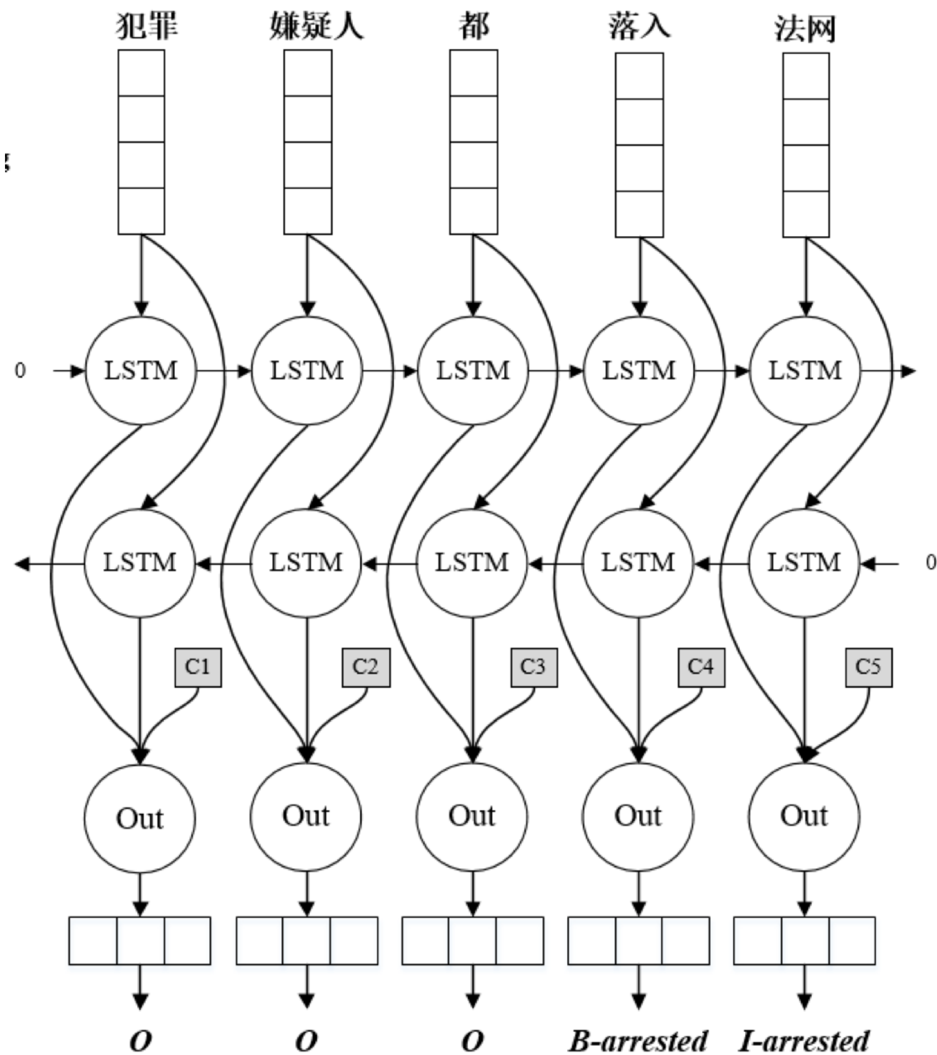
- **NN models work to help**
  - Feature extraction
  - Both local and global features (CNN, BLSTM)
  - Language issues: character level modeling

Models

- BLSTM+CNN for event trigger identification
- CNN to capture local context



[Zeng et al., 2016]



