Beyond conventional text generation

Aman @ Yiming Yang Lab seminar, 4/5/2022

LM-based generation models

- Typically involve a transformer-based language model + large parallel corpus
- **Established**
    - Pre-train —> Fine-tune —> test
- **Popular setups**
    - Unsupervised
    - GANs
    - Few-shot
    - Retrieval-augmented
    - Plan-based
    - Structured-generation
    - VAE/De-noising objectives
    - Structure-guided generation *
    - Memory-guided generation *
- **Today**
    - Few things not on the list
        - Text generation as optimization (aka Gradient-based approaches)
        - Stochastic processes

Outline

- Part A:
  - Text generation as optimization
- Part B
  - Modeling dynamics of text using stochastic processes
- Part C
  - Fast-slow graph generation

Text generation as optimization

Text generation as optimization

- **Given**
  - A large pre-trained language model p_θ
  - A set of constraints, specified as functions $f_1, f_2, \ldots, f_m, g_1, g_2, \ldots, g_n$
    - $g : X \times Y \to R$
      - Constraints on inputs x and output y (e.g., semantic similarity)
    - $f : Y \to R$
      - Constraints on just the output (e.g., fluency)
    - Constraints <=> Desired attributes
- **Generate**
  - samples (text) from p_θ **without re-training** that satisfy the constraints
- Admits several popular controllable text generation problems:
  - Make text polite
  - Include certain words in the output
  - Condition output on certain keywords

Text generation as optimization: y as a parameter

- $\mathcal{L}(x, y) = f_1(y) + g_1(x, y) + p_\theta(y \mid x)$
- Given an initial y^0
  - $y^{t+1} \leftarrow y^t - \eta \nabla_y(L)$
  - But y is discrete…
- Formality transfer:
  - Given y = "*give me the data*" generate *"please share the data*"
  - A formality constraint (e.g., $f_1(y)$ measures how polite the sentence y is)
  - What is $\nabla_y L$??
  - The intuition is well-motivated (how to change "*give me the data*" so that loss is lower), but not differentiable

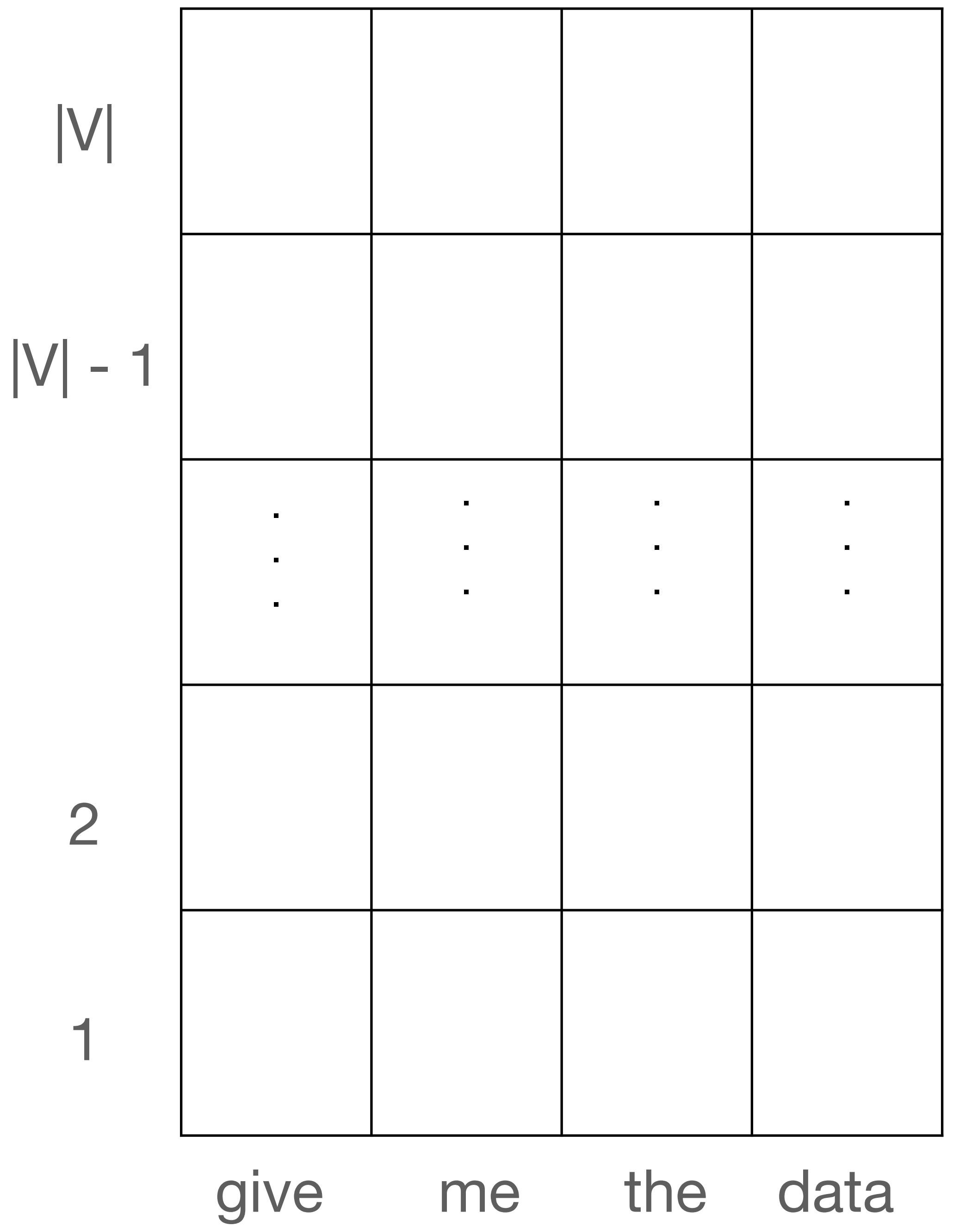
Recurring theme: y as a parameter

- Instead, y_i is treated as a simplex ($0 \leq y_{il} \leq 1, \ \sum_{l=1}^{|V|} y_{il} = 1$, |V| is the vocabulary size)

- In short, maintain a continuous representation of y so that it can be back propagated

- y a tensor like any other parameter, can be attached to the computation graph and backpropagated through

- Other solutions like Gumbel-softmax



Towards Decoding as Continuous Optimisation in Neural Machine Translation

Cong Duy Vu Hoang† and **Gholamreza Haffari**‡ and **Trevor Cohn**†

† University of Melbourne, Melbourne, VIC, Australia

‡ Monash University, Clayton, VIC, Australia

vhoang2@student.unimelb.edu.au, gholamreza.haffari@monash.edu, t.cohn@unimelb.edu.au

EMNLP 2017

Published as a conference paper at ICLR 2017

CATEGORICAL REPARAMETERIZATION WITH GUMBEL-SOFTMAX

Eric Jang
Google Brain
ejang@google.com

Shixiang Gu*
University of Cambridge
MPI Tübingen
sg717@cam.ac.uk

Ben Poole*
Stanford University
poole@cs.stanford.edu

Controlled Text Generation as Continuous Optimization with Multiple Constraints

Sachin Kumar[♣] **Eric Malmi**[◇] **Aliaksei Severyn**[◇] **Yulia Tsvetkov**[♠]
[♣]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[◇]Google Research
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
sachink@cs.cmu.edu, {emalmi, severyn}@google.com, yuliats@cs.washington.edu

Neurips 2021

Formulation

- Decoding as an optimization problem:
  - The language model will give some distribution over the input $p_\theta(y \mid x)$
  - Improve samples drawn from $p_\theta(y \mid x)$ without changing θ

$$\mathbf{y}^* = \arg\min_{\mathbf{y}\in\mathcal{Y}}(-\log p(\mathbf{y}|\mathbf{x}), f_1(\mathbf{y}), \ldots, f_u(\mathbf{y}), g_1(\mathbf{x},\mathbf{y}), \ldots, g_v(\mathbf{x},\mathbf{y}))$$

- **Initial attempt:** Treat constraints and likelihood as a function of ***y***, and perform gradient descent on y
- Optimization problem with constraints
  -

$$\arg\min_{y} -\alpha \log p(\mathbf{y}|\mathbf{x}) + \sum_{i=1}^{u} \lambda_i f_i(\mathbf{y}) + \sum_{j=1}^{v} \mu_j g_j(\mathbf{x},\mathbf{y}),$$

Issues with vanilla formulation

$$\arg\min_{y} -\alpha \log p(\mathbf{y}|\mathbf{x}) + \sum_{i=1}^{u} \lambda_i f_i(\mathbf{y}) + \sum_{j=1}^{v} \mu_j g_j(\mathbf{x}, \mathbf{y}),$$

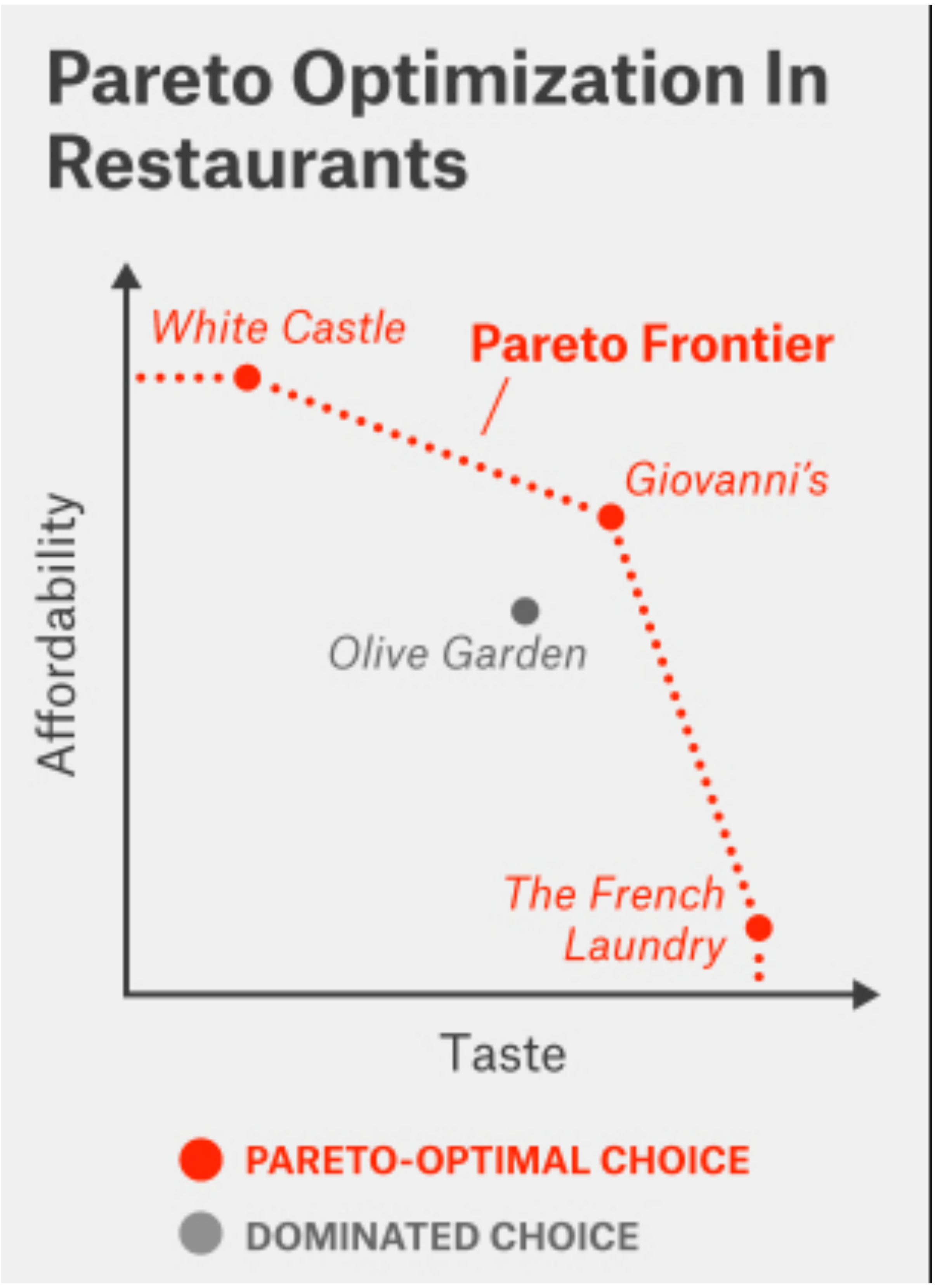
- Who decides weights over the constraints? different constraints may have different scales, some might be objective

- Non-convex “Pareto-front”, hard to achieve

- Reformulate again as a Lagrangian optimization problem:

$$\arg\min_{\mathbf{y}} - \log P(\mathbf{y}|\mathbf{x}) \text{ subject to}$$
$$f_i(\mathbf{y}) \leq \epsilon_i, i \in \{1, \cdots, u\}$$
$$g_j(\mathbf{x}, \mathbf{y}) \leq \xi_j, j \in \{1, \cdots, v\}.$$

  - Upper limits are easy to manually specify (e.g. probability of formality > 0.5) than weights



https://fivethirtyeight.com/wp-content/uploads/2015/02/silver-feature-pareto-1.png?w=305

Lagrangian formulation

- Lagrangian formulation

$$\mathcal{L}(y, \lambda_1, \ldots, \lambda_u, \mu_1, ...\mu_v) = -\log p(\mathbf{y}|\mathbf{x}) - \sum_{i=1}^{u} \lambda_i(\epsilon_i - f_i(y)) - \sum_{j=1}^{v} \mu_j(\xi_j - g_j(x, y))$$

$$\arg\min_{\mathbf{y}} -\log P(\mathbf{y}|\mathbf{x}) \text{ subject to}$$
$$f_i(\mathbf{y}) \leq \epsilon_i, i \in \{1, \cdots, u\}$$
$$g_j(\mathbf{x}, \mathbf{y}) \leq \xi_j, j \in \{1, \cdots, v\}.$$

