Deep Structured Prediction: Inference, Reparameterization and Applications

Yao Fu 符尧
The University of Edinburgh

Bytedance 2021.06.01

Motivation

Motivation

Today we will talk about Two important areas in NLP and ML …

Structured prediction

- CoreNLP: chunking, tagging, NER, parsing, information extraction
- General structured prediction in ML: categorical, sets, chains, trees, permutation, ranking, graph

… and their …

Applications

- CoreNLP pipelines
- Discovering latent structures of language
- Using latent structures for generation

Deep generative models

- VAEs for learning latent representations of data structures
- Structured latent variable models for text

… and more importantly, possibility to help …

Current DL limitations

- Compositionality
- Adversarial Robustness
- Prior knowledge
- Reasoning

Example: Named Entity Recognition

CITY O O O O Country

Beijing is the capital of China

Option 1: pretrained encoder + softmax (Bert paper)

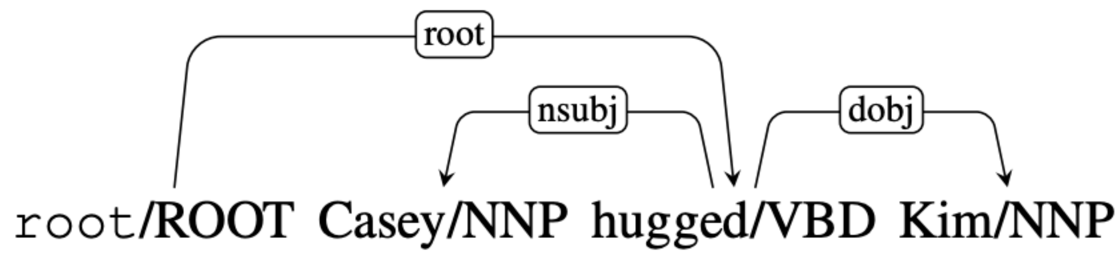
Option 2: pretrained encoder + linear-chain CRF (Wei et. al. NAACL 21, SOTA)

Easy, right?

But what if …

- Noisy supervision
- Unseen entity types
- Adversarial inputs
- Weak / distant / no supervision ?

Example: Dependency Parsing

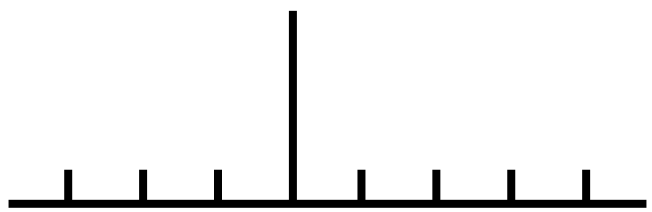


Option 1: pretrained encoder + biaffine (Dozat and Manning. ICLR 16)

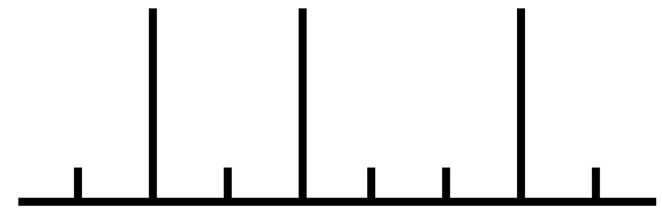
Option 2: pretrained encoder + TreeCRF (Zhang et. al. ACL 20, SOTA)

This one is not easy though

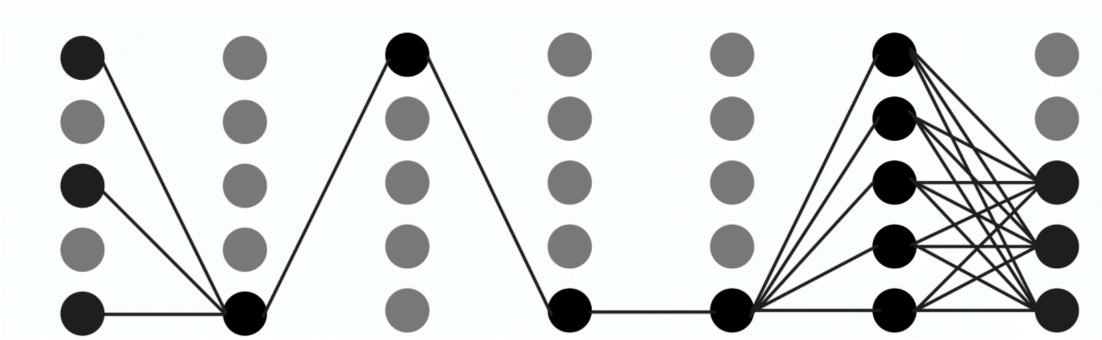
Structured prediction model families



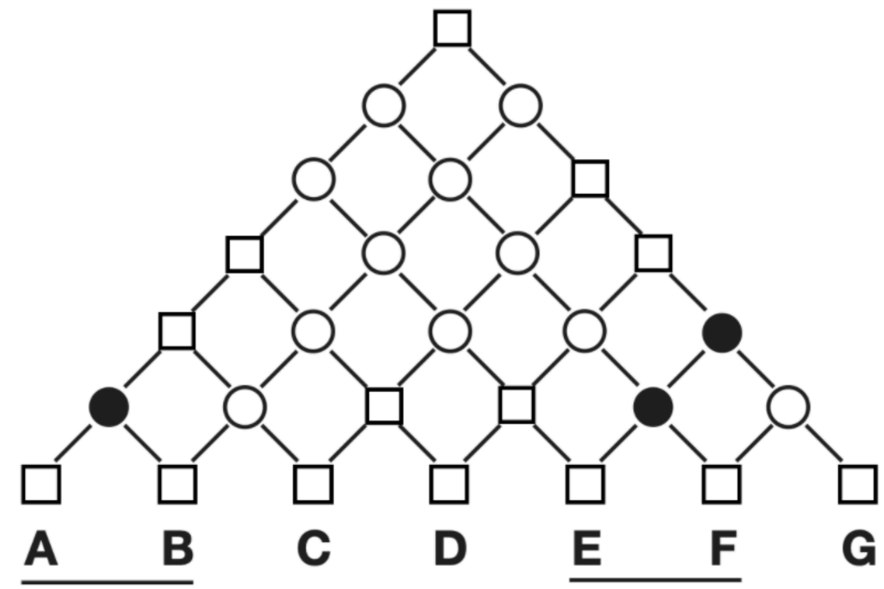
Categorical - softmax



Sets - topK
Bag of words models



Chains - linear-chain CRF
Not just tagging, but also non-autoregressive generation



Trees - Tree CRFs

and there is also: semi-markov, autoregressive, permutation, ranking, search .etc

Using CRFs

- View CRFs as the last layer of neural networks, and view it as an extension of softmax to combinatorial structures

- View all structured prediction models as a box of tools for any task involves structure

- Many SOTA models on classical structured prediction tasks are pretrained LM + CRF (and better than naive approaches like softmax)

- CRFs (as globally-normalized models) were traditionally slower than other locally-normalized models (like softmax), but recently improved with parallelization + tensorization with GPUs

Basics

Bayesian Learning

Bayesian generative model:

- X: data, Y: label/ structures
- Blei: "A model is a joint distribution"

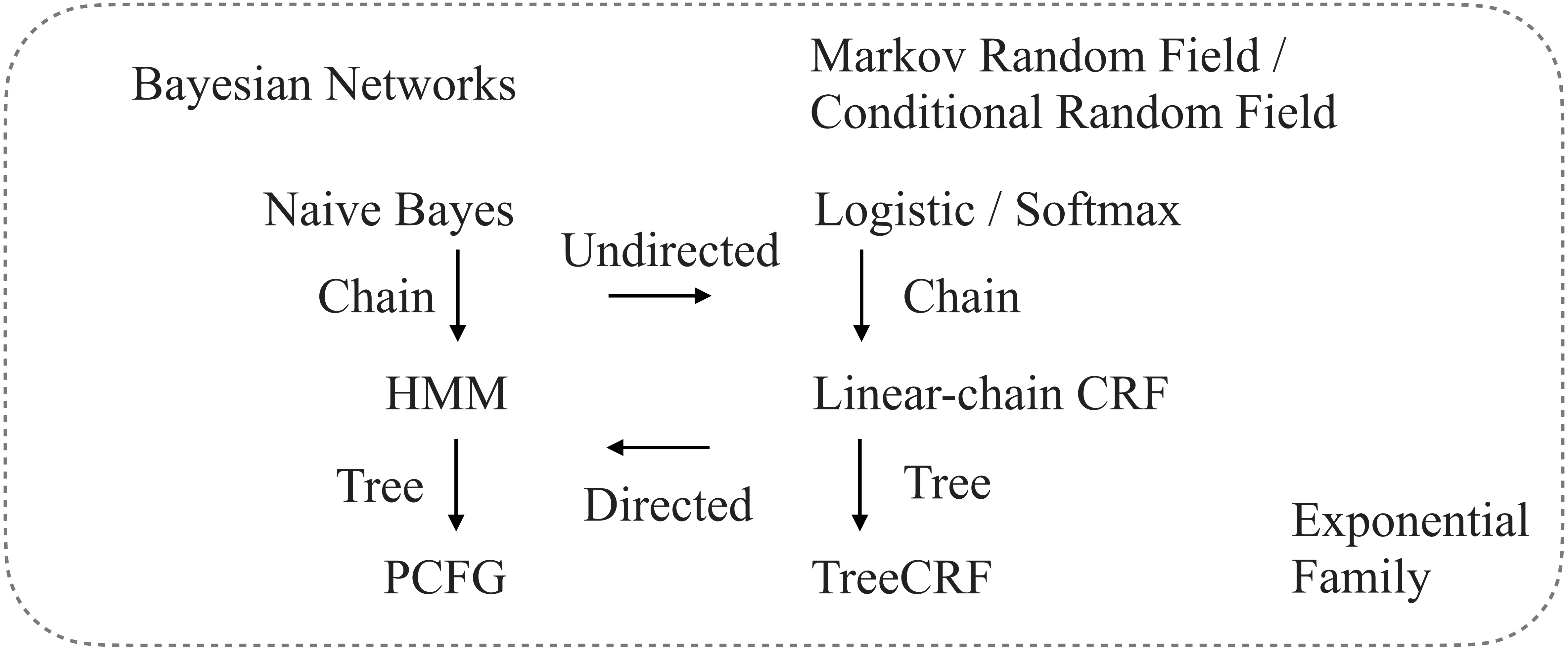
$$p_\theta(X, Y)$$

- We want the posterior distribution

$$\underbrace{p_\theta(Y \mid X)}_{\text{Posterior}} = \frac{\overbrace{p_\theta(X \mid Y)}^{\text{Conditional}}\,\overbrace{p_\theta(Y)}^{\text{Prior}}}{\underbrace{p(X)}_{\text{Marginal Likelihood}}} = \frac{p_\theta(X \mid Y)p_\theta(Y)}{\underbrace{\sum_y p(X, Y)}_{\text{Computationally expensive or intractable. E.g., y is a tree}}}$$

- View neural networks as input features, use graphical models to construct distributions
- We focus on distributions over discrete structures
- How to construct distributions over discrete structures?