Outline

Sequence Tagging

Tree/Graph Structure

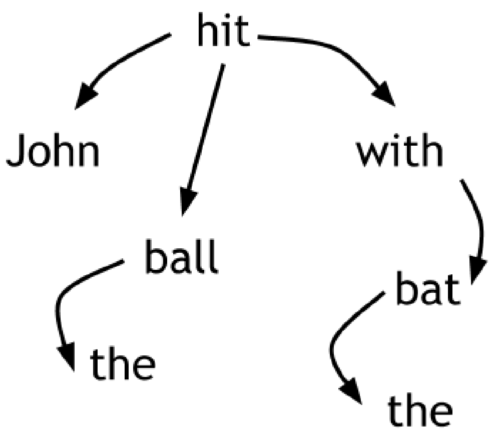
Dependency Parsing

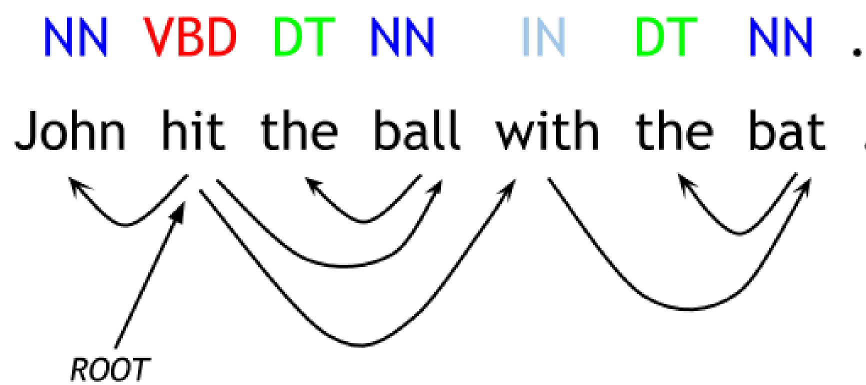
Semantic Role Labelling

Take-home Message

Tree/Graph Structures

- **One of most common/important structures in NLP**

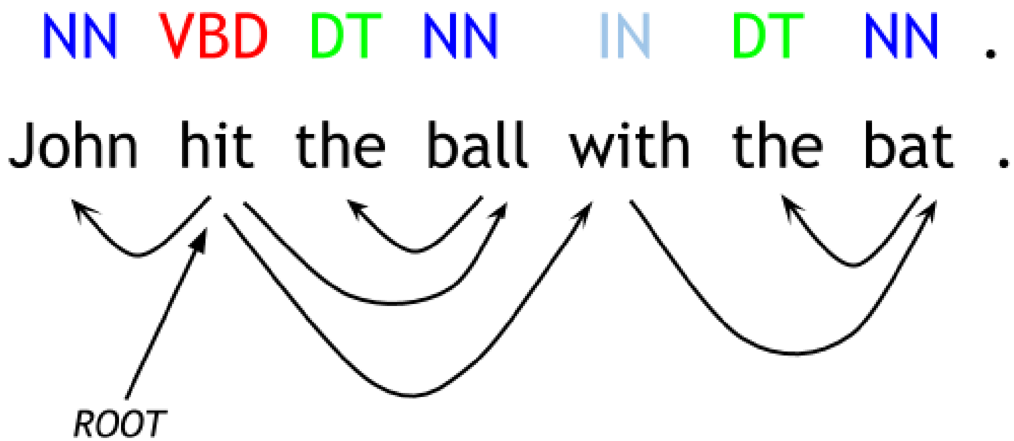




- **Both the tree/graph structures and their tags are latent**
- **Building bricks**
  - Syntactic parsing
  - Event extraction
  - Semantic role labeling
  - …

Dependency Parsing

- The Task
- Conventional Models
  - Transition based
  - Graph based
- However,
  - Feature engineering
  - Label bias ----> locally/globally normalized
- Neural Network Practice
  - Transition/Graph based
  - NN models to extract various features
  - choose from greedy search, beam search or approximate global normalization

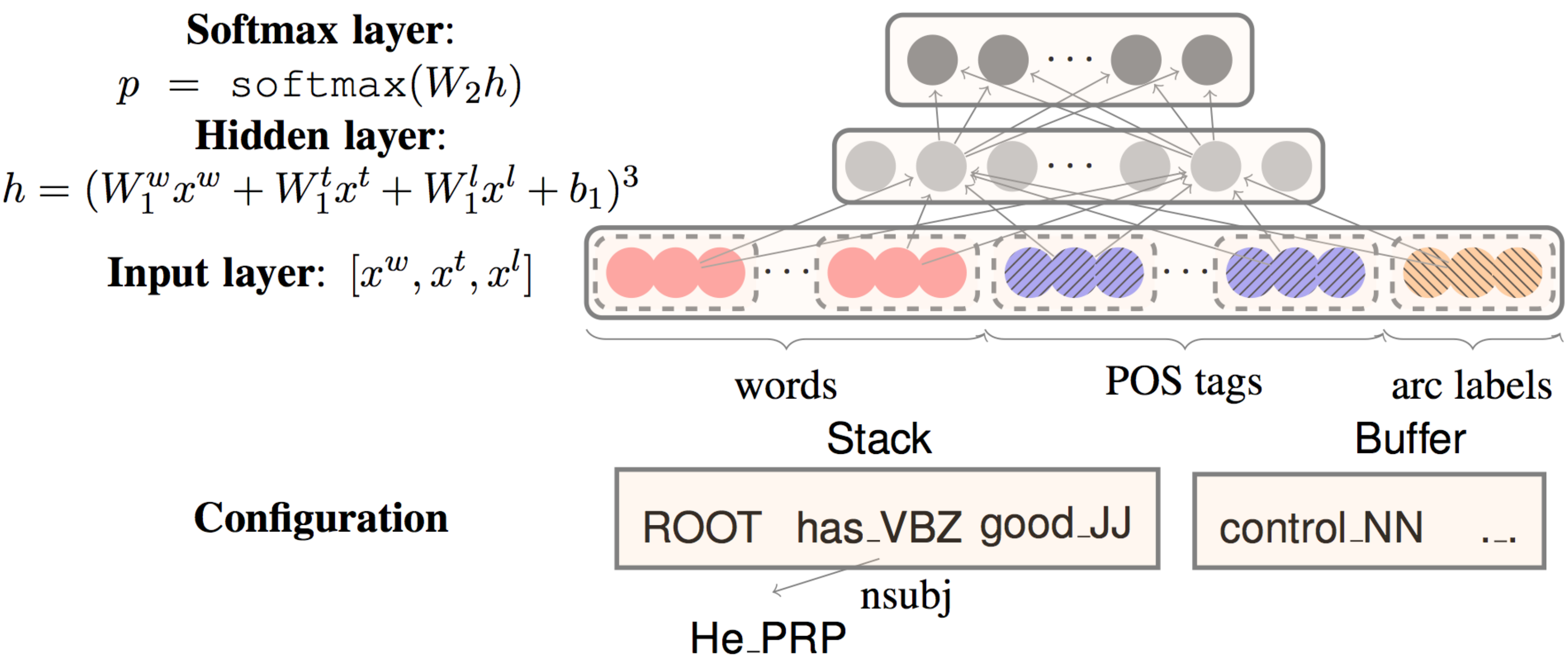


Transition based models

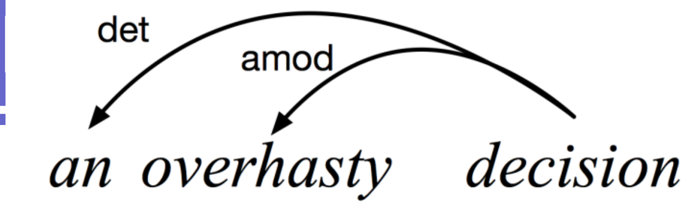
- Most DL based works in dependency parsing follows the transition based framework.
  - lower complexity
  - higher efficiency
  - more choices of features
- Follow normal transition styles
  - but, most are based on greedy search
- Various NN models used to produce dense features
  - normal NNs
  - LSTM
  - BLSTM, both word level and character level
- [Chen and Manning, 2014], [Weiss et al., 2015], [Dyer et al., 2015], [Ballesteros et al., 2015], [Zhou et la., 2015], [Kiperwasser and GoldBerg, 2016], [Andor et al., 2016]