- Typical solution:
  - $\arg\min_{\mathbf{y}} \max_{\lambda_i \geq 0, \mu_i \geq 0} \mathcal{L}(\mathbf{y}, \lambda_i, \mu_i)$
  - First solve the dual function to find Lagrangians, then plug Lagrangians to minimize
  - Similar issue as the previous formulation: first clipping the constraints *is not working*
- Final approach:
  - Method of multipliers: jointly optimize for both
  - $\mathbf{y}^{(t)} = \mathbf{y}^{(t-1)} - \eta_1 \nabla_{\mathbf{y}} \mathcal{L}, \lambda_i^t = \lambda_i^{t-1} + \eta_2 \nabla_{\lambda_i} \mathcal{L}, \mu_i^t = \mu_i^{t-1} + \eta_2 \nabla_{\mu_i} \mathcal{L}$
  - Increase the value of the multiplier with each gradient step as long as the constraint is violated (make the optimization process take constraints more seriously)

Algorithm 1: MuCoCO: detailed decoding algorithm

Input: input sequence $\mathbf{x}$, output length L, base model $\mathcal{G}$, attribute functions f_i and g_j and their respective initial and final thresholds, threshold update schedule, step sizes η_1, η_2;

Result: output sequence $\mathbf{y}$

For all $k \in \{1, \ldots, L\}$, initialize $\tilde{\mathbf{y}}_k^0$ uniformly over Δ_V;

For all $i \in \{1, \ldots u\}$ and $j \in \{1 \ldots v\}$, initialize λ_i^0, μ_i^0 as 0 and the thresholds ϵ_i^0, ξ_j^0 with the given values ;

for $t = 1, \ldots,$ MaxSteps **do**

```
// forward pass
```

for all k, compute $\hat{y}_k = \texttt{one-hot}(\arg\max \tilde{y}_k)$ and compute the loss $\mathcal{L}$ (using (5));

```
// backward pass
```

for all k, i and j, compute $\nabla_{\tilde{y}_k}^{t-1} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{y}}_k}, \nabla_{\lambda_i}^{t-1} = \frac{\partial \mathcal{L}}{\partial \lambda_i}, \nabla_{\mu_j}^{t-1} = \frac{\partial \mathcal{L}}{\partial \mu_j}$;

```
// Update the parameters
```

update $\tilde{y}_k^{(t+1)} \propto \tilde{y}_k^{(t)} \exp(1 - \eta_1 \nabla_{\tilde{y}_k} \mathcal{L})$;

update $\lambda_i^t = \max(0, \lambda_i^{t-1} + \eta_2 \nabla_{\lambda_i} \mathcal{L})$, and $\mu_i^t = \max(0, \mu_i^{t-1} + \eta_2 \nabla_{\mu_i} \mathcal{L})$;

update ϵ_i^t, ξ_j^t following the threshold update schedule

end

Can select one of "Pareto-optimal solutions"

return $\arg\min_t\{-\log p(\tilde{\mathbf{y}}^{(t)}|\mathbf{x}) : \forall i, f_i(\tilde{\mathbf{y}}^{(t)}) \leq \epsilon_i, \forall j, g_j(\mathbf{x}, \tilde{\mathbf{y}}^{(t)}) \leq \xi_j\}$;

Experiments

- Style transfer: make a given sentence more formal (*give me that data —> please share the data*)
- Task requirements:
    - Attribute transfer: the transferred sentence should be formal
    - Content preservation: the transferred sentence should not lose the original meaning
- Constraints, one per requirement:
    - $-\log(p_{formal}(\mathbf{y})) < -\log(0.5)$ = 0.3
        - Force $p_{formal}(\mathbf{y})$ to be greater than 0.5
        - $p_{formal}(\mathbf{y})$ using a classifier $\mathbf{y}$
        - Start with $-\log(p_{formal}(\mathbf{y})) < 10$ and slowly anneal
    - Cosine similarity: $USIM(x, y)$
        - $-USIM(x, y) < -0.15$

Results

Method	Constraint	Fluency	Transfer	Content Preservation (w.r.t. input)		Content Preservation (w.r.t. ref)	
				WSIM	USIM	WSIM	USIM
STRAP	None	91%	78%	0.69	0.77	0.72	0.80
FUDGE	FORMAL(y)	90%	85%	0.71	0.77	0.73	0.81
MUCOCO	FORMAL(y)	89%	93%	0.67	0.75	0.72	0.78
MUCOCO	USIM(x, y)	92%	85%	0.71	0.78	0.74	0.81
MUCOCO	USIM(x, y), WMD(x, y)	92%	87%	**0.73**	**0.79**	**0.77**	**0.86**
MUCOCO	SIM(x, y), WMD(x, y), FORMAL(y)	**93%**	**92%**	0.71	**0.79**	0.75	0.84

Table 1: Automatic evaluation of fluency, formality transfer, and content preservation for informal-to-formal style transfer models.

COLD Decoding: Energy-based Constrained Text Generation with Langevin Dynamics

Lianhui Qin [1] **Sean Welleck** [1 2] **Daniel Khashabi** [2] **Yejin Choi** [1 2]
[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Allen Institute for Artificial Intelligence

Introduction

- *COLD decoding is a flexible framework that can be applied directly to off-the-shelf left-to-right language models without the need for any task-specific fine-tuning*
- Setup:
    - Given
        - A language model
        - A set of constraints specified as differentiable functions of output
    - Generate samples that satisfy the constrain

Background

Energy distribution

- Energy-based distribution
  - $$p(y) = \frac{\exp^{-E(y)}}{\sum_y \exp^{-E(y)}}$$
  - E(y) is the “energy function”
    - Lower (more negative) energy states are more likely
    - Terminology from physics, low-value energy states are preferable (second-law of thermodynamics)
  - E(y) can be arbitrary in principle
- We don’t care about the actual distribution typically, only the samples

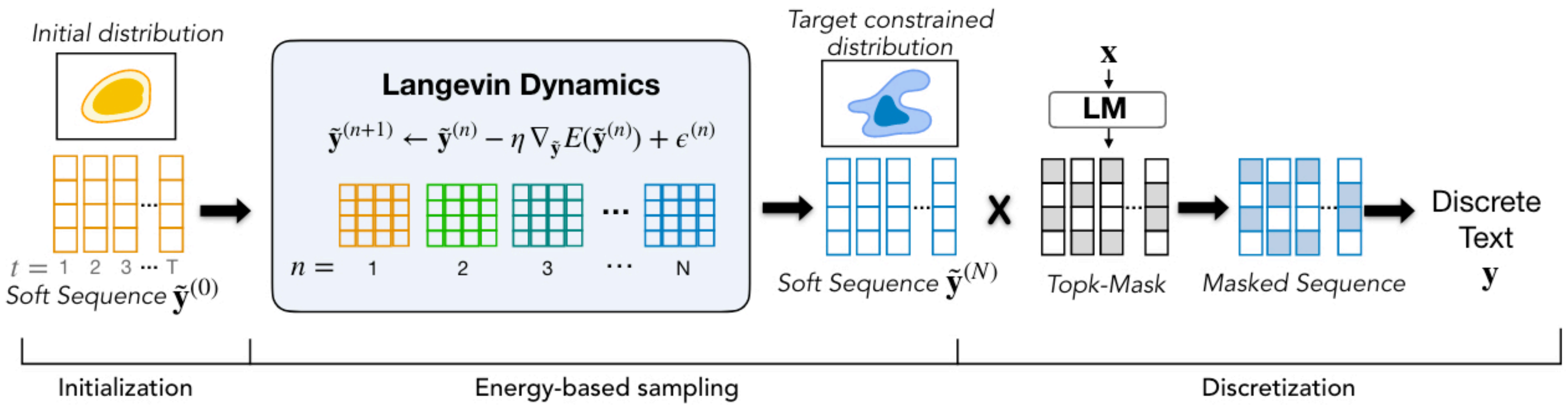
Encoding constraints in energy function

- Let $f_i(y)$ be a constraint function, which is high if constraint is better satisfied
  - $p(y) = \frac{\exp \sum_i \lambda_i f_i(y)}{Z}$
  - E.g., number of tokens in y that are one of {cat, dog, elephant}, more the better
- **Desired:**
  - A way to draw samples from the energy function

- **Langevin dynamics**
  - MCMC method, used for sampling from energy functions
  - Langevin dynamics is motivated and originally derived as a discretization of a stochastic differential equation whose equilibrium distribution is the posterior distribution
  - $y^{n+1} \leftarrow y^n - \eta \nabla_y E(y^n) + \epsilon^n$
    - η: step size, $\epsilon \sim \mathcal{N}(0,\sigma)$
    - Welling & Teh 2011 prove that $\lim_{n\to\infty} y^n \sim p$
  - Problem as usual:
    - $\nabla_y E(y)$
    - Differentiate w.r.t. concrete text ✔
    - Same solution as used by Kumar 2021 —> soft tokens ✘
    - Same constraint: the constrain functions need to be differentiable

https://www.stats.ox.ac.uk/~teh/research/compstats/WelTeh2011a.pdf

COLD decoding overview



Algorithm 1 Constrained Decoding w/ Langevin Dynamics.

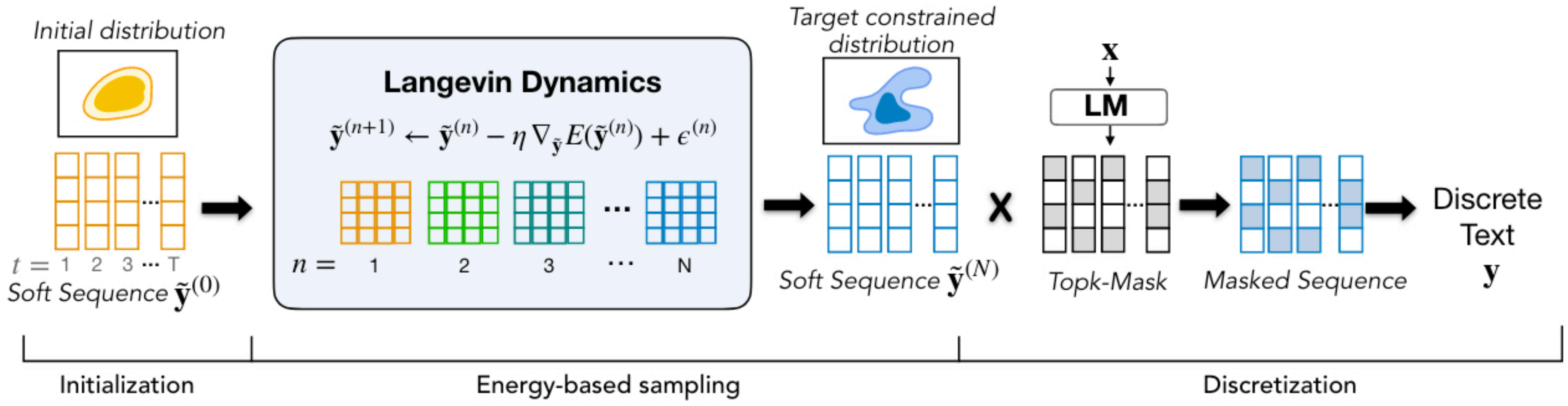
input Constraints $\{f_i\}$, length T, iterations N.
output Sample sequence $\mathbf{y}$.
 $\tilde{\mathbf{y}}_t^{(0)} \leftarrow$ `init()` for all position t `// init soft-tokens`
 for $n \in \{1, \dots, N\}$ **do**
  $E^{(n)} \leftarrow E(\tilde{\mathbf{y}}^{(n)}; \{f_i\})$ `// compute energy (§3.2)`
  $\tilde{\mathbf{y}}_t^{(n+1)} \leftarrow \tilde{\mathbf{y}}_t^{(n)} - \eta \nabla_{\tilde{\mathbf{y}}_t} E^{(n)} + \epsilon_t^{(n)}$ for all t `// update soft tokens (Eq.2)`
 end for
 $y_t = \arg\max_v \text{topk-filter}\left(\tilde{\mathbf{y}}_t^{(N)}(v)\right)$ for all t `// discretize (Eq.6)`
return: $\mathbf{y} = (y_1, \dots, y_T)$

Initialization. We initialize the soft sequence $\tilde{\mathbf{y}}$ by running greedy decoding with the LM p_{LM} to obtain output logits. In our preliminary experiments, the initialization strategy had limited influence on the generation results.