Graphical Models as Exponential Family



Use the exponential family to unify all above models
The challenge here is to do inference over these models

Bishop 06. Pattern Recognition and Machine Learning
Murphy 12. Machine Learning: a Probabilistic Perspective
Sutton 12. An introduction to Conditional Random Fields
Jordan 08. Graphical Model, Exponential Family, and Variational Inference

Inference over probabilistic models

View a generative model as a class
View inference as a set of member functions

Class $p_\theta(X, Y)$

Def Posterior(x, y)

return $p_\theta(y \mid x)$

Def Argmax(x)

return $y = \arg\max_y p_\theta(y \mid x)$

Def Marginal(x)

return $p_\theta(x) = \sum_y p_\theta(x, y)$

Def Entropy()

$\cdots$

Def Sample()

$\cdots$

Def Reparam-Sample()

$\cdots$

An inference operation is a function. Inside the function, is:

- When computation is tractable, then exact inference:
  - Simple computation
  - Dynamic Programming
- When computation is intractable, then approximate inference:
  - Search
  - Markov Chain Monte Carlo
  - Variational Inference

Today we focus on

$$\begin{tikzcd}
 & \text{Naive Bayes} \arrow[d, "\text{Chain}"'] \arrow[r, "\text{Undirected}"] & \text{Softmax} \arrow[d, "\text{Chain}"] & \\
\text{Viterbi / Forward} & \text{HMM} \arrow[d, "\text{Tree}"'] & \text{Linear-chain CRF} \arrow[d, "\text{Tree}"] \arrow[l, "\text{Directed}"'] & \text{Viterbi / Forward} \\
\text{CKY / Inside} & \text{PCFG} & \text{TreeCRF} & \text{CKY / Inside}
\end{tikzcd}$$

Inference algorithms	Directed models	Undirected models	Inference algorithms

Today we focus on

View a generative model as a class
View inference as a set of member functions

Class $p_\theta(X, Y)$

Def Argmax(x)

Viterbi / CKY

Def Marginal(x)

Forward-backward
Inside-outside

Def Sample()

Forward-filtering backward-sampling

Def Reparam-Sample()

Gumbel-FFBS

An inference operation is a function. Inside the function, is:

- When computation is tractable, then exact computation:
  - Simple computation
  - Dynamic Programming
- When computation is intractable, then approximate computation
  - Markov Chain Monte Carlo
  - Variational Inference

Now

Class $CRF(X, Y)$

Def Argmax(x)

Viterbi / CKY

Def Marginal(x)

Forward-backward

Inside-outside

Def Sample()

Forward-filtering backward-sampling

Def Reparam-Sample()

Gumbel-FFBS

Linear-chain CRF, a classical model for sequence labeling.

But what if noisy supervision?

Noisy Labeled NER with Confidence Estimation

Kun Liu*, Yao Fu*, Chuanqi Tan, Mosha Chen, Ningyu Zhang, Songfang Huang, Sheng Gao
NAACL 2021

Task: Noisy-labeled NER

	Brooklyn	and	Mary	live	in	New	York
Gold Labels	B-PER	O	B-PER	O	O	B-LOC	I-LOC
Noisy Labels	B-LOC	O	B-PER	O	O	O	B-LOC

- Real-world data inevitably involve annotation noise
- Noise could significantly harm model performance
- Challenge: do not know which labels are noisy
- Our method: confidence score estimation

Confidence Estimation

- Base model: BiLSTM-CRF

$$h = \mathrm{BiLSTM}(x) \qquad \Phi_i = \mathrm{Linear}(h_i) \qquad p(y\,|\,x) = \Phi(y)/Z \qquad \alpha, Z = \mathrm{Forward}(\Phi)$$

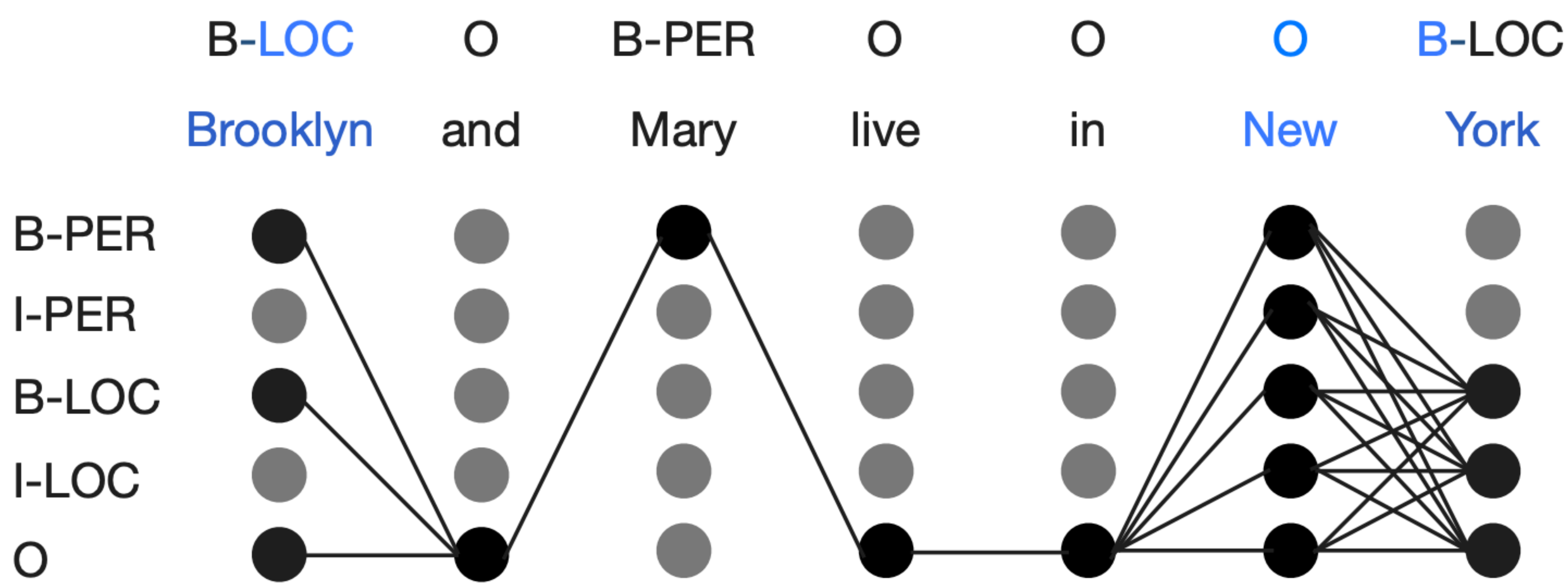
- Global strategy:
  - Intuition: annotators are more likely to make mistakes if they have already made mistakes on previous labels
  - Use CRF marginal probability, strong dependency between consecutive labels

$$\underbrace{s_i}_{\text{Confidence score}} = p(\hat{y}_i\,|\,x) \qquad p(y_i\,|\,x) = \underbrace{\alpha_i}_{\text{Forward variable}} \overbrace{\beta_i}^{\text{Backward variable}}/Z$$

- Local strategy:
  - Intuition: annotators make mistakes solely based on words, no matter whether they have already made mistakes previously
  - Use categorical distribution parameterized by a softmax, a noisy label only relies on the word context