Features

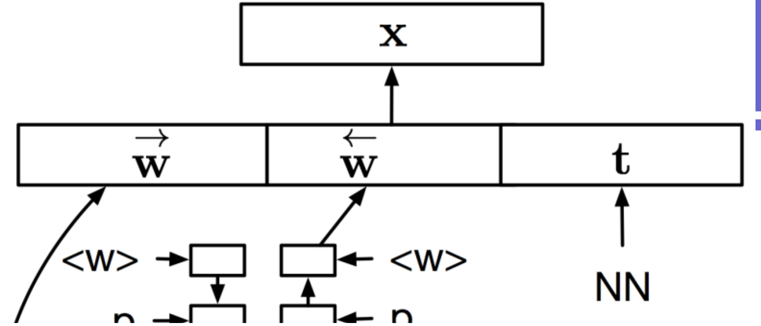
- Key features, such as words, POS and dep labels, as well as their combinations can be transformed into dense format through a neural network



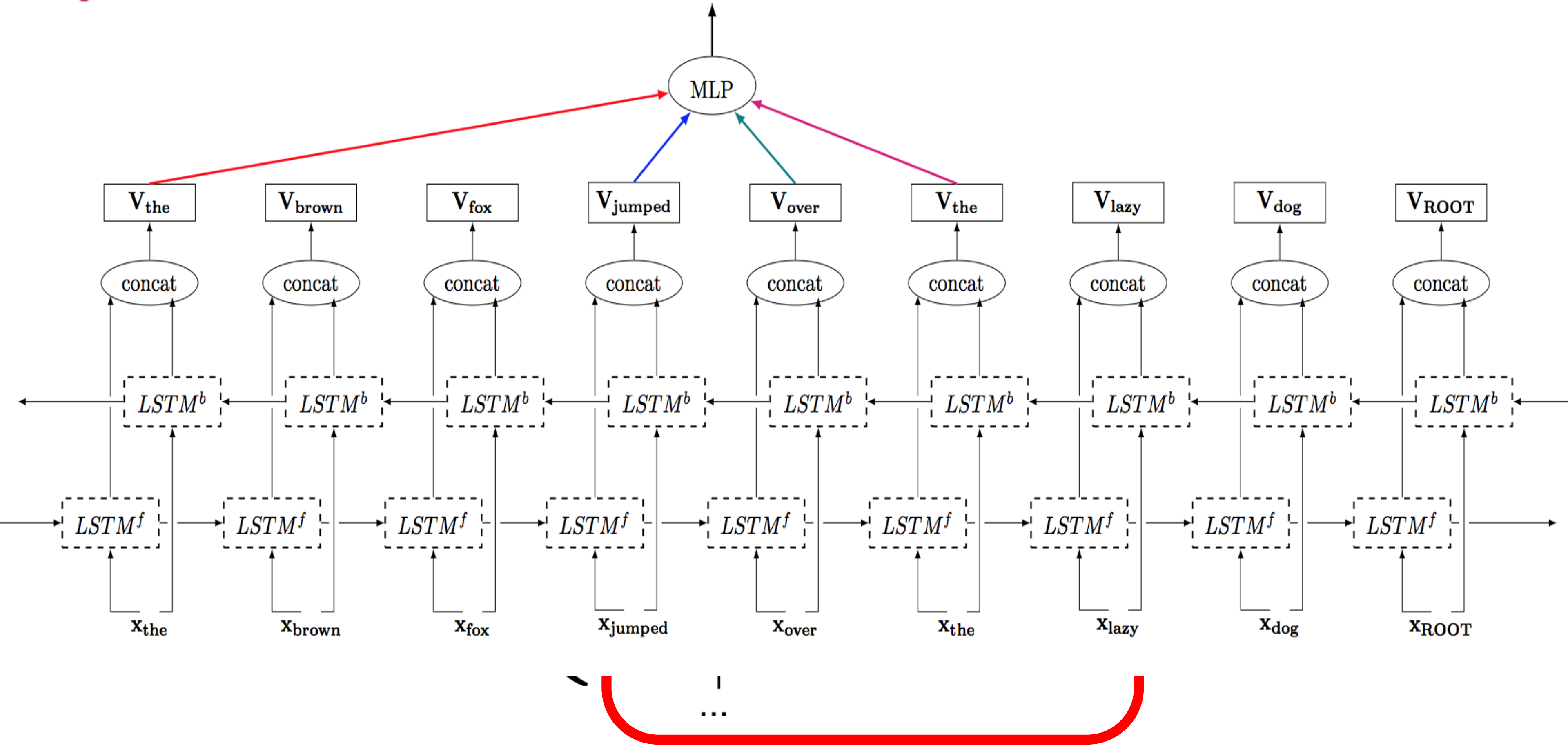
[Chen and Manning, 2014]



ured



Scoring:



[Dyer et al., 2015] [Ballesteros et al., 2015]
[Kiperwasser and GoldBerg, 2016]