From soft-tokens to hard tokens

- Top-k mask

$$y_t = \arg\max_{v \in \mathcal{V}_t^k} \tilde{\mathbf{y}}_t(v).$$



- Essentially, the language model gives an idea of what is right
  - Questionable, and ablations reveal that this is critical for their method

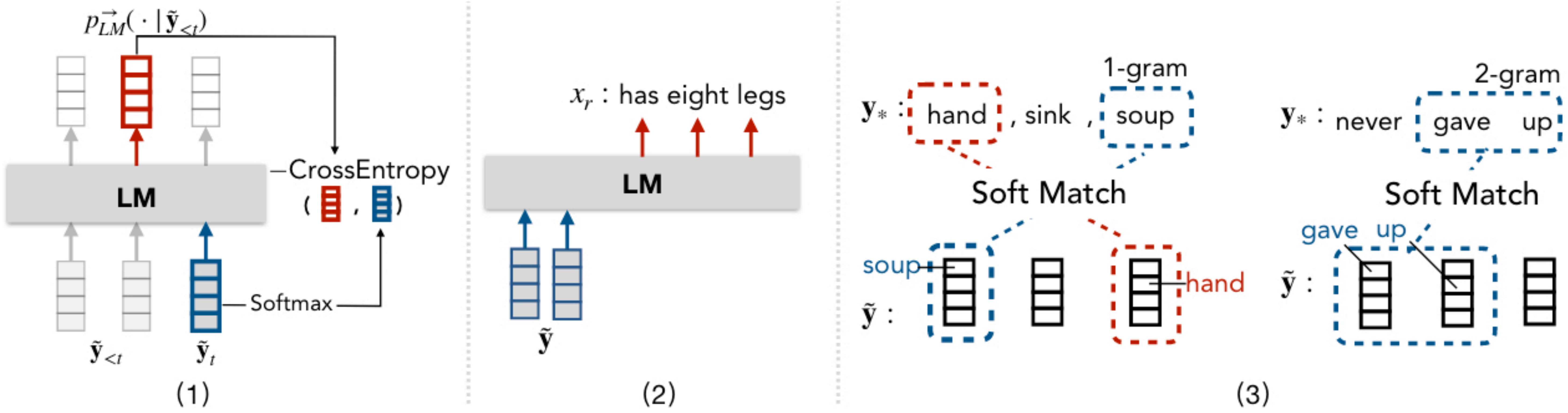
Experiments

Abductive reasoning

- Given a beginning (x_l: ***Tim wanted to learn astronomy***) and end sentence (x_r.***Tim worked hard in school to become one***), generate a bridge sentence that completes the reasoning chain
  - $\tilde{y}$***: Tim took admission in an astronomy program***
- Constraints:
  - The generated text $\tilde{y}$ should have a high-likelihood given the left and right sentences
  - $\tilde{y}$ should predict x_r
  - There should be some common keywords between $\tilde{y}$, x_l, x_r

Example of constraints

- Soft-fluency constraint: $$f_{\overrightarrow{\mathrm{LM}}}(\tilde{\mathbf{y}}) = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}} p_{\overrightarrow{\mathrm{LM}}}(v | \tilde{\mathbf{y}}_{<t}) \log \mathrm{softmax}\left(\tilde{\mathbf{y}}_t(v)\right)$$
    - To be used in conjunction with other constraints



Experiments

Counterfactual story generation

$$E(\tilde{\mathbf{y}}) = \lambda_a^{lr} f_{\text{LM}}^{\rightarrow}(\tilde{\mathbf{y}}; \mathbf{x}_l) + \lambda_a^{rl} f_{\text{LM}}^{\leftarrow}(\tilde{\mathbf{y}}; \mathbf{x}_r) + \lambda_b f_{\text{pred}}(\tilde{\mathbf{y}}; \mathbf{x}_r) + \lambda_c f_{\text{sim}}(\tilde{\mathbf{y}}; \text{kw}(\mathbf{x}_r) - \text{kw}(\mathbf{x}_l)).$$

Models	Automatic Eval				Human Eval			
	BLEU_4	ROUGE-L	CIDEr	BERTScore	Grammar	Left-coherence ($\mathbf{x}_l\mathbf{y}$)	Right-coherence ($\mathbf{y}\mathbf{x}_r$)	Overall-coherence ($\mathbf{x}_l\mathbf{y}\mathbf{x}_r$)
LEFT-ONLY	0.88	16.26	3.49	38.48	**4.57**	3.95	2.68	2.70
DELOREAN	1.60	19.06	7.88	41.74	4.30	**4.23**	2.83	2.87
COLD (ours)	**1.79**	**19.50**	**10.68**	**42.67**	4.44	4.00	**3.06**	**2.96**

- A number of other shortcuts to get it to work
  - Hyper-parameters appear to be highly tuned (0.3, 0.2, 0.02, 0.48), but no details how
  - The method only generates 10 tokens, and the rest are generated by standard language models

Discussion

- **Summary**
  - There are methods that can generate samples from pre-trained language models that satisfy certain constraints, without re-training
- **Difference between two works?**
  - Formulation and method, but conceptually identical
  - COLD allows drawing multiple samples, but also possible with MuCOCO
  - MuCOCO allows specifying different constraints on weights
- **Non-differentiable objectives**
  - Uses continuous sampling from black box models, but extremely slow

Mix and Match: Learning-free Controllable Text Generation using Energy Language Models

Fatemehsadat Mireshghallah[1], Kartik Goyal[2], Taylor Berg-Kirkpatrick[1]
[1] University of California San Diego, [2] Toyota Technological Institute at Chicago (TTIC)
[fatemeh, tberg]@ucsd.edu, kartikgo@ttic.edu

Modeling dynamics of text using stochastic processes

LANGUAGE MODELING VIA STOCHASTIC PROCESSES

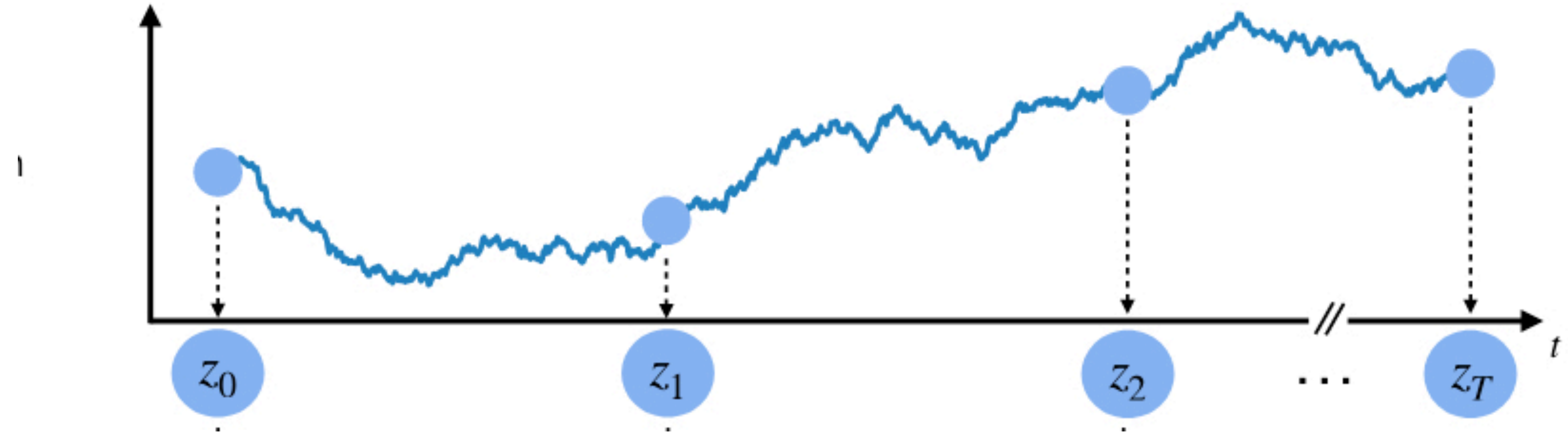
Rose E. Wang, Esin Durmus, Noah Goodman, Tatsunori B. Hashimoto
Stanford University
{rewang, edurmus, ngoodman,thashim}@stanford.edu

Introduction

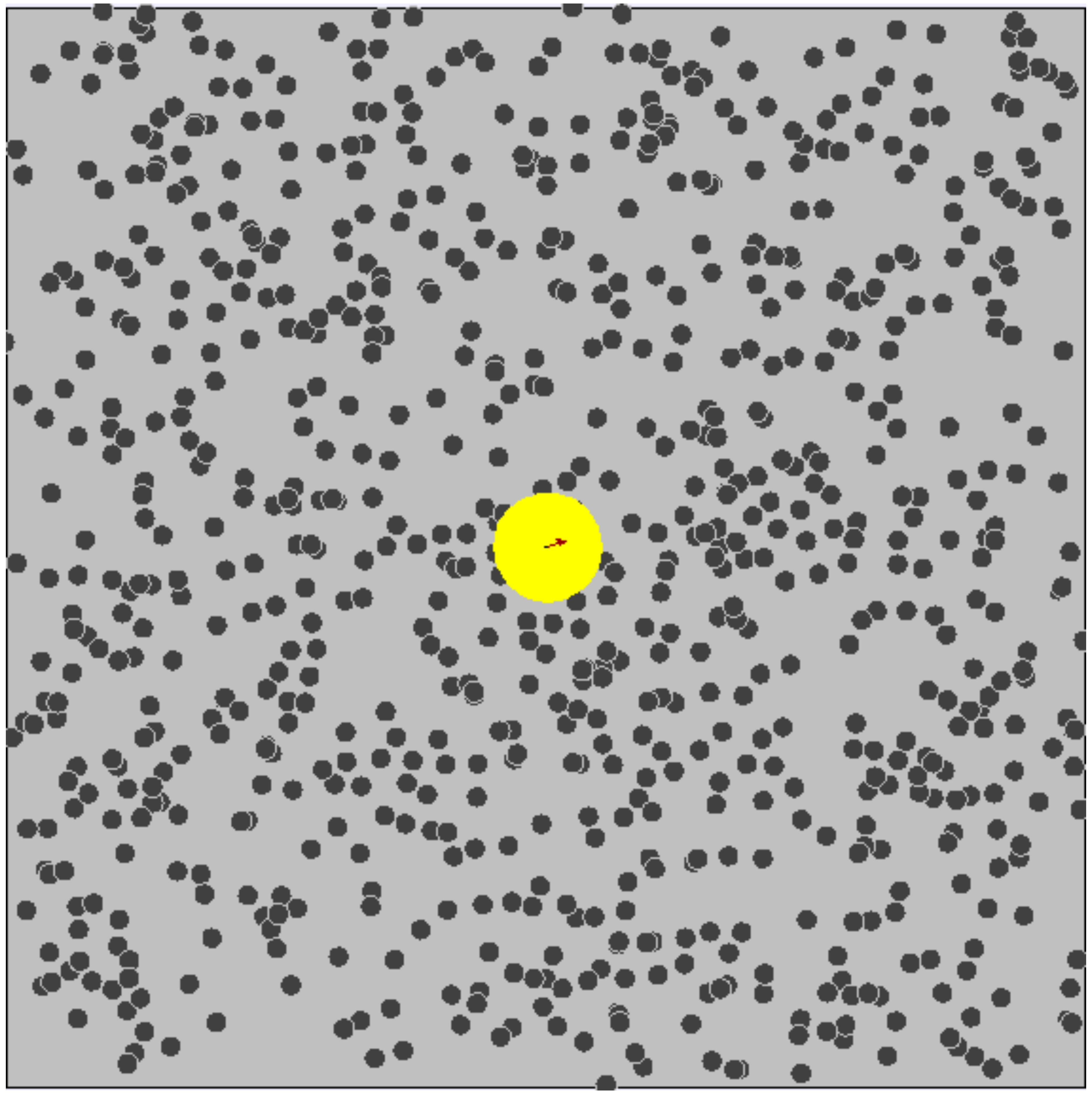
- Long-form generation is important
    - Story generation, document generation
- Current language models struggle
- Possible reason:
    - Auto-regressive models struggle to attend to longer sequences
    - No notion of “evolving” context

Overview

- Generate hidden states for each sentence of a long document
  - The representations should vary ***smoothly***
  - The representations should be ***grounded*** in the start and beginning



Brownian motion



Brownian motion

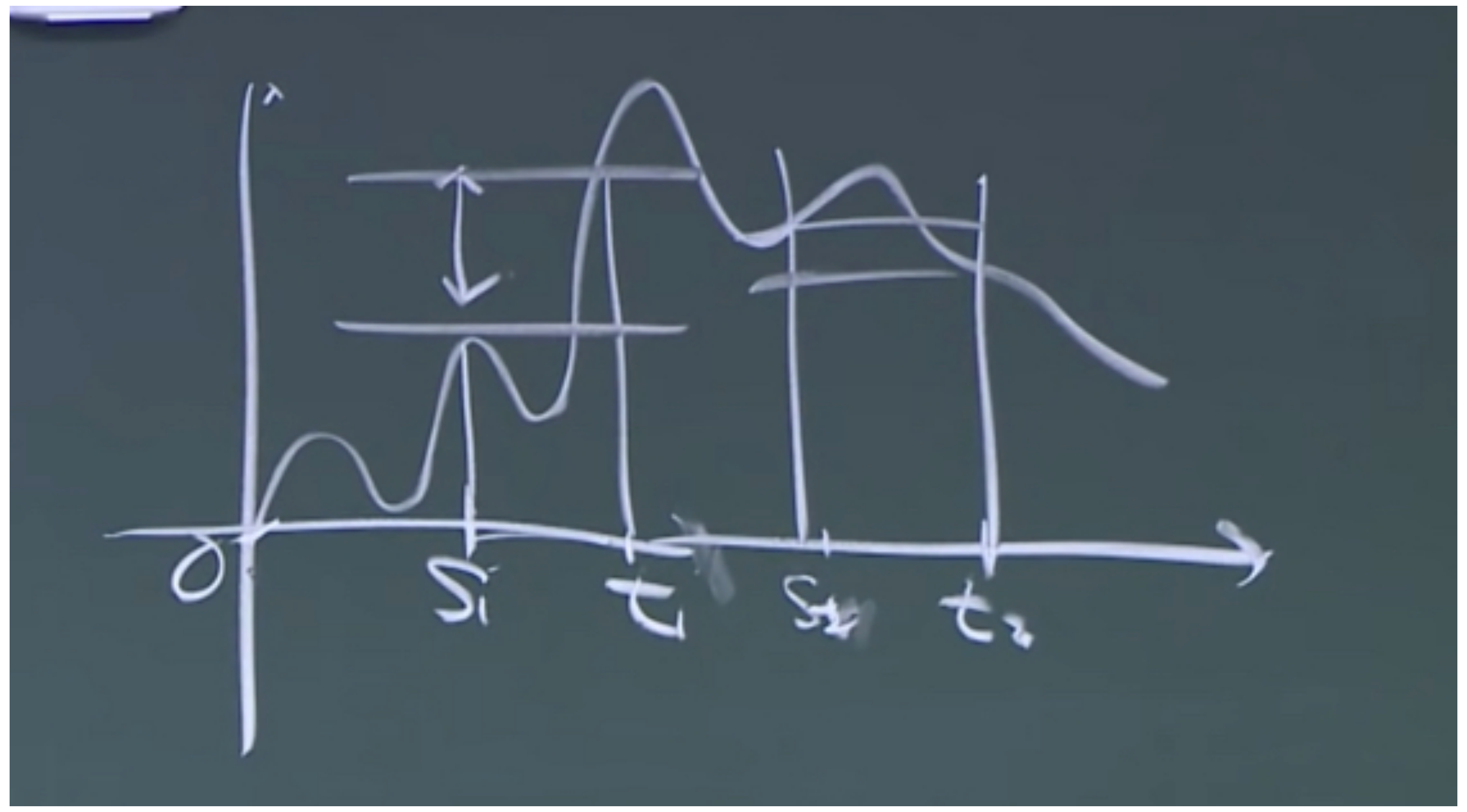
Theorem 2.1. *There exists a probability distribution over the set of continuous functions $B : \mathbb{R} \to \mathbb{R}$ satisfying the following conditions:*

