Scaling Structured Inference with Randomization

Yao Fu[1], John P. Cunningham[2], Mirella Lapata[1]

[1]University of Edinburgh [2]Columbia University

Review: classical graphical models for structured prediction

Graph	Model	Inference
Chains	HMMs/ CRFs	Forward-backward
Hypertrees	PCFGs/ TreeCRFs	Inside-outside
General graph	General Exponential Family	General sum-product

Very successful, but computation problem (on GPUs) when scaling to large state space

Computation problem with very many labels

HMM / Linear-chain CRF Forward-Backward time/ space

$$O(L\underline{N}^2)$$

PCFG / TreeCRF Inside-Outside time/ space:

$$O(L^2\underline{N}^3)$$

Goal: to scale structured prediction models to large set of labels on GPUs

Challenges in scaling and previous efforts

Different Graphs Structures

HMMs / PCFGs / Semi-Markov / General Graphs ...

Requirements of Existing Methods for scaling

Sparsity / Pre-clustering / Heuristics ...

Restrictions from Automatic Differentiation

All computation should be differentiable

Solution: Randomized Dynamic Programming

Our method

Different Graphs Structures

We handle them all

Requirements of Existing Methods

We have no pre-assumptions

Restrictions from Automatic Differentiation

Our method is fully compatible with AD
Thus can be seamlessly integrated with neural networks

A randomization solution

Our method: randomized sum-product

For each DP step,

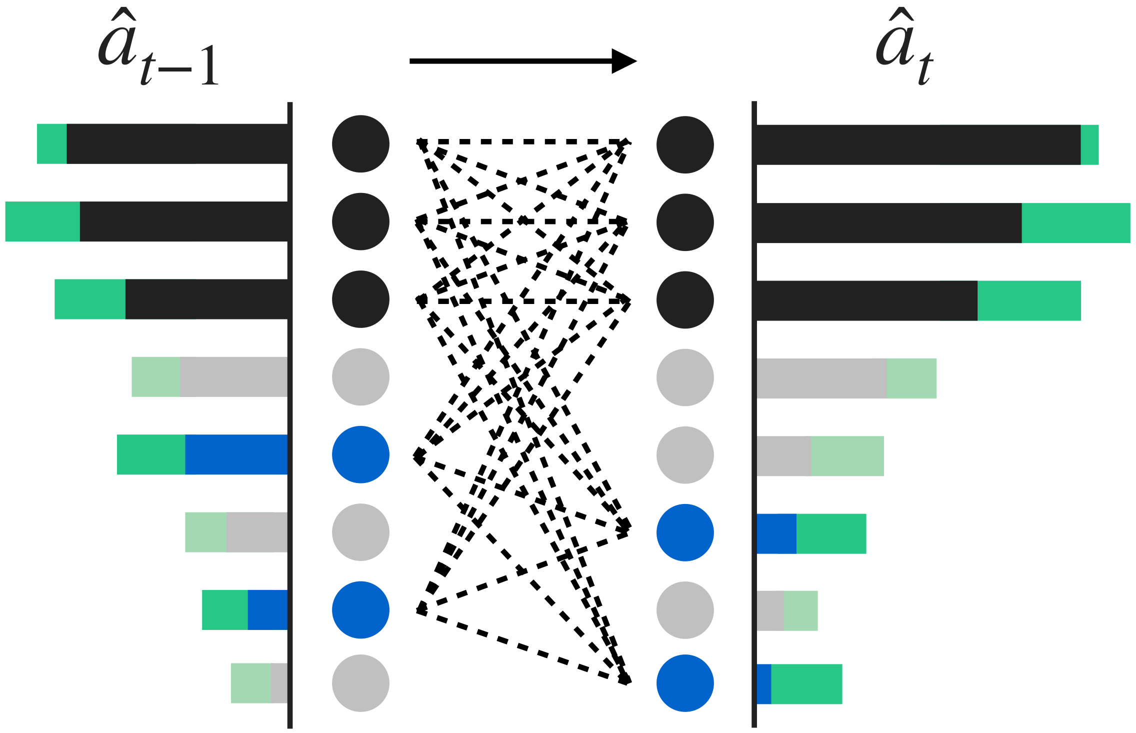
Weight at node ≈ sum of $\overset{\text{K1}}{\overline{\text{most probable states}}}$ combinations