$$s_i = p(\hat{y}_i\,|\,x) \qquad p(y_i\,|\,x) = \mathrm{Softmax}(\Phi_i)$$

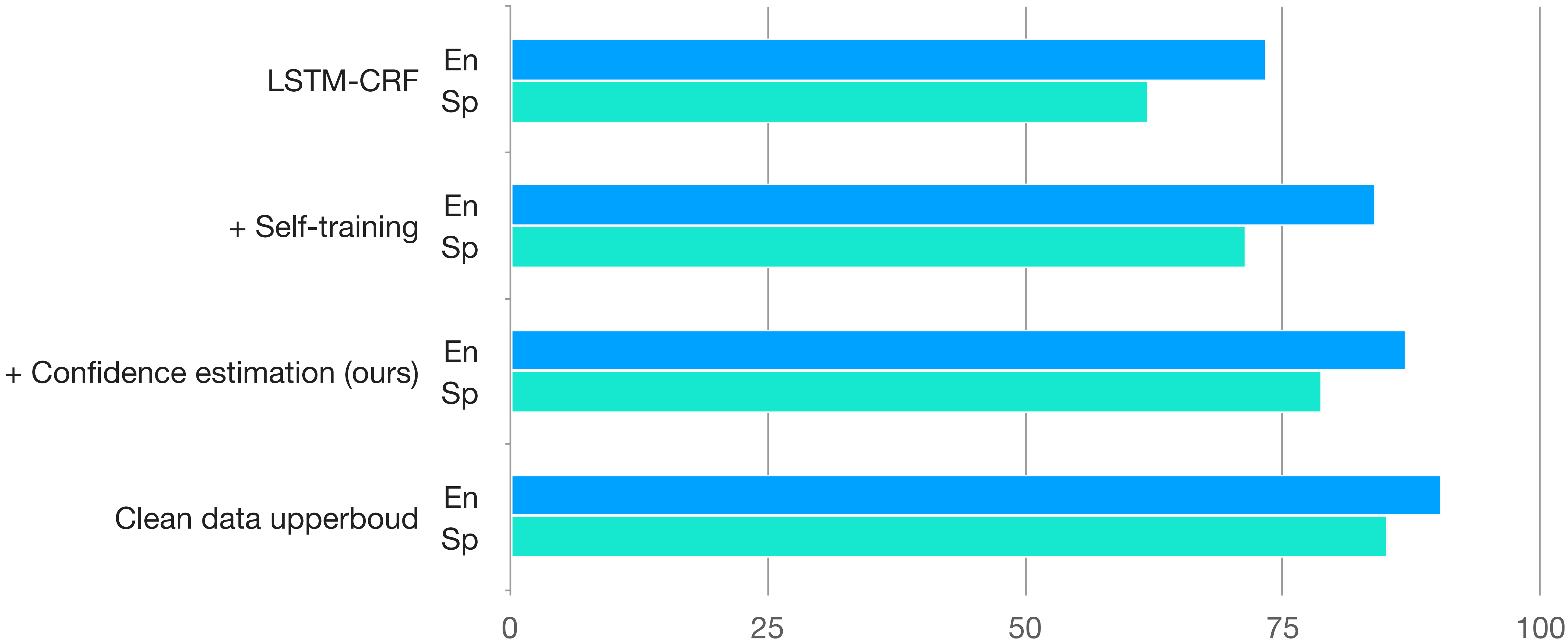
Partial Marginalization



- Update keep ratio r during training:
  - Keep cases with top r% confidence scores and view them as clean
  - View the rest (1-r)% as noisy
  - Use separate r for positive and negative
- Gradually decrease r with linear schedule

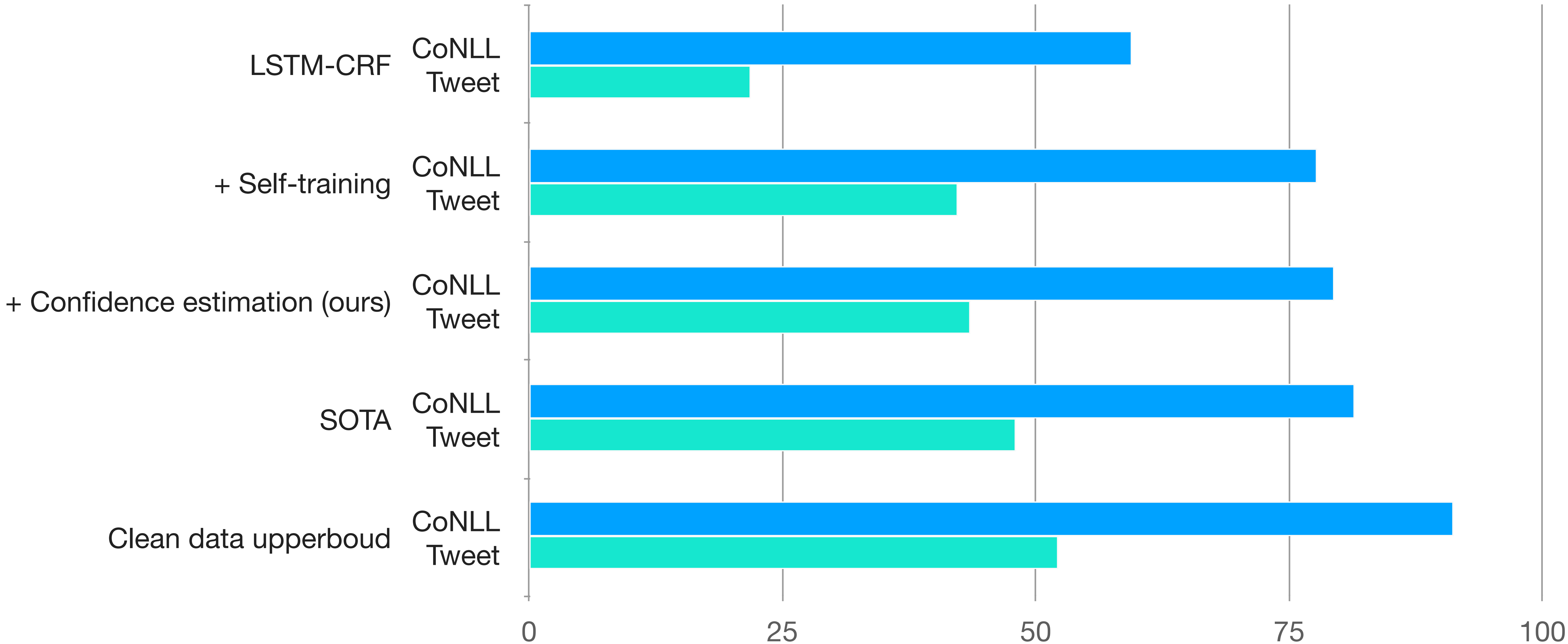
$$r_l(e) = 1 - \min\left\{\frac{e}{K}\tau_l, \tau_l\right\}, l \in \{\mathrm{p}, \mathrm{n}\}$$

Results - General Noise



- Baseline LSTM-CRF performs badly in noisy setting
- Self-training improves performance
- Our confidence estimation method (build upon self training) further improves performance
- Also notice if data is clean, then the simple model would perform the best

Results - Distant Supervision



- Baseline LSTM-CRF still performs badly
- Our confidence estimation method consistently improve performance upon self-training
- SOTA model use tailored methods in this setting, ours is orthogonal
- Again notice if data is clean, then the simple model would perform the best

Back to Confidence Estimation

- Base model: BiLSTM-CRF

  $h = \text{BiLSTM}(x) \quad \Phi_i = \text{Linear}(h_i) \quad p(y|x) = \Phi(y)/Z \quad \alpha, Z = \text{Forward}(\Phi)$

- Global strategy:
  - Intuition: annotators are more likely to make mistakes if they have already made mistakes on previous labels
  - Use CRF marginal probability, strong dependency between consecutive labels

$$\underbrace{s_i}_{\text{Confidence score}} = p(\hat{y}_i | x) \qquad p(y_i | x) = \underbrace{\alpha_i}_{\text{Forward variable}} \overbrace{\beta_i}^{\text{Backward variable}} / Z$$

How to compute the marginal probability?

Implementation of Marginal Computation: Parallelization + Automatic Differentiation

Eisner 16. Inside-Outside and Forward-Backward Algorithms Are Just Backprop
Rush 20. Torch-Struct: Deep Structured Prediction Library

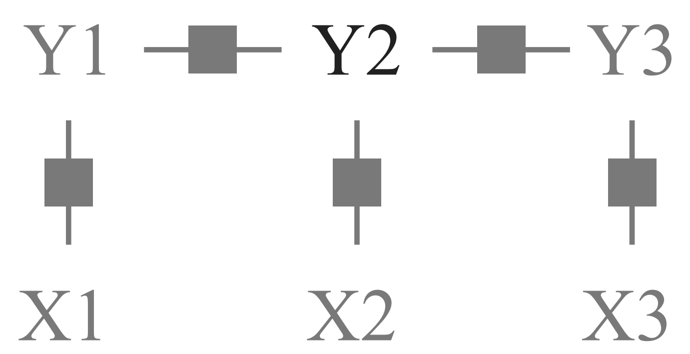
Marginal Probability

Computing the marginal probability is important for all probabilistic models

Class $p_\theta(X, Y)$

Def Marginal(x)

return $p_\theta(x) = \sum_y p_\theta(x, y)$



E.g., what is the marginal probability of the second word being a location?

$$p(Y_2 = \text{LOC} \mid x) = \sum_{y_1, y_3} p(Y_2 = \text{LOC}, Y_1 = y_1, Y_3 = y_3 \mid x)$$

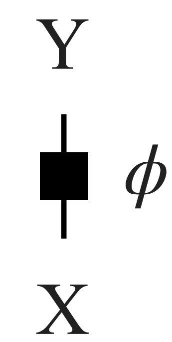
Starting from softmax

Class $p_\theta(X, Y)$

Def Posterior(x, y)

return $\log p_\theta(Y = y_i \mid x) = \log \mathsf{softmax}(\phi(y_i, x))$

$$= \phi(y_i, x) - \underbrace{\mathsf{logsumexp}_j(\phi(y_j, x))}_{=\log Z}$$



$$\frac{\partial \log Z}{\partial \phi(y_i, x)} = p(y_i \mid x)$$

Gradient of the log partition function Marginal probability of the factor

So if we want to compute the probability:

$p_\theta(Y = y_i \mid x) = \mathsf{softmax}(\phi(y_i, x))$

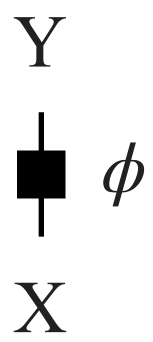
Method 1

$\log Z = \mathsf{logsumexp}_j(\phi(y_j, x))$

$p(y \mid x) = \nabla_\phi \log Z$

Method 2

Starting from softmax



$p_\theta(Y = y_i \mid x) = \mathsf{softmax}(\phi(y_i, x))$

Method 1

$\log Z = \mathsf{logsumexp}_j(\phi(y_j, x))$

$p(y \mid x) = \nabla_\phi \log Z$

Method 2

Class $p_\theta(X, Y)$

Def Posterior(x, y)

$\log Z = \mathsf{logsumexp}_j(\phi(y_j, x))$

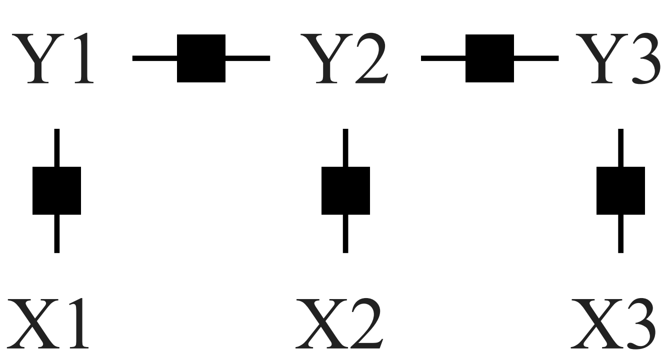
$\log Z.\mathsf{backward}()$

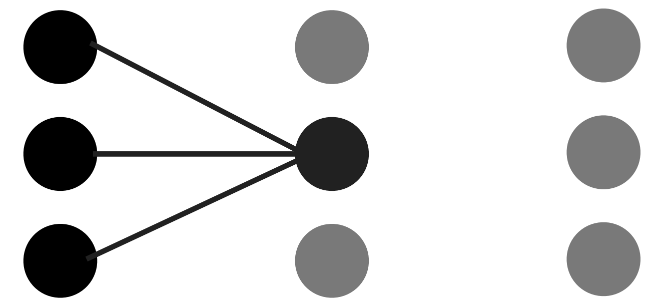
$p(y \mid x) = \phi.\mathsf{grad}$

So we can implement with autodiff

… But what is the use of it?

