Neural Particle Smoothing for Sampling from Conditional Sequence Models

Chu-Cheng Lin and Jason Eisner
JOHNS HOPKINS UNIVERSITY

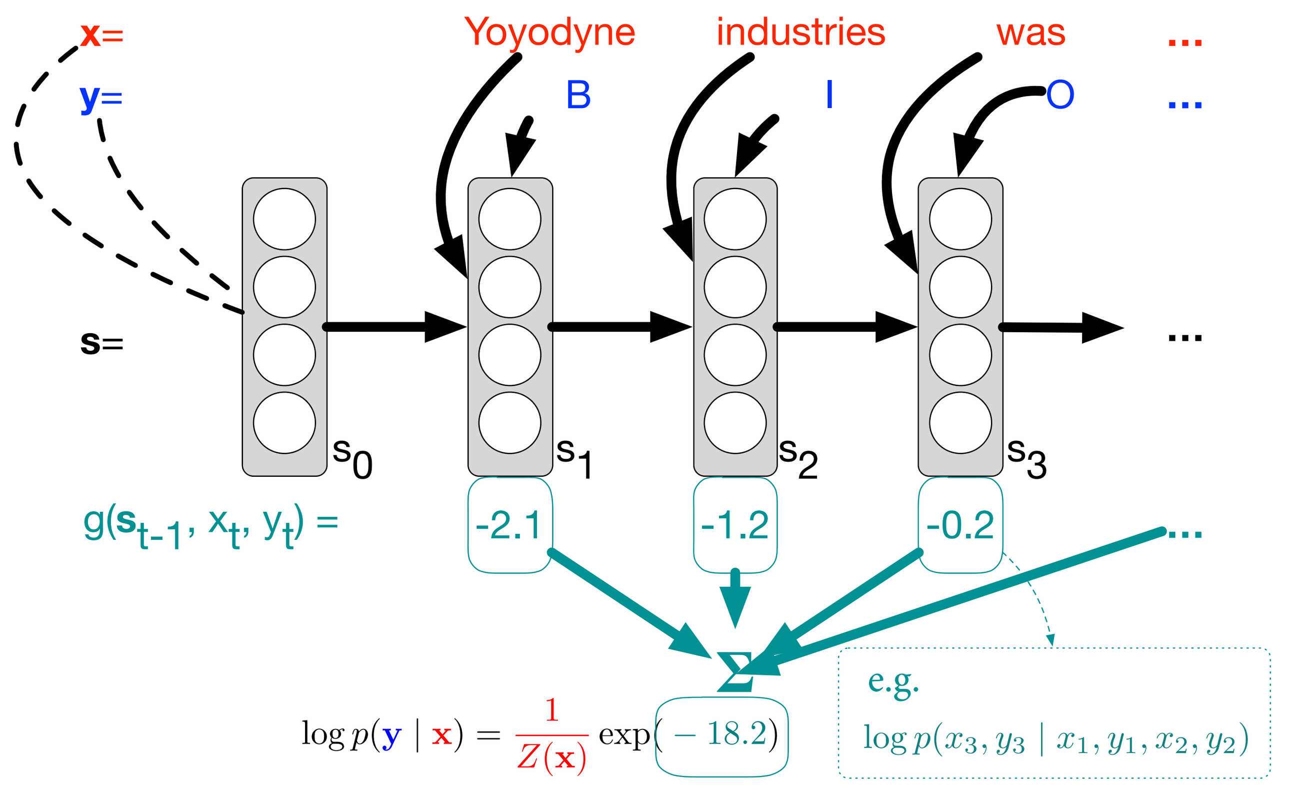
Quiz question: CRF is to dynamic programming as RNN is to …?