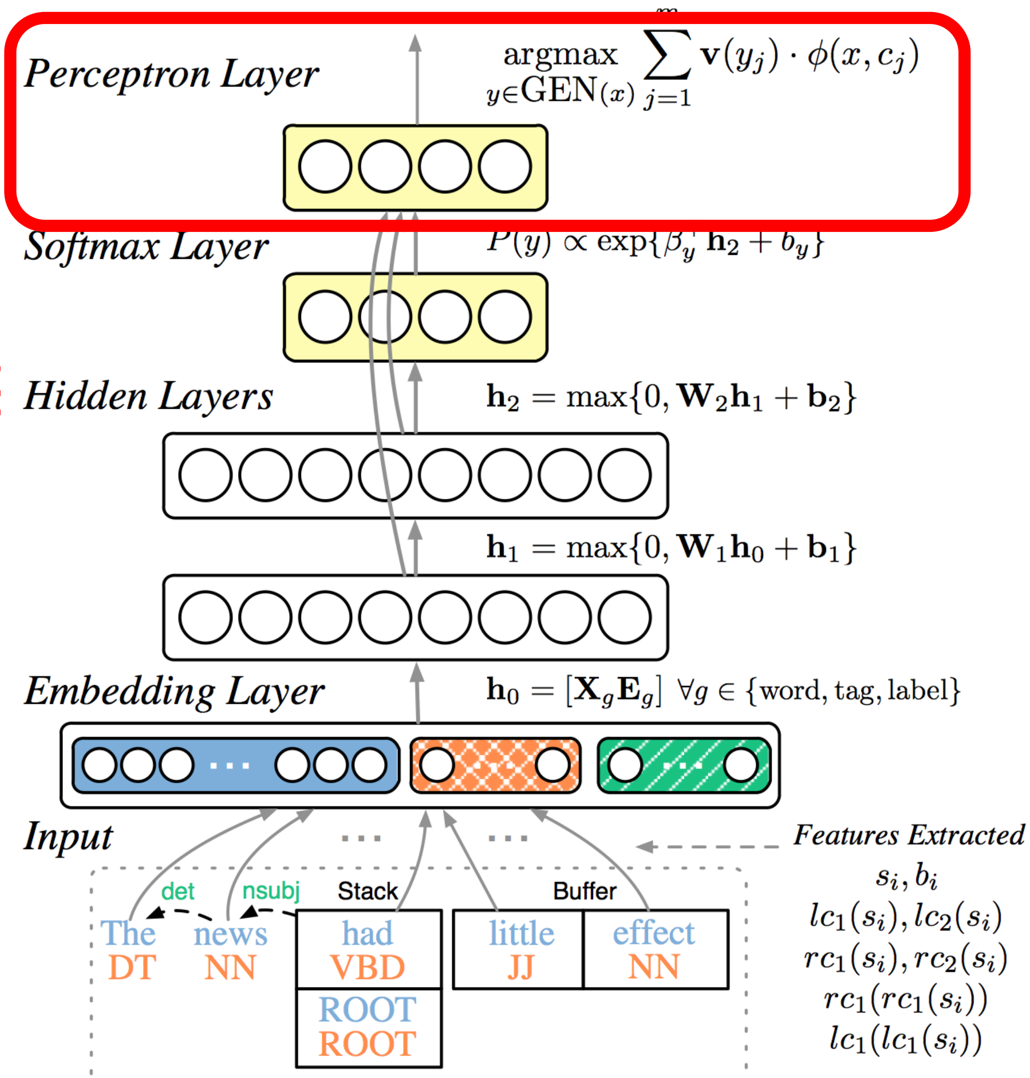
Greedy Search

- No global considerations about future decisions
  - look ahead with limited context
  - label bias
  - May be error propagations
- Training a global normalized model may not be trivial
  - Complexity
  - Space
- And, it is found that, in many parsers, beam search has minimal impact in the results. [Dyer et al., 2015]
- However, sometimes, better than beam-search based methods [Kiperwasser and GoldBerg, 2016]

Beam Search

- One solution is to use the output layers of the NN model to learn a structured perceptron with a beam search

- **gives 0.8%**



[Weiss et al., 2015]