From softmax to CRF





$$\alpha_t(i) = \sum_j \alpha_{t-1}(j)\phi_{t-1}(j,i)$$

Class $p_\theta(X, Y)$

Def Posterior(x, y)

$\log Z = \text{logsumexp}_j(\phi(y_j, x))$ $\longrightarrow$

$\log Z.\text{backward}()$

$p(y \mid x) = \phi.\text{grad}$

Class $CRF(X_{1:T}, Y_{1:T})$

Def Marginal(x, y)

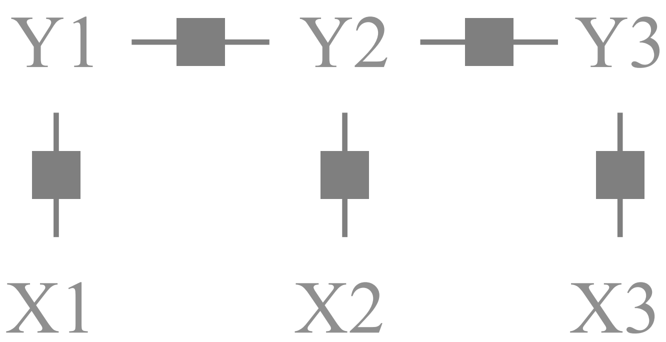
$\log Z = \text{Forward}(\phi(y, x))$

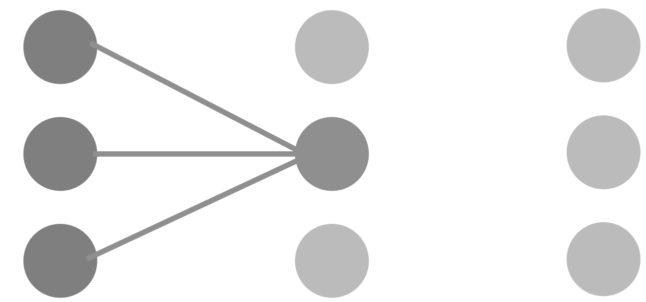
$\log Z.\text{backward}()$

$p(Y_t = i, Y_{t+1} = j \mid x) = \phi_t(i, j).\text{grad}$

$$p(Y_t = i \mid x) = \sum_j p(Y_t = i, Y_{t+1} = j \mid x)$$

From softmax to CRF





$$\alpha_t(i) = \sum_j \alpha_{t-1}(j)\phi_{t-1}(j, i)$$

Class $CRF(X_{1:T}, Y_{1:T})$

Def Marginal(x, y)

$$\alpha, \log Z = \text{Forward}(\phi(y, x))$$

$$\beta = \text{Backward}(\phi, \alpha)$$

$$p(Y_t = i \mid x) = \frac{\alpha_t(i)\beta_t(i)}{Z}$$

$\longrightarrow$

Class $CRF(X_{1:T}, Y_{1:T})$

Def Marginal(x, y)

$$\log Z = \text{Forward}(\phi(y, x))$$

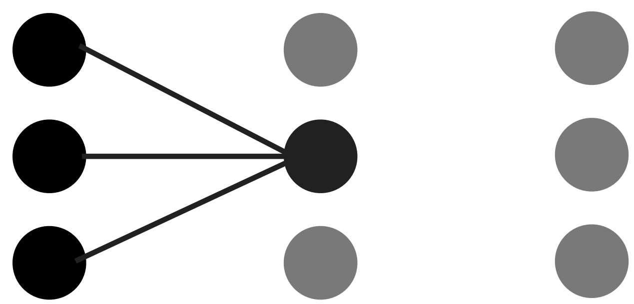
$$\log Z\,.\,\text{backward}()$$

$$p(Y_t = i, Y_{t+1} = j \mid x) = \phi_t(i, j)\,.\,\text{grad}$$

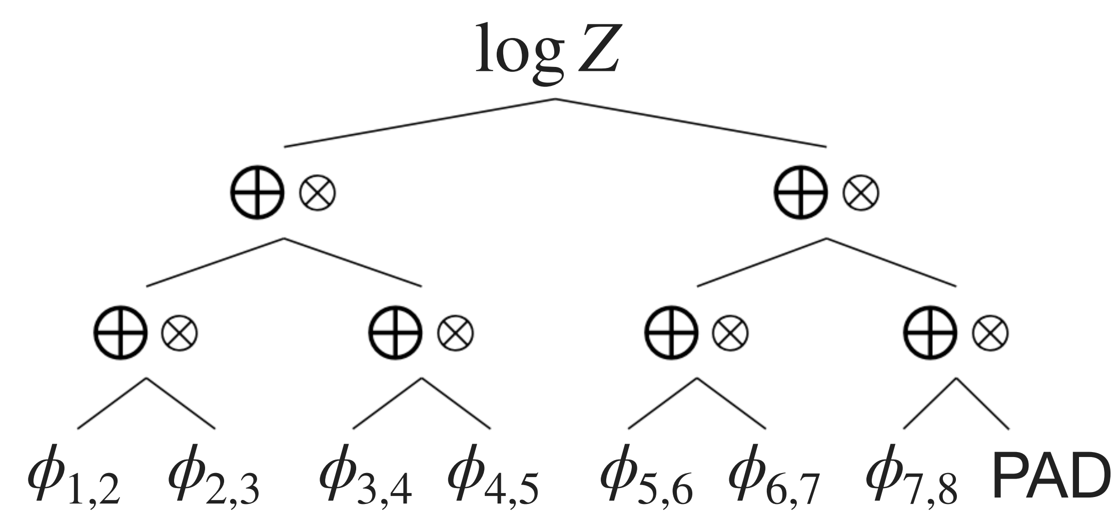
$$p(Y_t = i \mid x) = \sum_j p(Y_t = i, Y_{t+1} = j \mid x)$$

With autodiff we do not need to implement the troublesome Backward algorithm
Usually autodiff gives speedup with the optimization inside Pytorch (e.g., logsumexp)

Parallel Scanning



$$\alpha_t(i) = \sum_j \alpha_{t-1}(j)\phi_{t-1}(j,i)$$



Class $CRF(X_{1:T}, Y_{1:T})$

Def Marginal(x, y)

$\log Z = \text{Forward}(\phi(y,x))$

$\log Z.\text{backward}()$

$p(Y_t = i, Y_{t+1} = j \mid x) = \phi_t(i,j).\text{grad}$

$$p(Y_t = i \mid x) = \sum_j p(Y_t = i, Y_{t+1} = j \mid x)$$

Previously: Linear time complexity O(T)
Parallel scanning: O(log T)

With Parallel scanning + GPU + Autodiff we achieve both efficiency and performance

Sarkka and Garcia-Fernandez. 18. Temporal Parallelization of Bayesian Smoothers

Until now

Class $CRF(X, Y)$

Def Argmax(x)

Viterbi / CKY

Def Marginal(x)

Forward-backward
Inside-outside

Parallel Scanning + Autodiff + GPU

Performance SOTA, usually better than alternatives
As efficient as alternatives

Now

Class $CRF(X, Y)$

Def Argmax(x)

Viterbi / CKY

Def Marginal(x)

Forward-backward
Inside-outside

Def Sample()

Forward-filtering backward-sampling

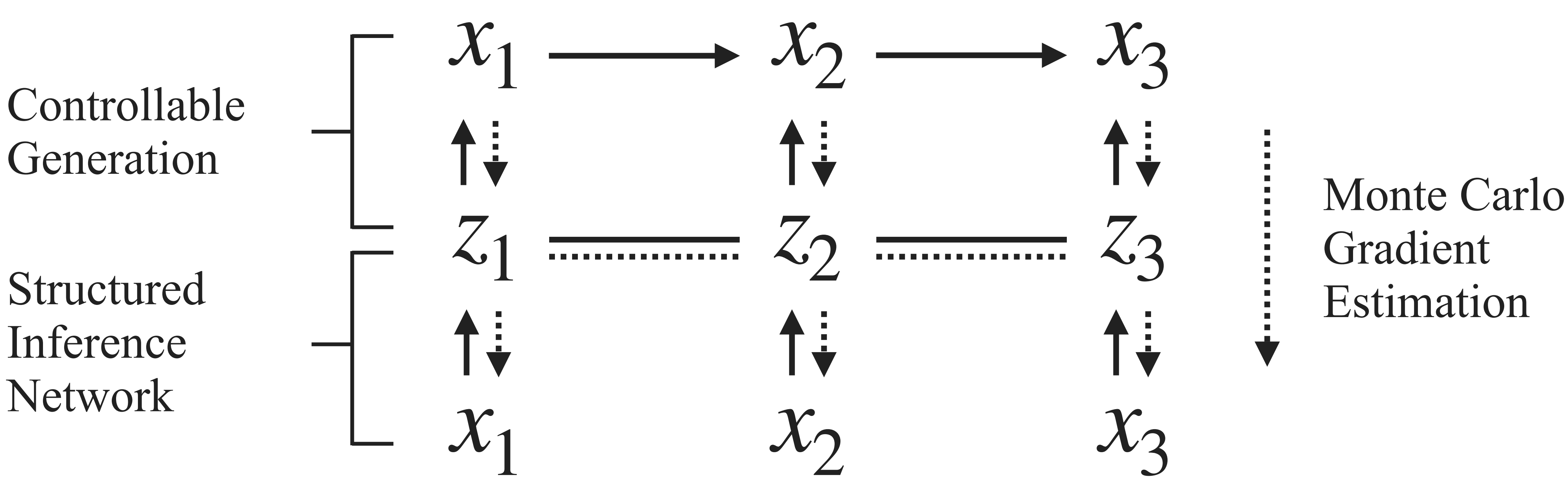
Def Reparam-Sample()