Hi, I built a fancy globally normalized neural model that is more powerful than CRFs! 😊
But oops, dynamic programming doesn't work anymore 😟
— *How will I compute my gradient for training?*
— *How will I figure out what my model predicts (conditioned on evidence)?*
— *How will I combine my model with other probability distributions?*

I guess I can sample from $p(\mathbf{y} \mid \mathbf{x})$, but how? My model specifies $\tilde{p}(\mathbf{x}, \mathbf{y})$

I know that $p(\mathbf{y} \mid \mathbf{x}) = \frac{\tilde{p}(\mathbf{x},\mathbf{y})}{\sum_{\mathbf{y}'} \tilde{p}(\mathbf{x},\mathbf{y}')}$, but I don't really feel like summing over the **exponentially many** $\mathbf{y}$'s. What can I do? 🤔

Incremental stateful global scoring models

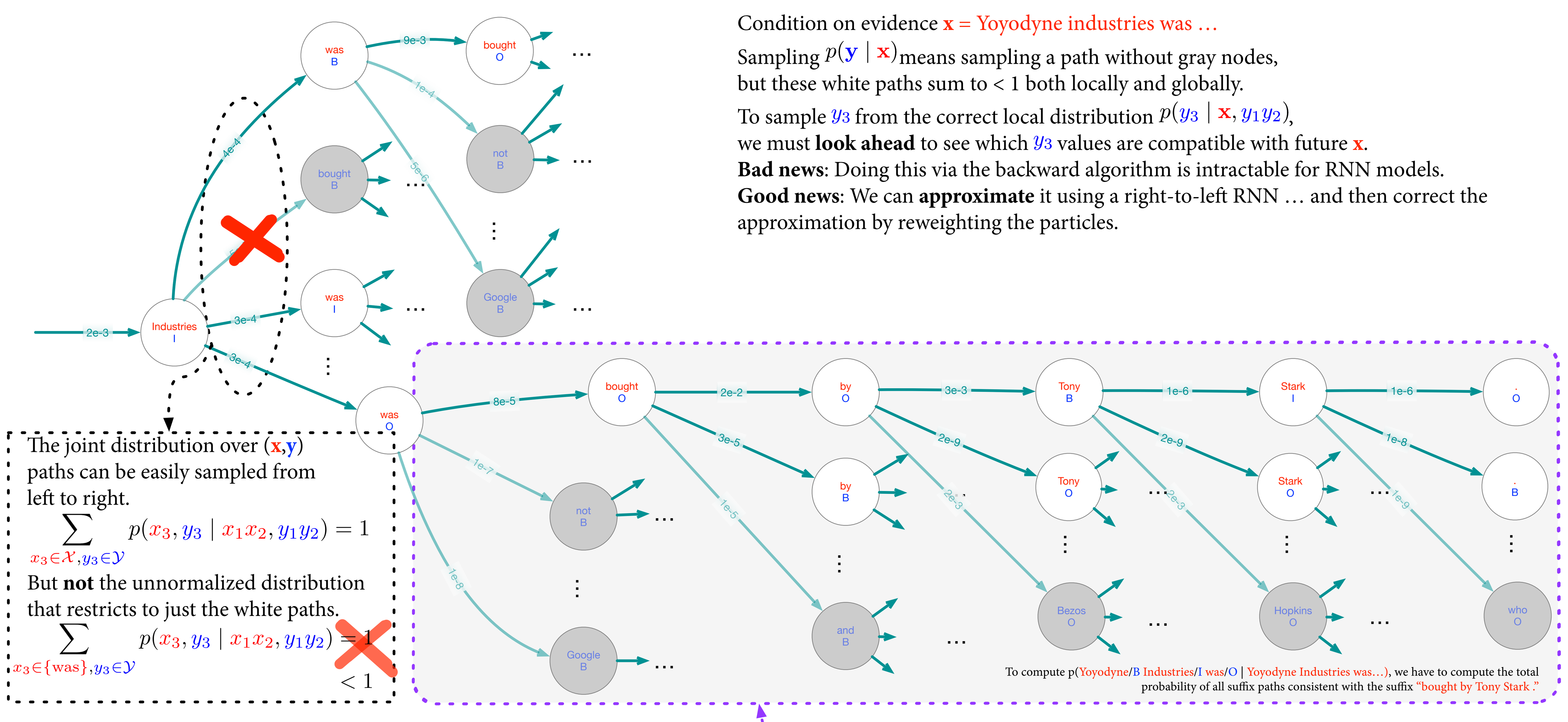


Can this describe a model more powerful than CRFs?
Could I sample "greedily" left to right? Why or why not?