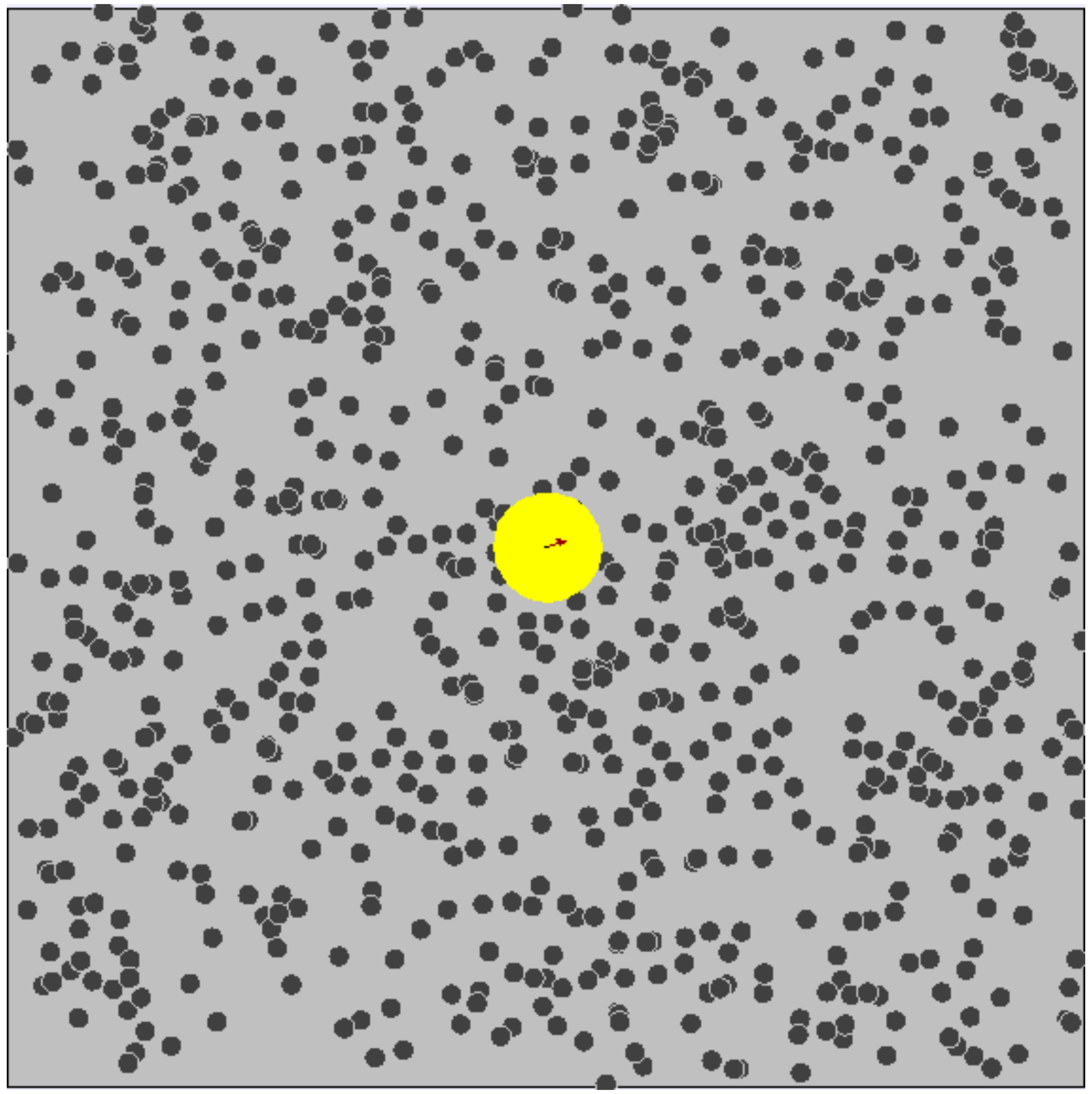
(i) $B(0) = 0$.

(ii) ***(stationary)*** *for all* $0 \leq s < t$*, the distribution of* $B(t) - B(s)$ *is the normal distribution with mean* 0 *and variance* $t - s$*, and*

(iii) ***(independent increment)*** *the random variables* $B(t_i) - B(s_i)$ *are mutually independent if the intervals* $[s_i, t_i]$ *are nonoverlapping.*

Distribution given by this theorem is brownian motion.

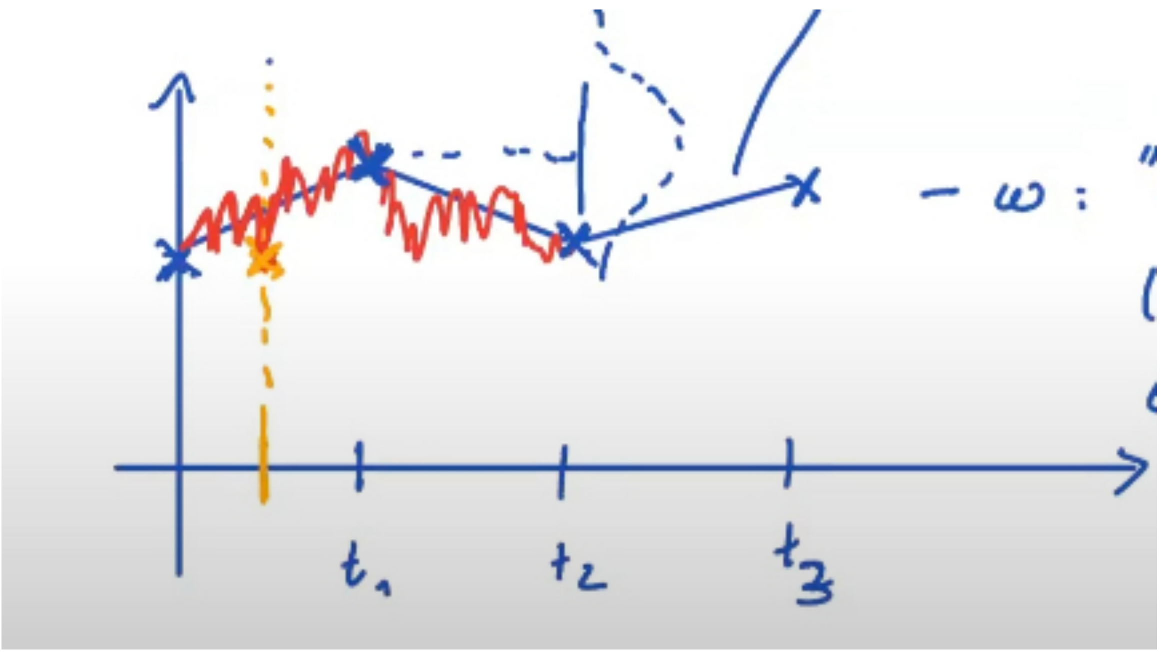




https://ocw.mit.edu/courses/18-s096-topics-in-mathematics-with-applications-in-finance-fall-2013/3b97c6b0c282dd9dc024c4c7ffe3fba8_MIT18_S096F13_lecnote17.pdf

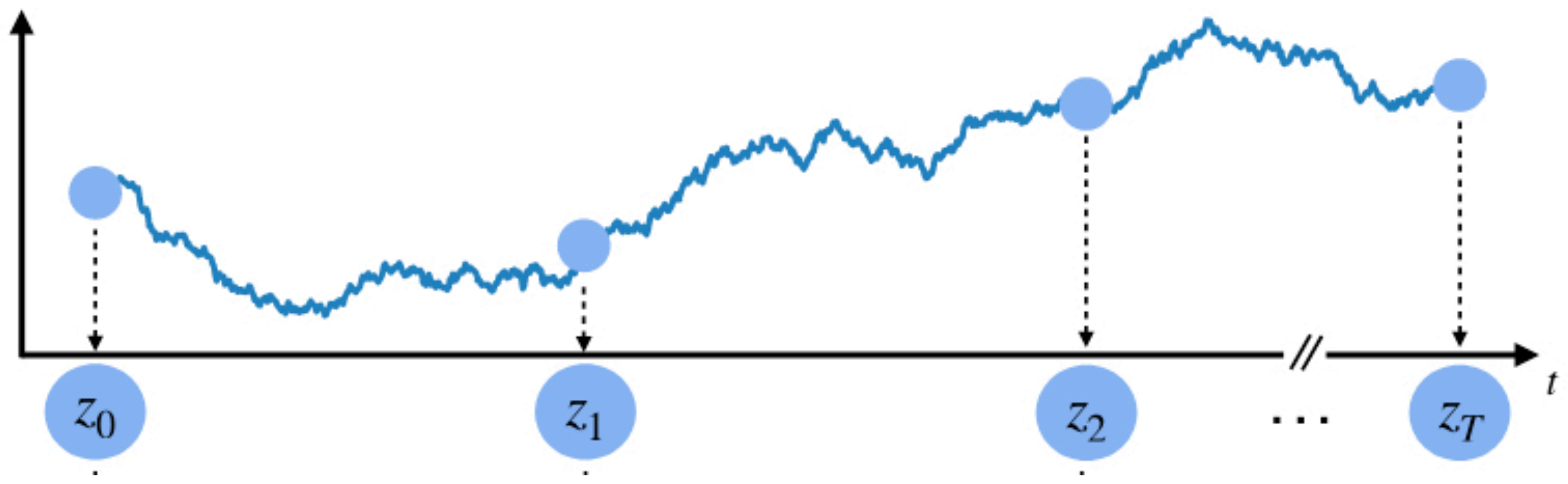
Brownian bridge

- We are given coarse points, and we want to "fill" the path between them with Brownian motion



- key idea: learn latents that create a brownian motion between two given points

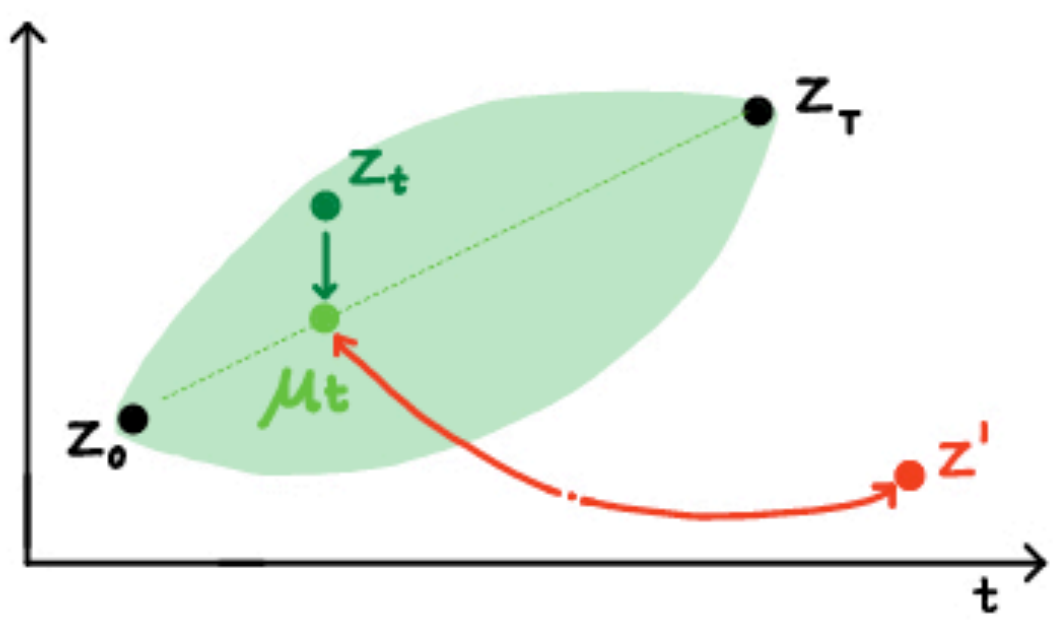
$$p(z_t|z_0, z_T) = \mathcal{N}\left(\left(1 - \frac{t}{T}\right)z_0 + \frac{t}{T}z_T, \frac{t(T-t)}{T}\right).$$