Gumbel-FFBS

Latent Template Induction with Gumbel-CRFs

Yao Fu, Chuanqi Tan, Mosha Chen, Yansong Feng, Alexander Rush
NeurIPS 2020

Controllable Text Generation with Latent Template



- Use templates z to control the structure of sentence x
- Infer z with linear-chain CRF
- Efficiently train z with reparameterized MC grad

Monte Carlo Gradient Estimation

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\log p(x|z)] \qquad \text{\# Goal of MC grad.}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\underbrace{\log p(x|z)}_{\text{Reward}} \cdot \underbrace{\nabla_\phi \log q(z|x)}_{\text{Score}}] \qquad \text{\# High var. hard to train}$$

$$= \mathbb{E}_{g(\epsilon)}[\nabla_\phi \log p(x|\underbrace{z(\epsilon, \phi)}_{\text{Reparameterization}})] \qquad \text{\# Lower var. more stable training}$$

$$= \mathbb{E}_{g(\epsilon)}[\nabla_z \log p(x|z) \odot \underbrace{\nabla_\phi \tilde{z}(\epsilon, \phi)}_{\text{Continuous Relaxation}}]$$

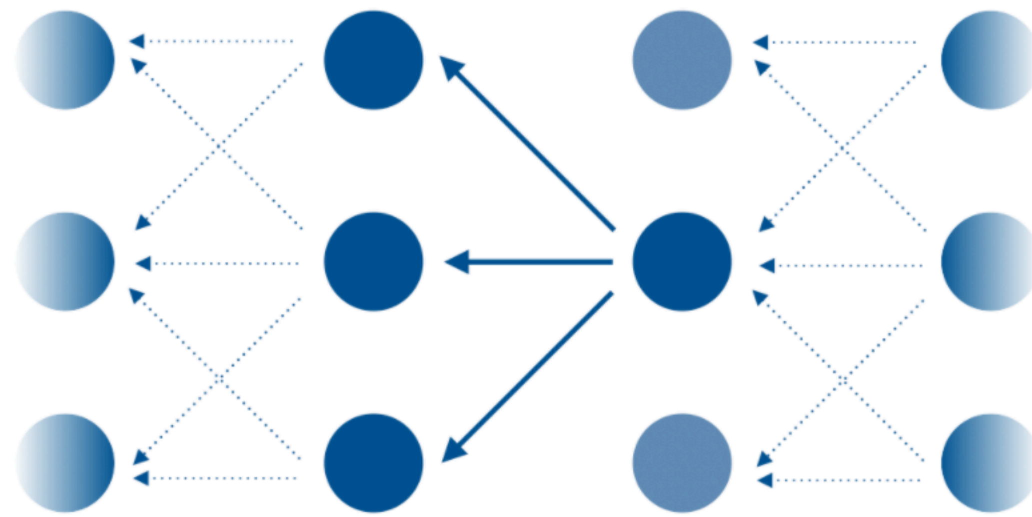
How to?

Large recent ML/ NLP trend

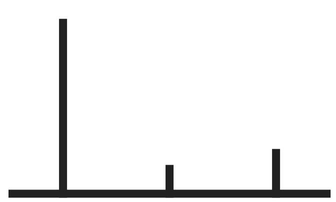
Quite challenging from many aspects

Mohamed et. al. 20. Monte Carlo Gradient Estimation in Machine Learning

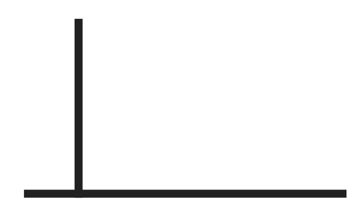
Gumbel-CRF Reparameterization



$$\hat{z}_t \sim p(z_t \mid \hat{z}_{t+1})$$



$$\tilde{z}_t = \text{Gumbel-Softmax}(p(z_t \mid \hat{z}_{t+1}))$$



$$\hat{z}_t = \text{Argmax}(\tilde{z}_t)$$

- Apply Gumbel to each FFBS step to get soft sample $\tilde{z}_t$
- Use Argmax to recover hard sample $\hat{z}_t$

Gumbel-CRF Reparameterization

Algorithm 1 Forward Filtering Backward Sampling

Input: $\Phi(z_{t-1}, z_t, x_t), t \in \{1, .., T\}, \alpha_{1:T}, Z$
Calculate $p(z_T|x) = \alpha_T / Z$
Sample $\hat{z}_T \sim p(z_T|x)$
for $t \leftarrow T-1, 1$ **do**
  $p(z_t|\hat{z}_{t+1}, x) = \frac{\Phi(z_t, \hat{z}_{t+1}, x_{t+1})\alpha_t(z_t)}{\alpha_{t+1}(\hat{z}_{t+1})}$
  Sample $\hat{z}_t \sim p(z_t|\hat{z}_{t+1}, x)$
end for
Return $\hat{z}_{1:T}$

Algorithm 2 Gumbel-CRF (Forward Filtering Backward Sampling with Gumbel-Softmax)

Input: $\Phi(z_{t-1}, z_t, x_t), t \in \{1, .., T\}, \alpha_{1:T}, Z$
Calculate:
  $\pi_T = \alpha_T / Z$
  $\tilde{z}_T = \text{softmax}((\log \pi_T + g)/\tau)$, $g \sim \text{G}(0)$
  $\hat{z}_T = \text{argmax}(\tilde{z}_T)$
for $t \leftarrow T-1, 1$ **do**
  $\pi_t = \frac{\Phi(z_t, \hat{z}_{t+1}, x_{t+1})\alpha_t(z_t)}{\alpha_{t+1}(\hat{z}_{t+1})}$
  $\tilde{z}_t = \text{softmax}((\log \pi_t + g)/\tau)$, $g \sim \text{G}(0)$
  $\hat{z}_t = \text{argmax}(\tilde{z}_t)$
end for
Return $\hat{z}_{1:T}, \tilde{z}_{1:T}$ ▷ $\tilde{z}$ is a relaxation for $\hat{z}$

- Apply Gumbel to each FFBS step to get soft sample $\tilde{z}_t$
- Use Argmax to recover hard sample $\hat{z}_t$

Alternative: REINFORCE

Estimators	Score /Reparam.	Seq. Level/ Stepwise	Unbiased MC Sample	Unbiased Grad.
REINFORCE-MS	Score	Seq.	Unbiased	Unbiased
REINFORCE-MS-C	Score	Seq.	Unbiased	Unbiased
PM-MRF	Reparam.	Step	Biased	Biased
PM-MRF-ST	Reparam.	Step	Biased	Biased
Gumbel-CRF	Reparam.	Step	Biased	Biased
Gumbel-CRF-ST	Reparam.	Step	Unbiased	Biased

(A) Characteristics of the estimators we compare

Gradient of REINFORCE:

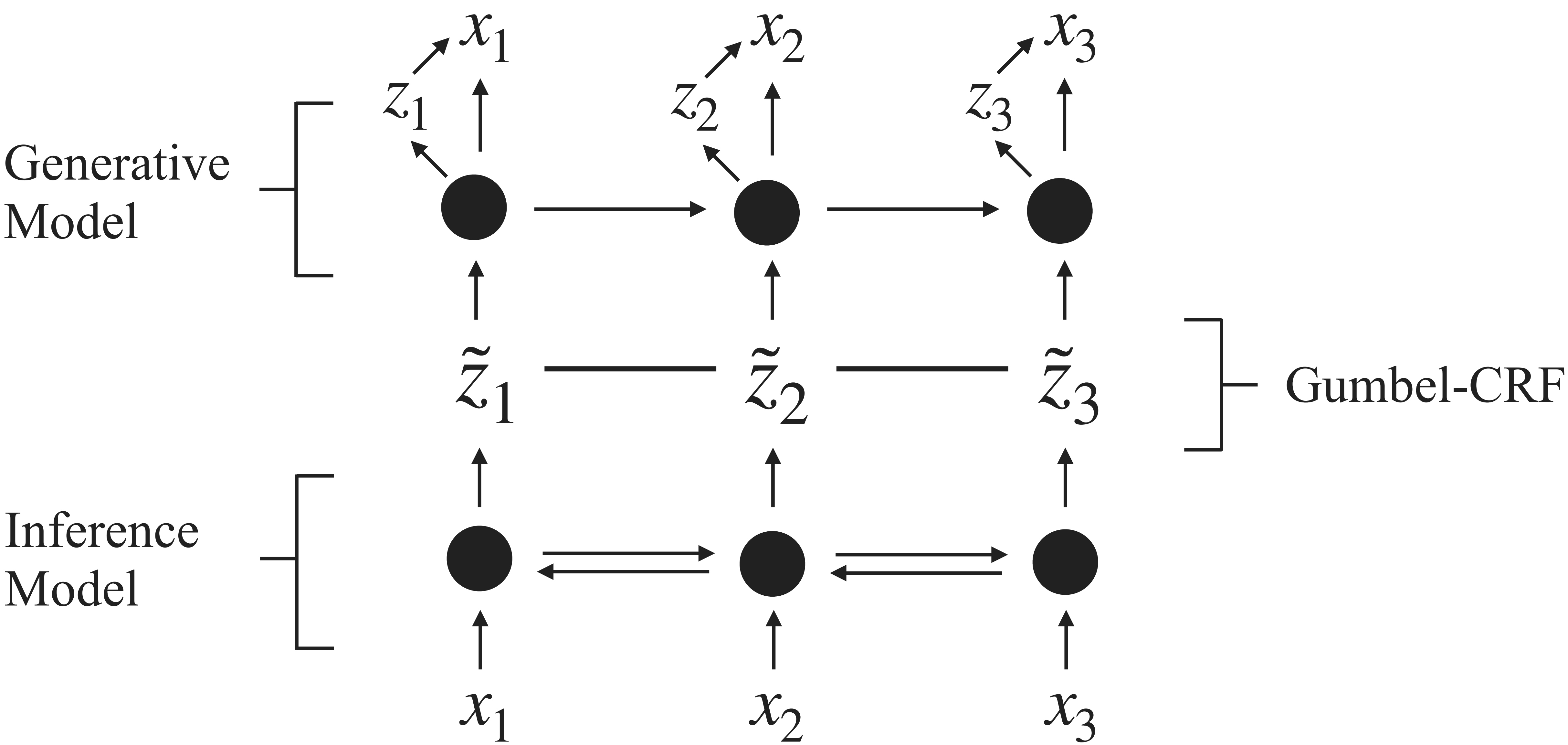
$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\log p(x, z)] = \mathbb{E}_{q_\phi(z|x)}[\log p(x, z)\, \nabla_\phi \log q_\phi(z \mid x)]$$