A comparison between inference methods for sequence models

	Left-to-right only	Left-to-right + lookahead	Data structure	Iteration step
Best path	Dijkstra's / Viterbi	A*	Best prefix path so far to each state	Extend a prefix chosen from a priority queue
Approx. best path	Beam search	Beam search + heuristic	M prefix paths of length t	Extend prefixes *exhaustively* to length t+1, then prune
Sampled path	Particle filtering	**Particle smoothing**	M prefix paths of length t	Extend each prefix *randomly* to length t+1, then reweight

Joint distribution = locally normalized distribution = greedy sampling = easy
Conditional distribution = globally normalized distribution = hard!



Condition on evidence **x** = Yoyodyne industries was …

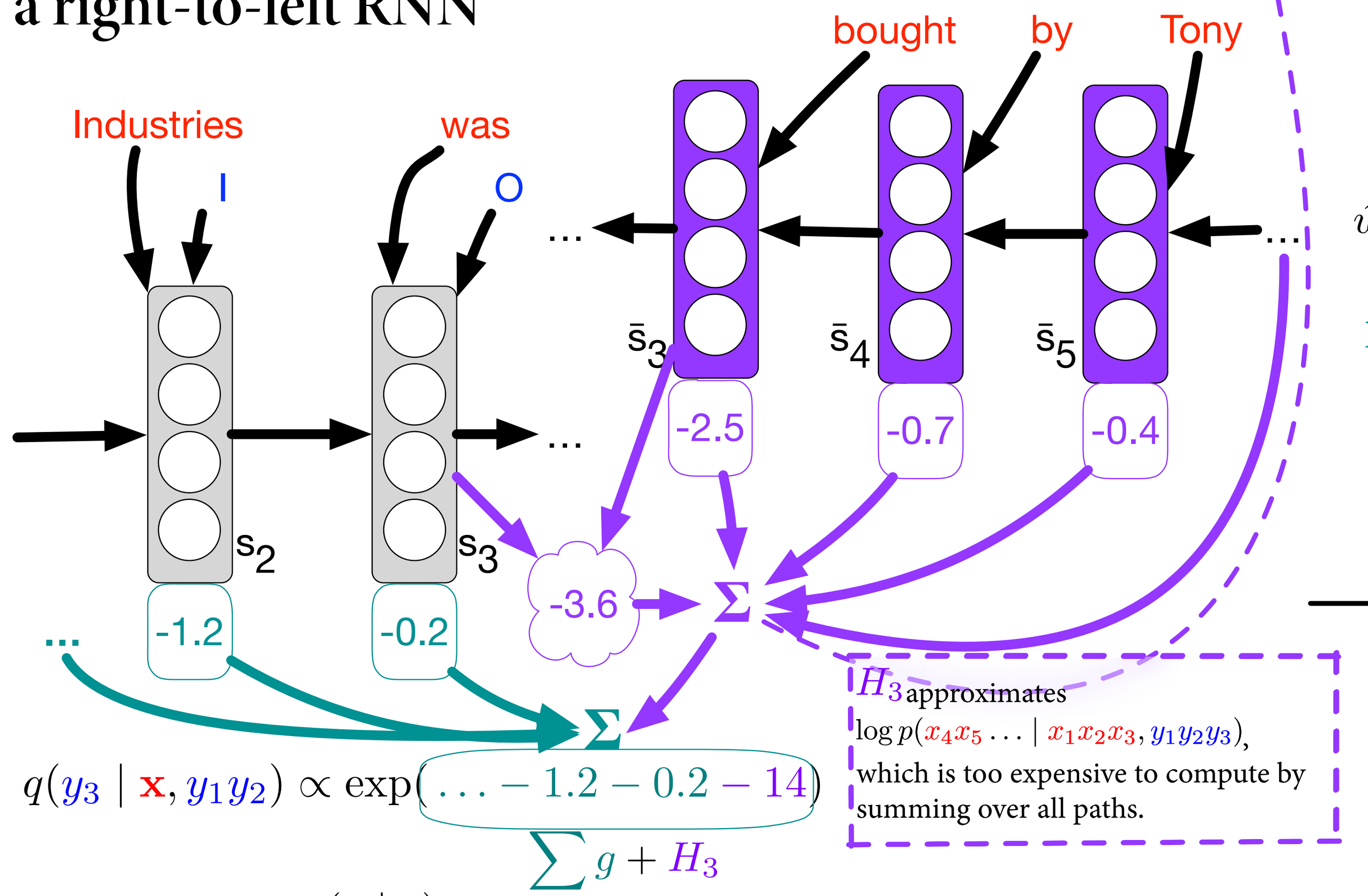
Sampling $p(\mathbf{y} \mid \mathbf{x})$ means sampling a path without gray nodes, but these white paths sum to < 1 both locally and globally.