No variance at the beginning or the end
The point is "**attached**"

Method outline

- Contrastive learning:
  - Three sentences x_0, x_t, x_T from a “document” in a sequence with hidden representations z_0, z_t, z_T
  - A random sentence x' with embedding z’




```
x_0: [USER] Hello, I'd like to buy tickets for tomorrow.
x_t: [ASSISTANT] What movie theater do you prefer?
x_T: [USER] Could you confirm my tickets just in case?

x': [USER] Hi, I'm looking to purchase tickets for my family.
```

$$\mathcal{L} = -\log \frac{\exp(\mathrm{d}(z_t, \mu_t))}{\exp(\mathrm{d}(z_t, \mu_t)) + \exp(\mathrm{d}(z', \mu_t))}$$

$$\mathcal{L}_N = \mathbb{E}_X \left[-\log \frac{\exp(\mathrm{d}(x_0, x_t, x_T; f_\theta))}{\sum_{(x_0, x_{t'}, x_T) \in \mathcal{B}} \exp(\mathrm{d}(x_0, x_{t'}, x_T; f_\theta))} \right], \text{ where}$$

$$\mathrm{d}(x_0, x_t, x_T; f_\theta) = -\frac{1}{2\sigma^2} \Bigg\| \underbrace{f_\theta(x_t)}_{z_t} - \underbrace{\left(1 - \frac{t}{T}\right) f_\theta(x_0) - \frac{t}{T} f_\theta(x_T)}_{} \Bigg\|_2^2$$

Generating text

- Train a model to generate the sentences conditioned on the latent embeddings

- During inference, generate text conditioned on these latent embeddings

Experiments

- Datasets
  - Wikisection: wikipedia articles on cities split by section, each article has four sections (abstract, history, geography, demographics)
  - Taskmaster-2 (TM-2) (Byrne et al., 2019) contains conversations on finding restaurants between an assistant and a user.
  - TicketTalk (Byrne et al., 2021) contains conversations on booking movie tickets between an assistant and a user. The assistant's and user's turns are similarly marked as in TM-2.
  - ROC Stories (Mostafazadeh et al., 2016) is a short 5-sentence stories dataset.

Results

- Can Time Control model local text dynamics?
- Encode two sentences x_t, x_{t+k} using their method and the baselines
- Order is shuffled, and encodings z_t, z_{t+k} are fed to a classifier

		Wikisection		TM-2		TicketTalk	
	Method	$k=5$	$k=10$	$k=5$	$k=10$	$k=5$	$k=10$
	GPT2	50.3 ± 5.8	50.2 ± 6.3	55.7 ± 5.3	63.6 ± 7.3	54.7 ± 6.1	65.0 ± 8.1
Classical methods	BERT	50.9 ± 4.9	47.8 ± 9.0	68.8 ± 3.5	80.7 ± 3.8	68.4 ± 5.1	80.4 ± 6.3
	ALBERT	49.9 ± 12.1	49.6 ± 18.0	$\mathbf{81.6 \pm 4.0}$	$\mathbf{86.1 \pm 7.3}$	$\mathbf{78.4 \pm 6.7}$	$\mathbf{89.4 \pm 3.1}$
	S-BERT	50.8 ± 6.0	48.0 ± 9.1	73.4 ± 3.5	$\mathbf{83.3 \pm 4.3}$	72.1 ± 5.3	84.2 ± 5.2
	Sim-CSE	49.1 ± 6.4	48.1 ± 8.5	75.4 ± 3.8	$\mathbf{86.2 \pm 3.9}$	$\mathbf{75.1 \pm 5.9}$	85.2 ± 3.1
	VAE (8)	49.5 ± 5.5	50.5 ± 5.1	50.5 ± 4.4	51.5 ± 6.0	49.9 ± 1.0	51.2 ± 1.0
	VAE (16)	50.1 ± 5.8	51.3 ± 4.7	48.8 ± 4.8	50.8 ± 4.9	50.1 ± 1.0	49.5 ± 1.0
	VAE (32)	50.5 ± 5.1	50.0 ± 6.0	48.0 ± 5.1	47.3 ± 5.9	50.0 ± 1.0	49.3 ± 1.0
	ID (8)	49.8 ± 5.9	50.1 ± 5.0	60.3 ± 5.2	65.2 ± 6.8	59.2 ± 1.9	66.5 ± 1.1
Implicit dynamics	ID (16)	$\mathbf{53.3 \pm 5.4}$	55.8 ± 6.2	60.5 ± 5.0	67.7 ± 6.8	60.3 ± 1.0	68.4 ± 6.4
	ID (32)	50.0 ± 5.0	50.1 ± 5.0	60.4 ± 5.3	67.6 ± 7.1	61.0 ± 1.0	67.9 ± 6.5
	BM (8)	49.8 ± 5.4	50.0 ± 5.4	49.8 ± 5.4	49.9 ± 5.2	49.7 ± 5.0	50.6 ± 5.8
$\mathrm{d}(x_t, x_{t'}; f_\theta) = -\frac{1}{2\sigma^2} \| \underbrace{f_\theta(x_{t'})}_{z_{t'}} - \underbrace{f_\theta(x_t)}_{z_t} \|_2^2$ Brownian motion	BM (16)	50.3 ± 5.5	50.5 ± 5.2	49.9 ± 4.3	51.1 ± 6.0	50.3 ± 4.6	50.8 ± 5.5
	BM (32)	49.3 ± 5.6	48.8 ± 5.8	49.5 ± 4.7	49.6 ± 5.2	49.5 ± 5.6	49.1 ± 6.1
	TC (8)	49.23 ± 5.72	48.3 ± 6.8	$\mathbf{77.6 \pm 7.8}$	$\mathbf{87.7 \pm 6.9}$	71.6 ± 2.9	82.9 ± 4.1
$\mathrm{d}(x_0, x_t, x_T; f_\theta) = -\frac{1}{2\sigma^2} \left\| \underbrace{f_\theta(x_t)}_{z_t} - \underbrace{\left(1 - \frac{t}{T}\right) f_\theta(x_0) - \frac{t}{T} f_\theta(x_T)} \right\|_2^2$ **Their method**	TC (16)	$\mathbf{57.25 \pm 5.30}$	$\mathbf{65.8 \pm 5.4}$	$\mathbf{78.2 \pm 8.1}$	$\mathbf{88.0 \pm 7.1}$	71.3 ± 3.3	82.9 ± 4.1
	TC (32)	50.1 ± 4.8	49.8 ± 5.8	$\mathbf{77.9 \pm 7.9}$	$\mathbf{87.9 \pm 7.4}$	$\mathbf{72.0 \pm 3.9}$	84.4 ± 3.9

Results

Infilling

- Five-sentence stories: x_0, x_1, x_2, x_3, x_4
- Generate $x_2 \mid x_0, x_1, x_3, x_4$
- Encode $z_0 \leftarrow x_0 || x_1$, $z_T \leftarrow x_3 || x_4$

Method	BLEU (↑)
LM	1.54 ± 0.02
ILM	3.03 ± 0.11
VAE (8)	0.75 ± 0.17
VAE (16)	0.62 ± 0.07
VAE (32)	0.03 ± 0.0
ID (8)	2.9 ± 0.3
ID (16)	0.9 ± 0.0
ID (32)	1.0 ± 0.1
TC (8)	3.80 ± 0.06
TC (16)	4.30 ± 0.02
TC (32)	$\mathbf{5.4 \pm 0.11}$

Table 2: BLEU on ground truth infill and generated sentence.

Potential next steps

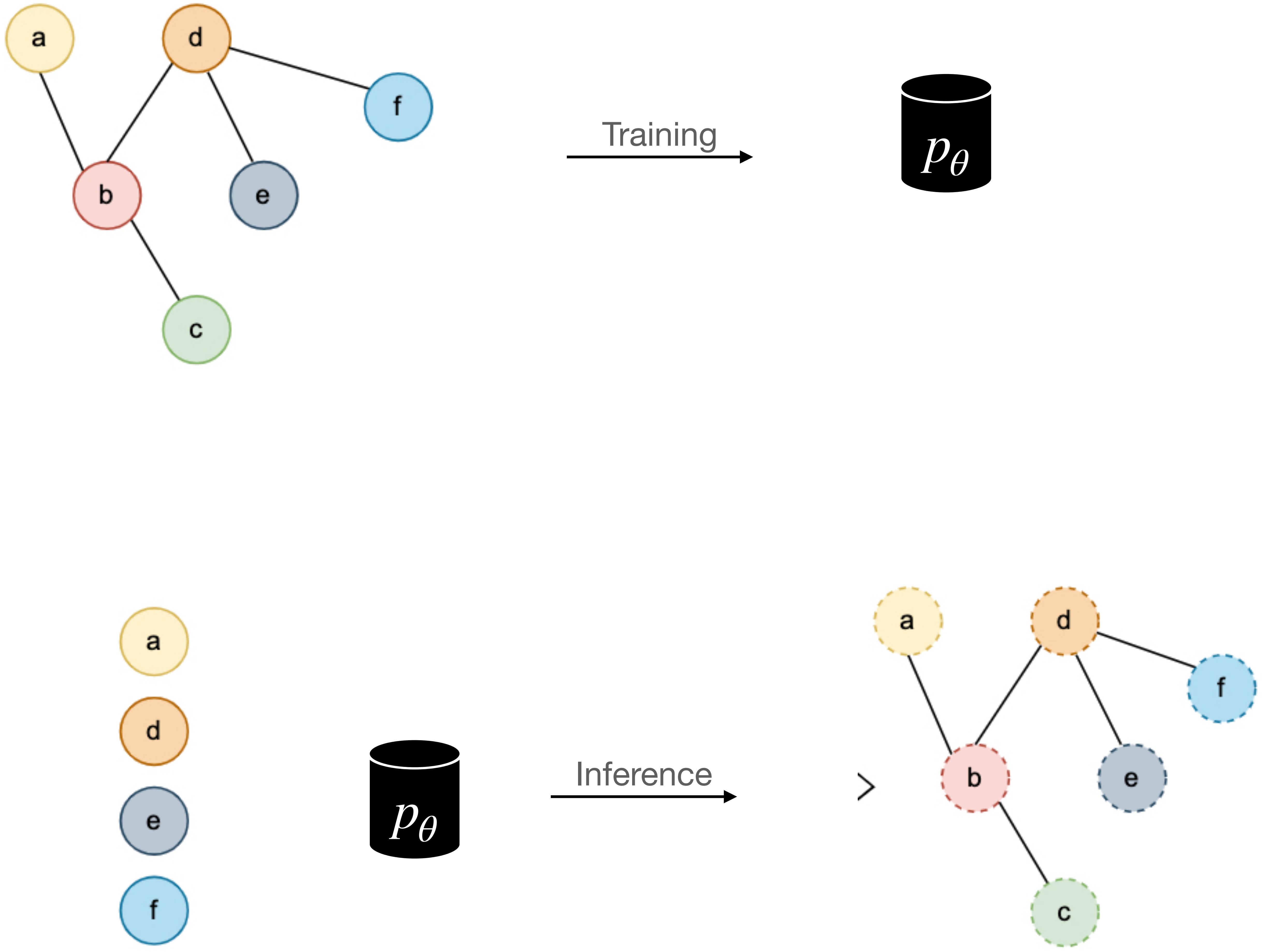
- Aligning brownian processes across sequences
- Pinning down to more than three points?

FLOWGEN: Fast and slow graph generation

Aman Madaan, Yiming Yang
(WIP)