Gradient of Gumbel-CRF:

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\log p(x, z)] = \mathbb{E}_{q_\phi(z|x)}[\Sigma_i \nabla_{\tilde{z}_i} \log p(x, z) \odot \nabla_\phi \tilde{z}_i]$$

Same time, closely related, more general setting:
Paulus et. al. 20. Gradient Estimation with Stochastic Softmax Tricks
Berthet et. al. 20. Learning with Differentiable Perturbed Optimizers
Blondel et. al. 20. Learning with Fenchel-Young Losses

VAE w. Gumbel-CRF



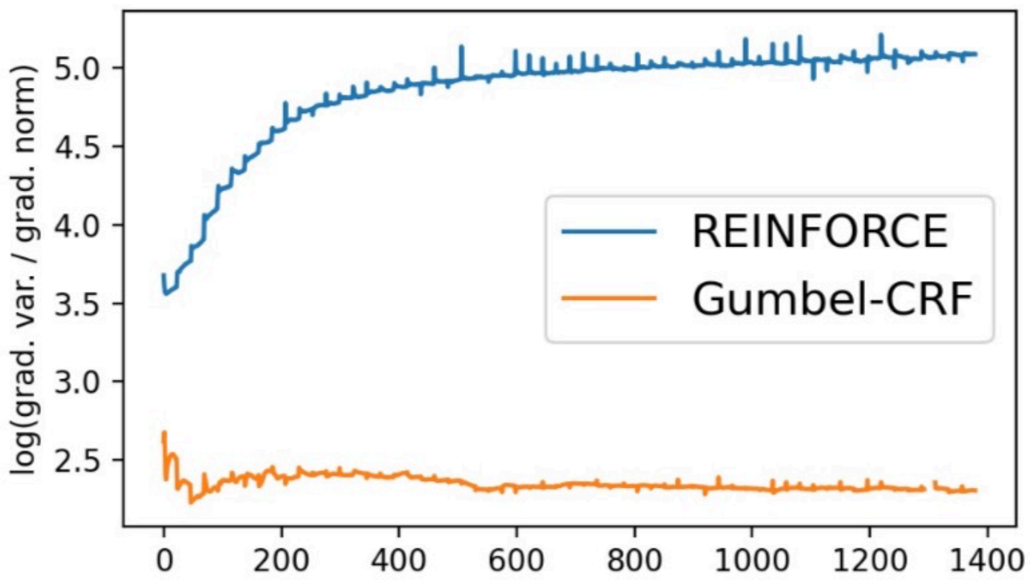
- Generative model autoregressive w.r.t. x and z (Li and Rush 2020)
- Inference model relaxed w. Gumbel-CRF

As a reparam.ed grad. estimator: density estimation

Table 1: Density Estimation Results. NLL is estimated with 100 importance samples. Models are selected from 3 different random seeds based on validation NLL. All metrics are evaluated on the discrete (exact) model.

Model	Neg. ELBO	NLL	PPL	Ent.	#sample
RNNLM	34.69	4.94	-	-	
PM-MRF	69.15	50.22	10.41	4.11	1
PM-MRF-ST	53.16	37.03	5.48	2.04	1
REINFORCE-MS	35.11	34.50	4.84	3.48	5
REINFORCE-MS-C	34.35	33.82	4.71	3.34	5
Gumbel-CRF (ours)	38.00	35.41	4.71	3.03	1
Gumbel-CRF-ST (ours)	34.18	33.13	4.54	3.26	1

- Gumbel-CRF ST version achieves best NLL and PPL w. less sample than baseline REINFORCE.



- Less variance than REINFORCE, more stable training

As a structured inference network: controllable generation

name: clowns | eattype: coffee shop | food: chinese | customer_rating: 1 out of 5 | area: riverside | near: clare hall

1. [there is a]$_{20}$ [coffee shop]$_{35}$ [in the]$_{9}$ [riverside]$_{35}$ [area ,]$_{12}$ [serves]$_{20}$ [chinese]$_{35}$ [food]$_{12}$ [. it is]$_{20}$ [called]$_{35}$ [clowns]$_{44}$ [. is]$_{20}$ [near]$_{35}$ [clare hall]$_{44}$ [. It has a customer rating]$_{20}$ [of 1 out of 5]$_{8}$ [.]$_{20}$
2. [clowns]$_{44}$ [is a]$_{20}$ [expensive]$_{12}$ [coffee shop]$_{35}$ [located]$_{12}$ [in]$_{9}$ [riverside]$_{35}$ [area]$_{12}$ [.]$_{20}$
3. [clowns]$_{44}$ [is a]$_{20}$ [coffee shop]$_{35}$ [in the riverside]$_{9}$ [. it is]$_{20}$ [family friendly]$_{12}$ [and has a]$_{20}$ [1]$_{45}$ [out of 5]$_{8}$ [stars]$_{12}$ [rating .]$_{20}$

name: browns cambridge | eattype: coffee shop | food: chinese | customer_rating: 1 out of 5 | area: riverside | familyfriendly: yes | near: crowne plaza hote

1. [browns cambridge]$_{44}$ [offers]$_{12}$ [chinese]$_{35}$ [food]$_{12}$ [near]$_{35}$ [crowne plaza hotel]$_{44}$ [in]$_{35}$ [riverside]$_{9}$ [. it is a]$_{20}$ [coffee shop]$_{35}$ [, not children friendly]$_{12}$ [and has a]$_{20}$ [5]$_{45}$ [out of 5]$_{8}$ [rating .]$_{20}$
2. [there is a]$_{20}$ [moderately priced restaurant]$_{12}$ [that serves]$_{20}$ [chinese]$_{35}$ [food]$_{12}$ [called]$_{35}$ [browns cambridge]$_{44}$ [coffee]$_{9}$ [. it has a customer rating]$_{20}$ [of 5 out of 5.]$_{8}$ [it is]$_{20}$ [not family-friendly]$_{12}$ [. it is]$_{20}$ [located]$_{12}$ [near]$_{35}$ [crowne plaza]$_{44}$
3. [browns cambridge]$_{44}$ [is a]$_{20}$ [chinese coffee shop]$_{35}$ [located]$_{12}$ [in]$_{9}$ [riverside near]$_{35}$ [crowne plaza hotel]$_{44}$ [. it has a]$_{20}$ [customer rating]$_{20}$ [of 5 out of]$_{8}$ [5]$_{44}$ [and is]$_{20}$ [not family-friendly]$_{12}$ [.]$_{20}$

Bigram	Sentence Segments	4gram	Sentence Segments	
(A) 12-35	1. located near	(D) 35-44-12-20	1. near the city center	
	2. restaurant near		2. near café rouge, there is a	
	3. restaurant located near		3. in the city center, it is	ngrams w. semantically similar segments
(B) 20-8	1. has a customer rating of	(E) 44-20-35-20	1. french food at a moderate	
	2. has a customer rating of 5 out of		2. french food for a moderate	
	3. and with a customer rating of		3. fast food restaurant with a moderate	
(C) 20-12	1. is located	(F) 12-20-12-20	1. food with a price range of	ngrams w. semantically different segments
	2. is a family friendly		2. price range and family friendly	

Until Now

Class $CRF(X, Y)$

Def Sample()
Forward-filtering backward-sampling

Embed Gumbel-Softmax into each sampling step

Gradients with lower variance

Def Reparam-Sample()
Gumbel-FFBS

Useful for inducing sentence templates

Def Argmax(x)
Viterbi-backtracking

Def Marginal(x)
Forward-backward

Def Entropy()
Forward-backward styled DP

Furthermore: all algorithms can be unified with semiring and share the same DP graph.

When reloading the backward function in autodiff, we achieve difference inference operations with the same implementation

Rush 20. Torch-Struct: Deep Structured Prediction Library

Now

Class $TreeCRF(X, Y)$

Def Argmax(x)

Viterbi / CKY

Def Marginal(x)

Forward-backward
Inside-outside

Def Sample()

Forward-filtering backward-sampling

Def Reparam-Sample()