To sample y_3 from the correct local distribution $p(y_3 \mid \mathbf{x}, y_1 y_2)$, we must **look ahead** to see which y_3 values are compatible with future **x**.

Bad news: Doing this via the backward algorithm is intractable for RNN models.

Good news: We can **approximate** it using a right-to-left RNN … and then correct the approximation by reweighting the particles.

Approximating the backward algorithm with a right-to-left RNN



We approximate $p(\mathbf{y} \mid \mathbf{x})$ with a proposal distribution
$q(\mathbf{y} \mid \mathbf{x}) = q(y_1 \mid \mathbf{x}) q(y_2 \mid \mathbf{x}, y_1) q(y_3 \mid \mathbf{x}, y_1 y_2) \ldots$

We then use importance reweighting to (mostly) correct for the approximation. This becomes accurate as the number of particles goes to infinity. We train H_t to minimize the estimated KL-divergence between q and the true distribution p: $\frac{\mathrm{KL}(p \| q) + \mathrm{KL}(q \| p)}{2}$

Sampling and weight updating steps in particle smoothing

$$\tilde{w}_t = w_{t-1} \cdot \frac{\exp\left(\sum_{i=1}^{t} g(s_{i-1}, x_i, y_i) + H_t\right)}{\exp\left(\sum_{i=1}^{t-1} g(s_{i-1}, x_i, y_i) + H_{t-1}\right) q(y_t \mid \mathbf{x}, \mathbf{y}_{:t-1})}$$