Graph generation

- Overview: learn a model for a (typically large) graph



Initial nodes

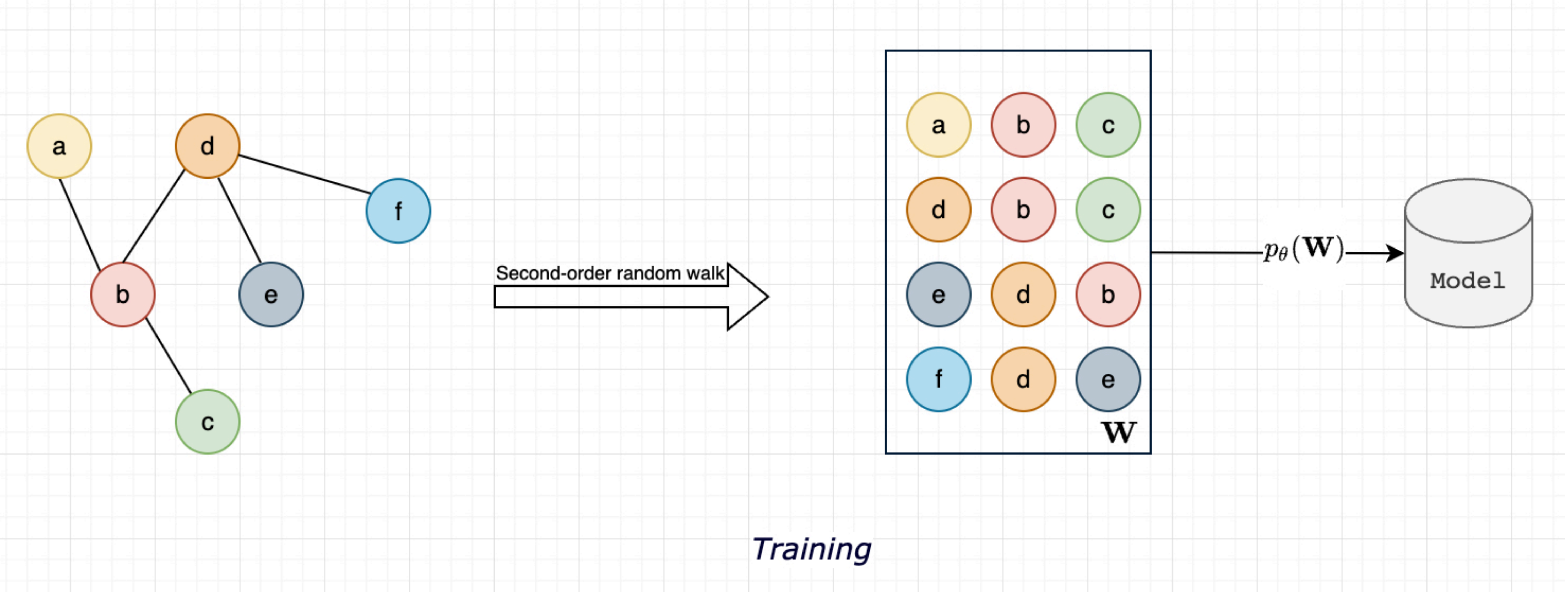
Reconstructed graph

Evaluating graph generation models

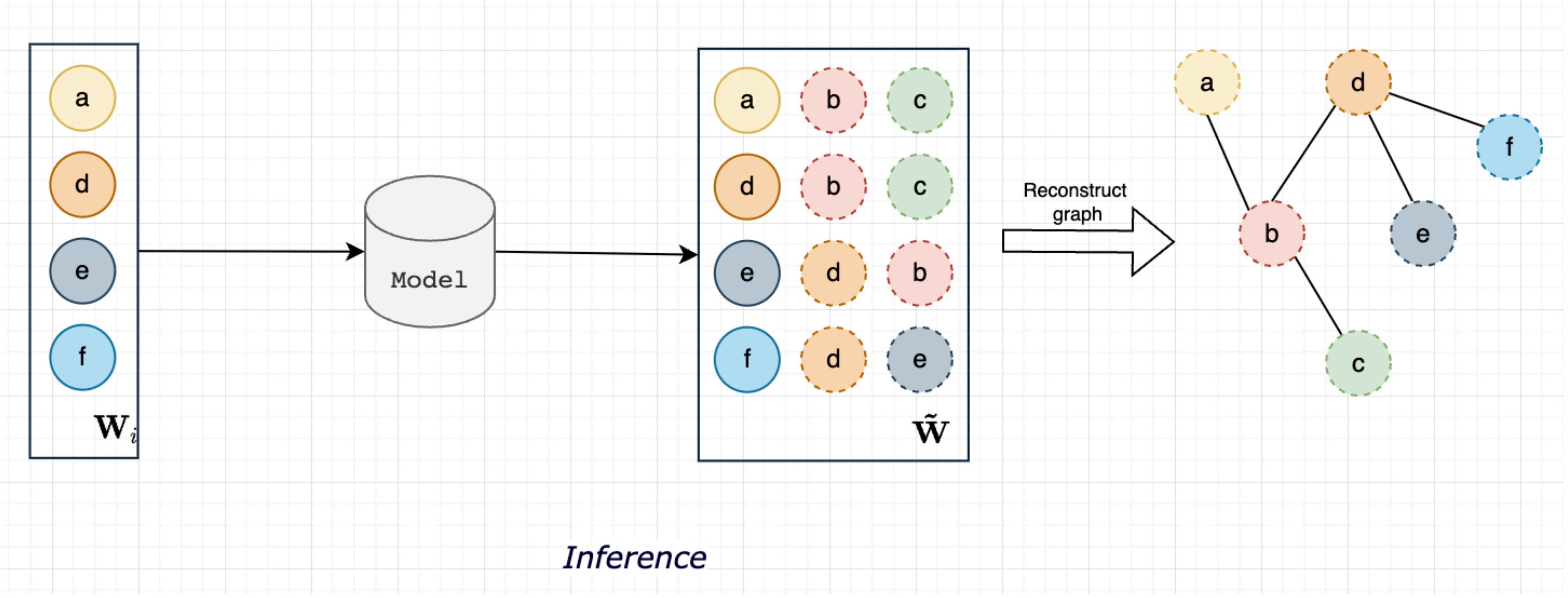
- **Structural:** how closely does **G'** *look like* **G**?
    - Max degree
    - Triangle count
    - Clustering coefficient
    - Can be gamed by a model that memorizes all the edges

- **Downstream task:**
    - Tests generalization
    - During training, entire **G** is not used and some edges are held out
    - What % of those edges are generated in **G'?**

Autoregressive graph generation (our method)



N layer transformer architecture with self-attention



Alternative interpretation: matrix completion

Autoregressive graph generation

Results

- Outperforms existing baselines + Not sensitive to hyper-parameters tuning

Method	CORAML	CITESEER	POLBLOGS
Adamic/Adar	92.16	88.69	85.43
DC-SBM	96.03	94.77	95.46
node2vec	92.19	95.29	85.10
VGAE	95.79	95.11	93.73
NetGAN	95.19	96.30	**95.51**
FLOWGEN (ours)	**96.93**	**97.03**	95.00

Table 3: Area under the curve (AUC) for link prediction. FLOWGEN outperforms or matches strong baselines.

Graph	Max. degree	Assort-ativity	Triangle Count	Power law exp.	Inter-comm unity density	Intra-comm. unity denisty	Cluster-ing coeff.	Charac. path len.
CORA-ML	240	-0.075	2,814	1.860	4.3e-4	1.7e-3	2.73e-3	5.61
Netgan	**233**	-0.066	1,588	1.793	6.0e-4	**1.4e-3**	2.44e-3	5.20
FLOWGEN (ours)	224	**-0.080**	**2,123**	**1.857**	**5.4e-4**	1.3e-3	**2.50e-3**	**5.40**

Table 2: Comparison of FLOWGEN with Netgan for structural metrics for CORAML. The ground truth values are listed in the top-row, and the value closer to the ground truth is highlighted in bold. FLOWGEN closely matches the ground truth graph on a larger number of metrics.

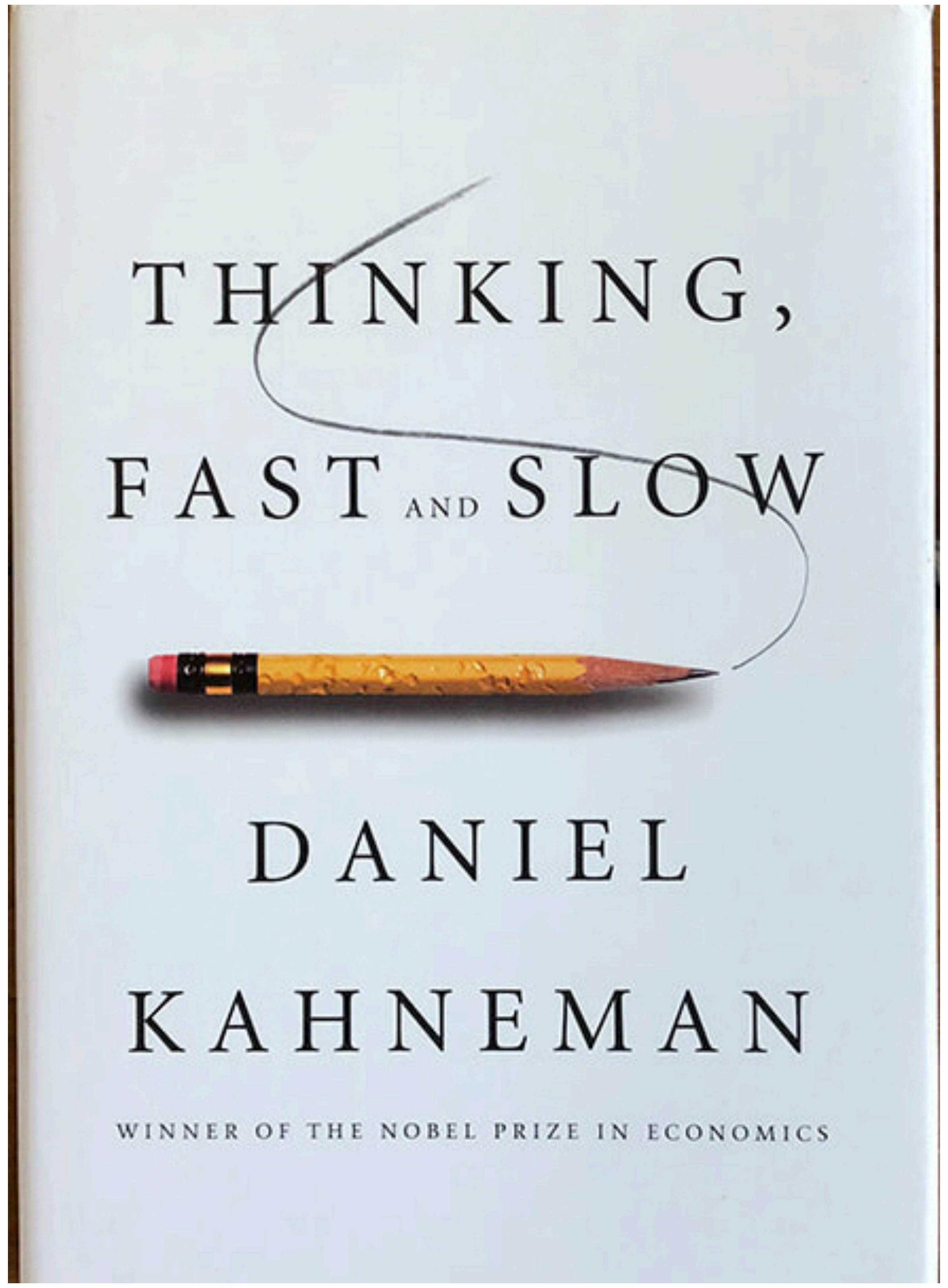
	N_{LCC}	E_{LCC}
CORAML	2,810	7,981
CITESEER	2,110	3,757
POLBLOGS	1,222	16,714

$$2 * 2 = ?$$

19 * 3 = ?

Dual-process theory of mind

- Humans use different metaphorical parts of the brain to solve problems of various difficulties:
    - System 1: ***fast***, instinctive, pattern-matcher, good for easy problems
    - System 2: ***slow***, analytical, deep-thinker, good for hard problems
- Current machine learning models:
    - Always use the same hammer for problems of all levels of difficulty
- Practical implications:
    - More efficient systems
    - Critical in the age of large language models

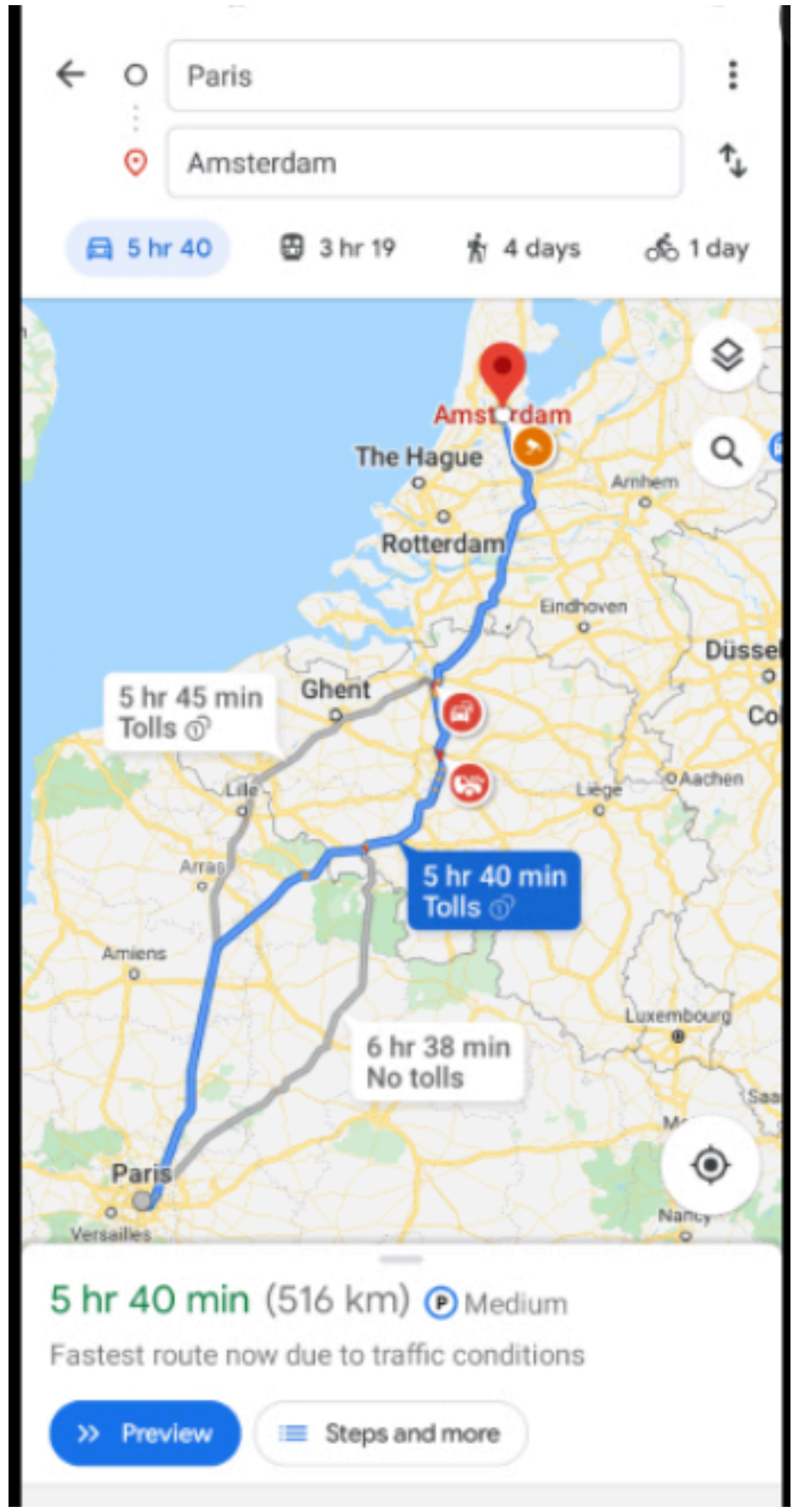


FLOWGEN

Generating graphs fast and slow

- Our method relies on using random walks for reconstructing the graph

- A random walk begins from a fixed, given node v_1
    - Generating the 2nd point requires reasoning about 2-hops $p(v_2 \mid v_1)$
    - Generating the 3rd point requires $p(v_3 \mid v_2, v_1)$
    - Intuitively, the process gets more difficult with sequence
- Need assistance only when generating walks in the later part