Approximate Global Normalization

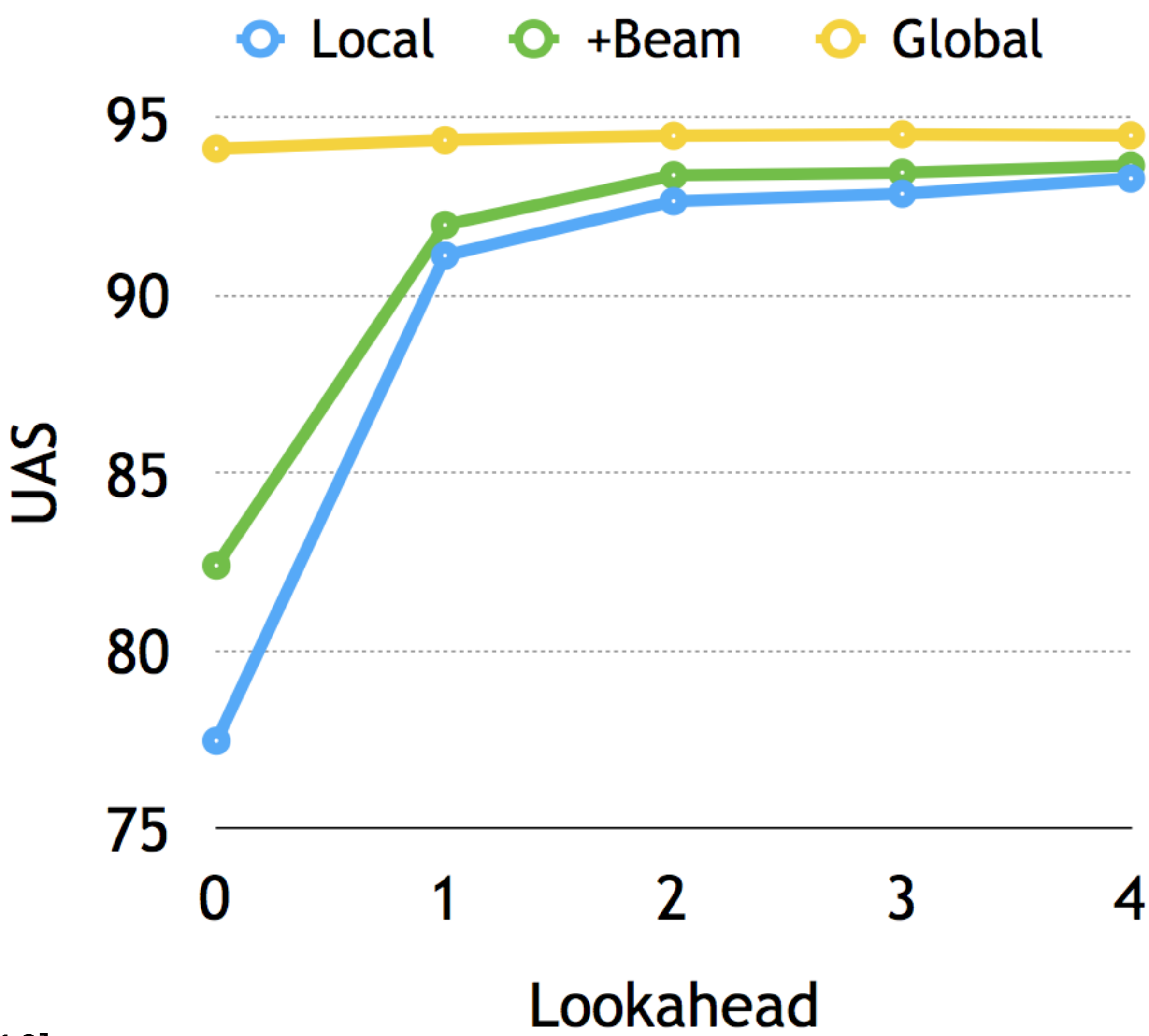
- Local normalized models may suffer from local optimal (greedy search), label bias, etc.
- Perform global normalization exactly
  - maximize the whole action sequence
- It is not trivial !
  - too many possible action sequences
  - expensive (impossible) to enumerate and compute
- Or, do it approximately ?

Approximate Global Normalization

- **Contrastive Learning** [Hinton, 2002; LeCun and Huang, 2005; Liang and Jordan, 2008; Vickrey et al., 2010]
  - reward observed data
  - penalize noisy data
- In the beam search case: [Zhou et al., 2014]
  - Give gold sequence with higher probability
  - Give incorrect sequences in the beam with lower probabilities
  - Early update may be helpful
  - gives more than **1.5%**

Approximate Global Normalization

- The importance of looking ahead



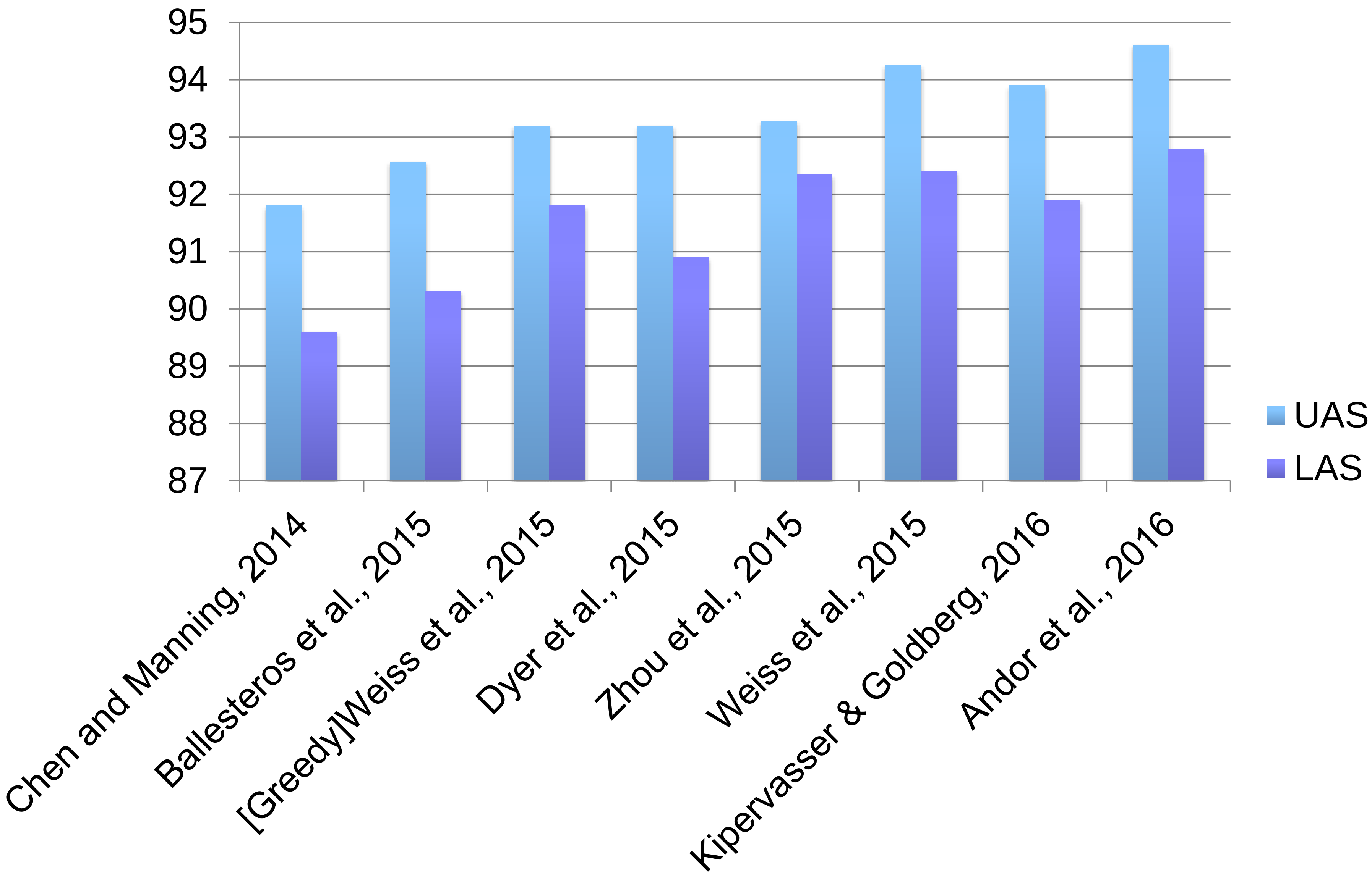
[Andor et al., 2016]