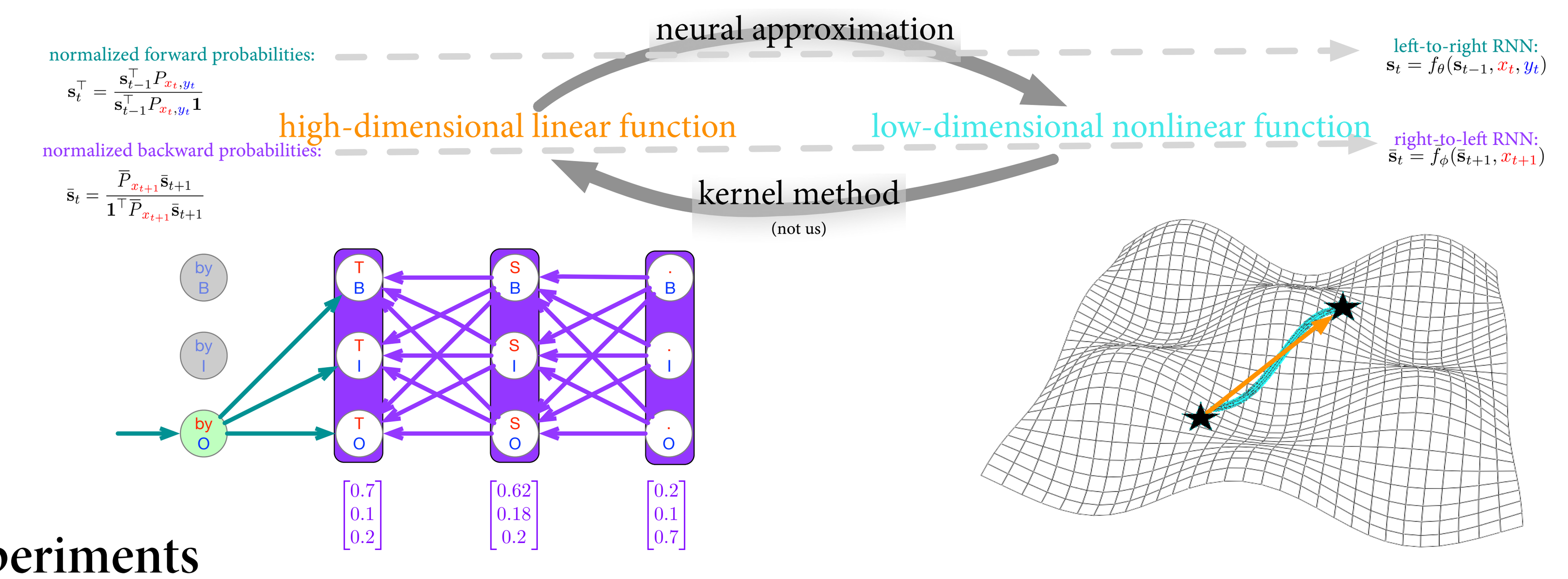
current estimated prefix probability; previous estimated prefix probability; proposal probability

$$w_t = \frac{\tilde{w}_t}{\sum_{\tilde{w}'_t} \tilde{w}'_t}$$

original weight	original particle	① sample extension from q	② reweight particle	③ normalize weight across particles	extended particle
0.2	B I I	B / I / O	0.0003	0.002	B I I B I
0.3	B I B	B / I / O	0.1	0.798	B I B B B
0.5	B I O	B / I / O	0.025	0.2	B I O B I
(sums to 1)				(sums to 1)	

Justifying the neural approximation of dynamic programming

If the model were an HMM emitting (x_t, y_t) pairs, instead of an RNN, our architecture would be exact.
The hidden vectors s_t and $\bar{s}_t$ would represent forward and backward distributions over the hidden HMM state, and would be updated linearly at each t.
(Updates are deterministic because these are belief states, i.e., distributions over states.)
Our RNN is trying to approximate some 10000000-dimensional (?) HMM using only a 50-dimensional vector.
We hope the belief states tend to fall near a 50-dimensional manifold so that s_t can give the manifold coordinates.
The deterministic update in that coordinate space becomes nonlinear.



Experiments

We evaluate how good our sampler is by evaluating how similar its distribution is to the true distribution, in terms of KL-divergence. We experiment on two different model formulations, and three tasks:

— **English stressed syllable prediction** t u ʃ ɛ t / 0 2 0 1 0

— **Chinese social media NER** … qu le lun dun … / … O O B I …

— **Source separation** s ɪ ə g ɹ ɹ ə i n ə o ʊ l ɫ ə / 1 1 1 2 2 1 2 1 2 1 2 2 2 1 2 / pour fill the the glass rum / 1 2 2 1 2 1

The particle smoothing samplers (**PS**/**PS:R**) consistently perform better than particle filtering baselines (**PF**/**PF:R**).