FLOWGEN

Overview

- Train two models: ***fast*** (transformer with 1 layer) and ***slow*** (transformer with 6 layers)
- Start random walk generation with a ***fast*** model, ***switch*** to a slow model ***when the walk is starting to feel lost***
- ***Switch***
  - Easy because of auto-regressive setup: the walk generated so far is a prefix to the slower model
- ***When the walk is starting to feel lost***
  - In general, known to be a challenging issue
  - Relates to uncertainty estimation, model calibration

How Can We Know *When* Language Models Know?
On the Calibration of Language Models for Question Answering

Zhengbao Jiang[†], **Jun Araki**[‡], **Haibo Ding**[‡], **Graham Neubig**[†]
[†]Languages Technologies Institute, Carnegie Mellon University, United States
[‡]Bosch Research, United States
{zhengbaj,gneubig}@cs.cmu.edu
{jun.araki,haibo.ding}@us.bosch.com

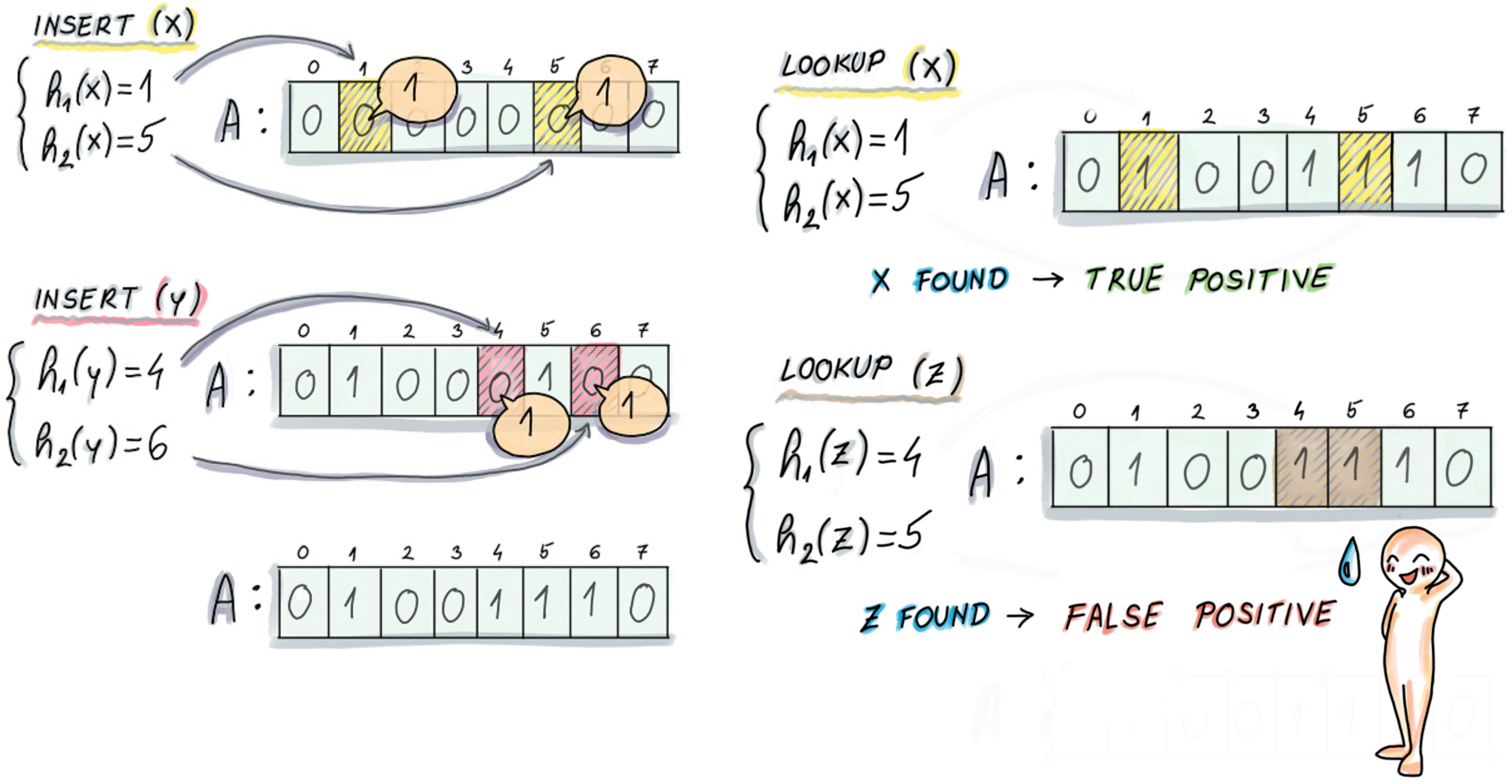
FLOWGEN

How do we know when the model is getting lost?

- Define $N = [v_i, v_{i+1}, v_{i+2}, v_{i+3}]$ to be a neighborhood of four consecutive nodes in the true graph, $\mathscr{R}$ to be all the random walks of length 4
- We want an estimate of $P(N \in \mathscr{R})$
  - For smaller graphs, $\mathscr{R}$ can be enumerated and fed to a balanced-binary tree for efficient search (aka hashmap)
  - Not an option for our case
    - Graphs are large, exponential possible $\mathscr{R}$
    - Can use distributed lookup caches, but if switch detection takes all the time, we will miss out on any gains by fast + slow interplay
- Want a quick estimate of $P(N \in \mathscr{R})$
  - Noisy estimate is okay
  - **Bloom-filters** are meant for exactly this use case
- ***Interpretation:*** random walks are brownian motion in the limit, we want to determine if a set of consecutive points was sampled from the process defining brownian motion

Bloom filters

- Probabilistic data structures for efficiently answering set-membership queries
- Parameters:
  - m bits
  - k hash functions
  - n elements to be inserted
- Tunable False positive rate: $\epsilon \approx (1 - e^{\frac{-kn}{m}})^k$

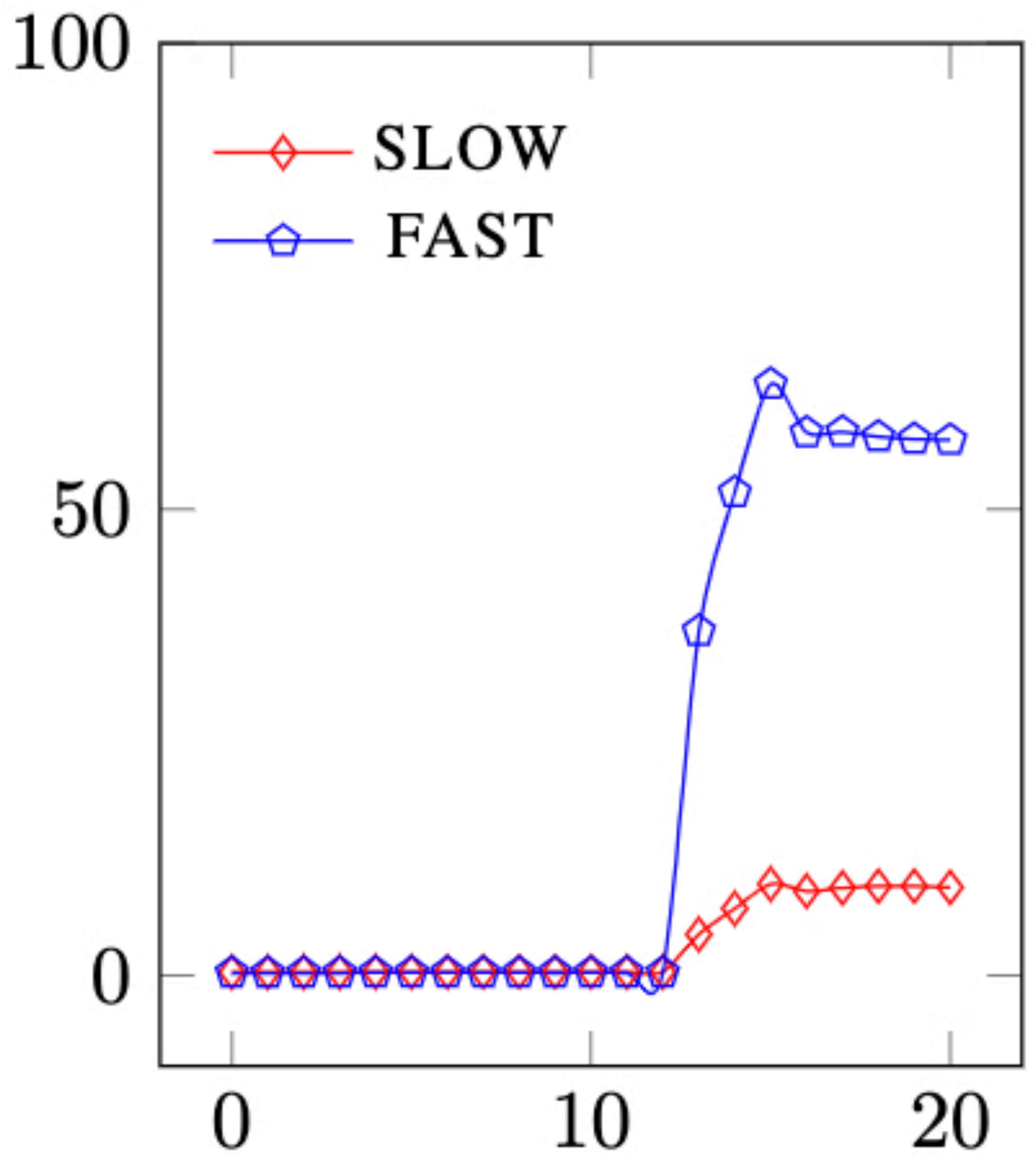


https://freecontent.manning.com/wp-content/uploads/bloom-filters_02.png

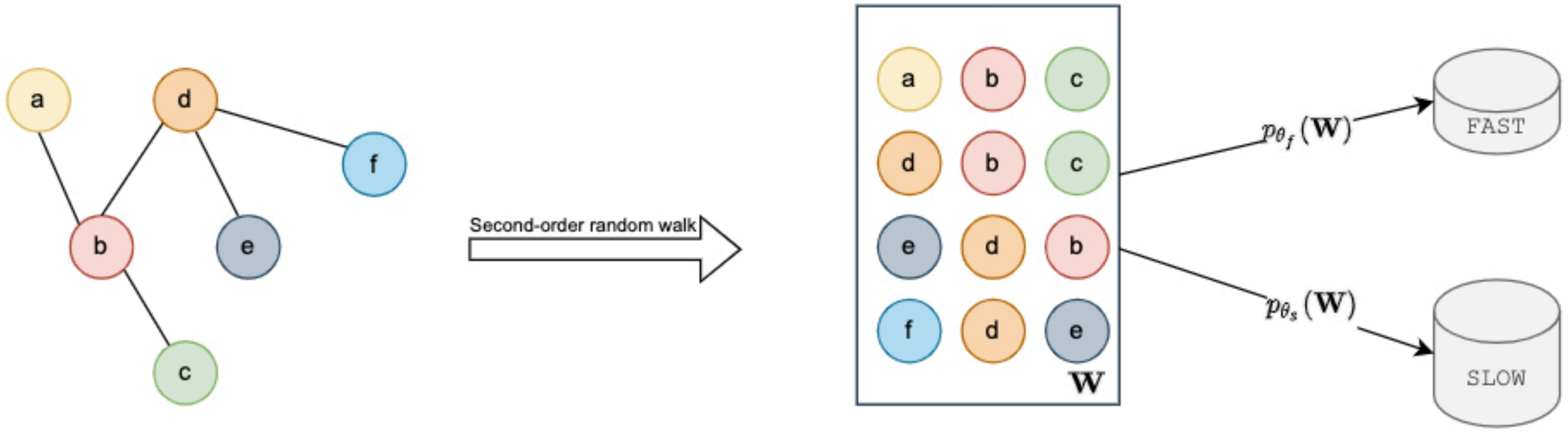
FLOWGEN

Detecting exploration

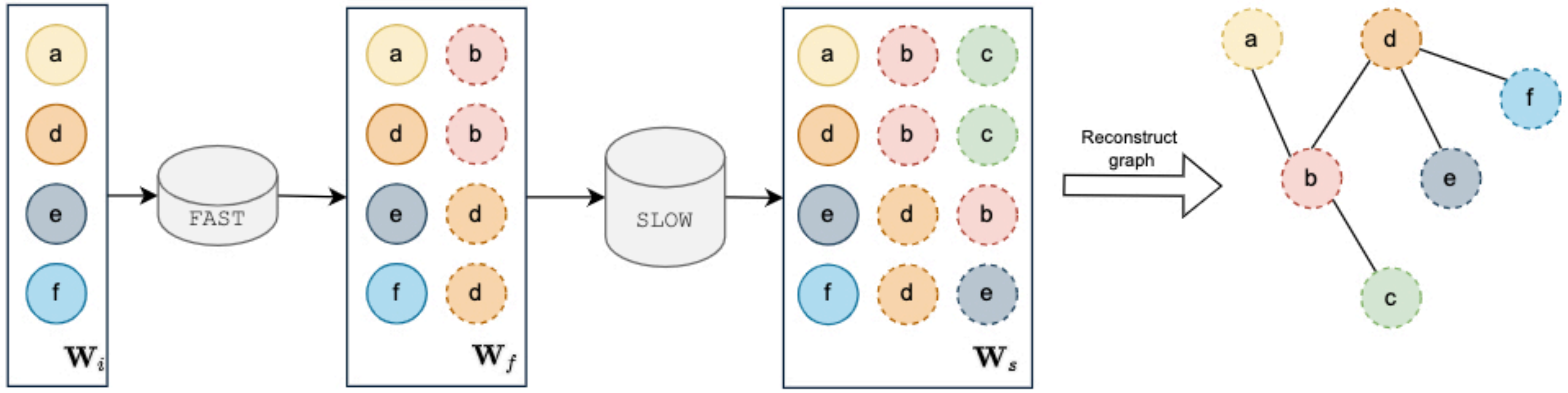
- Store a large fraction of neighborhoods in the training data in a bloom filter $\mathscr{B}$
- Approximately 1 bit per walk
    - 130x less space to store the entire graph for doing search queries
- Random walk can be in “exploring” or “exploiting” phase:
    - When a neighborhood is not found in $\mathscr{B}$, random walk is “exploring”, otherwise random walk is exploiting
    - Let $EL(N)$ be 1 if the neighborhood is likely to be exploratory
    - $\dfrac{dEL(N)}{dt}$ rate of change of exploration with time
    - Switch at $t = argmax_t \dfrac{dEL(N)}{dt}$