Approximate Global Normalization

- **Global Training**
  - Backward propagation with beam search
  - with early update
  - slow
- **Works for multiple tasks:**
  - Dependency parsing
  - POS tagging
  - Sentence Compression
- **Insight:**
  - Global models are more expressive

[Andor et al., 2016]

On WSJ


95
94
93
92
91
90
89
88
87
Chen and Manning, 2014
Ballesteros et al., 2015
[Greedy]Weiss et al., 2015
Dyer et al., 2015
Zhou et al., 2015
Weiss et al., 2015
Kipervasser & Goldberg, 2016
Andor et al., 2016
UAS
LAS

Semantic Role Labeling

- **The Task**

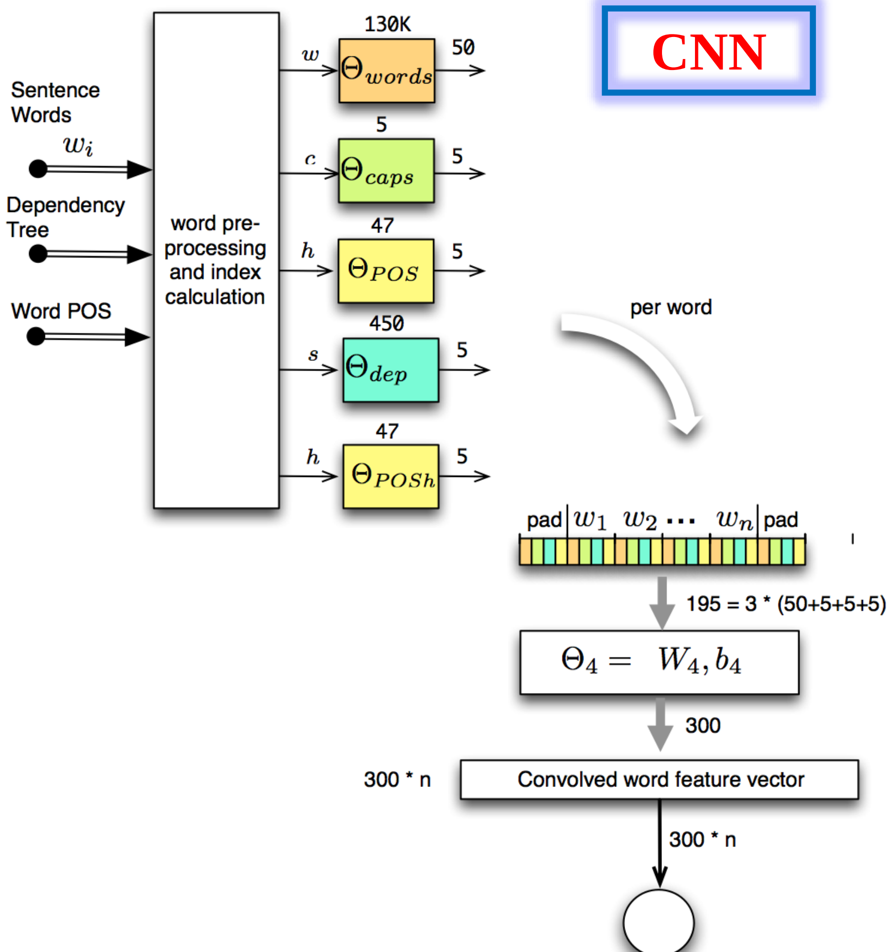
[He$_{A0}$] had trouble **raising** [funds$_{A1}$]. [Roth and Lapata, 2016]