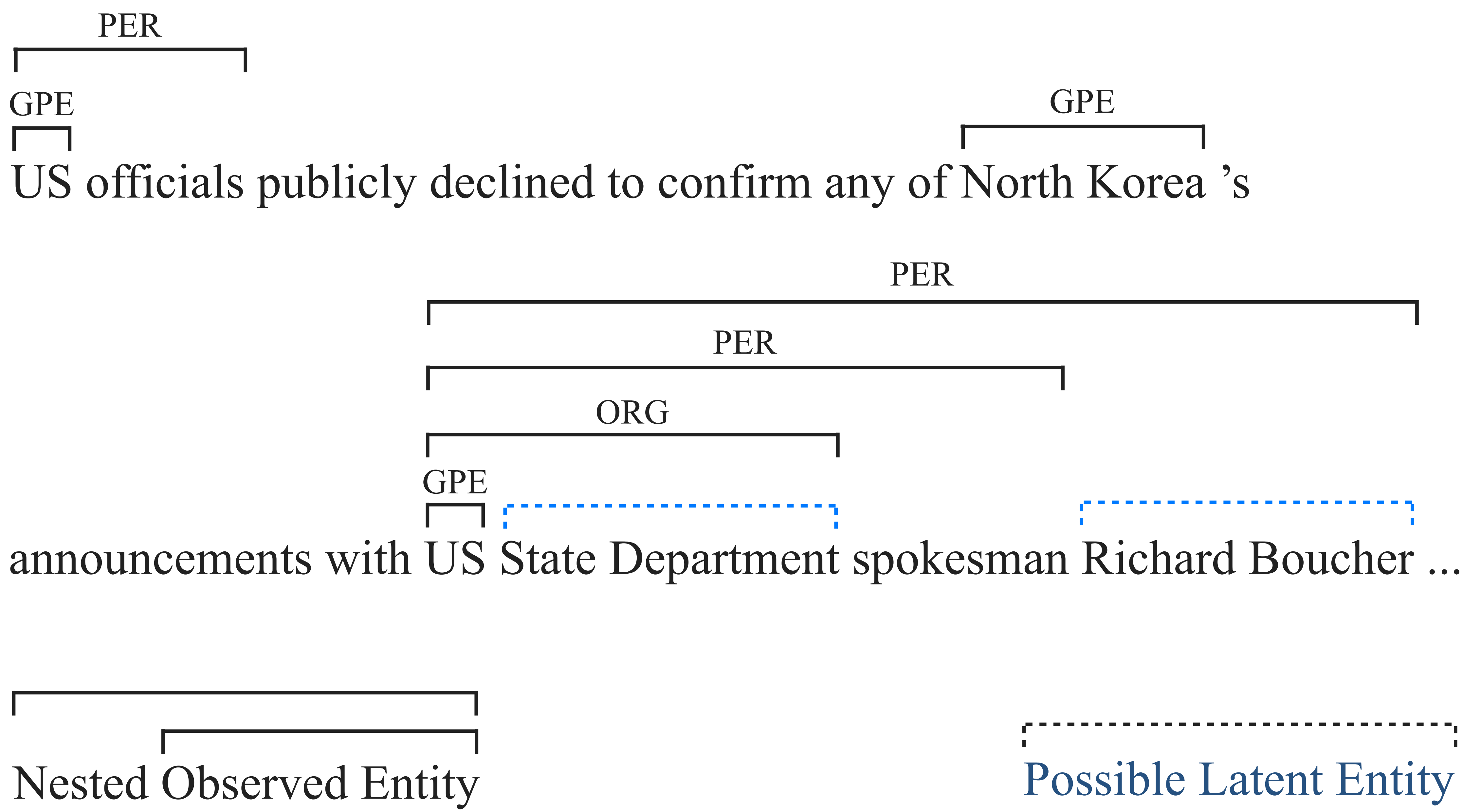
Gumbel-FFBS

Nested Named Entity Recognition with Partially-Observed TreeCRFs

Yao Fu, Chuanqi Tan, Mosha Chen, Songfang Huang, Fei Huang

AAAI 2020

Task: Nested NER



Formulation: constituency parsing with partially observed trees

TreeCRFs

TreeCRFs with partially observed trees:
jointly model observed and latent

… but inefficient (major drawback in previous literature)

No clear batchification:
different sentences, different partial trees
different DP graph

↓

Batchified computation
Unified DP graph

$O(n^3)$ for each sentence

↓

$O(n \log n)$
by parallelized summation

We propose efficient partial marginalization with
Masked Inside

Model

Biaffine Scoring:

Encoder states for word i and j

$$s_{ijk} = h_i W_k h_j + w_k^1 h_i + w_k^2 h_j + b_k$$

Score for an entity
from i to j with label k

Inference use CKY to get max. prob. trees

MLE Training:

Conditional probability for a partial tree

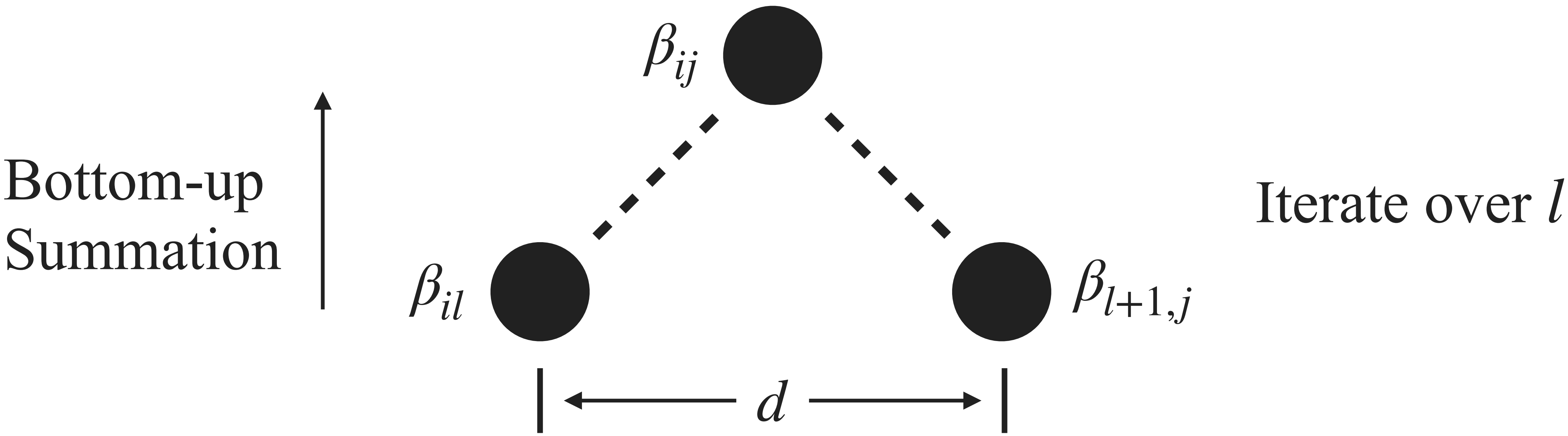
Computed by the Inside Algorithm

$$\log p(T \mid x) = \sum_{t \in \tilde{T}} s(t) - \log Z$$

The set of all possible full trees by completing the partial tree
Computed by an Inside-styled DP

How to do this partial marginalization efficiently?

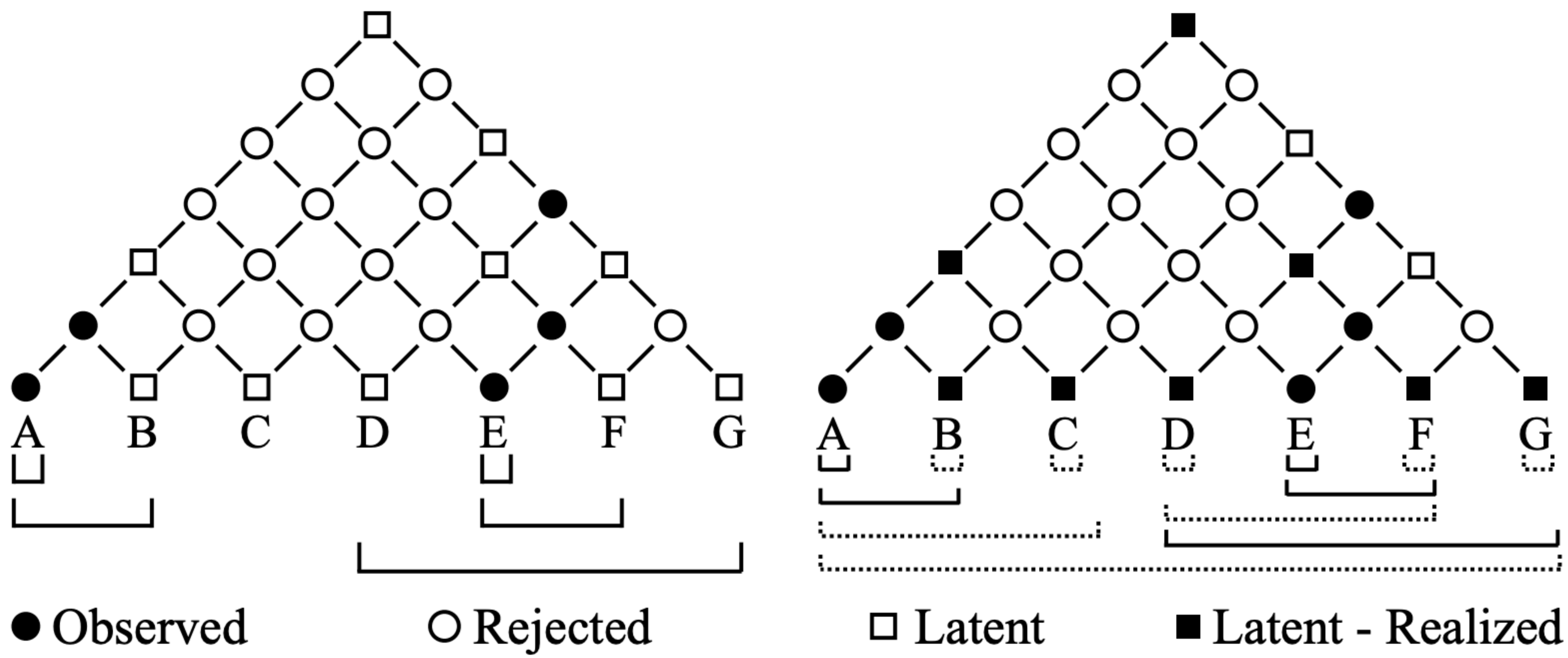
Recap: Inside Algorithm



$$\beta_{ijk} = \exp(s_{ijk}) \cdot \sum_{l=i}^{j-1} \sum_{k_1,k_2} \beta_{ilk_1} \beta_{l+1,j,k_2}$$

- Bachified by nature: different sentences use same DP graph
- Improved recently: $O(n \log n)$ (Zhang e.t al. 20, Rush 20)
- Not for partial marginalization: different sentences use different DP graph
- Can we transform partial marginalization to similar form?