+ $\underset{\text{K2}}{\underline{\text{randomly sampled states}}}$ from the rest

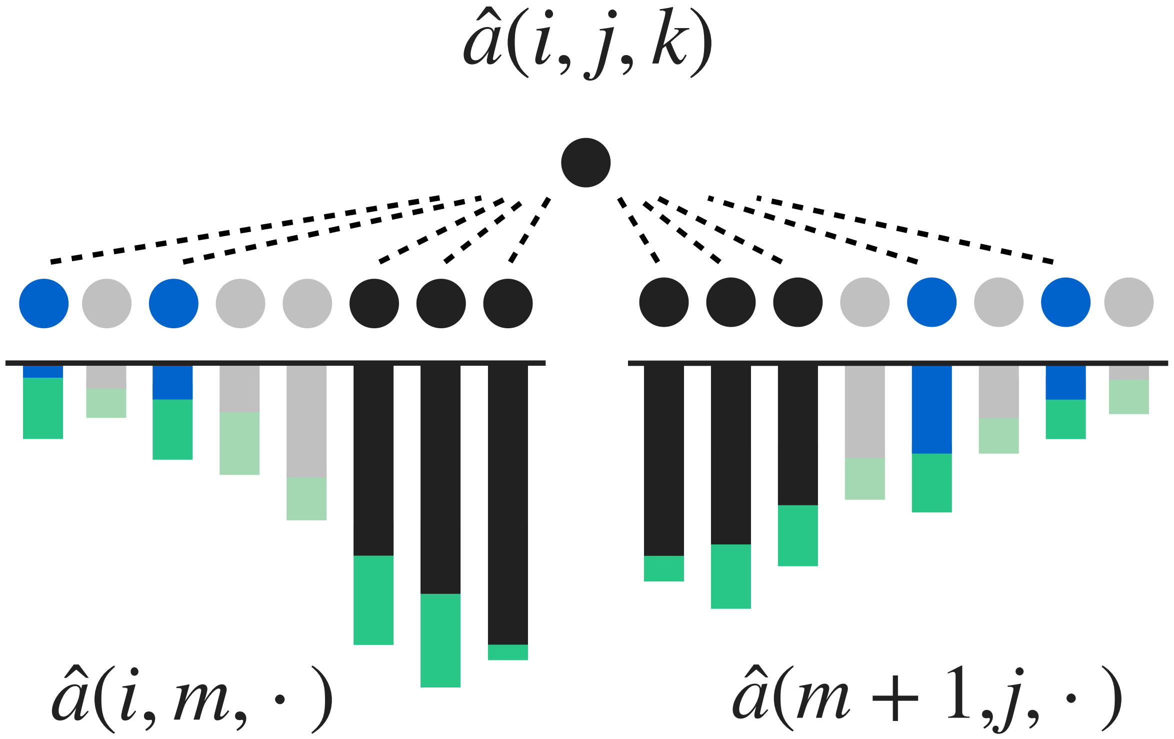
Computation reduction: K1 + K2 << N

Applying randomized sum-product to chains and trees

HMMs / Linear-chain CRFs
Randomized Forward



PCFGs / TreeCRFs
Randomized Inside



■ TopK summand 🟦 Sampled summand ▨ Dropped summand

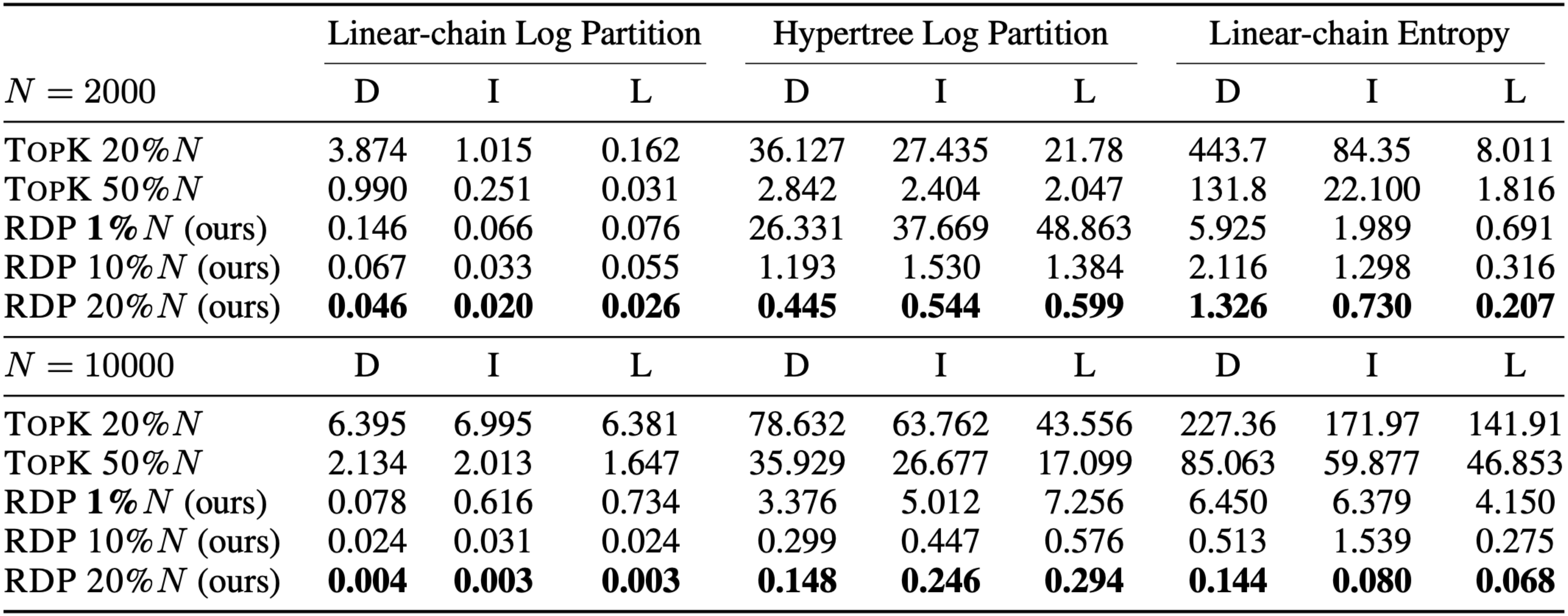
🟩 Gap to oracle ● TopK state 🔵 Sampled state

More details in paper

Variance Reduction with Importance Sampling and Rao-Blackwellization

RDP for entropy and sampling

Performance



$N = 2000$	Linear-chain Log Partition D	Linear-chain Log Partition I	Linear-chain Log Partition L	Hypertree Log Partition D	Hypertree Log Partition I	Hypertree Log Partition L	Linear-chain Entropy D	Linear-chain Entropy I	Linear-chain Entropy L
TopK 20%N	3.874	1.015	0.162	36.127	27.435	21.78	443.7	84.35	8.011
TopK 50%N	0.990	0.251	0.031	2.842	2.404	2.047	131.8	22.100	1.816
RDP **1%**N (ours)	0.146	0.066	0.076	26.331	37.669	48.863	5.925	1.989	0.691
RDP 10%N (ours)	0.067	0.033	0.055	1.193	1.530	1.384	2.116	1.298	0.316
RDP 20%N (ours)	**0.046**	**0.020**	**0.026**	**0.445**	**0.544**	**0.599**	**1.326**	**0.730**	**0.207**
$N = 10000$	D	I	L	D	I	L	D	I	L
TopK 20%N	6.395	6.995	6.381	78.632	63.762	43.556	227.36	171.97	141.91
TopK 50%N	2.134	2.013	1.647	35.929	26.677	17.099	85.063	59.877	46.853
RDP **1%**N (ours)	0.078	0.616	0.734	3.376	5.012	7.256	6.450	6.379	4.150
RDP 10%N (ours)	0.024	0.031	0.024	0.299	0.447	0.576	0.513	1.539	0.275
RDP 20%N (ours)	**0.004**	**0.003**	**0.003**	**0.148**	**0.246**	**0.294**	**0.144**	**0.080**	**0.068**

Applicable to

- different structures: chains and trees
- different inference: log partition function / entropy

Performance

- Smaller mean square error than baseline topK summation
- Memory requirement as small as 1% of full states N

More experiment details in paper

Bias-Variance Decomposition

Integrating with neural networks

Conclusion

Using randomization to approximate sum-product dynamic programming inference

RDP has Advantages in
(1). Memory saving and statistically principled bias-variance control
(2). Compatibility to a wide range of models and inference, as well as automatic differentiation

We hope our work would open new possibilities in large-scale differentiable structured predictions

Thank you!