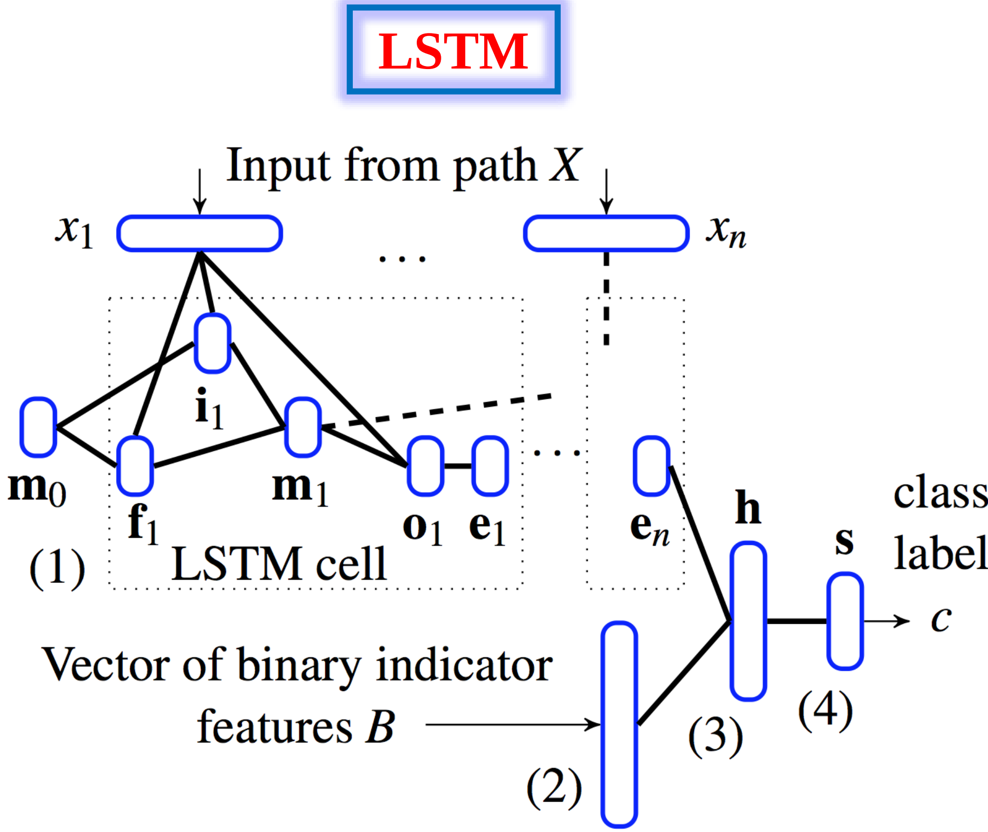
- **Conventional Models**
  - Pipeline
  - predicate identification, argument identification, argument classification
- **However,**
  - Feature engineering, local features, global features, ...
  - Pipeline
- **Neural Practice**
  - NN models to extract various features, or as local classifiers
  - In a sequence labeling style, an End-to-End system

Pipeline System

- **Step by step**: predicate identification, argument identification, argument classification