- Generate 10k walks from fast and slow model, detect when they switch by finding the rate of new neighborhoods



Training



Inference

Results

Same accuracy at 50% less time

	FAST	SLOW	FLOWGEN
CORAML (500k)	76.8 (288)	**92.2** (806)	90.8 (484)
CORAML (100M)	91.5 (50k)	96.7 (180k)	**96.9** (110k)
CITESEER (500k)	90.9 (313)	**94.6** (862)	93.3 (687)
CITESEER (100M)	96.1 (62k)	**96.8** (172k)	96.5 (137k)
POLBLOGS (500k)	61.9 (309)	85.4 (854)	**86.5** (686)
POLBLOGS (100M)	66.2 (48k)	**93.8** (156k)	**93.8** (108k)

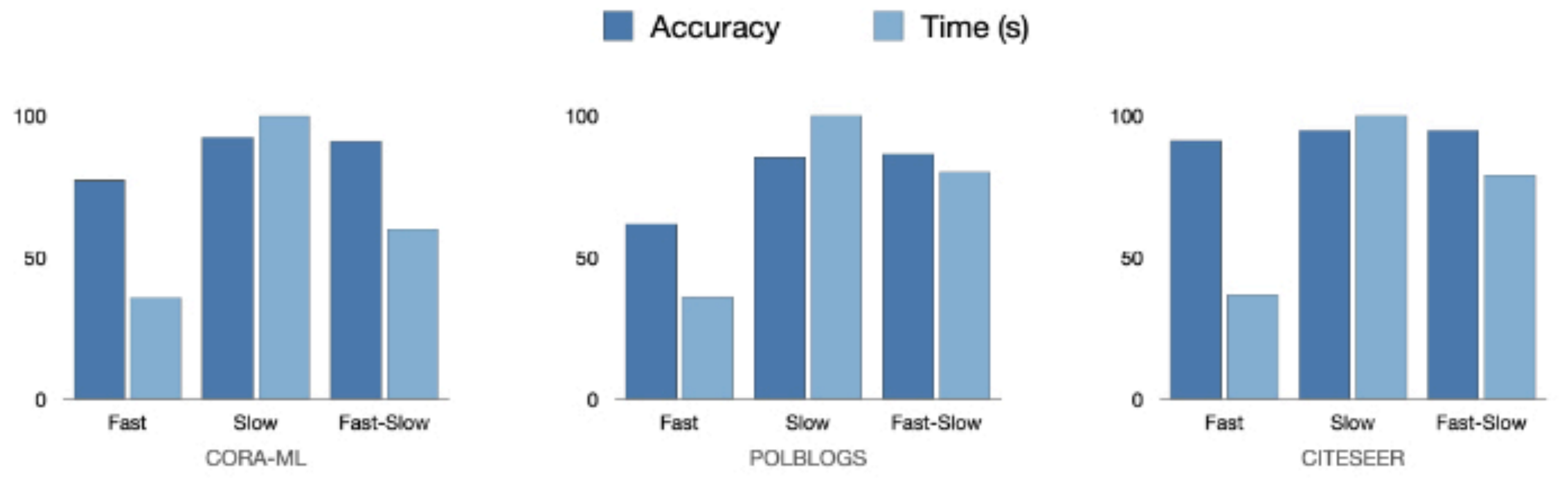
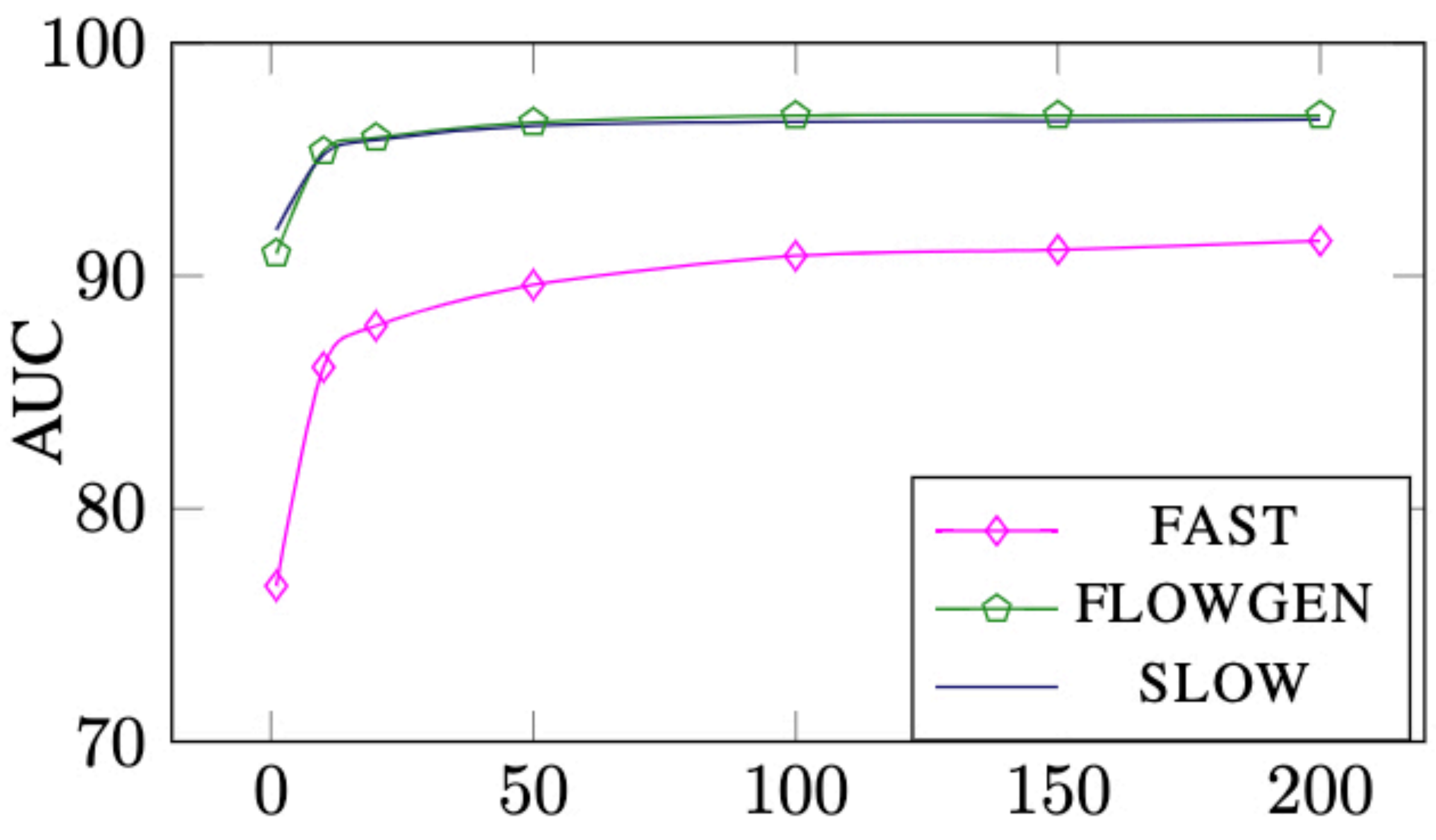
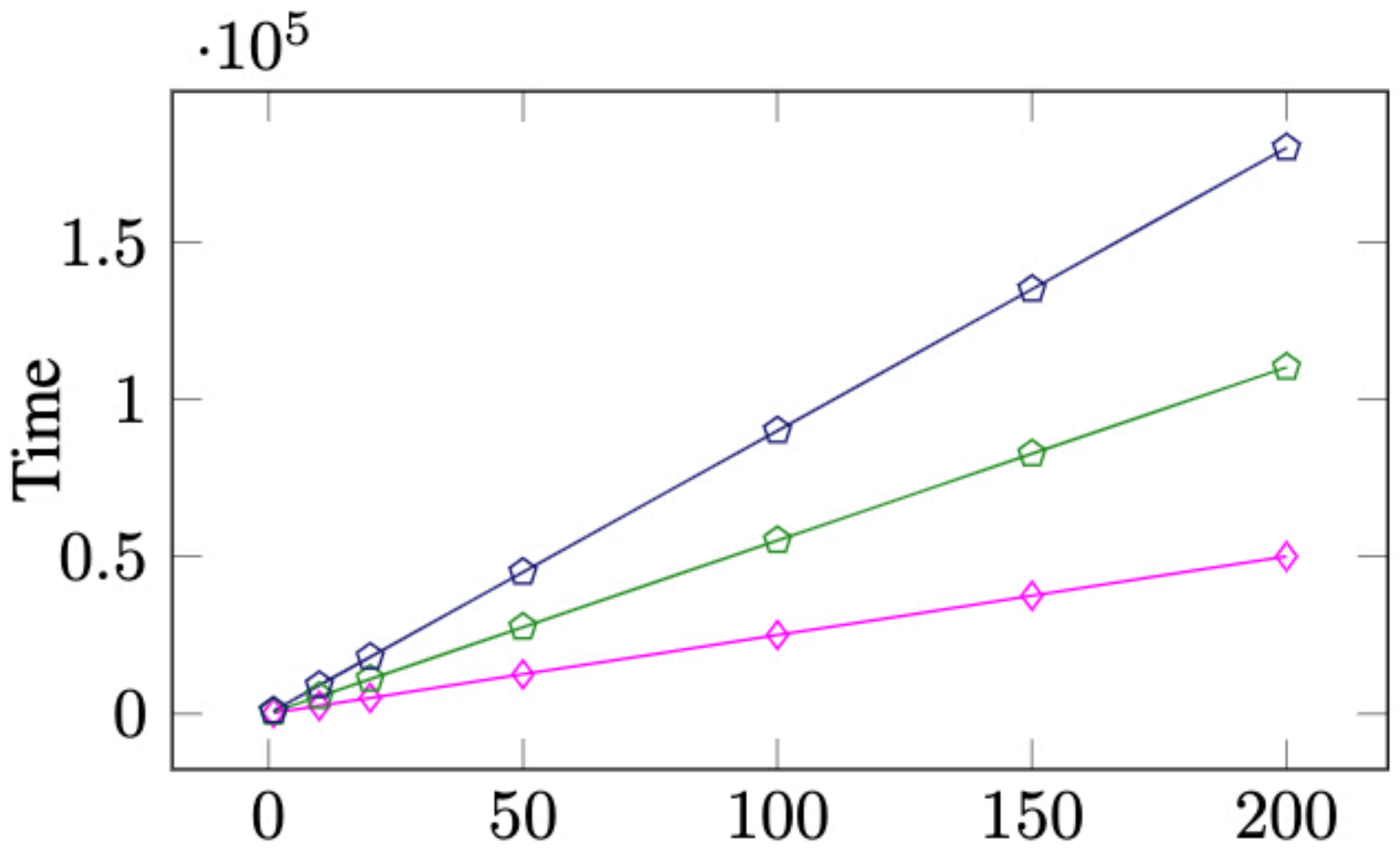


Figure 4: Average precision vs. time taken for the three settings. The FAST and SLOW model speed-accuracy trade-off is apparent: FAST model is fast but less accurate (average precision $\sim$ 75%, compared to the SLOW model which is slower but has average precision of 92%. FLOW combines the strengths of the two modes: it achieves an accuracy of 90% while being $\sim$ 50% faster than the SLOW model. Note that the time is normalized relative to SLOW (SLOW takes 100% of the time).





Take home message

- Beyond vanilla seq2seq
    - Gradient-based optimization
    - Sampling from energy models
    - Gradient-free sampling
    - Generation as brownian process
    - Fast-slow generation with bloom filters
- Next steps:
    - Gradient based methods that actually work
    - Fitting a brownian motion model to the random walk for determining the switch point more efficiently