Inference over different types of nodes



Left: an observed partial tree; right: a full tree compatible with the left

… by realizing latent nodes

Masked Inside

$$\beta_{ijk} = \exp(s_{ijk})\cdots$$
Likelihood evaluation

$$\beta_{ijk} = 0$$
Rejection

$$\beta_{ijk} = \sum_{k \in \mathscr{L}_l} \exp(s_{ijk})\cdots$$
Partial Marginalization

$$\beta_{ijk} = \sum_{k} m_{ijk} \cdot \exp(s_{ijk})\cdots$$

$$m_{ijk} = 1, m_{ijk'} = 0$$
k observed tag, k' all other tags

Likelihood evaluation

$$\forall k, m_{ijk} = 0$$
k all tags

Rejection

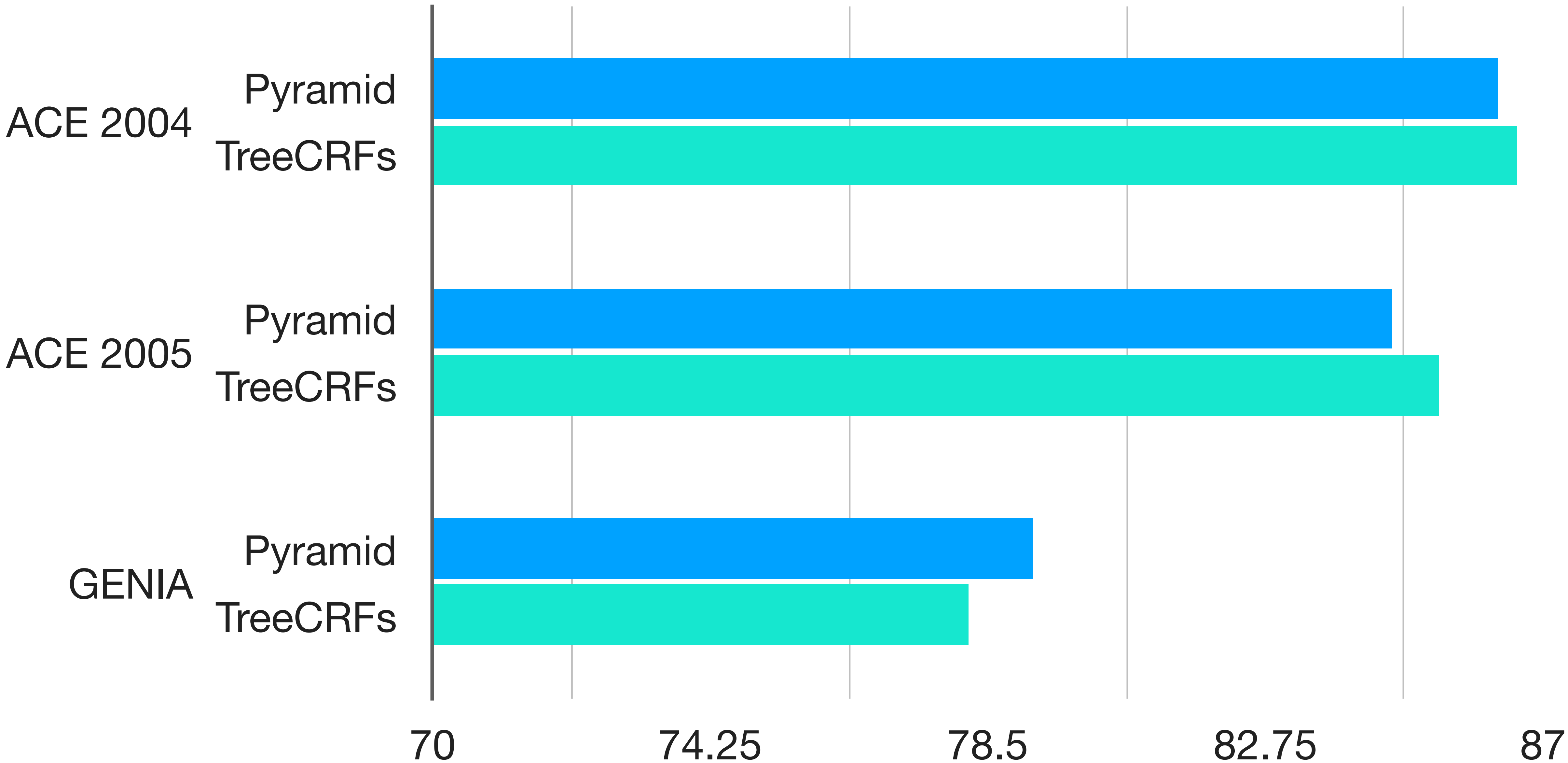
$$\forall k_1, m_{ijk_1} = 1, \forall k_2, m_{ijk_2} = 0$$
k_1 latent tags, k_2 observed tags

Partial Marginalization

Different masks recover different inference operations

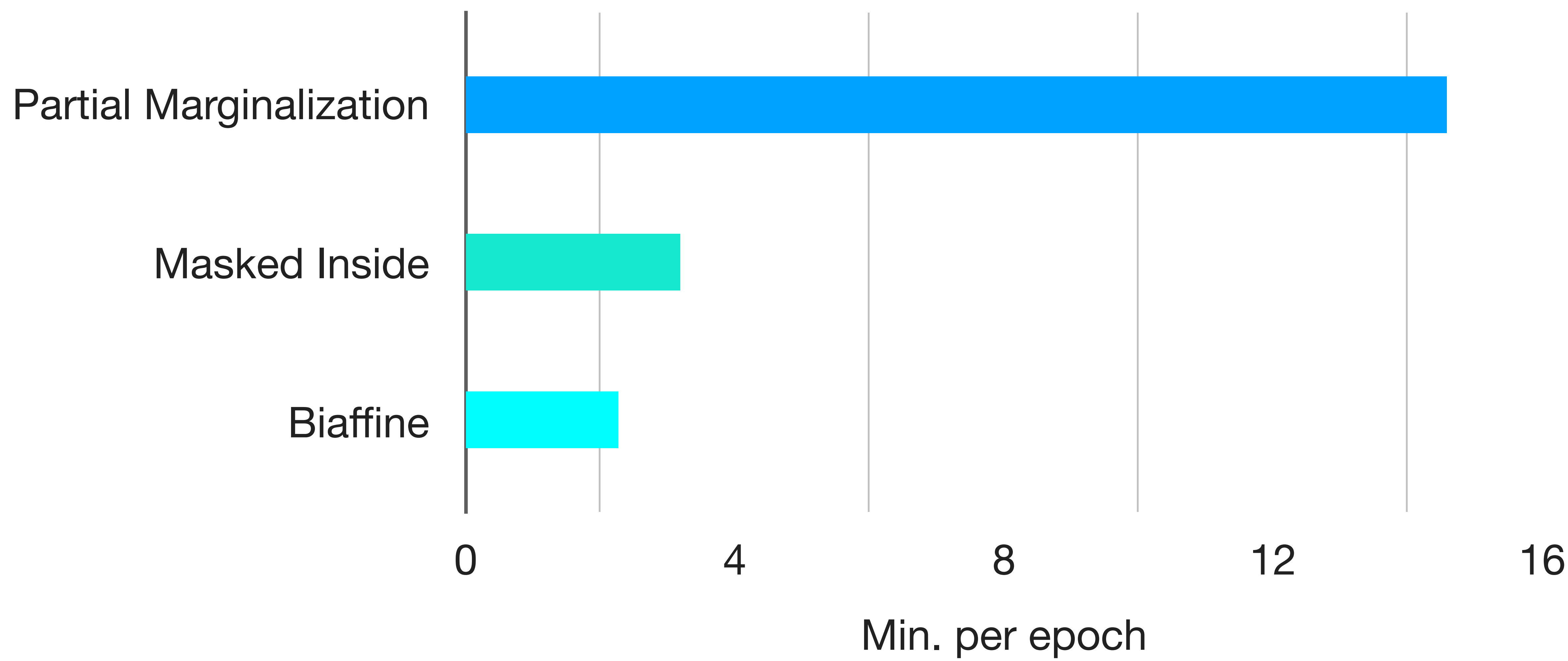
Unified DP graph for different trees, so parallelizable and tensorizable

Performance



Pyramid: tailored layer-by-layer architecture for nested entities
GENIA: imbalanced tags. Future improvement direction

Efficiency



Biaffine: local softmax dist., O(1) complexity, upper bound, worse performance than TreeCRFs

Ours: better performance than Biaffine, slightly slower because of O(n log n) complexity

Conclusion

Conclusion

Softmax

↓ Chain

Linear-chain CRF — Viterbi / Forward, Gumbel-FFBS — Parallel scanning — Fully / noisy / distant supervised: Seq. tagging. Unsupervised: latent structure induction

↓ Tree

TreeCRF — CKY / Inside, Masked Inside — Tensorization — Supervised: Parsing. Partially supervised: nested NER. Unsupervised: grammar induction

Undirected models	Inference algorithms	Implementation: Semiring DP + Autodiff + Parallization + Tensorization	Applications
Softmax → Chain → Linear-chain CRF	Viterbi / Forward Gumbel-FFBS	Parallel scanning	Fully / noisy / distant supervised: Seq. tagging. Unsupervised: latent structure induction
→ Tree → TreeCRF	CKY / Inside Masked Inside	Tensorization	Supervised: Parsing Partially supervised: nested NER Unsupervised: grammar induction

A box of structured prediction models for various learning settings

Thanks

References

Books
Bishop 06. Pattern Recognition and Machine Learning
Murphy 12. Machine Learning: a Probabilistic Perspective
Sutton 12. An introduction to Conditional Random Fields
Jordan 08. Graphical Model, Exponential Family, and Variational Inference

Papers
Wei et. al. Masked Conditional Random Fields for Sequence Labeling. NAACL 21
Dozat and Manning. Deep Biaffine Attention for Neural Dependency Parsing. ICLR 2017
Zhang et. al. Efficient Second-Order TreeCRF for Neural Dependency Parsing. ACL 2020
Jason Eisner. Inside-Outside and Forward-Backward Algorithms Are Just Backprop. EMNLP 2016
Alexander Rush. Torch-Struct: Deep Structured Prediction Library. ACL 2020
Sarkka and Garcia-Fernandez. Temporal Parallelization of Bayesian Smoothers. 2018
Li and Rush. Posterior Control of Blackbox Generation. ACL 2020.
Mohamed et. al. Monte Carlo Gradient Estimation in Machine Learning. JMLR 2020
Paulus et. al. Gradient Estimation with Stochastic Softmax Tricks. NeurIPS 2020
Berthet et. al. Learning with Differentiable Perturbed Optimizers. NeurIPS 2020
Blondel et. al. Learning with Fenchel-Young Losses. JMLR 2020
Grathwohl. et. al. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. ICLR 2018