- **Each step:** local classifiers with feature extraction, using CNN, or LSTM

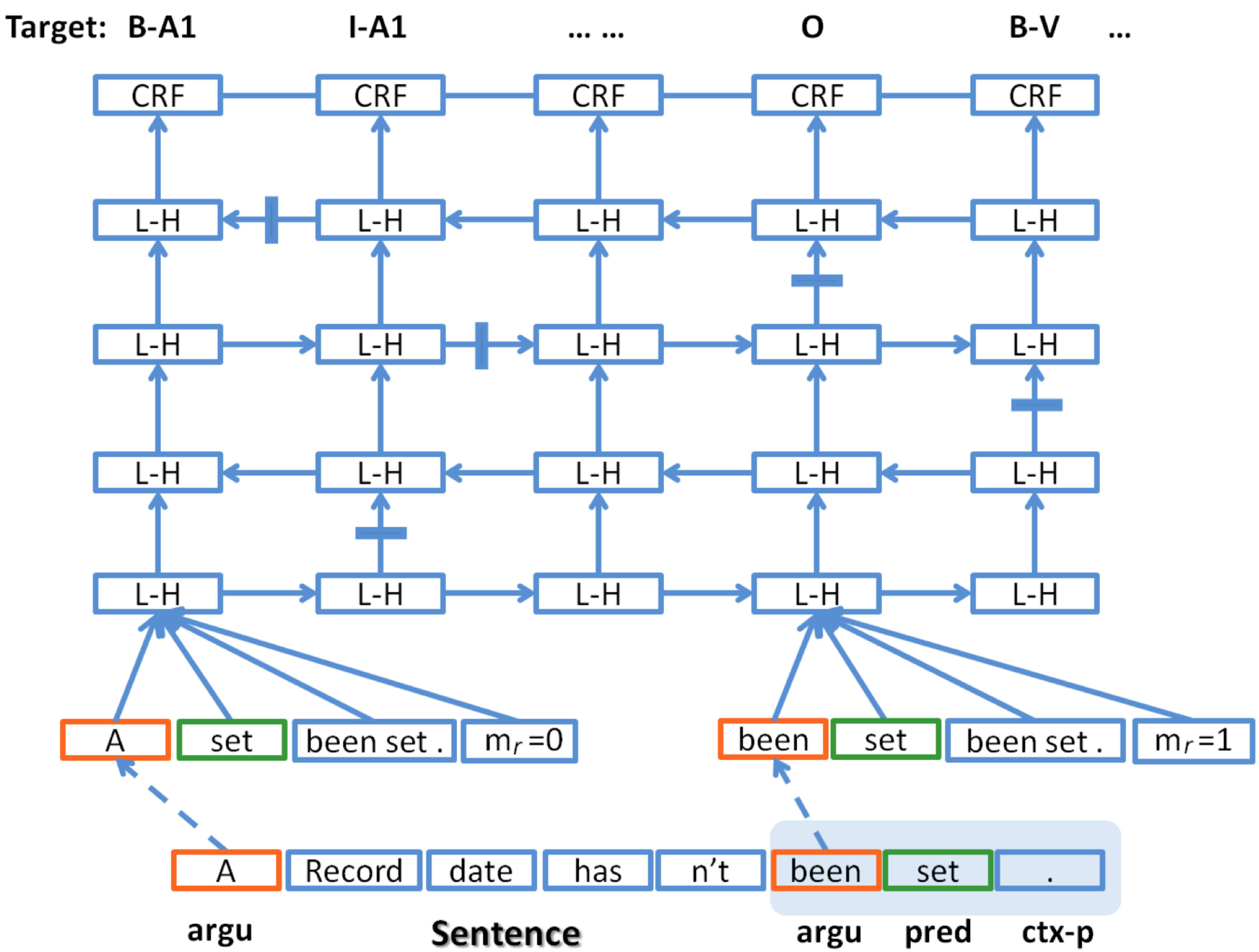




[Fortland and Martin, 2015] [Roth and Lapata, 2016]

End-to-End System

- Multi-Layered LSTM with CRF to **Sequence Tagging**.



[Zhou and Xu, 2016]

CNN v.s. LSTM

- **Handle long distance dependencies**
- **CNN**
  - good at capturing local features, or work on dependency path
  - not so good at End-to-End systems, or extracting features from plain word sequence for SRL
- **LSTM**
  - good at capturing both local features, and global information, either for local decisions, or sentence level re-ranking
  - more powerful in capturing features of various levels
  - deep LSTM can also be used to build End-to-End SRL systems

Outline

Sequence Tagging

Tree/Graph Structure

Take-home Messages

How-tos

- **Standard settings for Sequence Tagging**
  - BLSTM + CRF
  - character modeling or CNN for fine modeling
  - fused with successful traditional features
  - perhaps, beam search for various features
  - But Use as many useful resources as you can !
- **More complex structures**
  - BLSTM + Transition base systems
  - choose from greedy search / beam search
  - But

Choose globally normalized models if you can!

Thanks

Reference

- Alexandre Passos, Vineet Kumar, Andrew McCallum, Lexicon Infused Phrase Embeddings for Named Entity Resolution, In Proceedings of the Eighteenth Conference on Computational Language Learning, pages 78–86, Baltimore, Maryland USA, June 26-27 2014.
- Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1354–1359.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pages 334–343.
- Cıcero dos Santos and Victor Guimaraes, Boosting Named Entity Recognition with Neural Character Embeddings, In Proceedings of the Fifth Named Entity Workshop, joint with 53rd ACL and the 7th IJCNLP, pages 25–33, Beijing, China, July 26-31, 2015.
- Cıcero Nogueira dos Santos and Bianca Zadrozny, Learning Character-level Representations for Part-of-Speech Tagging, In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014.
- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov and Michael Collin, Globally Normalized Transition-Based Neural Networks, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2442–2452, Berlin, Germany, August 7-12, 2016
- Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 740–750.
- David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pages 323–333.

Reference

- Deng Cai and Hai Zhao, Neural Word Segmentation Learning for Chinese, In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 409–420, Berlin, Germany, August 7-12, 2016
- Eliyahu Kiperwasser and Yoav Goldberg, Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations, arXiv:1603.04351v2
- Hao Zhou, Yue Zhang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pages 1213–1222.
- In Transactions of the Association for Computational Linguistics, vol. 4, pp. 357–370, 2016
- Jason P.C. Chiu and Eric Nichols, Named Entity Recognition with Bidirectional LSTM-CNNs,
- Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labelling using recurrent neural networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1127–1137.
- Jingjing Xu and Xu Sun, Dependency-based Gated Recursive Neural Network for Chinese Word Segmentation, In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 567–572, Berlin, Germany, August 7-12, 2016
- Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013a. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In Proceedings of the EMNLP 2013, pages 311–321, Seattle, Washington, USA, October.
- Matthieu Labeau, Kevin Loser, Alexandre Allauzen, Non-lexical neural architecture for fine-grained POS Tagging, In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 232–237, Lisbon, Portugal, 17-21 September 2015

Reference

- Meishan Zhang, Yue Zhang, Guohong Fu, Transition-Based Neural Word Segmentation, In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 421–431, Berlin, Germany, August 7-12, 2016.
- Michael Roth and Mirella Lapata, Neural Semantic Role Labeling with Dependency Path Embeddings, In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1192–1202, Berlin, Germany, August 7-12, 2016
- Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 349–359.
- Nicholas FitzGerald, Oscar Ta ̈ckstro ̈m, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 960–970, Lisbon, Portugal.
- Peilu Wang, Yao Qian, Frank K. Soong, Lei He, Hai Zhao, A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding, arXiv:1511.00215v1
- Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.
- Wang Ling, Tiago Luıs, Luıs Marujo, Ramon Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, Isabel Trancoso, Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation, In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1520–1530, Lisbon, Portugal, 17-21 September 2015.
- Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In Proceedings of the 52nd ACL, pages 293–303, Baltimore, Maryland, June.

Reference

- Wenzhe Pei, Tao Ge, Baobao Chang, An Effective Neural Network Model for Graph-based Dependency Parsing, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 313–322, Beijing, China, July 26-31, 2015.

- William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics, pages 279–288, Denver, Colorado.

- Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In Proceedings of the 53nd ACL, pages 1744–1753, July.

- Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In Proceedings of the 2015 EMNLP, pages 1197–1206, September.

- Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast on-line training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In Proceedings of the 50th ACL, pages 253– 262, July.

- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidi- rectional LSTM-CRF models for sequence tagging. CoRR,abs/1508.01991.